# Semantic-based Framework for the Generation of Travel Demand

Gregory L. Albiston

School of Science and Technology

A thesis submitted in partial fulfilment of the requirements of Nottingham Trent University for the degree of

*Doctor of Philosophy*

January 2019

# Copyright statement

# Acknowledgements

Firstly, I would like to acknowledge my gratitude to my supervisor Dr. Evtim Peytchev for his support of my Ph.D. study and related research. I would also like to acknowledge my co-supervisor Dr. Taha Osman for his insight, discussion, and feedback during my research and preparation of this thesis. Finally, the companionship and support of my partner, friends, family, and peers have helped sustain me throughout the years of study and it is this for which I am most grateful.

"All models are wrong but some are useful."
Prof. George E. P. Box, statistician

"Everything changes and nothing stands still."
Heraclitus of Ephesus, philosopher

# List of Acronyms

| | |
|---|---|
| AAA | Anyone can say Anything about Any topic |
| ANN | Artificial Neural Network |
| API | Application Programming Interface |
| CEMDAP | Comprehensive Econometric Micro-simulator for Daily Activity-travel Patterns |
| CLI | Command Line Interface |
| CPM | Computational Process Model |
| CSV | Comma Separated Values |
| CWA | Closed World Assumption |
| ETL | Extract Transform Load |
| GIS | Geographic Information System |
| GML | Geographic Markup Language |
| GPS | Global Positioning System |
| GUI | Graphical User Interface |
| HTTP | Hypertext Transfer Protocol |
| INSPIRE | Infrastructure for Spatial Information in Europe |
| IRI | Internationalized Resource Identifier |
| JADE | Java Agent DEvelopment Framework |
| JSON | JavaScript Object Notation |
| MATSim | Multi-Agent Transport Simulation Toolkit |
| NNA | Nonunique Naming Assumption |
| OGC | Open Geospatial Consortium |
| OLO | Ordered List Ontology |

| | |
|---|---|
| ONS | Office of National Statistics |
| OS | Ordnance Survey |
| OSM | Open Street Map |
| OWA | Open World Assumption |
| OWL | Web Ontology Language |
| RDF | Resource Description Framework |
| RDFS | Resource Description Framework Schema |
| REST | Representational State Transfer |
| RMI | Remote Method Invocation |
| RPC | Remote Procedure Call |
| RUM | Random Utility Model |
| SAWSDL | Semantic Annotations for WSDL and XML Schema |
| SHACL | Shapes Constraint Language |
| SOAP | Simple Object Access Protocol |
| SPARQL | SPARQL Protocol and RDF Query Language |
| SPIN | SPARQL Inference Notation |
| SQL | Structure Query Language |
| SUMO | Simulation of Urban Mobility |
| SWRL | Semantic Web Rule Language |
| TRANSIMS | TRansportation ANalysis SIMulation System |
| TSV | Tab Separated Values |
| URI | Unique Resource Identifier |
| URL | Unique Resource Location |
| URN | Unique Resource Name |
| XML | Extensible Markup Language |
| XSLT | Extensible Stylesheet Language Transformations |
| WWW | World Wide Wide |
| WKT | Well Known Text |

# Abstract

Traffic and transportation have a wide-ranging impact on the daily lives of the human population and society. Activity-based travel demand generation models and traffic simulators are tools that have been developed to investigate traffic and transport problems and assist in developing solutions.

The closer modelling of human behaviour, the emergence of new technologies and the availability of more detailed datasets is leading to greater modelling complexity. The robustness of conclusions in investigations is supported by comparison of multiple techniques and models yet variations in the platform, data requirements and dataset availability present barriers to their breadth. This thesis investigates the development of a Semantic Web framework for activity-based travel demand generation.

It is proposed that the application of a knowledge-based approach and development of an orchestrating framework will enable a loosely coupled modular architecture. This approach will reduce the burden in preparing and accessing datasets through the construction of a platform-independent knowledge-base and facilitate switching between modules and datasets.

The principal contributions of this work are the application of a knowledge-based approach to travel demand generation; the development of a Semantic-based framework to control the configuration of the process and the design; and demonstration of the Semantic-based framework through the implementation and evaluation of the modular travel demand generation process, including integration with two third-party traffic simulators.

The investigation found that the proposed approach can be successfully applied to model and control the travel demand generation process. Multiple configurations were explored, including utilising network communications, and found that this had a noticeable impact on execution duration but also the potential for mitigation through distributed computing.

This presents the opportunity for an online infrastructure of datasets and module implementations for travel demand generation that users can select and access through the framework. This infrastructure would remove the need for ad hoc interfaces; data format conversion or platform dependence to facilitate the process of traffic modelling becoming quicker and more robust.

# Table of Contents

# List of Figures

# List of Tables

# Listings

# Chapter 1

# Introduction

This thesis proposes a core data model for travel demand modelling and defines a framework to support the selection of alternative techniques and data sources. It is proposed that the core data model for activity-based models and traffic simulations will improve the interoperability of data sources and modelling techniques through a common knowledge-base. The supporting framework will enable users to configure and control the knowledge-base and the applied techniques through a consistent mechanism. It is intended that these improvements will reduce the time and resources required to assemble a real-world traffic simulation and facilitate greater comparison and validation between models and implementations.

In this chapter, there will be a discussion of the motivational context for this work along with the primary features of the traffic and transport modelling domain and Semantic Web technologies. There will then be an outline of the problem and proposed solution before considering the research questions and contributions described in the remainder of the thesis.

## 1.1 Motivation

The consequences of traffic and transportation are directly and indirectly experienced by the human population every day. This experience can have consequences in terms of travel time or financial cost but can also have an impact upon an individual's health, environment and economic opportunity [1]. Therefore, it is

important to develop high-quality solutions to the problems faced. The understanding of traveller behaviour is seen as crucial to managing and developing transport infrastructure and the adaptation to emerging technologies [2]. One tool in developing solutions to these problems are travel demand models and traffic simulations, which seek to create representations of human travel within the physical environment to evaluate the likely outcome of transport policy decisions. These systems also provide input into other domains for modelling and informing policy decisions and so have a wide-ranging utilisation and impact [3].

To address these matters there has been a multi-decade research into the modelling and simulation of travel and transport. The complexity of the travel experience, the wide range of contexts, the involvement of human behaviour and the emergence of new technology has led to the development of numerous models, techniques and implementations. The stages of the process have often been addressed on an individual basis requiring additional efforts to integrate together or have been developed as proprietary closed solutions that are not widely available [4].

At each stage of the process there are numerous design and implementation decisions, which can produce diverse, but valid, outcomes that are best understood through the consideration of multiple scenarios, repetitions and implementations. Each stage has diverse implementation decisions, technologies and designs that make comparison and utilisation difficult. The systems produced have a complexity and cost in time and resources that hinders the comparison between approaches, adoption of best practice, development of new techniques.

Moreover, the conceptual and practical need exists for all models to be imperfect simplifications of the physical environment and social behaviour they target [5]. The need for verification and comparison between these systems and their models will remain regardless of their increasing complexity and the computational resources available to solve them. This need will likely increase as greater complexity leads to greater challenges in the explanation of outcomes [6].

This lack of integration and comparison exists despite the fundamental similarity between models and techniques. Therefore, the practical integration should be achievable with several examples performed on a one-to-one basis. Yet, this has not resolved the ongoing issue of integrating many models, switching between

these models and the models being available for further re-use. Achieving an integration requires overcoming a host of practical implementation issues including converting between data formats and managing multiple operating systems, database platforms and programming languages.

The proposal and introduction of overarching interfaces that satisfies all the potential variations that exist is complex, difficult to maintain and highly unlikely to be sustainable. Instead a solution is needed that orientates the different models to the fundamental concepts of the domain and then allows the flexibility to adjust for variations between models without them conflicting.

It is proposed that Semantic Web technologies can be used to develop a framework that will assist in integrating datasets together into a coherent knowledge-base. Upon this knowledge-base can then be applied the modular components required to generate and simulate travel demand. The extensible nature of the Semantic Web will enable modules with different design assumptions and data requirements to operate alongside each other upon the common knowledge-base. Therefore, reducing the burden for comparison between implementations.

The Semantic Web is designed upon principles of open, structured and re-usable data to facilitate machine to machine interactions. These principles will enable the publication of high-quality input datasets for direct consumption and utilisation as part of the local knowledge-base, rather than the current reliance on human-driven sourcing, cleaning and conversion. Datasets can be published at a national level with users able to select subsets based upon their interest area.

Further, the online design of Semantic Web technologies will enable module components to operate upon remotely held data, such that the knowledge-base need no longer be a single physical entity. This would enable the modular components of the process to be remote services that are directed to the data to act upon by the user, rather than being locally installed and configured instances. This would alleviate the technical and resource burden from users, facilitate reproducible results and allow domain-experts to develop best practice techniques. Removing these burdens, raising the quality of data and encouraging comparison between techniques will allow greater focus upon effective solutions and help alleviate the risk of error in outcomes.

The diversity of datasets, design choices and modelling approaches means

that a single unified data model and set of component interfaces is a complex objective to achieve and likely to be unsustainable as new data sources and conceptual models emerge. Therefore, a more flexible approach would be suggested that enables users to adapt data and modules to their knowledge-base as they investigate alternative scenarios and techniques.

This work proposes a Semantic Web-based framework that identifies data structures for fundamental components of activity-based travel demand models. It is proposed that the semantic modelling of these concepts into a structured knowledge-base will allow the development of modular components that can be more easily interchanged, while still allowing users control over the organisation and structure of their experimental scenarios.

It is intended that the modelling process can become less burdensome and more flexible for incorporating new concepts and approaches based upon the knowledge-base. Further, it is proposed that a Semantic Web basis will enable the assembly of models and simulations from both local and online sources to enable quicker modelling, knowledge inferencing and improved comparability of results.

In other words, a knowledge model of the travel demand problem domain will allow models to be aligned around their similarities. Each model can then expand this knowledge model for their own purposes and focus. Users may still have to align models, but by mediating knowledge concepts rather than implementation details. The introduction of more flexible integration can then allow the multiple stages of the process to be further decomposed from a few large monolithic steps to many smaller steps. This can then allow greater comparison of the alternative design choices between models.

## 1.2 Traffic and Transport Modelling Domain

Traffic and transport planning is a wide-ranging field that impacts on the daily lives of most members of society through transport congestion, infrastructure investment, safety, pollution etc. [1, 3] The understanding of traveller behaviour is seen as crucial to managing and developing transport infrastructure and adapting to emerging technologies. The Traffic and Transport Modelling domain seek to

support this by modelling the complexity of the physical environment and human behaviour.

This section provides further detail relating to activity-based demand models. It outlines how human travel can be defined and described; some of the general terminology used for travel demand models; the general considerations of activity-based demand models; and the main approaches to developing activity-based demand models.

### 1.2.1 Representation of Journeys

The conceptual evolution of Travel Demand Modelling from trips to activities can be seen in Figure 1.1, based upon [7]. In trip-based models, the focus has been on the travel portion of a person's day as they commuted to and from employment or education, with other activities a secondary consideration. A broader perspective of the journey is provided by considering the continuous time period of a tour, encompassing both travel and activity. In an activity-based model, the travel and activity are considered separately with timing constraints imposed by the type of activity being undertaken.


Figure 1.1: Diagram of journey representations indicating the level of detail required in their construction. (Based upon [7].)

The three representations can be further described as follows:

- Trip Based: A trip is a sequence of one or more stages, continuous movement using one mode of transport or vehicle, between two activities that are in different locations. The simplest representation and easiest to model but lacking the depth and nuance of human activities.

- Tour Based: A tour is a sequence of trips starting and ending at the same location. A trip chain may also be modelled which ends at a different location to the start. This approach captures the reliance of trips upon each other and that there may be sub-trips which take place beyond the initial activity, such as travel for lunch at work.

- Activity Based: An activity is a continuous interaction with the physical environment, service or person by an individual. A schedule is developed for individuals, grouped into households, that takes into consideration the time and location constraints of household members and negotiates the conflicting journey demands. A fuller representation of the process for journey planning and scheduling.

This focus on activities, termed Activity Based Modelling, introduces a need for a broader range of information about the potential activities and their locations. Further there is a need to consider human behaviour in selecting and scheduling activities that can be limited by fixed or flexible constraints, e.g. shop opening hours or contracted employment hours, as well as the negotiation amongst individuals within households for the use of resources, e.g. vehicle availability, and skills, e.g. adult drivers to transport adult non-drivers and children. It should also be considered that once a plan has been established it is not fixed but subject to change due to travel delays and other factors which means there is feedback and review of the plan as it progresses over a day.

The advantages of activity-based models include developing behavioural realism, integrity between components, greater spatial and temporal resolutions, and supporting dis-aggregate traffic micro-simulators. However, due to the complexity of the domain and ongoing research to identify definitive approaches, these

objectives have been met to varying degrees through a number of different implementations [8, 9]. It has also been commented that the modelling of human behaviour is lacking in most transport models [4]. The main design approaches for activity-based models are discussed further in Section 1.2.4 after discussion of modelling terminology and considerations in the following sections.

### 1.2.2 Terminology of Travel Demand Models

The modelling of traffic and transport has developed over several decades with the terminology used to describe its components also developing. This section outlines the concepts and terms related to describing travel and activity as used in general for travel demand models [7, 10].

- Activity: a continuous interaction with the physical environment, service or person for a time interval. This includes any idle/non-travel waiting before or during the activity.
- Stage: a continuous movement with one mode of transport, and one vehicle (if used). It includes any idle waiting immediately before or during the movement.
- Trip: a continuous sequence of stages between two activities.
- Tour: a sequence of trips starting and ending at the same location.
- Journey: a sequence of trips starting and ending at the reference location of the person.
- Reference Location: the location from which a person starts and ends their journey. Typically the person's main residence, but could be multiple locations for holiday home owners, stop-over commuters, away from home students or children of separated parents.

Figure 1.2 illustrates these terms when considering a typical routine for a person in employment followed by a leisure activity. The person's commute to and from work is undertaken by transitioning from one mode to another, e.g.

walking to a bus stop and catching a bus. In this figure, there is no consideration of the modes or timings of the journeys or activities.

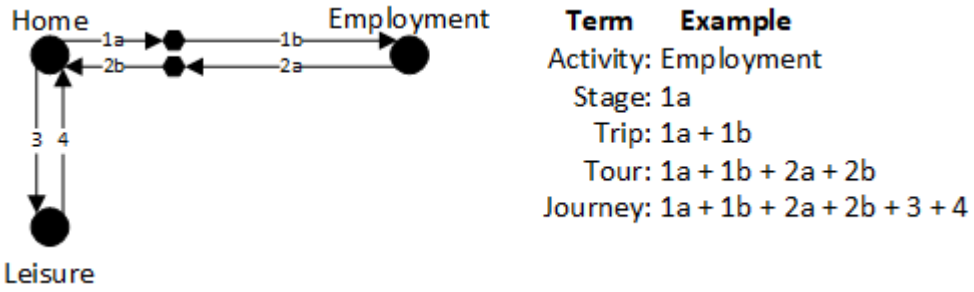


Figure 1.2: Diagram of a daily commute followed by leisure activity to illustrate travel terminology.

### 1.2.3 Modelling Considerations of Activity-Based Demand Models

The consideration of activities as the source of travel demand presents several modelling aspects [7, 9, 10]. The general term *activity* can encompass the whole scope of human undertakings. Therefore, activities are not all equal in priority, performed by all people or executed in the same order. The characteristics of activities will vary in duration, proximity, occurrence, capacity and financial cost. Certain activities are undertaken in isolation for short-term needs while others form part of long-term projects to satisfy personal objectives. Activities can be performed solo or coordinated within a group of individuals, e.g. familial households or commercial organisations. The ordering of activities can also be influenced by group dynamics with one member being required to escort another member to an activity prior to undertaking their own activity, e.g. parents taking children to school.

These different aspects of activities introduce multiple considerations. It is impractical to itemise and define all potential activities and therefore groupings of the most important are required. A criticism of previous travel demand models has been a focus upon commuter activities [11, 12]. This focus is despite recent research for trip purposes and distance travelled showing commuting shar-

ing an equivalent proportion with other activities, as shown in Table 1.1, and that approximately 5% of the UK population work from home [13], so will not be commuting but still undertake other activities. These broad categories could also include more specific activity types that a user may wish to explore.

| Trip Purpose | Trips (%) | Distance (%) |
|---|---|---|
| Leisure inc. entertainment, holidays, sport and day trips | 17 | 21 |
| Commuting | 15 | 20 |
| Visit Friends | 15 | 19 |
| Personal business and other escort | 18 | 14 |
| Shopping | 19 | 11 |
| Personal trips in course of work | 3 | 10 |
| Education inc. escort | 12 | 5 |

Table 1.1: Table of trip purpose by proportion of average number of trips and distance travelled. [14]

Broadening the range of activities under consideration also has an impact on the time periods of modelling. The primary focus upon commuter, freight and school journeys places an emphasis on the weekday morning and afternoon peak periods [11, 12]. Yet research into sources of travel demand shows the breadth of activities taking place over the day [15], as illustrated in Figure 1.3 and Figure 1.4.

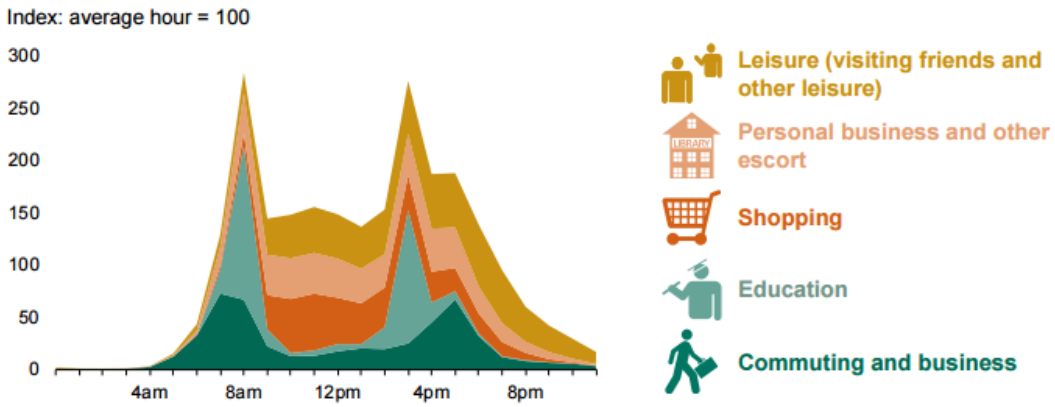


Figure 1.3: Graph of trips in progress by start time and purpose, Monday to Friday. [15]

Both figures show trip survey data for the United Kingdom and clearly follow the classic *M-shaped curve* of travel demand [16]. This *M-shaped curve* exists

during the working week when there is low demand during the morning, midday, evening and night, but peaks and troughs around the two rush-hour periods in the morning and afternoon. Figure 1.4 also shows the weekend days with the peak occurring around midday and so producing a different travel demand.

It should be noted that these distributions have been indexed against the average volume of trips and so do not show the absolute volumes between weekday and weekend, i.e. many more trips could be taking place during the weekdays than the weekend. It is also worth noting that Figure 1.3 has a large number of trips attributed to the **education** activity, which is a seasonal activity. During holiday periods, e.g. summer and New Year, those in education will engage in other activities with different location and timings. These factors highlight that more sophisticated travel models must represent a range of activities over a wide proportion of the day with intra-day, inter-day and inter-week variations.

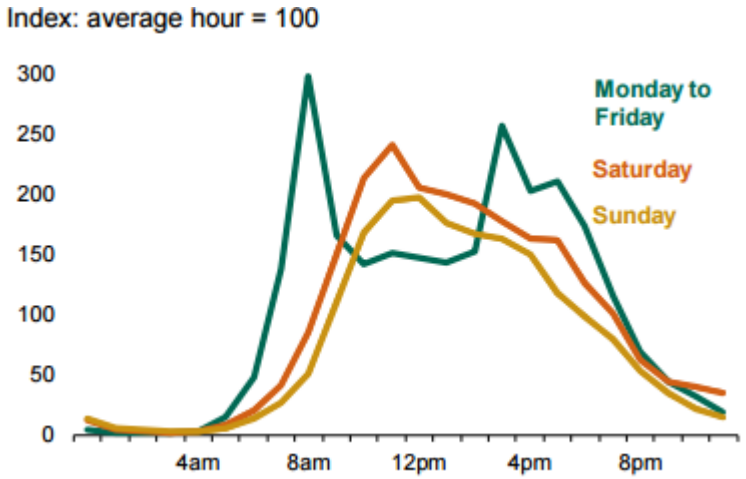


Figure 1.4: Graph of trips in progress by time of data and day of week. [15]

These factors are further complemented by other decision making influences. Many activities can only take place at a location when it is open or during certain periods of the day, e.g. daylight hours. These physical locations will also have a finite capacity and may have an associated financial cost which affects their popularity. The popularity of a location may also have less quantifiable factors that can draw individuals to it regardless of closer options that would be more expedient, e.g. service quality or reputation. Therefore, there are practical limits

to the modelling realism.

### 1.2.4 Types of Activity-Based Demand Models

The development of activity-based demand models represents a paradigm shift in the approach to modelling travel demand. The developed approaches to implementing this paradigm are diverse as there are a range of design considerations to model, as described in Section 1.2.3.

The process of traffic modelling of real-world scenarios requires a number of stages that can generally be categorised as data collection, population synthesis, demand modelling and traffic simulation as illustrated in Figure 1.5. At each stage, a range of options and techniques are available to the modeller, e.g. data formats, data granularity and modelling parameters, that require adaptation to integrate.

The burden of adapting and aligning between the stages limits innovation and quality improvements despite fundamental components of the domain existing [17]. The specific data requirements vary between implementation and design approaches, but the general requirements are regarded as being similar across activity-based and trip-based models [9]. Broadly these requirements are travel diaries, population and household census, land-use parcel data and transport network infrastructure, i.e. roads and public transit connections.

In an ideal scenario, every individual is able to complete all trips with maximum efficiency, i.e. shortest travel time, and minimal inconvenience, i.e. departing at the time and using the mode of their choice. However, the finite capacity of the road network and public transit services means that this is not always possible. Therefore, an individual must make a behavioural response in order to compensate or adjust to select an alternative trip. These responses have been identified as [18]:

- Ignore: continue the trip as planned.
- Negation: abandon the trip.
- Modal: switch from current mode to an alternative.

- Spatial: select an alternative route.
- Temporal: change the departure time.

Travel demand models seek to incorporate these behavioural responses through the features and design principles that they implement. Below are brief outlines of the predominant approaches for activity-based travel demand models although implemented models may incorporate multiple approaches [8, 19].

The unifying basis of these different modelling approaches are the activities that a human is undertaking in their daily lives. It is proposed that a common knowledge-base could support the data requirements of multiple demand model implementations to enable common concepts, e.g. population, road network, and locations, to be re-used while also providing implementation specific data.


Figure 1.5: Basic integrated activity-based micro-simulation components. (Based upon [9].)

#### 1.2.4.1 Constraints Based

A schedule is developed and checked for its feasibility according to travel and time constraints. Space-time prisms examine the feasibility of travelling between two locations in the specified time-frame according to maximum speed. A set of

activities are provided with timings, duration and tolerances. All combinations of route, mode and locations between the activities are found as a sequence of travel. A sequence that does not violate the constraints of the activities and travel times is then selected.

#### 1.2.4.2 Discrete Choice/Econometric

A person is assumed to be an entirely rational entity that chooses from a finite set of probability weighted choices. Typically, the probabilities are calculated to maximise utility using attributes related to the choice and person. Taking the case of a single choice as in Equation (1), an individual ($J$) has multiple influencing attributes ($x$). The fitted model's observed coefficients ($\beta^{\tau}$) are multiplied by the attributes and summed to derive the observed value ($V$). Negative coefficients are applied when a smaller attribute value is preferred e.g. cost, time or distance.

$$U_J = \beta_J^{\tau} x_J + \epsilon_J = V_J + \epsilon_J \tag{1}$$

$$P_i = \frac{e^{V_i}}{\sum_j e^{V_j}} \tag{2}$$

This observed value ($V$) and the unknown error term of unobserved variables ($\epsilon$) form the choice utility ($U$). The probability of selecting a choice ($i$) is found from the sum of the choice set for an individual ($j$) as in Equation (2), where in a multinomial logit model the closed form probability does not require the error terms and follows a logit probability [9]. A single choice probability is the exponential of its observed value normalised across the sum of exponential observed values of all choices in the set.

#### 1.2.4.3 Computational Process Model (CPM)

Computational Process Models, also known as rules-based approaches, were introduced as an alternative to the unrealistic behavioural assumption of utility maximisation. The basis for decision making is derived from heuristic responses so that behaviour is more similar to habitual rather than self-optimising. The rules are activated by contextual variables through an *if... then...* structure.

A ruleset is considered *complete* if a rule activates for all variable cases and *consistent* if a single rule activates in each case [20]. Semantic Web rules engines and reasoners support the description and processing of rules in a range of languages. This approach allows exploration of how a decision maker formulates and executes the schedules by capturing the explicit scheduling constraints. They are able to model interdependent decisions as well as behavioural principles. The modelling focus tends to be on activity scheduling rather than activity generation. They have further been categorised into weak and strong CPMs [21].

- Weak CPM: Apply heuristics consisting of a sequential or partially sequential decision making process. There is assumption of utility maximisation, or other rational method, at the level of individual decision steps, e.g. [22–24].
- Strong CPM: Employ a production system, or other rule-based approach, at the level of individual decision steps, e.g. [20, 25–28]. The production systems are often modelled based upon decision trees [20] or heuristic rules [25, 28].

#### 1.2.4.4 Agent-Based

An extension to activity-based models is incorporating agent-based design to further explicitly model individuals as entities which interact and respond to the dynamic environment to produce emergent behaviour [19]. This is in contrast to the passive continuation of the predetermined plan of earlier approaches. In principle, this re-planning process could be achieved in other approaches but is not highlighted in the examined literature. The agent-based design has been explored to address shortcomings in activity-based models through improved modelling of emergent phenomena, population heterogeneity and complex behaviour and learning [19].

The agent paradigm seeks to model autonomous individuals in the domain of interest. The state and behaviour of these individuals flexibly change based on previous experience and the contextual environment. Humans display agency and conceptual models have applied terms such as *mind* and *body* to encapsulate

*behaviour*, *actions*, *sensors* and *effectors* [29]. These agents are partitioned into *regions* that forms the environment within which they communicate and interact with other agents. In a holonic agent-system the agents can themselves be composed of other agents whose actions fulfil their own goals and those of the over-arching agents [30]. This, therefore, defines abstract entities as agents e.g. commercial organisations are agents composed of their employees. By defining discrete individuals and regions these systems can be organised and executed as distributed systems with agents transferring from region to region. The focus on individual entities in the environment aligns with the micro-simulation paradigm.

The advantages of agent-based modelling have been found in situations where complex interactions are taking place between agents; there is an important spatial component; the agent population has heterogeneous characteristics and agents exhibit complex behaviours involving learning and adaptation. Therefore, the approach has been applied to the traffic and transport domain. However, challenges are presented in the computational complexity of modelling large systems; the outcome patterns of interactions are unpredictable; and predicting overall system behaviour is extremely difficult due to the strong likelihood of emergent behaviour. [19]

The definition of agent-based systems in traffic and transport has been implemented in a variety of different manners. However, the following general characteristics have been identified [19, 31]:

- Self-Contained: identifiable and discrete with a set of characteristics and behaviour rules.
- Autonomous: exhibit control over their own actions and are able to make decisions.
- Proactive: react to external events to achieve their goals.
- Social: interact and communicate with other agents to accomplish their task and achieve the complete goal of the system.
- Flexible: ability to learn and adapt its behaviour based on experience.

In practice, a range of approaches have been applied to modelling the activity and travel of people as software agents. This includes executing a travel plan but making no changes until a review at the end of the day to select an alternative plan [32]. In other cases a more detailed approach is applied with agents responding to the changing environment during simulation, termed the *nanoscopic* level, through adjustments to driver behaviour while following a plan of travel for the day, termed the *microscopic* level [33].

## 1.3 Utilising Semantic Web Technologies

The Semantic Web is not a single technology but a hierarchical collection of formal standards and recommendations with supporting tool implementations. Its objective is to enable the structuring of data for automated interpretation and facilitate exchange between applications [34, 35]. Further developments include extending these standards to produce complete Semantic Web applications. The components of the Semantic Web are platform and programming language independent for transferability between implementing tool-sets.

The basis of Semantic Web technologies is the modelling of data to develop a knowledge model. The development of a knowledge model enables the identification and expression of common concepts and their relationships. Contextual facts can be asserted to construct a knowledge-base. Semantic modelling upon a knowledge-base using the relations, and their defined meanings, enables the inferencing of additional implied facts or the identification of inconsistencies.

Knowledge-bases can then be shared and utilised as the basis to develop applications and task solving models. Sharing and reuse of the commonly defined concepts and relations is achieved through vocabularies, also termed ontologies [36]. These concepts and relations can be applied to provide consistent understanding and structure giving interoperability between knowledge-bases. Numerous vocabularies have been developed including spatial (GeoSPARQL [37]), temporal (OWL Time [38]) and also for specific domains (transport domain topics include traffic disruption [39], automotive [40], infrastructure [41], and buildings [42]).

This enables the incorporation of facts from multiple knowledge-bases to extend a dataset and for knowledge-bases to adhere to a published structure. These

relations can then be used as the basis for additional concepts, either in depth of detail or breadth of coverage. The expected shape of the data can also be expressed as part of the vocabulary to form a schema that ensures the frequency of relations, the content of instance data and the inter-relationship of concepts are as intended. The structure and shape of the knowledge-base can be automatically interpreted to ensure data quality compliance, logical consistency and derive new statements. The Semantic Web uses these modelling benefits to retrieve, join and transform data for the user's application.

The primary objectives of this work are to provide a data-focused, modular approach that can be explored and adapted by the user. An immediate benefit of a Semantic Web approach is the storage and organisation of the diverse datasets required by the travel demand generation process. The extensible graph structure of a graph database combined with an engineered schema forms a knowledge-base, which can be partitioned into multiple logical graphs. Semantic Web technologies can be applied to the structured data of the knowledge-base to obtain inferences or apply rules to describe conditions and outcomes. By allowing the practitioner to specify their own vocabulary and rules the demand model can be expanded and customised to incorporate new behaviours and effects. This could incorporate the activities of interest, parameters of individuals or the behaviour and decision-making process.

The use of rules in activity-based demand models forms the basis of Computational Process Models and therefore potential exists for designing modules utilising Semantic Web rules languages. The focus during this investigation has been upon SPARQL based querying with RDFS inferencing as the combination provides flexibility and control without certain complexities introduced by OWL schemas or rules-based extensions, e.g. modelling restrictions [38], Open World Assumption and computational complexity. However, overlap exists such that certain outcomes could be achieved using several alternative Semantic Web technologies each with their own advantages. Key components of the Semantic Web are briefly outlined in the following sections to provide context of the terminology and further describe their application.

### 1.3.1 Resource Description Framework (RDF)

This fundamental data structure of the Semantic Web uses a directed labelled graph approach based upon *subject-predicate-object* triples [39]. This allows an extensible and adaptable structure to represent concepts, data and relationships as a metadata standard. New concepts can be incorporated by modifying the structural schema without needing to alter the underlying data. Each part of the triple is either a unique *resource*, described by a Unique Resource Identifier (URI), or a *literal* data value and type, such as a string or integer. The *resources* can represent real objects or abstract concepts and are cross-referenced with other resources to form a graph structure.

The triples of the knowledge-base can be persistently stored in a graph database, more specifically a triplestore or quadstore, as one or more URI named graphs to allow data partitioning. Queries can be performed on a single, collection or all graphs in the database. This means that sets of triples can be conveniently accessed, exported or deleted. This provides a mechanism to isolate different scenarios, multiple instances of the same scenario and to keep invariant data, e.g. population characteristics and road network, separate from generated variant data, e.g. a household's travel plans. Graph databases represent an alternative to traditional relational databases and are capable of storing and searching billions of triples. Data and schema can also be expressed separately so that the same data can be used to derive alternative inferences or structure for a knowledge-base.

The *subject-predicate-object* triples of the graph database aligns to the column and row structure of the tables within a relational database. Each unique *predicate* expresses a different column and each *subject* forms a new row of the primary key column. Each *class* of the *subject* is a separate table. The combination of *subject-predicate* corresponds to a row/column cell within the table and the *object* as the cell's content. However, the graph structure allows the dataset to contain an unlimited number of triples for a particular *subject* varying in either *predicate* or *object*, following the AAA principle discussed further in Section 4.2. Thus the graph database can support describing relations in an arbitrary manner for an arbitrary number of classes.

In relational databases, the structure is fixed according to the tables of the

schema and can only incorporate additional *predicates*/columns through programmatic intervention on the database. Therefore, a relational database is reliant upon the schema being fixed during the design process while a graph database can accept any additional relations provided by the user or other sources. The schema of the knowledge-base is also expressed as triples in a graph database and so forms part of the data content. This schema can be used to apply structure and shape by expressing constraints to identify violations, e.g. cardinality, class membership or datatype.

### 1.3.2 SPARQL Protocol and RDF Query Language (SPARQL)

The SPARQL query language provides a mechanism to explore, retrieve and modify RDF data as well as derive additional information, such as arithmetic or aggregation operations found in the relational database Structured Query Language (SQL). SPARQL endpoints can be deployed on to graph databases to process queries and return data for local or remote usage. The triple pattern of RDF provides a consistent graph matching structure to queries, while the syntax is standardised and platform independent.

The protocol permits the direct use of remote SPARQL endpoints through Federated Queries over the Hypertext Transfer Protocol (HTTP) [43]. This means a single query could access an authoritative external data source, such as road network data from a Transport Authority, and apply it to locally stored data. Alternatively, computational processing can be split into stages or partitioned among discrete hardware, e.g. spatial queries for different regions held on separate hardware. Data can be dynamically retrieved when required for processing so that the latest version is used rather than relying on periodic static releases with obsolete data.

The query protocol allows the underlying graph database to contain additional data not required in the current context but relevant for alternative uses. Therefore, a single coherent knowledge-base can be maintained rather than separate sets of data sources for multiple models. Extensions can be implemented as *property* and *filter* functions to increase the functionality of queries on the

graph database. This allows the triggering of complex data processing and data generation using standard query syntax.

### 1.3.3 Schema Languages

Inference reasoners apply schemas onto datasets to automatically verify logical consistency and infer knowledge, such as class membership and relationships. The schemas follow a standard vocabulary to describe relationships and inferences. The Resource Description Framework Schema (RDFS) is a data modelling vocabulary to extend RDF by describing groups of related resources and the relationships between them and provides many fundamental inferences [40]. The Web Ontology Language (OWL) language is a collection of logic based semantics to describe knowledge and relationships with greater expressiveness than RDFS [41]. The Shapes Constraints Language (SHACL) provides conditions for validating RDF data and describing rules and functions in SPARQL to ensure the graph data is structured correctly [42]. The practical considerations relating to these schema languages is discussed later in further detail (Section 4.2).

### 1.3.4 Rule Languages

Semantic Web Rule Language (SWRL), and other rule languages, allow the definition of rules to complement schema languages and express additional relationships within the schema using if...then... structured statements. The specified rules syntax may provide specific functions that the interpreting engine can execute but there is limited or no control over rule execution.

The SPARQL Inference Notation (SPIN) framework is a standards submission that utilises SPARQL syntax for the execution of queries as rules [43]. Rules can be formed into templates or functions for reuse in other rules while execution order and frequency can be controlled. Several schema and rule languages can be embedded and shared as part of the schema. This allows data structure and processing to be examined, shared and executed together across platforms.

## 1.4 Problem Statement

There are several challenges presented by the development and utilisation of travel demand models and traffic simulators. There is a diverse array of data, techniques and solutions within the domain while the design and modelling decisions applied to implementations have evolved as research has progressed. This presents challenges to users in selecting the appropriate approach and ensuring verification of outcomes across multiple implementations and models.

There is limited integration between implementations of models and traffic simulators so that users must employ ad hoc techniques to obtain and exchange data. Further, data may not be published in a format supported by an implementation or aligned to its schema and so must be converted by a third-party tool or an ad hoc technique. The development of these ad hoc techniques requires a thorough understanding of both the data and implementation with the risk of introducing error but also reducing investigative resources.

The data requirements of implementations are set to further increase. Future developments will see more complex modelling of human behaviour; the diversification of datasets; and the incorporation of new technologies into policy-making. This will further complicate the selection process of implementations and the verification of their outcomes.

Finally, the increasing behavioural complexity and data requirements will likely lead to further computational requirements. The development and implementation of activity-based models have already been hindered by their computational demands as more tractable solutions were explored.

## 1.5 Proposed Solution

The modelling of traffic and transport is a wide-ranging domain that has a direct impact on people's lives, through investment, policy decisions and the environment, as well as influencing other domains of research. It encompasses a wide breadth and depth of information that is only likely to increase and produce greater burdens on the development and utilisation of models. Therefore, a systematic approach is required in the organisation and access of this data so that

quality outcomes of reduced error, model comparison and reproducible results can be achieved. It is proposed that following a knowledge-based approach will allow the orientation of the modelling process upon the data rather than the system interfaces and boundaries between tools and models.

This will produce a constructed knowledge-base using a schema, which describes the relationships between fundamental concepts, and the investigative data. The functionality of the modelling process would be defined as loosely coupled inter-operable modules, which are configured through external module parameters also stored in the knowledge-base. The schema will encompass a core schema extended to include additional concepts required by the modules. The core schema would allow publishers to produce data in directly consumable formats with minimal transformation, while its extension will enable modules to achieve more diverse functionality. The use of a metadata standard for the core schema will assist in the proper use and interpretation of the data and linkages to other datasets and schemas. This would promote re-use of data, assist comparison of techniques and reduce the burden of solution assembly upon users to achieve quicker, more comparable and less error-prone investigative analysis.

The proposed modules can be implemented as innovative approaches or wrap existing tools to incorporate state of the art research. These modules encompass a broad range of tasks that operate upon the data of the knowledge-base and so may, in turn, be decomposed into sub-modules. Enabling users to select and switch between modules, sub-modules and data sources will encourage wider verification of models, implementations and scenario outcomes through alternative configurations. By defining a common mechanism to mediate this process the burden of selecting, transforming and processing data will be reduced. A framework will be developed that will allow users to specify how data is selected; the modules to process the data; the location of data sources and module services; and the validity of the user configuration.

## 1.6 Research Questions

It has been proposed in the previous section that an underpinning knowledge-base can provide the basis for undertaking traffic and transport demand modelling in-

corporating a range of models and data. Therefore, this thesis is based on the following hypothesis:

"The application of a knowledge-based approach and orchestrating framework will enable a loosely coupled modular architecture for activity-based travel demand generation and traffic simulation."

Consideration of the above hypothesis provides a set of research questions to direct the focus of the investigation. The selected research questions for further exploration are:

RQ1 How can a loosely coupled modular Semantic Web knowledge-base be applied to travel demand modelling and traffic simulation?

RQ2 What data concepts are required to construct a knowledge-base for travel demand modelling and traffic simulation?

RQ3 How can alternative techniques and data be selected using a Semantic Web knowledge-base?

RQ4 Can a Semantic Web framework be implemented for the generation of travel demand?

## 1.7 Thesis Contributions to Knowledge

The principal contributions of this work can be summarised as:

- Applying a knowledge-based approach to the process of travel demand generation.
- Development of a Semantic-based framework for travel demand generation.
- Design and demonstration of a Semantic-based travel demand generation framework.

## 1.8 Research Methodology

The research methodology adopted for this project is based on a building a software artefact to demonstrate the application of Semantic Web technologies to Activity-Based Travel Demand Generation. The following research activities were undertaken during the course of the project:

### Literature Review

The research involved an extensive literature review into the fields of Activity-Based Travel Demand, Traffic Simulation, Population Synthesis and Semantic Web. This review was carried out to develop a foundation based on existing approaches and incorporate current research progress. There was further review undertaken into published datasets, tools, ontologies and other relevant material, including population census, travel surveys, and transport infrastructure, to support the design and construction of the schema, prototype and evaluation scenarios.

### Analysis and Design

The outputs from the review of existing literature, tools and datasets were analysed as part of the design process. This process established the necessary requirements, features and organisation of the schema, framework and prototype. The outputs of the review were also used to identify the research questions for investigation. The design process also included the comparison and selection of tools and programming languages, such as Semantic Web library, traffic simulators, and software project management tools, that would be utilised during the prototype implementation.

### Iterative Development

The implementation of the prototype was undertaken following an iterative approach. The discrete modules were developed in a series of phases to develop and refine functionality. The design and limitations of these modules were reviewed

and developed during each iteration.

The prototype was implemented using the Java programming language, which was selected due to its maturity and performance, as a Gradle multi-part project. The selection of Gradle allowed the development of each module as a discrete software artefact and a build management environment to manage software library dependencies. These libraries were selected to assist the pace and robustness of development and included the Semantic Web library Apache Jena, which was selected due to its standards compliance and wide usage in Semantic Web research and applications, along with the SUMO and MATSim traffic simulators which feature prominently in Traffic and Transport research.

A test-driven development approach was applied through unit testing to develop the functionality of the modules and mitigate against error and regression during development. This was supported by version control software to manage the evolving source code and documentation. These software project management processes were applied to raise the quality of the development and assist in the later publication of source code and re-use. Developments from this phase also lead to contributions and feedback to open source projects and standards (see Appendix A).

### Evaluation

The evaluation of the prototype was performed in two phases. The first phase considers the application of a knowledge-modelling and modular approach by considering the travel demand and traffic simulation output from a constructed scenario. The second phase considers the performance of the framework in fulfilling travel demand generation in alternative configurations. Consideration was also given to the significant challenges and issues encountered during the development process. Testing and evaluation of discrete pieces of functionality were applied during the iterative development through unit testing.

## 1.9 Thesis Structure

The remaining chapters of this thesis are organised as follows:

- Chapter 2 discusses the identified related work in the context of travel demand modelling, traffic simulation and Semantic Web.
- Chapter 3 describes the proposed framework and how a Semantic Web knowledge-base can be applied to travel demand modelling and traffic simulation.
- Chapter 4 presents the developed common schema of data concepts, the basis of its development and organisation into relevant domains.
- Chapter 5 provides the design of the orchestrating framework, necessary processes and discusses the alternative configurations it supports.
- Chapter 6 describes the implemented prototype for travel demand generation using a Semantic Web framework.
- Chapter 7 evaluates the implemented prototype for the generation of travel demand with two third-party traffic simulators and compares the performance of the prototype in alternative configurations.
- Chapter 8 concludes the research, summarises the main outcomes and outlines suggestions for further work.

# Chapter 2

# Related Works

## 2.1 Introduction

This chapter discusses the identified related work by examining the overall context of travel demand modelling. There is consideration of the development of these models, integration with traffic simulators and identification of future requirements. Examination is also undertaken of existing applications of Semantic Web technologies to transport domain. This is then followed by discussion of the challenges presented by existing travel demand models.

## 2.2 Context of Travel Demand Modelling

The impact of transport upon the human population is wide ranging and influential with a daily impact on the lives of most members of society. This can be experienced directly through traffic congestion and infrastructure investment but also indirectly through travel safety and pollution [1]. Traveller behaviour both influences and is influenced by the economic development and land-use of their environment. The development of transport policy is supported by travel demand modelling and traffic simulation to manage and develop transport infrastructure and adapt to emerging technologies [3].

This requires the construction of complex systems that combine multiple concepts, datasets and techniques resulting in the reconciliation of data from multiple

sources and tools to enable their integration. However, the fundamental concepts and environment being modelled are consistent suggesting a common data model is achievable. The generation of travel demand is one stage in the process of constructing traffic simulations to investigate transport planning problems by modelling the movement of residents, non-residents and freight [11]. The process of model assembly for an integrated activity-based micro-simulation encompasses several stages: population synthesis, activity-based demand model and traffic simulation [9].

The process of population synthesis is the generation of individuals, and their relevant characteristics, to reside in the geography of the examined scenario. The purpose of the demand model is to produce planned journeys undertaken by these individuals through the geography. These journeys utilise different modes of transport at different times of the day or as part of the movement of goods between locations [7]. Traffic simulation explores the capability of the road and transport infrastructure to supply capacity to meet the demand for travel and transport. For each of these factors the focus, scope and complexity of models and tools can vary considerable.

The representation of traffic demand has moved from a trip based approach, which emphasises transport utilisation, to an activity based approach, which focuses upon the individual. This has seen a transition in transport research from the 4-step model (trip generation, trip distribution, mode choice and route assignment) of the 1970s, with a focus upon trips and journeys, to modern activity-based approaches [7], with consideration of human activities and travel. This transition is a recognition that the demand for travel originates from the need to move people and goods between daily activities rather than being inherent within the transport network [11].

These more recent activity-based approaches more closely capture human behaviour, decision making and planning. However, the resulting increase in data requirements and computational complexity has limited their significant progress until the last two decades [44]. The increase in computing power over recent decades has enabled this transition from macro-level simulation, based on mathematical equations applied to aggregate distributions, to micro-level simulation, focused upon individual entities. The micro-level simulation provides closer mod-

elling of the physical environment and incorporation of supplementary data into the models and simulators but with a corresponding increase in data complexity [45].

There have been a limited number of practical examples of activity-based models outside of key large North American cities, which use proprietary or in-house implementations [9, 21]. This lack of adoption of activity-based models has partly been attributed to perceptions of additional data requirements and implementation burden, despite identified alignments with traditional models [9], and the computational complexity of activity-based models [44]. These models and tools have been developed as proprietary closed source developments, which limits their accessibility and cannot be easily extended, or represent fresh academic efforts with a specific research focus that have not been concerned, or designed, for sharing or re-use [4].

There has also been limited general availability of traffic demand data sources for research with a reliance upon unpublished local datasets or stochastic generation [46, 47]. This has in part been attributed to the conflicting priorities of maintaining individuals' data-privacy and the need for spatial granularity in activity-based micro-simulation.

The development of travel demand models involves tuning parameters for their specific geographic scenario. This has led to research into their transferability to another geographic context [21, 47–49], but not necessarily producing a generic or readily transferable environment for wider research to take place. Investigations have found some components to be transferable while other components may require parameter re-calibration [21]. However, in some cases specific geographic related design assumptions have been incorporated which limit their transferability [47].

There have also been investigations into direct integrations between more general travel demand models and traffic simulators [50–52]. These demonstrated that integration is achievable, but highlighted that developing interfaces between modules was the most time consuming task and identified designing interfaces for modules in different programming languages and platforms as an area of future work. Ideally a model should be readily accessible for re-use in a number of different scenarios, such as geographical locations, with only adjustment to the

input data.

The accessibility and transferability of travel demand models and traffic simulators is of importance during investigations. Simulation software should be readily able to be verified and validated [53] against alternative implementations of the same or different models so that their relative merits can be explored. This includes the need to check different implementations have replicated the same model through reproduction of results or the alignment of alternative models exploring the same target phenomenon. Given that all models and simulations are incomplete representations of the physical world, it is good practice for comparison of results to be applied across different implementations.

This is particularly the case in activity-based models that are often reliant upon population sampling to reduce data gathering and computational complexity but at the risk of introducing greater uncertainty [8]. A range of model designs are also employed for determining variables such as mode, destination and scheduling time period [9]. Therefore, users should expect to be able to repeatedly perform their investigations across multiple frameworks with minimum investment of time and resources.

Setting up multiple travel demand and traffic simulators can require repeated data conversion that consumes user time and also a thorough understanding of each implementation. Adaptation may be required to each stage's schema and file format. The analysis of framework output also requires conversion of the results multiple times to the desired analytical format. The time and resources required to prepare and deploy such complex systems, whether 4-step or activity-based, limits validation and verification between alternative models.

These factors have meant that the utilisation of a traffic model and simulator is a significant commitment requiring long term investment in time and resources to deploy and then acquire, clean and appropriately format the input datasets for its specific requirements. Undertaking this for a single simulator can be difficult to achieve and maintain with meaningful verification, multiple test scenarios and constantly evolving data. Therefore, multiple models and simulators are often not readily available to provide direct comparable results during research.

The development away from the macro-simulation 4-step models to micro-simulation activity-based models has placed greater attention upon human be-

haviour in travel. The primary focus of transport modelling and taffic simulation has been upon car and freight transportation to evaluate traffic flow and the impact of traffic control measures [11]. This focus on car transportation is viewed as untenable due to its environmental, economic and health impacts with alternative healthier and safer modes needing to be modelled and investigated [1].

The emphasis on vehicles and traffic control has meant that the modelling of human behaviour is lacking in most transport simulators [4]. The continued human involvement in transport and traffic decisions will necessitate micro-level simulation yet the verification of human behaviour at micro-level is considered difficult on a purely empirical basis [54].

There are also further limitations in the scope of activities and travel patterns modelled compared to those which travel research has shown are exhibited by individuals. There has been a primary focus upon commuter and freight transportation simulation, as previously mentioned, which along with school journeys places an emphasis on weekday morning and afternoon peak activities and travel [11, 12].

This is despite research into sources of travel demand showing a broader range of activities and time periods being undertaken by individuals [14] and increasing complexity in activity patterns [44]. Therefore, transport models must develop to encompass a greater variety and volume of data in order to capture and express the increased breadth of activities and time periods.

The time-frame of scheduling and planning has generally focused on the short term, i.e. days, weeks or months, of traffic simulation and its use for exploring traffic congestion and travel patterns. The long term view would incorporate changes to land use, e.g. new roads and facilities, resources, e.g. car purchases, and demographics, e.g. births, marriages, ageing, and deaths, and their impact on travel demand [9]. This has led to the proposal of more complex modelling and simulations where social and economic decisions are closely modelled to produce a simulated *artifical* society [55].

The activity choices and time periods form part of the human behavioural responses, which have been noted as lacking in traffic demand and simulation [4]. These behavioural responses include route choice which has been modelled from the perspective of rational self-optimisation [8]. Inconsistencies between modelled

and human behaviour have been attributed to varying perception of an *optimal* route, reduced information and the penalising effects of congestion [7].

Yet research has shown that humans are imperfect decision makers [56] and apply procedure rather than substantive rationality [6]. This has led to wide range of modelling approaches which can incorporate multiple paradigms for human behaviour [8]. Alternative approaches have sought to use rules-based responses to the input stimulus [20] or to incorporate agent-based systems which display autonomy, pro-activity and communication to meet their goals [19]. Therefore, there is a breadth of modelling approaches to capturing the complexity of human behaviour and decision making.

Conversely, there are a variety of implementation and design choices within each class of modelling approach. Further, the feedback loop of data for travel experience, as opposed to travel planning, from traffic simulation to activity based models is often absent [9]. This additional data would enable and sustain human based behaviour, e.g. habits and learning, and enable new observations to emerge over repeated cycles.

The need for wider consideration in modes of transport is also reflected in new technological developments that will influence human travel and policy making. These technological developments, e.g. automated driving; vehicle to vehicle communication; and vehicle to infrastructure communication, present new opportunities for co-ordination between participants in the transport environment, but also introduce challenges in their deployment that traffic and travel modelling and simulation can assist in addressing [4, 12].

Developments in web-based social networking applications and platforms also offer new and wider access to transport methods, such as car-pooling and car/lift-sharing services, that could help alleviate road congestion but require more diverse modelling [57]. Therefore, the data and modelling requirements being sought by users will further increase complexity.

These technologies also provide new sources of data that can complement travel demand models, e.g. GPS traces [58]. These can provide greater coverage of traffic flow data than provided by traditional static traffic sensors, e.g. induction loops and cameras, or surveys by not being fixed to specific points in the road network. However, data privacy concerns can limit scope and publication of

datasets [59, 60].

The data produced by these types of sensors concerns traffic volume, but not the travel purpose, i.e. origin and destination activities. This reduces the applicability and sensitivity for modelling policy, infrastructure and land-use changes [9]. Therefore, they do not represent a complete replacement of activity-demand modelling, but can provide complimentary data.

This wide variety of techniques and data requirements illustrates the need for a knowledge model of the domain. The development of a knowledge model for travel demand requires the identification of the problem domain's common concepts and their relationships. Research has been undertaken to develop standardised and consistent definitions for transport modelling of movement and activity [10]. This is further supported by the publication of transportation and land use related ontologies [39, 61, 62].

The EC INSPIRE project [63] seeks to provide a standardised spatial infrastructure data format across the EU, including transport networks, with work in progress to develop RDF vocabularies [41]. Other research efforts investigated the additional data requirements to incorporate transport models into the CityGML format [17], the alignment of CityGML with ontology based approaches [42] and the conversion of the GML format to RDF [64]. To the best of our knowledge, there are no published works focusing on traffic demand modelling.

The Semantic Web has been developed to transition the World Wide Web (WWW) from a distributed network of web-pages and documents focused around presentation of information to openly accessible data repositories focused upon structured information [36]. This will enable the development of more sophisticated computer to computer interactions and computer support for decision-making [34, 35].

The development of the Semantic Web is based on a hierarchical collection of formal standards and tools that support and extend the functionality offered to achieve the proposed vision. The components of the Semantic Web are platform and programming language independent for transferability between implementing tool-sets with new and revised standards being developed as the need is identified.

These interoperable tool-sets remove the need for systems to develop their own interfaces between technological components. Therefore, alternative imple-

mentations can be utilised consistently across operating system and programming language platforms. The Resource Description Format (RDF) provides a common data format for the representation of data and schema concepts using a graph structure. This RDF data can be serialised in file formats or persistently stored in graphstores that can be extended in schema without modifying the database structure, unlike in traditional relational databases. The SPARQL query language allows graph structured data to be queried and retrieved in closed offline and open online contexts, unlike Structured Query Language (SQL) used with relational databases which was developed for offline usage.

The design of the Semantic Web is based on structured data, machine to machine processing and open data exchange between applications [65] so that data can be reused and flexibly adapted. Technologies have also been developed to facilitate the processing of existing data sources in flat files [66], structured files [67] and relational databases [68] into Semantic Web formats. Semantic Web standards support the retrieval of data from local and remote online sources [43].

The use of a Semantic Web approach also enables the incorporation and linking with facts from other data sources as part of the data model. It is further envisaged that autonomous agents will be able to interpret ontologically structured data and make reasoned decisions [65], which is in keeping with micro-simulation and activity-based modelling's focus upon individuals.

This is achieved through ontologies expressing a vocabulary enabling communication between agents, even if they do not share a global theory, as a commitment to the vocabulary means it will be used coherently and consistently [69]. This use of ontologies has been applied to develop web service descriptions of Semantic Web Services by describing the semantics of data and behaviour of services so assisting interoperability and ultimately leading to their automated discovery, negotiation, composition and invocation [70].

There has been development of frameworks to support these web services for biomedical datasets. These have been based on retrieval of single triples [71] thereby lacking the flexibility to retrieve additional data or the execution of SPARQL queries. Other efforts have extended this approach to provide a SPARQL query mechanism to retrieve grounded facts, but with the user being unaware and unable to control the origin and provenance of the data and ser-

vices [72]. In the travel demand generation process a user would be specifically targeting datasets and techniques relevant to their investigative interest. The approach also does not use the standardised SPARQL mechanisms for retrieving remote data and instead reformulates queries into requests for individual triples, incurring additional network overhead and limiting query expressivity.

Research has been taking place into transferring existing data sources and datasets to the Semantic Web. This includes publishing demographic census and similar data that form inputs to traffic demand modelling [73, 74]. A key aspect of Semantic Web technologies is an extendible data model and the ability to share information for reuse in a flexible and convenient manner [75].

The application of the Semantic Web in the transport domain has seen the development of several traffic prediction and routing systems [76–80]. These were able to utilise diverse information sources to improve prediction accuracy by exploiting the underlying semantics of the data. A number of issues were highlighted including data quality, data assimilation, scalability and time reasoning within macroscopic simulations and the need for developing generalised approaches and tools.

Initiatives such as the UK's data.gov.uk [81] are making a broader range of data sources easily accessible but the incorporation into traffic models continues to be an ad hoc process. The recent trends of Big Data, Open Linked Data [82] and volunteer initiatives, such as Open Street Map [83], has seen the increased gathering, processing and availability of large detailed datasets. This presents an opportunity for traffic demand models to incorporate a greater range of information and modelling processes. Travel demand models have also been identified as having contributions to domains outside of transportation, e.g. environment and health [8]. This leads to greater associated complexity in the breadth and depth of data.

## 2.3 Challenges of Travel Demand Modelling

Several challenges are presented to users when selecting and utilising travel demand model and traffic simulator frameworks. These frameworks are generally developed as collections of tools and models that fulfil the distinct functions of

the modelling process [8]. Due to development focus, one tool in the collection may provide advanced features or design while another is more limited, e.g. supported transport modes, activity model, routing algorithms or human behaviour model. Therefore, a user must evaluate between frameworks and compromise on certain features to utilise others.

An alternative approach is for frameworks to be designed as modular software applications [4, 32]. These define interfaces and objects that are extended by new modules as part of the application. This requires a high quality approach to the development of the software archiecture and interfaces to ensure that variations in modelling paradigm and scale are accomodated [4]. Later developments of modular interfaces can require modules to be redeveloped to enable compatibility with newer features and functionality. This places an ongoing maintenance responsibility upon the developers and can prevent users from utilising unmaintained modules, or require multiple versions of the core software.

The file formats supported by a framework may also force its selection. Geospatial and road network data are provided in a wide range of standard file formats with each framework supporting a subset and potentially its own bespoke format. A user may not have the technical skills or resources to convert their own data file format into one of the supported formats. Other input data will rely on the framework's generation tools or need conversion to each framework's schema as no common standard exists.

Given that all models and simulations are incomplete representations of the physical world, it is good practice for comparison of results to be applied across different implementations [53]. This is particularly the case in activity-based models that are often reliant upon population sampling to reduce data gathering and computational complexity but introduces greater uncertainty [8].

A range of model designs are also employed for determining variables such as mode, destination and scheduling time period [9]. Therefore, users should expect to be able to perform their investigations multiple times and across multiple frameworks with minimum investment of time and resources. Setting up multiple configurations can require repeated data conversion that consumes user time and a thorough understanding of each implementation. The analysis of framework output also requires conversion of the results multiple times to the desired

analytical format.

There is potential to convert the output of one framework's tool for reuse in another framework as a one-to-one integration [47, 51, 52], although that may not be their intended design. However, the user would again need to have a thorough understanding of both tools' configurations to avoid error and reconcile design or data differences.

Frameworks are also developed in a variety of programming languages and platforms. The transference of data between tools or utilising domain libraries may require the user performing manual processes unsuitable for large numbers of investigative cases and risking introducing error. In each of these cases there is a required level of user technical skill and resources that may not be available and diminishes the investigative portion of a project.

The process of selecting obtaining and preparing input data for these frameworks also places a burden upon the user. The MatSIM traffic simulator [32] does not provide tools for the generation of activity-based travel demand, while the SUMO traffic simulator [46] provides a limited tool based upon even distribution of aggregate population statistics, a subset of activities and single mode journeys. Geographic, road network and population data are external to the framework and their tools. Each required dataset is typically published by a different agency or organisation.

A user will need to identify the required data for their target area, source from the multiple providers, clean, combine and reformulate to then use in their model and selected framework. The data requirements between users are likely to be very similar with only relatively specific enhancements for their interest area. Yet no unified datasets or combining mechanisms have been identified.

Each of these identified challenges requires investing additional resources and potentially developing ad hoc solutions which could compromise the investigation by not accurately deploying the appropriate travel demand models and traffic simulators. Users also face the barrier that completing these activities for a single framework does not automatically allow many frameworks to be utilised in an investigation. Traffic demand modelling has also relied on close relationships with transport authorities to produce specific local demand models or generate random traffic flows from the limited public data [46], which limits the reproduction of

previous research and the progress or quality of new research.

The development of travel demand models also requires several further features. There has been a primary focus upon week day commuter car and freight transportation to evaluate the impact of traffic control [11] yet changing working patterns and business hours are emerging [12] with commuting only representing 15% of trips and 20% of distance travelled [14]. Technological developments are also presenting alternatives for coordination and planning [4, 57], such as car-pooling, car-sharing, automated vehicles and vehicle communication, which need to be modelled.

There is an identified need for models to be further developed to cover multiple days, improve cooperation between household members, incorporate social network relationships and develop non-utilitarian human behaviour and decision making [8]. Human behaviour modelling has tended to apply a single or a few approaches to all individuals and not consider all the contextual information that could be utilised [9]. Further, human behaviour itself can exhibit irrational and subjective choices that vary in context and experience [6, 56]. These present problems in the prediction and validation of transport systems [19]. These design goals further increase the breadth and depth of data requiring management and increase computational complexity resulting in longer model run times [19].

## 2.4 Conclusion

The previous discussion has highlighted that the development of travel demand models and traffic simulations are expected to lead to increased complexity and corresponding increases in data requirements. This is driven by the need for greater modelling of human behaviour and providing more diverse behavioural responses for the different transport participants. This in turn will assist in supporting the need to explore wider policy making to support alternative modes of transport from the traditional car and freight; the impact of emerging technologies; and the increased complexity of human activity patterns. Therefore, there is a need for modelling technologies and solutions that can support the organisation of this diverse data and the access to emerging solutions.

It has also been identified that the complexity and approaches of the transport

domain and modelling human behaviour are best explored when comparison is made across multiple models and implementations. This enables the identification of relative modelling strengths and weaknesses to support robust conclusions. However, the current burden of data preparation and integration between stages inhibits investigations. This is despite models and simulators being successfully integrated on typically one-to-one basis.

The Semantic Web technology has been identified as a solution for supporting the resolution of such issues. The technology provides a knowledge modelling approach to assist in the description and sharing of data. This enables common concepts to be re-used between modelling components and simulators, while also allowing variations and extension to these concepts. There have already been successful efforts to apply Semantic Web technologies in other transport related solutions. Vocabularies and datasets for the transport domain have also been published online and these provide the potential for quicker access to data and the aligning of concepts, so that interoperation of models and simulators becomes more widespread.

The service orientated architecture provides for accessing this data and also the basis for separating the modelling components into discrete services, so that data and processing can be distributed and redirected. This provides the potential for decomposing the numerous modelling choices and alternatives of travel demand modelling into discrete services. This will mean that users can select and assemble the models for their investigations rather than being reliant upon the traditional single monolithic implementation. Therefore, Semantic Web technologies can provide a basis for achieving the proposed knowledge-based approach and is the technology of choice for further investigation in this thesis.

# Chapter 3

# Architecture of the Proposed Semantic-based Travel Demand Generation Framework

## 3.1 Introduction

This chapter seeks to establish the overall design of the framework to address the challenge of applying a loosely coupled modular Semantic Web knowledge-base to demand modelling and traffic simulation as stated in research question RQ1. It provides an overview of both Semantic Web technologies and travel demand modelling and then considers the implications for the framework. Current travel demand models cover a broad range of design decisions and concepts that it is impractical to exhaustively explore and discuss. Therefore, general components are identified with specific reference to aspects of activity-based models, but potential exists to apply the framework more generally. Future developments to transport models will likely increase data complexity that an extendible knowledge-base can assist in managing.

The many tools and implementations currently provided by travel demand models and traffic simulator would each conceptually form modules of the framework. These modules would then interact through a core extendible schema to facilitate interoperability and minimise user intervention (Chapter 4). The pur-

pose of the framework is to enable users to retrieve, transform and re-use their data across multiple travel demand, traffic simulator and supporting modules for their investigations (Chapter 5), while also incorporating their own data and techniques.

The following sections of this chapter provide a motivating scenario and general overview to establish the context of the proposed framework. This is followed by examination of the proposed framework by considering in more detail the stages of the travel demand generation process; potential modules for travel demand generation; and the alternative configurations to which the framework can be applied. It is intended that this will highlight the breadth and scope of the proposed framework and how it will address the problem.

## 3.2 Design of Framework for Travel Demand Generation

The core stages of an integrated travel demand model are the sequence of population synthesis, travel demand generation and traffic simulation [9] as previously illustrated in Figure 1.5. Data from each discrete stage is passed to the next stage with iterative feedback sometimes occurring from traffic simulation back into travel demand model. Each stage is also reliant upon a variety of input datasets, such as demographics, network supply, travel diaries and land use. The proposed framework introduces a semantic modelling layer between the stages as shown in Figure 3.1.

The introduction of this framework layer would enable conversion of the input and output of each stage into the common RDF format and the formation of an underpinning knowledge-base. The common format and knowledge-base will enable different implementations of the stages to be interchanged more easily as shown in Figure 3.2. The user is able to select different modules of functionality from a menu of modules for each stage. This will allow a broader range of techniques to be evaluated and analysed during an investigation on the same knowledge-base, rather than utilising a single set of integrated components or multiple discrete instances.

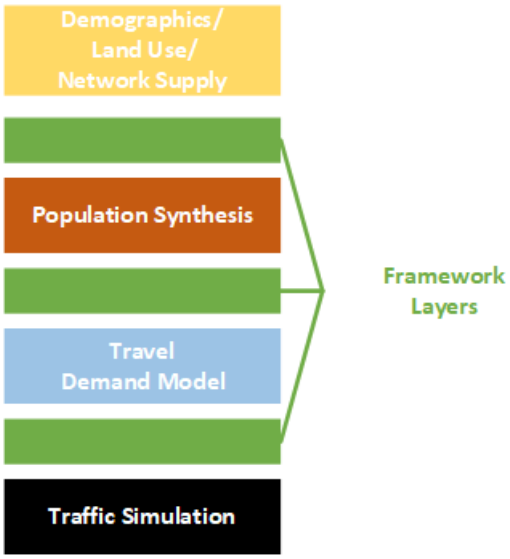


Figure 3.1: Diagram of Travel Demand model sequence showing Framework layers.

The developers of modules are no longer required to select a modelling framework and adhere to its platform, interface and design decisions. Instead they can design their module based on the RDF data inputs and outputs from the knowledge-base. A module implemented in one programming language can interact, or be replaced, by another in a different programming language. Existing implementations can potentially be incorporated by wrapping with an RDF interface to convert between the knowledge-base and the module. These interfaces provide an Extract Transform Load (ETL) process that can utilise existing tools to convert to RDF through Data Materialisation or provide virtual triples of a dataset stored in a relational database through On-Demand Mapping [68]. A single set of interfaces to the platform-agnostic knowledge-base would need to be maintained.

The use of a common knowledge-base also provides for the primary stages to be decomposed further into sub-modules. The main stages in Figure 3.2 represent complex processes which draw upon ancillary data of the knowledge-base. This presents opportunity for different combinations of sub-techniques to be evaluated during an investigation, e.g. replacing a search strategy or choice model. Investigation and implementation of new ideas could also focus upon specific elements

of the larger set of components.

These sub-modules would interact with existing implementations through the knowledge-base data model rather than reimplementing entire stages of the process or extending an existing implementation. This should facilitate exploring new techniques and identifying best practice. The storage of data in the knowledge-base also allows the sharing of both scenario and results data for sharing, re-use and comparison.

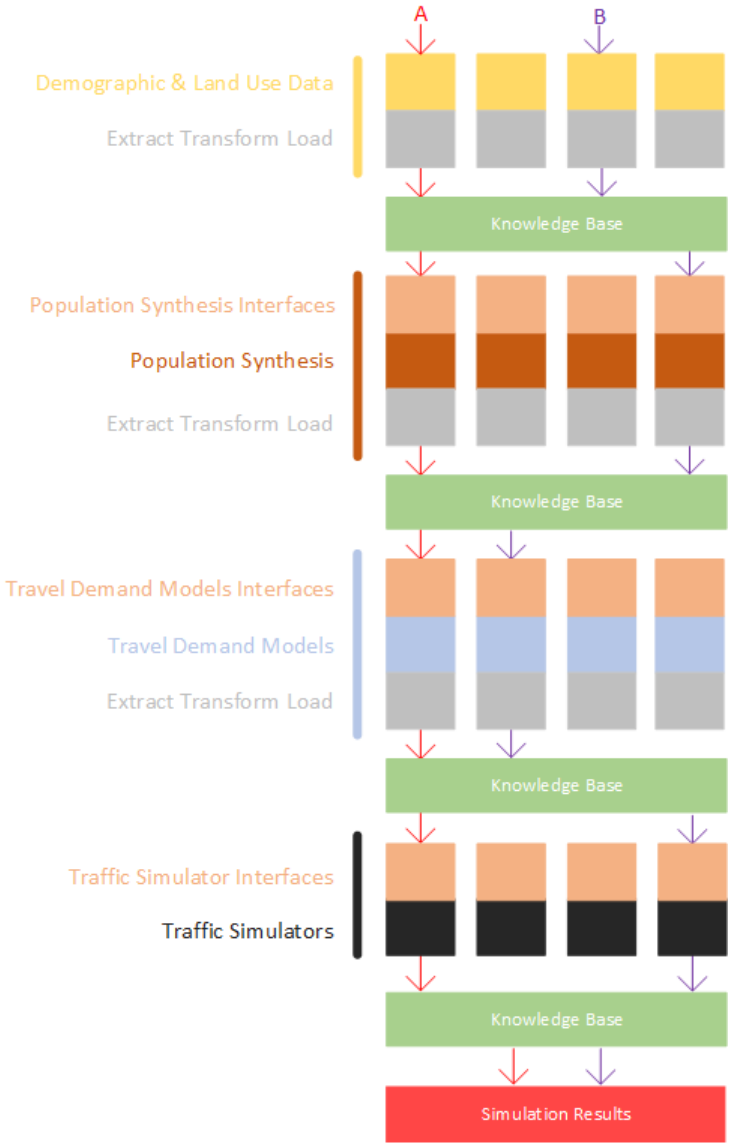


Figure 3.2: Diagram of selecting alternative implementations during the stages of the modelling process.

The main components and flow of the travel demand modelling framework are shown in Figure 3.3. The framework incorporates the construction of the knowledge-base from input datasets and techniques, e.g. population synthesis and activity pattern generation. The constructed knowledge-base is then used to inform the travel demand model to generate trips and journeys for the scenario. These journeys are simulated for their physical interactions with each other and

the road transport environment by a traffic simulator.

The control and manipulation of the knowledge-base and the module stages is facilitated through SPARQL queries to retrieve and transform the required data. As illustrated in Figure 3.2, the modules could represent one or more implementations, e.g. network routing could have different implementations for different modes to provide diversity within a scenario or two implementations which cover multiple modes that are compared during an investigation.

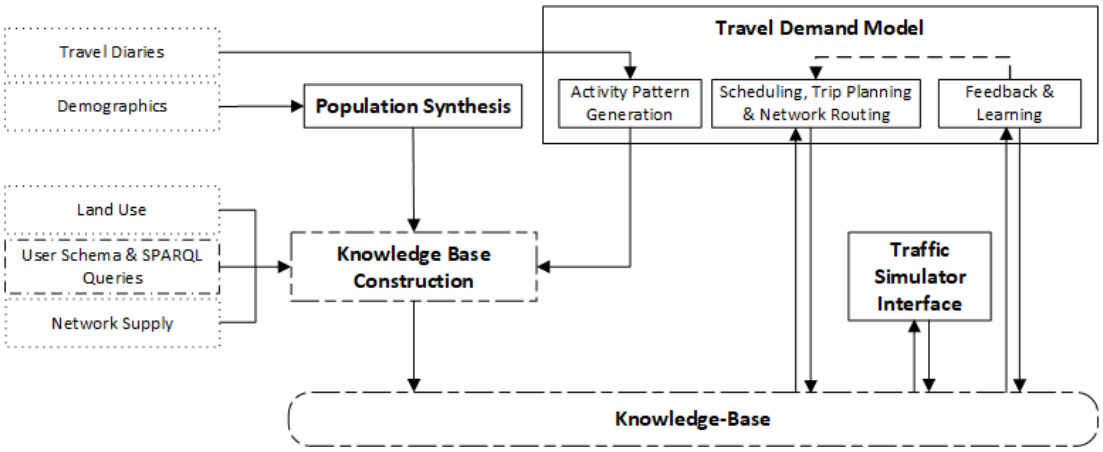


Figure 3.3: Diagram of main stages for Travel Demand modelling framework.

An immediate benefit of this approach is providing the storage of the diverse data required to satisfy the different stages in the extensible graph structure of a graph database. The combination of this data with an engineered schema forms the knowledge-base. This knowledge-base provides the structure for interactions between different modules and data concepts, e.g. the Trip Planning module requesting an estimated travel time from the Network Routing module.

The input and output data from each stage, scenario or execution can be partitioned into different named graphs within the graph database of the knowledge-base for reuse or extraction through SPARQL queries. This can assist with managing data disposal and alternative datasets, such as repeated scenarios, alternative time periods, different geographic areas, design implementations and user requests. A single knowledge-base containing multiple graphs can be constructed rather than multiple sets of input files or databases.

Users can convert existing datasets or the output from external tools into RDF

and import into the graph database. The extendible nature also allows additional data for topics of specific research interest, e.g. vehicle characteristics or social network relationships, to be linked and stored in the same graph database without interference to the execution of the travel demand modules.

This means that implementations can extend the core knowledge-base but still operate on the same graph database. Therefore, modules with alternative design principles or providing new functionality, e.g. environmental or communication models for vehicles, can be applied to the same set of underlying data. The structure of input data can be transformed using SPARQL queries or linked to existing concepts by extending the core schema.

Execution of modules is achieved in SPARQL queries using property functions for each module. Therefore, the SPARQL query can control the data selected and its processing. This allows the user to change between modules by simply modifying the query. Modules can also be selected according to the characteristics of the data through class membership, data properties or inter-relationships (Section 5.3.10). This can address the identified limitation of travel demand models applying a single behavioural approach to all participants (Section 1.4).

An example of this is shown in Listing 3.1 where the modules are implemented as property functions and can be identified by the *mod* namespace prefix. In the example, two sets of persons are selected from the population, one based on their membership of an income quartile class and the other based upon being an employee with an income greater than the stated threshold. Each set of persons has a different routing approach selected to generate travel demand based on their contextual data, which has been a criticism of many travel demand models. Therefore, different behaviours can be described for subsets within a population but as part of the same query. Modules can interact based on the data provided rather than how that data is sourced.

Another advantage of SPARQL is that it supports federated queries that can retrieve data in both local and remote online graph databases. External knowledge-bases set up as SPARQL endpoints, e.g. LinkedGeoData for land use and network infrastructure data [61], can be queried to retrieve relevant RDF triples and accelerate knowledge-base construction.

```
PREFIX mod: <http://example.org/module#>
PREFIX ex: <http://example.org/local#>

SELECT ?person ?route
WHERE{
    {
        ?person a ex:Quartile4Income .
        ?route mod:routingMethodA (?person ?start ?end).
    }UNION{
        ?person a ex:Employee .
        ?person ex:income ?income .
        FILTER(?income > 50000)
        ?route mod:routingMethodB (?person ?start ?end).
    }
}
```

Listing 3.1: Example SPARQL query for selecting alternative routing modules based on characteristics.

A step further is remote services providing property functions that perform the functional stages of travel demand modelling based on input parameters. Therefore, a user focused upon only the simulation stage output could write a query to target remote services of data and modules providing configuration parameters then only store the configuration and results locally.

The execution of queries can be separated into multiple stages to allow retrieval and manipulation of partial data in the knowledge-base or to use alternative sub-modules. Partial data can be supplied as URI references accompanied by the URL reference of the remote service. Modules can query against the service URL to retrieve the data associated with the parameter URI and then perform their task.

The overall approach of using remote services would require data retention policies as generated data could be stored by the remote service. Potentially only a small proportion of URI references for the generated data would be transferred

between services. A user wishing to retain a complete record in perpetuity may need to request the relevant data from the services, which would be simplified by storing each user request in its own named graph.

The alignment of these remote services would require an understanding of their data requirements and behaviour. Service and data descriptions form part of the SPARQL standard as part of the Semantic Web's open network design. Remote services could also provide RDF responses to document their functionality and design assumptions to assist users. The description of data semantics and service behaviour through ontologies and annotations has been developed in standards, such as SAWSDL, to enable the machine interpretation and ultimately lead to automated discovery, negotiation, composition and invocation of these services [70]. Therefore, mechanisms exist and are being developed that could support this approach.

National, or international, knowledge-bases would allow re-use of quality and consistent datasets while remote modules can provide best-practice implementations. Currently users construct a local dataset and model for their specific problem or geographic area by establishing their own infrastructure and sourcing, processing and transforming data files for the selected tools. The proposed approach would reduce these requirements as the data and interfaces are already designed around the same data paradigm of RDF. Users would be able to select a local, remote or hybrid execution of the travel demand generation based upon their need and focus.

To conclude, the knowledge-based approach to integrating the population synthesis, travel demand and traffic simulation processes results in a novel framework that improves the integration between independently developed implementations; assists users to manage and structure local data; allows users flexibility to compare alternative implementations; and provide the basis for accessing standardised datasets, scenarios and tools as on-line internet services.

## 3.3 Application of Framework for Generation of Travel Demand

This section provides more detail on the general components identified in Figure 3.3. These components are further decomposed into key stages as shown in Figure 3.4 to provide more detail on the processes taking place. These component modules could incorporate multiple sub-stages in their implementation.

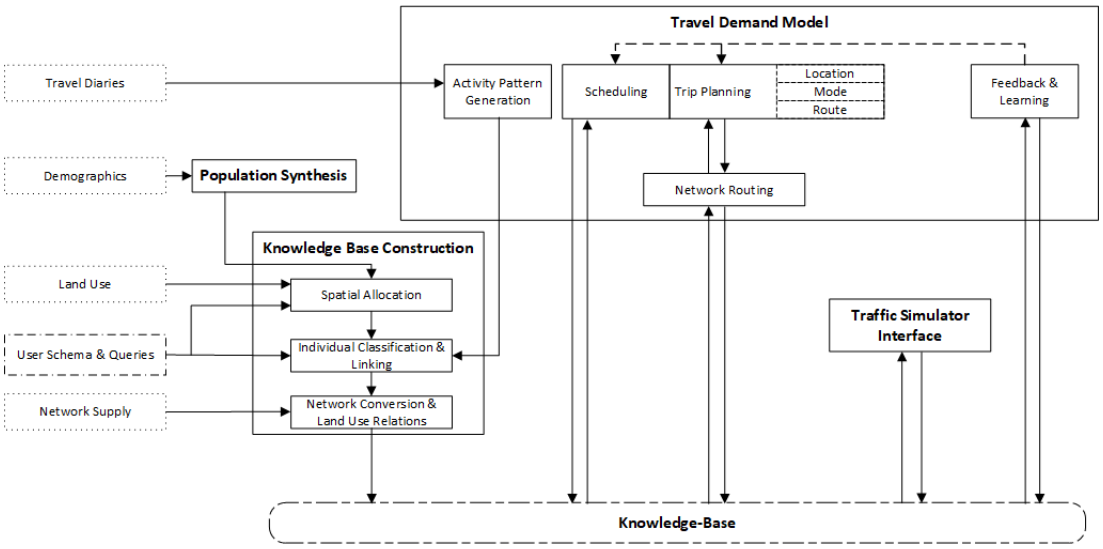


Figure 3.4: Diagram of modular stages for Travel Demand modelling framework.

### 3.3.1 Population Synthesis

This stage provides the conversion of aggregate demographic data into a disaggregated set of persons grouped into households. Each person and household are described by a set of characteristic variables. The synthesised population and additional data sources may require further reorganisation and cross-reference prior to use by the travel demand model (Section 3.3.2). In some existing travel demand models, an integrated synthesiser is available, but the user is then limited to the chosen algorithm and implementation choices in this active field of research.

In the proposed approach, any population synthesiser can be utilised once its output is serialised into RDF and published locally into the knowledge-base

or retrieved from a remote SPARQL endpoint. The population's characteristic variables can be reformulated to a user's chosen schema as part of constructing the knowledge-base.

### 3.3.2 Knowledge-Base Construction

The purpose of this stage is to bring together the synthetic population, activity patterns, land use, network infrastructure and local modelling of the problem domain. The user can develop a formal schema that extends the core schema (Chapter 4) using inference languages and apply a reasoner to the data to perform inferencing. The use of the inference languages has successfully resolved reconciliation between heterogeneous information sources, allowed sourcing of structured data from public datasets, e.g. Linked Open Data, and produce new and interesting facts as inferences [76, 77]. These inferences can be asserted into the knowledge-base to persist or form virtual triples, i.e. triples that are removed when the reasoner is removed.

The inferencing process can automatically allocate instances to classes, create data properties, infer relationships and identify contradictory data in the knowledge-base, e.g. a child who possesses a driving licence. Class membership can be assigned based upon relationships and data e.g. vehicle ownership, age or income. Alternatively, SPARQL queries, or a rules language engine, can perform these tasks with varying flexibility and restrictions. SPARQL queries can also permanently transform, remove or add data to the knowledge-base.

SPARQL queries can be stored as text files and can be applied manually to the knowledge-base or applied programmatically. Therefore, the queries can be easily distributed to share best practice and users can implement local modifications before applying to the knowledge-base.

Traffic and transport, particularly in micro-simulation, heavily utilise spatial relationships between objects. Graph databases can be extended to natively support these spatial relationships, as in some relational databases. These extensions, such as those complying with the GeoSPARQL standard [37], allow the knowledge-base to be searched for data based on the spatial relationships using SPARQL queries without the need for external spatial processing (Section

4.4.1). Three general stages of knowledge-base construction have been identified as shown in Figure 3.4.

#### 3.3.2.1 Spatial Allocation

The synthesised population must be aligned with the land use data for the zones and regions of interest. The generated households are produced for an entire zone but must be allocated to the individual physical locations of homes within the zone. The allocation process between households and homes can be achieved through user defined SPARQL queries, a library of best practice approaches or implemented modules for more sophisticated techniques. This step is described in literature as one of the final steps in population synthesis but there is limited reporting of techniques and theoretical results [84].

The decoupling of the Spatial Allocation task from the Population Synthesis process allows the user to adapt the allocation process to their chosen schema, available data and selected approach, rather than that implemented by the selected Population Synthesis tool. For example, an allocation based upon house prices and number of bedrooms would produce a different knowledge-base instance to another allocation approach that utilises household composition. In this way numerous alternatives can be explored with consideration of the varying impact on the traffic simulation results.

#### 3.3.2.2 Individual Classification and Linking

In existing activity-based models, the functionality is typically based upon hard-coded characteristic values, such as specific household compositions or types of locations, or a fixed set of configurable parameters, such as school and retirement age. This functionality includes decision making processes with a criticism that existing models apply a single or a few approaches to all individuals [9], i.e. all employees make travels decisions in an identical manner or that the factors in a commuter's and tourist's decision making are identical.

It is proposed that these shortcomings can be overcome through local user control and generic module design. The user's control, over the local schema and in manipulating the knowledge-base data, provides choice in the characteristic

values and their inter-relationships that are present in the user's scenarios. For instance, a core schema would define location and activity, but it is the user's local schema that extends these to define the specific types of locations and activities. Therefore, it is the user that determines the design of the local schema and data rather than fitting the data to a modules' design assumptions.

Modules should be designed against generic concepts, rather than specific instances, so that the user has as much flexibility as possible. For example, a routing module would not specify the modes of transport supported but instead the data parameters required to perform routing for any mode and its design assumptions. The user would then select modules based on available data and investigation design, e.g. one user may consider it adequate that car and bicycles use the same routing approach while another would select a dedicated module for each. Certain cases may require a module to extend the breadth of the core schema, but should be minimised, e.g. vehicle routing module that considers road conditions, such as low bridges, requires additional vehicle characteristics not required by a generic router.

In the Semantic Web, classes are sets of individuals and can be sub-classed to any hierarchical depth [85]. A person, household or activity are all examples of individuals in this context. An individual can belong to multiple classes that can be asserted or inferred using a reasoner, SPARQL query or rules engine. Classification can be based on context using the individual's existing class membership and the values, cardinality and inter-connecting properties. Illustrative examples are:

1. A person with access to a car and possessing a driving licence belongs to the Car Driver class.

2. A person aged between 5 and 11 belongs to the Primary School Student class.

3. Retail location selling luxury items and open after 6 pm assigned Affluent Evening Retail activity type.

4. Leisure activities sub-classed into Exercise, Sport or Culture classes but also Indoor or Outdoor.

5. People working at the same location are inferred to be colleagues of each other.

The hierarchy depth of classification and properties becomes a user choice to an arbitrary level of detail based on data sources, design assumptions and implementation context. e.g. the core class and property triple "Person *hasActivityAt Location*" is sub-typed as "Employee *hasEmplyomentAt* Workplace" and "Student *hasEducationAt* School". The user asserts the data according to the sub-types, but generic modules can still access the core concepts through inferred memberships.

Filtering according to temporal, e.g. opening hours, or spatial, e.g. activity location, or any other characteristic allows different contexts to exist within the same knowledge-base. Locations modelled with an area of effect, e.g. school catchment or retail operational area, enable partitioning and selection rather than assuming a pervasive effect as in many existing models.

The allocation of an individual to a class, or their existing relationships to other individuals, can be used to apply default values or create new relationships, e.g. a person belonging to a household is inferred to be resident at the household's location. The creation of new relationships can also be constrained by applying filtering. Persons could be associated or limited to activities in a certain geographic area or specific types.

Different derived schema and data will produce alternative knowledge-bases that still function with the generic modules. Modules that extend the core schema would operate on the same knowledge-base without interference to generic modules.

#### 3.3.2.3 Network Conversion and Land Use Relations

This stage consists of two parts: the conversion and addition of road network, and other transport infrastructure, data into RDF format and the linking of land use data to the road network. This process can involve conversion of location addresses and post-codes to spatial coordinates in the selected coordinate reference system. Formats for road network information typically follow a node (junction) and edge (road) graph structure but there is a need for a standardised RDF

vocabulary for transport networks and supporting infrastructure.

The INSPIRE project [63] includes a transport infrastructure theme. Work is in progress to develop RDF vocabularies for INSPIRE [41], but no vocabulary yet encompasses the whole transport domain for simulation purposes. Other research has investigated additional data requirements of transport models for CityGML [17] and the conversion of GML to RDF [64]. A standardised schema and tools would allow routing operations and interpreting of road semantics to be performed on the knowledge-base without the current dependency on a specific traffic simulator or GIS system.

Once the network infrastructure has been stored in the knowledge-base then geospatial relationships are identified between infrastructure and land use locations. This primarily consists of identifying the proximity of roads, buildings and public transport access points to each other.

### 3.3.3 Travel Demand Model

A variety of travel demand models have been developed based upon several different techniques, such as the Four Step Model, Activity Based and Agent Based. The focus in this work is on constructing activity and travel schedules based on template activity patterns. The generation of activity pattern templates takes place prior to the knowledge-base construction as they serve as an input to that stage. However, since travel demand models exist that do not require activity patterns it is discussed at this stage.

#### 3.3.3.1 Activity Pattern Generation

The activity pattern is a typical data structure of agent-based models and provide a template, or skeleton [19, 20], from which a person's activity and travel schedule are assembled according to the context, see Figure 3.5. The generic templates are populated with contextual instance data to form a schedule. In the example, assumptions are made that the initial and final activities are extended to fill the entire scenario time-period, but alternative implementations may make different assumptions. The locations and travel choices will create varying travel durations between activities. Waiting times are included in the activity or travel stage [10]

to remove empty periods, but time gap filling approaches vary, e.g. extend activity duration; allow travel time contingency or plan extra activities.

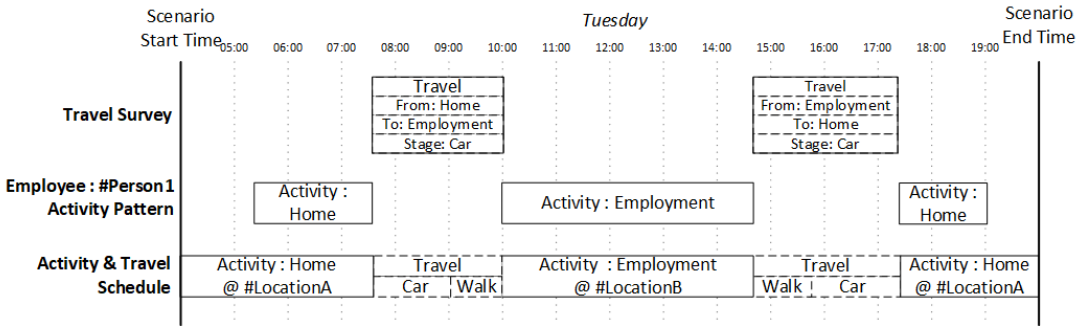


Figure 3.5: Diagram of time-line representation of a travel survey as the basis for an Activity Pattern template.

The patterns available may be fixed [8] or derived from travel diaries of a sample population using classification algorithms [20, 86]. The travel diaries may be used to derive the activity choices, durations, indicative start and end times, journey mode and household co-operation. Sets of activity patterns can be associated with households and individuals in the synthetic population based upon the corresponding characteristics. Any activity generator could be utilised once its output is serialised to RDF and then aligned with the knowledge-base schema.

#### 3.3.3.2 Scheduling

The activity patterns are applied to a context, i.e. population, geography, and scenario, different from that in which they were derived to form a schedule. Adapting to an alternative context requires consideration of preserving minimum activity duration; tolerance for timing slippage; and extending or including additional activities to fill time gaps. The scheduling process typically covers a single day, but future developments include multi-day scheduling; improved co-operation between household members; and the incorporation of social network data [8]. Schedulers also need to ensure that consistent travel takes place for tours that return to same location and journeys that return to an individual's reference location [10].

Activity prioritisation is used in some scheduling approaches to allow co-operation between household members' schedules. Mandatory activities, e.g. education and employment, are determined first with invariant start and finish timings. Maintenance, e.g. food shopping, and discretionary, e.g. leisure, activities are then assigned with flexible strategies for duration and inclusion [9]. Schedule coordination, such as adults escorting school children and shared vehicle usage, is modelled by mandatory activities being scheduled on an individual basis and then reviewed for cooperative travel across the household before allocating lower priority activities.

#### 3.3.3.3 Trip Planning

This stage is a key distinction between travel demand models with travel decisions typically consisting of activity location; trip mode and activity time frame [87]. The scheduling stage determines the short term planning over a day or week, while this stage considers the near term planning of individual trips within the day. Design decisions are influenced by choice type and resolution order due to their interdependence and impact on later decisions [9, 20, 88]. Route choice is a further development in activity-based models [8, 9], but already a feature in some agent-based models [19]. These decision-making and trip planning processes represent a range of design approaches (Section 1.2).

#### 3.3.3.4 Network Routing

The topology of the transport network has an influence on the travel decisions taken by persons and the connectivity between locations, infrastructure and services. Many transport simulators provide tools to perform routing using a variety of algorithms, e.g. A-star and Dijkstra. This module interprets the network supply information in the knowledge-base to inform the travel demand models and removes dependency on transport simulator tools. The removal of this dependency will enable alternative algorithms to be considered that are developed for large road network datasets [89, 90] or satellite navigation systems [91] and which may be the focus or requirement of an investigation.

This module can be executed either prior or dynamically during demand mod-

elling, with generated routes stored in the knowledge-base for re-use or reference. However, the prior option requires an exhaustive set of all route combinations which becomes very large as the number of points of interest increases.

An area for future work is consideration of semantics present in road network datasets, such as temporal context, trip purpose and physical characteristics. These features are lacking in the examined routing tools such that routes ignore private or resident only access, road closures at specific time periods, weather events, traffic signalling, previous travel experience, and tall vehicles under low height bridges etc. These are characteristics that affect routing and can be accommodated in the proposed knowledge-base. Alternative services could also be modelled, such as taxi services, car sharing, lift sharing and autonomous vehicles. Therefore, this module presents a wide set of complex factors that would benefit from being developed separately to traffic simulators so that greater comparison can be made between implementations and better inform the demand modelling stages.

#### 3.3.3.5 Feedback and Learning

The process of feedback and learning is based upon the relative success of the proposed travel plan. The outcome of the simulation process is fed back into the travel demand model to inform the decision-making process. Generally learning and adjustment to schedule and trips is performed following a batch simulation of a whole schedule [32]. However, iterative schedule adjustment due to travel delays during simulation have been developed [8, 19].

The whole knowledge-base can be made available for interface with simulator APIs [92] or as part of an artificial transport framework [33]. Therefore, the simulation stage can be performed as an iterative step-wise process with travel demand being adapted as simulation conditions change. This would facilitate both the iterative inter-simulation and reactive intra-simulation rescheduling.

### 3.3.4 Travel Simulator Interface

The outcome of a travel demand model is a person's activity and travel schedule. Interfacing directly with traffic simulators, or aggregation of travel demand into

Origin/Destination matrices, can be achieved through knowledge-base query and data conversion into the required format. The information required by a specific traffic simulator may not require the complete schedule with requirements varying. For example, MATSim [32] as a minimum requires people to be identified with a plan of activity type, end time, travel mode and location in XML format. SUMO [46] requires both person schedules and vehicle routing with start and stop locations and departure times in XML format. TRANSIMS [86] requires person and vehicle information including the household, person, purpose, mode, vehicle, start and end locations with departure and arrival times in CSV format.

Network supply information is an additional input that is already required in the knowledge-base for travel demand generation. Simulators typically support their own bespoke file format for configuration parameters and network supply, but some standard network topology formats are supported. Therefore, interfacing to a simulator will require specific interfacing modules, but some topology serialisations would be re-usable.

## 3.4 Design of Framework Software Application and Configuration

The previous sections have discussed the framework by considering the module components that are required for generation of travel demand. In this section there will be examination of the organisation and execution of the framework by considering the design of the framework as a software application and the potential physical configurations. These alternative configurations offer flexibility to the user in how data is obtained and modules are utilised to undertake the travel demand generation process.

### 3.4.1 Design of Framework Software Application

This section identifies the software components that are required for the framework. These software components are the building blocks of the framework and the travel demand generation process and therefore it is important to provide

clarification of their role. The three primary software components are the User Application, Semantic Web library and the Graphstore as shown in Figure 3.6.

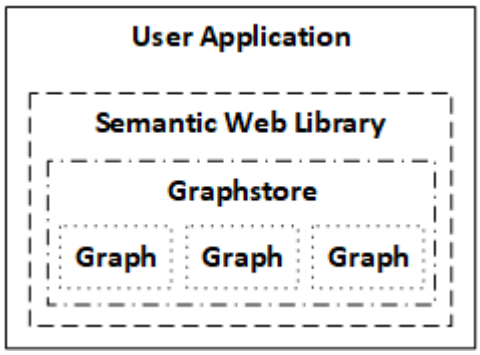


Figure 3.6: Diagram of the software components of a user application.

- User Application: represents the entry point for executing the travel demand generation process. The User Application could provide a Graphical User Interface (GUI) or Command Line Interface (CLI) but may be programming code custom written by the user. The User Application will load any local knowledge-base with dataset or configuration data required for execution.

- Semantic Web Library: provides an Application Programming Interface (API) that complies with the published standards, e.g. RDF and SPARQL. The User Application utilises this library to access Semantic Web technologies. The library is dependent upon a specific platform and programming language. However, the Semantic Web data formats are interoperable so that the output of one Semantic Web library can be utilised by a different library.

- Graphstore: the storage provider for one or more RDF graphs. The graph data may be held in-memory or persistently on disk depending on the design. Implementations are specific to each Semantic Web library but the contents can be serialised to an interoperable Semantic Web format. The knowledge-base, the collection of knowledge and data, is manifested by the data held in one or more Graphstores.

The interoperability of the Semantic Web standards decouples applications located on the same or different physical machines. They can each have different standards compliant Semantic Web libraries and Graphstores as illustrated in Figure 3.7. Each software component can be developed in a discrete manner. Therefore, developers can make their own contributions without concern for compliance with specific interfaces or using a single platform. Following a common schema, as discussed in Chapter 4, simplifies the exchange of data between discrete components. However, compliance to the data schema is not mandated as discussed further in Chapter 5.

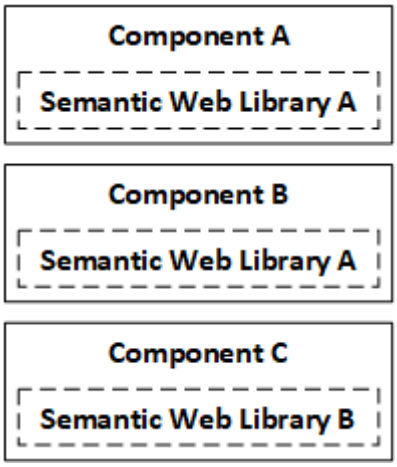


Figure 3.7: Diagram of the alternative component configurations using different Semantic Web libraries.

Applying the discrete component design allows the functionality of the different process stages to be separated from the user application, termed in this work a *module*. Similarly, the data that forms the knowledge-base in the Graphstore can be separated from the User Application. The knowledge-base can be further separated across multiple physical Graphstores. The arrangement of these two components are shown in Figure 3.8.

The Semantic Web is designed for on-line interoperability using the HTTP communication protocol. Therefore, Semantic Web libraries provide functionality for sending and receiving HTTP requests as part of the HTTP servers. The HTTP server provides a service for accepting SPARQL queries and responding with the results from those queries. This functionality can be used by both these

components to make them remotely accessible. Similarly, a User Application, as shown in Figure 3.6, would have access to using the HTTP server and constructing HTTP requests and responses.

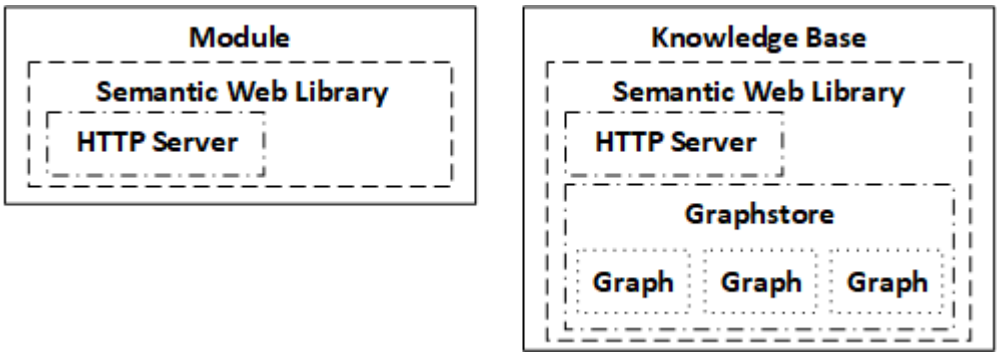


Figure 3.8: Diagram of the software components of a module and knowledge-base.

The three components of User Application, Module and Remote knowledge-base can therefore be arranged in a variety of configurations. These configurations influence the physical arrangement of computers that can be used to execute the framework and will be discussed further in the next section.

### 3.4.2 Configuration of Framework Components

The previous section identified three software components that are utilised within the framework: User Application, Module and Remote knowledge-base. This sections will discuss how these components can be configured to provide alternative physical arrangements. These arrangements provide choice to the user in where they obtain the data and functionality to execute the travel demand generation process. The wider the choice and the less burdensome the access process then the greater potential for robust investigations. There will be consideration across four main configuration examples although further variations, e.g. mixes of local and remote knowledge-base, could be applied.

#### 3.4.2.1 Local Knowledge-Base and Local Modules Configuration

The first configuration to be considered, and most straightforward, is where the User Application, Modules and knowledge-base are all located on a single physical

computer. This represents a conventional set-up where an application is supplied with configuration and data files to be executed in a local environment and is illustrated in Figure 3.9.

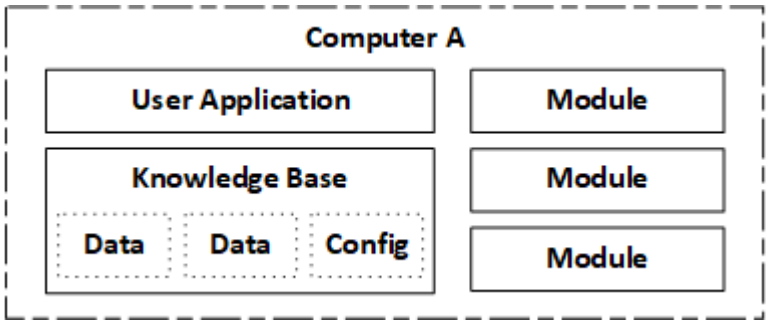


Figure 3.9: Diagram of the local configuration using a single computer.

A graphstore is able to store multiple graphs of data. Similarly, multiple Modules may be present to be executed. Therefore, a configuration needs to be expressed by the user to select the correct graphs, functionality, and queries to be used in a specific execution. This configuration is described by a RDF data structure, termed *Framework Configuration*, called *Config* in the diagram. The further details of the *Framework Configuration* are discussed in more detail in Section 5.3. However, in this context it provides a directory to locate data and queries to be used. The User Application can access all the required data and modules locally using the provided configuration.

#### 3.4.2.2 Remote Knowledge-Base and Local Modules Configuration

The next configuration considers where data is located on remote computers as published RDF datasets. The term *remotely accessible* or *remote* are applied for any service that is available outside of the application. Therefore, it is remote to the application rather than remote to the physical computer. The use of the term *Computer* in these diagrams is a simplification for clarity.

The data is accessible through SPARQL endpoints provide by HTTP servers and can be retrieved using *Federated Queries*. The RDF graph structure permits some or all of the data to be located remotely and combined with local data and modules. This is illustrated in Figure 3.10 where part of the knowledge-base has

been located on a separate computer and is accessed over a HTTP connection.

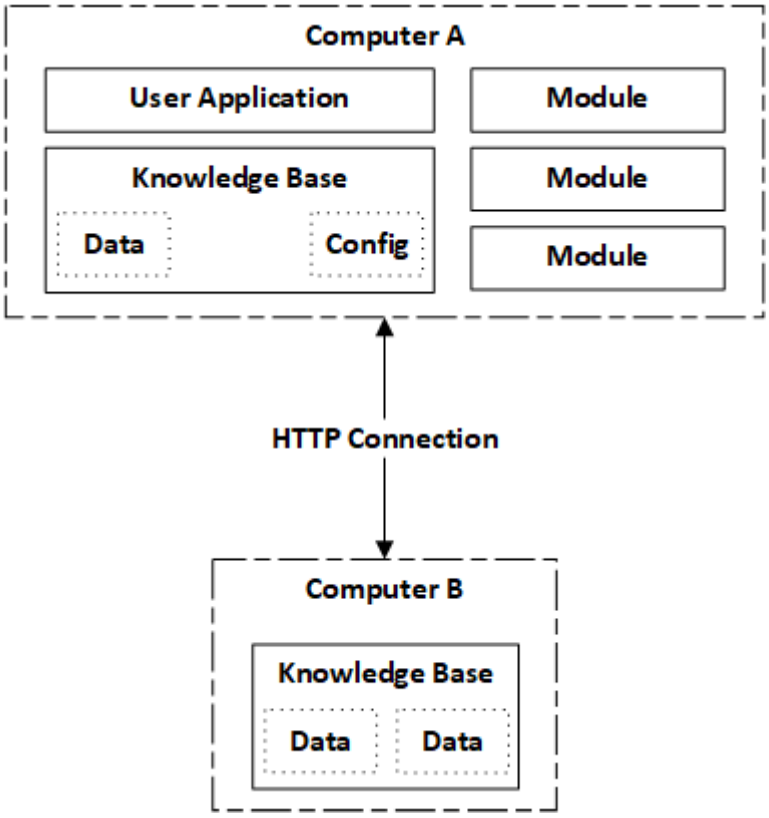


Figure 3.10: Diagram of a remote configuration with data held on a remote computer.

The knowledge-base in Computer B contains two graphs of data. These could be different sets of data with each being used during execution. Alternatively they could be different versions of the same data with each being used in separate *Framework Configurations*. Each dataset is referenced in the *Framework Configuration* by their Unique Resource Identifier (URI).

#### 3.4.2.3 Local Knowledge-Base and Remote Modules Configuration

The positioning of components is not limited to the data of the knowledge-base. The Modules in Figure 3.10 are positioned locally and accessing remote data. Therefore, the inverse would also hold with the Modules positioned and executed remotely on data held locally, as illustrated in Figure 3.11.

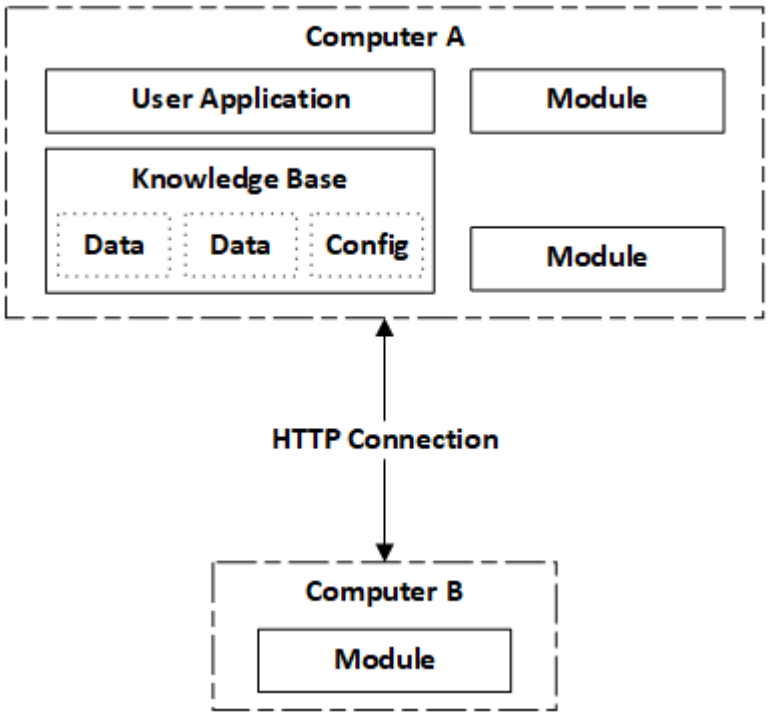


Figure 3.11: Diagram of a remote configuration with module held on a remote computer.

The *Framework Configuration* supplies the information required to access the graphs of data in a generic Semantic Web-based approach. This enables developers to implement their models and functionality in their own environment. They do not have to publish across multiple platforms or provide documentation on how to deploy the module. Instead resources can focus on explaining the design, data requirements and implementation of the module to inform users and obtain feedback for future developments.

The user does not have to deploy the module locally avoiding the time and resources required to address configuration or environment issues. There can instead be a focus on preparing the data relevant to their investigative scenarios. The utilisation of a new version or alternative module requires only adjustment of the URI referring to the module and its hosting service.

#### 3.4.2.4 Remote Knowledge-Base and Remote Modules Configuration

The final configuration is when both data and modules are positioned remotely. Since both the data and modules can be configured to be remotely retrieved then it is not required that either are local. In this case only the *Framework Configuration* is held locally by the user.

The *User Application* retrieves any data it requires from the remote knowledge-base, e.g. households in an area, to initiate or fulfil the data requirements of the initial remote Modules. However, once begun these Modules may also make direct data requests of the remote knowledge-base. This is achieved by referring back to the *Framework Configuration*. The local knowledge-base could then be the recipient of the final results of the travel demand generation process or retrieve them from where they are remotely held once completed.

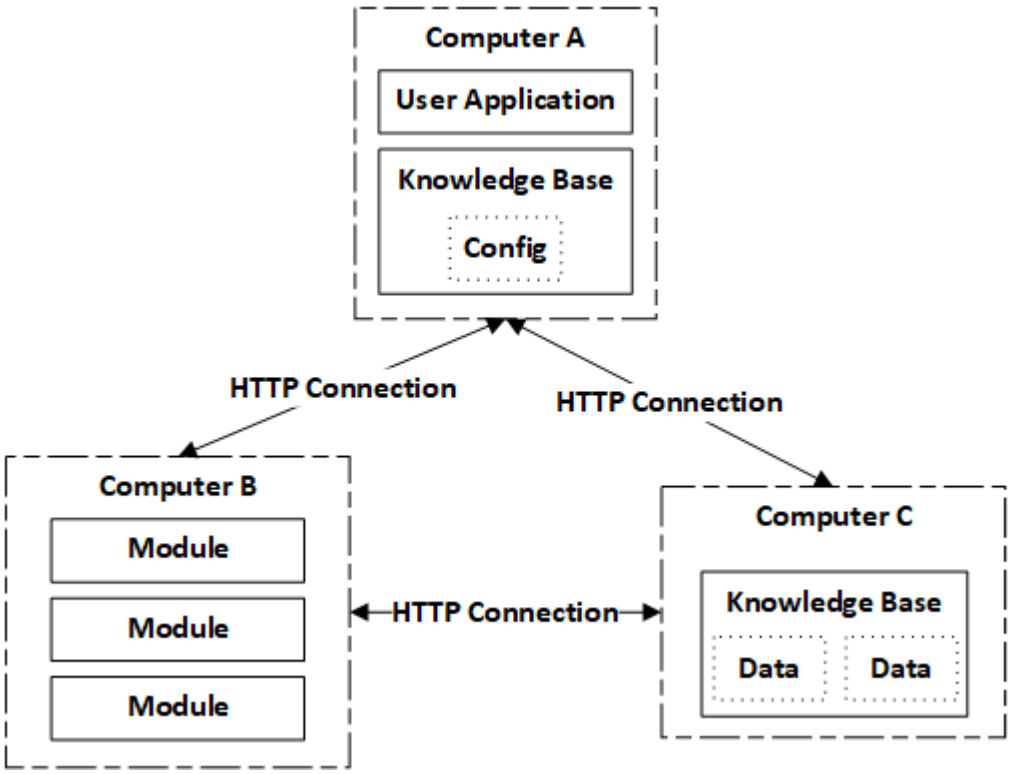


Figure 3.12: Diagram of a remote configuration with data and module held on remote computers.

#### 3.4.2.5 Implications of Remote Configurations

These remote configurations can be scaled across as many computers or services as required to perform the process, e.g. modules and datasets could be stored on the same computer or each hosted by a different computer. By defining Modules and knowledge-bases as services the functionality and data can be sub-divided as an implementation requires with the user able to supply the *Framework Configuration* required to satisfy it. For example, Module A is implemented with three sub-modules: Module B, Module C and Module D. The user constructs a configuration that redirects Module A to use Module C1. The Module C1 has two sub-modules and the user redirects one of these to their own implementation of the functionality while continuing to use the other default sub-module.

The HTTP request is sent using a URL that could be responded to by an on-line service or by a local service on the same physical computer. Similarly, requests sent to different URLs could be serviced by a single physical computer. Alternatively, requests to a single URL could be serviced by a cluster of computers with requests being allocated to balance workload across the cluster. The transmitter of the request does not know, or need to know, where the response is actually coming from or how it has been handled, although there is the need to ensure the response is trusted (Section 5.5).

This means that the physical location and configuration of a service can be changed without any impact upon the requesting application and so decoupling requester from responder. These general benefits of a service orientated architecture mean that the framework can be re-organised and up-scaled. Therefore, development of complex and computational demanding functionality, such as human behaviour models, can be developed and then executed upon clusters of distributed computers.

The potential to host multiple configurations provides benefits beyond conventional set-ups for the travel demand generation process. It does not restrict a user to pre-specified set-ups and allows easier access to data sources and module implementations. This in turn reduces the burden of setting up and executing the travel demand generation process, e.g. by removing the extract transform load (ETL) process (Section 5.2), so that investigation can be undertaken quicker.

The control of these configurations and enabling switching between them without breaking the execution process is vitally important. There is also a computational cost in preparing and transmitting HTTP requests that can impact execution times. The developed approach, the *Framework Configuration*, for addressing these matters are discussed further in Section 5.3.

## 3.5 Chapter Summary

This chapter has introduced the design of the framework components to address Research Question RQ1 by identifying the knowledge-base as the underlying data repository for the travel demand modelling process. The main stages of the travel demand process have been further described as a set of module components. These modules interact through the common data concepts that have been modelled in the knowledge-base.

It has been identified that the existing three stage process of Population Synthesis, Travel Demand Model and Traffic Simulation needs to complimented by a Knowledge-Base Construction stage. In this stage the input datasets are aligned to the common schema and with each other. This can include the spatial allocation of households to residences; classification of concepts into the user's schema based on properties and characteristics; and the reconciliation of land use with road network infrastructure.

The development of a common schema will facilitate interoperability between modules and reduce the need for user intervention in transforming concepts contained in the datasets. In Chapter 4 there is discussion of the developed common schema of data concepts to investigate Research Question RQ2, while Chapter 5 will consider how adaptations can be made when modules do not follow the same schema.

The Travel Demand Model stage was further decomposed into five modules to develop an activity-based approach. These modules were identified as examples of functionality to which alternative techniques could be applied. The inclusion of alternative and existing travel demand models would substitute some or all of these modules by ensuring that data concepts at the boundary of the module or stage are satisfied. Chapter 6 discusses in further detail the implementation of

the Scheduler, Trip Planning, Network Routing, and Traffic Simulator Interface modules in the development of the prototype.

There was also discussion of the application design and physical configuration of the framework which proposed both local and remote configurations. An objective of the framework is to provide users with greater control over the selection of modules and the sources of data. Therefore, accessing remote sources supports this objective but also introduces the potential for user error in access and integration. Chapter 5 investigates ensuring data and query quality provided by none expert users and also discusses Research Question RQ3 for the selection of these alternative sources.

# Chapter 4

# Semantic Modelling of Travel Demand Generation Data

## 4.1 Introduction

The previous chapter discussed the components for travel demand generation with focus on activity-based models and how a knowledge-based and Semantic Web approach can be applied. This chapter discusses the knowledge modelling of the data concepts identified for the framework's components (see Figure 3.3). These data concepts form a schema describing the framework's main concepts and their interaction as the basis for addressing research question RQ2. This schema forms the basis of the implemented modular prototype design discussed in Chapters 6 and 7.

The chapter begins with discussion of general principles and design patterns for semantic web schema design. The top-level data concepts for travel demand modelling are then described before more detailed examination of the fundamental data concepts which are widely re-used before focusing upon the identified data concepts. The data concepts have been split into those travel demand concepts that should be generally recognisable from real-world experience and those that are specifically related to the travel demand modelling and simulation process.

Throughout illustrative examples are provided to demonstrate how the schema can be extended by a user for their investigative scenario. There is then con-

sideration of the utilisation of the schema through detailed examination of its application to key classes and the inter-relationships between concepts. Finally, there is discussion of the graph organisation of the knowledge-base and how it can be applied to organise data into distinct datasets for operational use.

### Notation of Schema Diagrams

This chapter and later chapters use graph diagrams of the schema and supporting examples to show class, instance and property relations as illustrated in Figure 4.1. The notation shows classes as nodes (boxes) linked by edges (lines) of *object property* or *sub-class* relations. *Object property* relationships are shown primarily through edges but secondary *object* and *data* properties are listed below the class names with corresponding class or datatype.

Property cardinalities are also shown to indicate the schema shape and enable data validation. Only namespaces from public vocabularies are shown in the diagrams to assist their identification. Classes are shown by a circle and properties by a rectangle. The illustrative example diagrams show individuals, or instances, of classes using the *rdf:type* property and indicated by a diamond symbol.

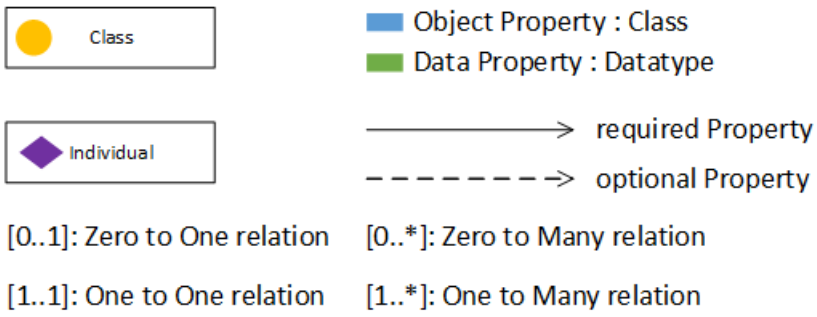


Figure 4.1: Diagram of notation used in schema diagrams.

## 4.2 Semantic Web Schema Design

The main technologies of the Semantic Web have previously been discussed with regards to their functionality (Chapter 3) and their general application (Section 1.3). In this section there will be discussion of the principles upon which the

Semantic Web has been developed in relation to data modelling. It will also outline the particular design considerations and published public vocabularies that have been applied during the development of the framework schema which is described later (Section 4.5).

### 4.2.1 Semantic Web Principles

The core technology of the Semantic Web is the Resource Description Framework (RDF) from which a variety of technologies have been developed to fulfil specific purposes. Therefore, the selection of Semantic Web technologies is dependent upon the purpose. If an application does not require querying of data then it does not require SPARQL. If an application does not require Description Logic inferencing then it does not require OWL.

The choice to not utilise a technology, e.g. OWL, does not preclude the use of another technology, e.g. graphstore data storage and SPARQL data retrieval. There is also overlap between technologies where the same outcome can be achieved using different technologies, e.g. OWL, SPARQL, SHACL and SPIN can all be used to infer data and determine class membership.

The premise of the Semantic Web is based upon the design principles of modelling for re-use and on-line access to data. Basing the framework upon the Semantic Web enables sharing of data and schemas in an inter-operable manner while allowing the users and module developers choice in the technologies they apply. In this section there will be an outline of three principle areas of the Semantic Web to highlight the impact that they have on the development and usage of the schema and framework.

#### Anyone can say Anything about Any topic (AAA)

The premise of the World Wide Web is that the publication of information is controlled by the information producer. This means that there is not a central authority determining the accuracy or appropriateness of information and it is instead a consideration of the consumer. This does not mean that information cannot be produced and attributed to authoritative sources, but that any information can be produced by any provider. This principle has been termed as

*Anyone can say Anything about Any topic* and means in practice that information from different sources can be contradictory, inaccurate, deceptive or out dated [36].

Designing to the underlying AAA principle leads to the notions of alternative models to represent the same concepts and producing models that can be re-used and extended. The support of alternative data models can be seen in the design of the Resource Description Framework where properties can be asserted for classes without constraint or contradiction until a perspective of the context is applied through a schema or language, e.g. RDFS or OWL.

The re-use and extension of data models allows sharing of best practices and consistent structuring of data, while enabling its application for new purposes. An anti-pattern of the sharing for re-use modelling process is *creeping conceptualisation* where the developed schema exhaustively anticipates all applications of the model [36]. It has been sought to avoid this anti-pattern with the proposed schema by limiting it to the essential classes and properties.

In the following explanation of the schema a distinction is made between those elements which are deemed necessary for the proposed framework and providing illustrative examples intended to give context or understanding. Indeed it is intended that one of the benefits of applying a Semantic Web approach to the traffic and transport simulation domain is to allow flexibility in the data model so that a wide variety of implemented models and tools can be utilised by allowing alternative properties and additional classes to exist alongside each other.

### Non-unique Naming Assumption (NNA)

The *AAA principle* discussed previously introduces another aspect of the Semantic Web which has implications for inferencing. The decentralised approach allows publishers to describe concepts and models in the manner of their own choosing. Therefore, two publishers may select different names (URIs) to identify identical resources. Both identifiers are correct in their own context so it cannot be assumed that different identifiers refer to different resources, unless it is established that the two resources are not the same. This is different to other modelling approaches where it is assumed that resources are not the same, unless

it is established that they are the same.

### Open World Assumption (OWA) and Closed World Assumption (CWA)

The Semantic Web is designed upon a principle of a distributed network. In such an environment, information can be separated and split over the numerous participants of the network. Each participant may only hold partial information about any resource or a participant may not be accessible to provide their information.

This has lead to the development of the principle Open World Assumption (OWA) and is applied in some schema languages, e.g. OWL [93]. The application of these schema languages allows the inferring of additional facts not present in the data but implied by the structure and content. The OWA principle means that conclusions cannot be made about a resource if it relies on the assumption that complete information is available. Contradictory information may be held on another part of the network which would invalidate the previous conclusion.

This has implications for inferencing such as the absence of a statement in a dataset does not mean the negative of the statement can be assumed, e.g. the absence of a visitor record for a location does not mean that there has never been any visitors. Applying this assumption reduces the inferences, i.e. conclusions, that may be found when applying a schema to a dataset but it does ensure that those conclusions will never be contradicted. There can also be further complications introduced that increase the complexity and place restrictions on modelling achieved by the schema language [94].

This is quiet different to Object-Orientated Languages where a Closed World Assumption (CWA) is applied [85] and has also been adopted for some Semantic Web technologies [95, 96]. Following this principle, it is assumed that the information available is complete for the resource. In CWA it is accepted that contradictory information may arise later but arriving at the conclusions at the current time is more important. This means that more definite inferences can be made, e.g. no visitor records for a location means that the location has had no visitors. It is commented that in many pratical applications OWA does not make a difference or can be ignored in favour of CWA [36].

### 4.2.2 N-ary Relationships

The fundamental structure used in RDF and other schema language is the *triple* [97]. These *triples* express statements of the form *subject*, *predicate* and *object*. These form a *binary* relation between the *subject* and the *object*. For example, *Peter owns X123* is a statement that identifies the *owns*(predicate) relation between *Peter* (subject) and *X123* (object). Additional statements can be created that expresses Peter's ownership of multiple cars as shown in Figure 4.2.

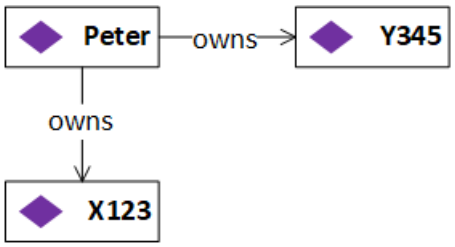


Figure 4.2: Diagram of schema multiple relations of same property.

Difficulty arises when additional information is expressed about the relationship between Peter and his cars. For example, the date that the car was purchased by Peter. The *n-ary* relation has been proposed as a design pattern to resolve this issue [98]. There are two forms to the pattern based on creating a class and introducing a list. The latter is proposed for certain specific cases of *n-ary* relations and is not considered further as the selected approach in the core schema relating to lists is to use the Ordered List vocabulary (Section 4.2.3) due to its benefits with SPARQL queries.

In the *class-creation* approach a connecting class is created to form a bridging relation between the related individuals. There are two approaches proposed based upon whether there is a *distinguished participant* as shown in Figure 4.3. The selection of either is reliant upon modelling context with both approaches being applied in the schema.

In the *distinguished participant* approach it can be seen that Peter is the subject at the root of the graph, while in the *no distinguished participant* the emphasis is placed upon the created *Car Purchase* class as the subject with both expressing the same information. It can also be seen in this example that for both approaches that there is no direct relation between Peter and the cars,

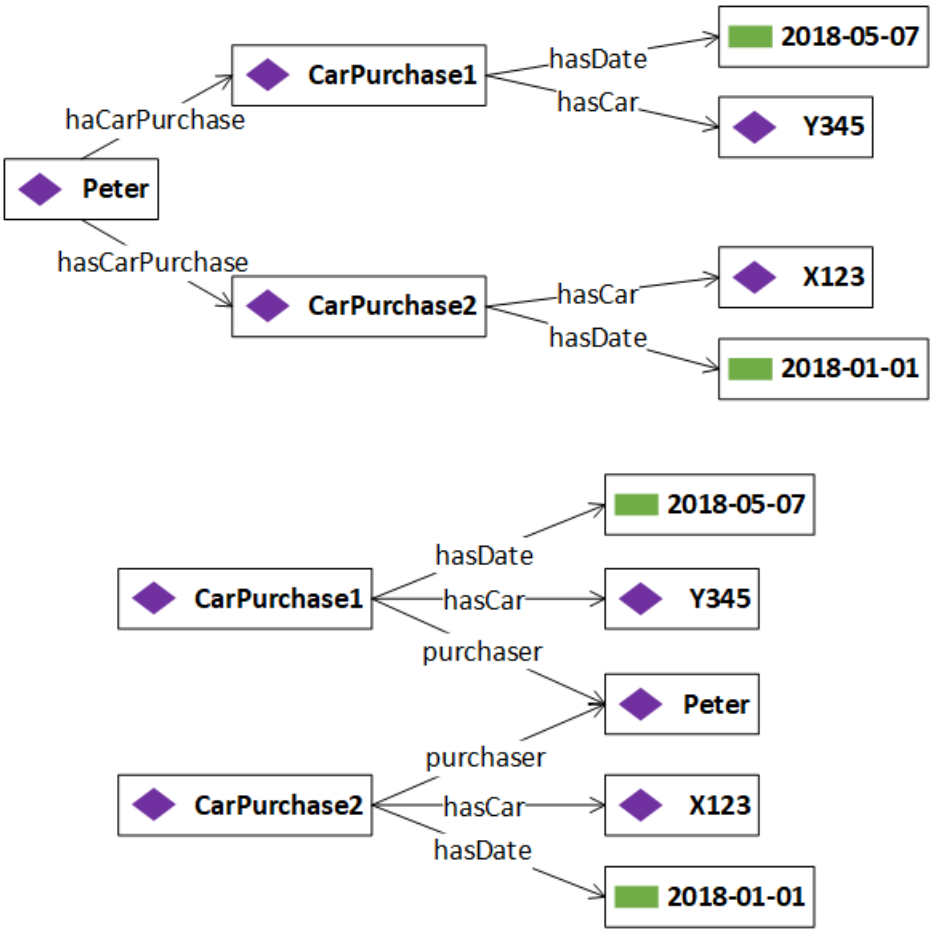


Figure 4.3: Diagram of example N-ary patterns for *distinguished participant* (top) and *no distinguished participant* (bottom).

which would have to be asserted or inferred if required.

It is noted that adopting the *class-creation* approach for *n-ary* relationships increases the complexity of expressing class restrictions in OWL. However, the desired constructs can still be achieved and usage of OWL with the framework and schema is an optional user choice.

An alternative approach to *n-ary* relations is the use of *reification* which was included in the original RDF specification, but discontinued in the later version [97, 99]. The *reification* approach was intended to provide additional information about a specific statement, e.g. the origin of a statement, rather than additional statements in the knowledge-base. The *reification* design pattern was found to

cause issues for inferencing with reasoners and distrupted the graph structure leading to its discontinuation [99]. Therefore, it has not been applied as a design pattern in the core schema.

### 4.2.3 Ordered Lists

This section discusses the approach that has been applied for organising items in an ordered list. A common data structure is the need to group together items that have a related purpose, generally termed a *collection*. The RDF [100] and RDFS [101] vocabulary provide several different types of collections which vary in semantics according to whether repetitions are permitted and the items are ordered.

The use of lists, which provide an ordered sequence, occurs frequently in the core schema to assist in organising items, e.g. the time intervals of a schedule. In some cases these items could be grouped as an unordered collection and sorted by characteristics when retrieved. However, this introduces additional processing that could be avoided when the collection is created by applying a list structure.

The RDF and RDFS collections relating to lists are minimalist approaches which define the list as a chain pointing from element to element. Each property of the chain is defined according to its position in the chain. This makes these structures inefficient to retrieve in SPARQL, which can resolved through implementation specific extensions, but are not part of the SPARQL standard.

The Ordered List Ontology [102] defines a vocabulary, illustrated in Figure 4.4, based upon a root resource to the list that has a series of slots. This root resource can be further extended by other classes and properties. These slots have properties relating to their stored item, i.e. the contents of the list, its integer index position in the list and consistent properties to form a chain structure with other slots, in the same manner as the general RDF list structure. The number of slots in the list, and therefore items, is also a property of the root resource.

The SPARQL protocol provides syntax for ordering by property values so that query results are returned in an ordered manner. This means that data following the Ordered List Ontology structure can be returned in order of index or specific items can be retrieved based on the index of their slot. A single item can also be

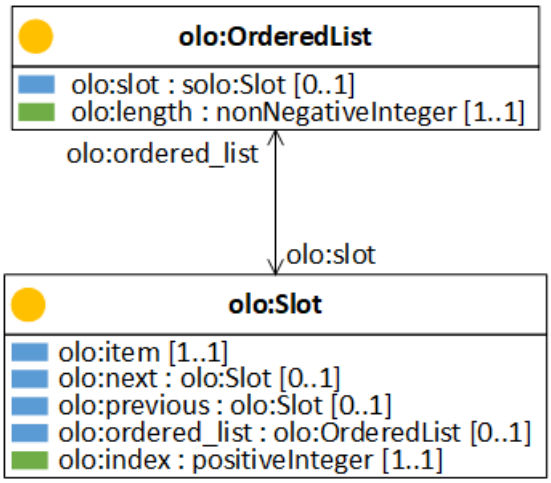


Figure 4.4: Schema for published Ordered List Ontology vocabulary. [102]

referred to in multiple lists. Therefore, applying this vocabulary provides simplification to SPARQL querying without relying upon implementation extensions for a commonly applied data structure.

### 4.2.4 Value Set Design Pattern

This section discuses a design approach that has been applied repeatedly in the core schema. It outlines the basis for this approach in the context of Semantic Web language restrictions and assisting with usability of the schema.

The Semantic Web defines several languages that provide rules and conditions to structure RDF data and form the basis for inferencing, including Resource Description Framework Schema (RDFS)[101] and Web Ontology Language (OWL)[103]. There are three profiles of the OWL language: Full, Description Logic (DL) and Lite. Each offer different restrictions and features which can influence computational completeness and decidability. The Description Logic profile is the basis of the OWL2 language [93] which superseded OWL. Both the OWL and RDFS languages are based on the Resource Description Framework (RDF).

RDF provides a structure to data based upon triple statement of *subject, property, object*. The subject and property must be *resources* while the object can be a *resource* or a *literal*. *Resources* are represented by Unique Resource

Identifiers (URI) while *literals* are strings with corresponding datatype.

*Classes*, which form sets to group items together, are themselves a type of *resource*. Instances of these *classes* are termed *individuals*. In designing a schema two common features are referring to a group of individuals, i.e. a class, and defining common characteristics for the group. An example of this would be the following statements:

*Example Statements:*

*Peter is qualified to drive cars and vans.*
*Peter owns the car X123.*

These statements identify a specific car that Peter owns and the class of vehicles that he is qualified to drive. These could be represented by the schema illustrated in Figure 4.5.

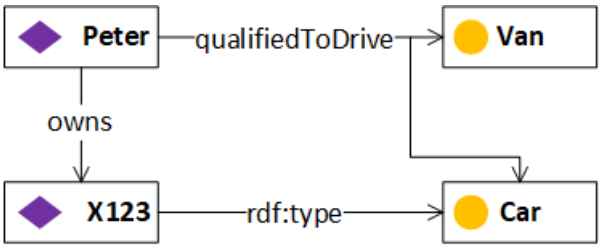


Figure 4.5: Diagram of schema for example statements complying with RDF standard.

A triple of the form *Peter qualifiedToDrive Car* is permitted in OWL Full and RDFS but is not permitted in OWL DL, OWL Lite or OWL2 [104]. In these latter languages there are restriction that separate classes, properties, individuals and data values. This means that *classes* cannot be the subject of *properties* except for specific exceptions, such as *rdf:type*. Applying a reasoner based upon these languages would result in failure and many Semantic Web tools are developed based upon the restrictions of OWL DL or OWL2 [103]. Adopting this approach in the core schema would therefore preclude usage of these languages and many tools by users.

To comply with these restrictions a Value Set, or Enumeration, design pattern [104, 105] has been applied. In this design pattern the classes of interest are

asserted as individuals as illustrated in Figure 4.6. The asserted individuals would themselves be used as the object of statements about the general group.

These individuals often represent sibling classes and so themselves could form a class. In addition to complying with all the language restrictions this approach does not prevent the user from applying their own classes to group individuals, e.g. *Person* for *Peter* and *Car* for *X123*. To avoid confusion between the class and individual the term *type* has been used in the core schema.

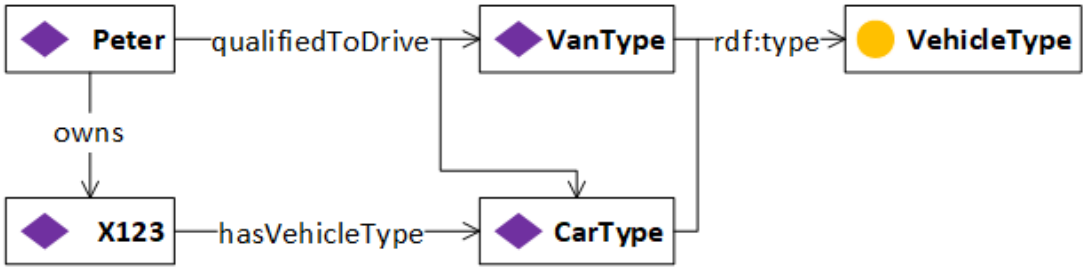


Figure 4.6: Diagram of schema for example statements applying the value set design pattern to comply with OWL DL/OWL2.

The second identified modelling feature is defining common characteristics for a group. This assists in maintaining consistency if values are modified and reduces repetition in the data. The properties available for a user to define the relations between the subject and object of a triple are *object* and *datatype* properties.

In OWL DL, OWL Lite and OWL2, these properties are restricted to being between individuals and not classes. This presents the previous issue in using these properties on classes. This can be resolved by using the *type* individuals asserted to represent classes to also be the subject of the properties for the common characteristics of the group members. In conclusion, the use of the Value Set design pattern ensures language compliance for the user; permits the classification of individuals into the user's own classes; and allows the defining of common characteristics.

## 4.3 General Data Concepts for Travel Demand

The previous chapter discussed the identified modules of the framework for travel demand and described their purposes. These module components are intended

to interact through the data present in the knowledge-base to support the overall objective of travel demand modelling. The data requirements of the modelling process incorporate a wide range of physical and abstract entities.

This section will provide general description of these entities and their organisation into separate domains. These domains are broader than the specific concepts of the core schema which are described in the following sections. Additional data and concepts are intended to align under these general concepts, but are not required to operate the core schema with the module components, e.g. vehicle emissions data.

A summary list of examples and key terms is provided, but is not intended to be exhaustive. Figure 4.7 illustrates the top level domain concepts that have been identified. Interconnections exist between these domains, particularly for all domains to the geospatial and temporal, but these are not shown for clarity.


Figure 4.7: Diagram of schema concept domains.

Travel demand is based upon the movement of people and goods from one location to another. The purpose of movement for people can broadly be categorised as people accessing goods or services provided by organisations; fulfilling or accessing occupations with organisations; and social interactions with other

people. The purpose of movement of goods is for delivery to people or to organisations where they may be consumed or further transported at a later time.

Fulfilling these activities requires travel through a geospatial environment from place to place. This travel utilises transport vehicles and infrastructure with the available options and duration varying temporally.

The following statements have been used as a conceptual basis for the identified groupings:

- People have demographic characteristics.
- People form relationships with other people.
- People have access to transport choices.
- People make activity choices influenced by demographic characteristics.
- People reside in places as household groupings.
- People travel to places for activity choices.
- Organisation provide activity choices and goods at places.
- People and organisations require goods.
- People and organisations utilise vehicles for transport and movement of goods.
- Vehicles operate within transport infrastructure.
- Environments vary geospatially and temporally.

## Geospatial and Temporal

These concepts are incorporated into all the other top level concepts and represent that traffic demand modelling and simulation takes place over a spatial environment with varying time component. The spatial environment requires the capturing of specific points and more complex line-string and polygon shapes. The interaction of these shapes through their spatial relationships, e.g. shapes

overlapping or containing each other, enables relevant selections and interactions to be made.

The representation of time variation requires the capturing of both specific points in time such as an event/instant or the interval in time when a condition may or may not be available. The tracking of temporal factors has an impact both within a scenario and between scenarios. This can include the selection of activities to undertake, availability of transport infrastructure or comparing the performance between simulated and planned trips.

- Point
- Linestring
- Polygon
- Instant
- Interval

## People

This concept represents both the specific details about an individual, e.g. age, household composition and occupation, along with the aggregated population demographics for an area from which individuals are derived during the Population Synthesis component. It is the key concept with interconnections to all other concepts as the selection choices and composition of the environment are derived from people. People individually, or in groups, form households that are used as a common collection in demographics and simulations as it can determine the resources and decision making requirements that an individual faces.

- Age
- Gender
- Employment
- Household composition

- Vehicle ownership
- Occupation type
- Driving licence coverage
- Inter-personal relationships

## Spatial

This concept captures the physical locations which people travel between and spend their time. It could include buildings providing services, such as retail, entertainment and leisure, or occupation, such as employment, education and volunteering, or the home residence, such as a houses and apartments. Residing in these places represents that the person no longer needs to travel for a period of time while an activity is undertaken. However, the capacity to supply an activity is finite. The availability of activities is also time variant depending upon the opening hours of the place. It should also be noted that a place can fulfil multiple activities or services, both in terms of the organisations present and the purpose of the individual, i.e. a retail place is also an employment place.

- Buildings
- Land use
- Education
- Employment
- Leisure
- Retail
- Housing
- Administrative organisational areas

## Activities

The undertaking of activities by humans is the source of travel demand. The nature of activities can encompass the full range of human enterprise. These activities take place in different locations, are available at different times of the day and occur upon different days. The choice of activity selected by an individual varies according to their circumstances and preferences. The structure of a schedule may be derived from a template of activities.

- Type
- Location
- Availability
- Duration
- Templates

## Network Infrastructure

This concept captures the transportation infrastructure necessary for people to undertake travel, such as road network, railway stations, bus stops and car parking, along with supporting information, such as road semantics and traffic signalling.

- Road, cycle and pedestrian networks
- Train station and railways
- Tram lines and stops
- Bus stations and stops
- Ferry and airport terminals
- Public transport routing and scheduling
- Car parking facilities

- Petrol stations
- Traffic signals and sequencing
- Access permissions and restrictions

## Vehicles

This concept represents the physical mode of transport utilised by people or goods and their characteristics. In some case this is tightly bound to the infrastructure available, such as rail or ferry travel, while in others it is dependent upon the choices and resources available to a person. An output of traffic simulation is the economic and environmental impact of travel caused by vehicles, e.g. emissions and fuel efficiency. Given the increasing automation and diversifying energy sources of motorised vehicles it is likely that this concept will need to extend beyond the purely motorised aspect to incorporate communication and autonomy.

- Physical characteristics
- Performance characteristics
- Economic characteristics

## Goods

This concept represents the physical products that people and organisations consume and are moved between locations by organisations. The movement of goods creates demand for travel according to the places that supply and demand them. Organisations organise logistics to seek to optimise the speed of movement and reduce costs. This domain is not a particular focus of this work and is incorporated in the abstract sense of freight vehicles moving between locations without consideration of the drivers and inputs for that behaviour. Its inclusion is to recognise its place in the overall process.

## Organisations

Organisations represent collections of individuals that are unified under a common purpose. Individuals may belong to multiple organisations and fulfil different roles within them. Organisations can be broadly categorised into the public, private and voluntary sector according to their purpose and configuration. These sectors would satisfy the employment, education, retail or leisure activities that people seek to undertake, but also include the households into which individuals are grouped. An organisation may consume or supply services from one or more places while a place may support one or more organisations.

In turn the organisation will potentially produce, consume and transport many goods and potentially provide employment or services to many people. Organisations may also be in direct control of the movements of their vehicles or elements of the transport infrastructure. Therefore, the organisation is a unifying concept that draws together and interacts with all the different concepts identified.

- Public sector
- Private sector
- Voluntary sector

## Demand Modelling and Simulation

The process of travel demand generation and simulation incorporates a number of abstract data concepts. These abstract concepts can include the rules or parameters for decision making; the activity and travel schedule; or the potential routes for travel through a transport network.

- Scenario parameters
- Routes
- Schedules
- Mode costs
- Behaviour rules

- Scheduling rules
- Simulation results

The following three sections will examine these general concepts in greater detail and demonstrate how they have been modelled in the schema. The sections have been organised to discuss the fundamental geospatial and temporal concepts; the grounded concepts drawn from the physical world; and finally the abstract concepts derived for travel demand modelling and simulation.

## 4.4 The Temporal and Geospatial Modelling of Travel Demand

The overall purpose of travel demand models is the modelling of the real world that is experienced within a spatial environment that varies through time. Therefore, these two concepts and how they are represented is a fundamental and recurring theme. The handling of these concepts will have an effect on the accuracy, consistency, versatility and re-usability of the schema. This section discusses these two fundamental concepts and outlines the adopted approaches taken in the schema.

### 4.4.1 Geospatial

The examination of travel demand generation is inextricably linked to consideration of geospatial data. Activities and travel performed by people takes place in the physical environment with positioning and interactions having an influence upon outcomes. This influence can be found in all stages of the process, including the preparation of the knowledge-base, generation of the travel demand, executing traffic simulation and the analysis of the modelling results.

The representation of geospatial data covers a wide range of factors and numerous standards have been developed. In many cases there is a distinction between the abstract notion of an object (*feature*) and the physical description of its geospatial shape (*geometry*). This enables different *geometry* representations

of the *feature* to exist, which may vary in the level of detail, coordinate reference system or data format.

The GeoSPARQL standard [37] has been developed to provide an RDF data model that is compatible with the SPARQL protocol. This is shown in Figure 4.8 by the two related classes of *Feature* and *Geometry*. The *Geometry* class has further meta-data properties for the serialisation of its geometry shape. These features and geometries can be compared for their spatial relationships, e.g. points within a polygon, which are much broader than tests of distance or equivalence.

These spatial relations have usage in tasks relating to preparing the knowledge-base as discussed previously (Section 3.3.2). However, there are other forms of functionality that are outside the standard, such as determining which side of a line a point is positioned or the distance a point is positioned along a line, that are needed for accurate micro-simulation and need to be provided by modules.

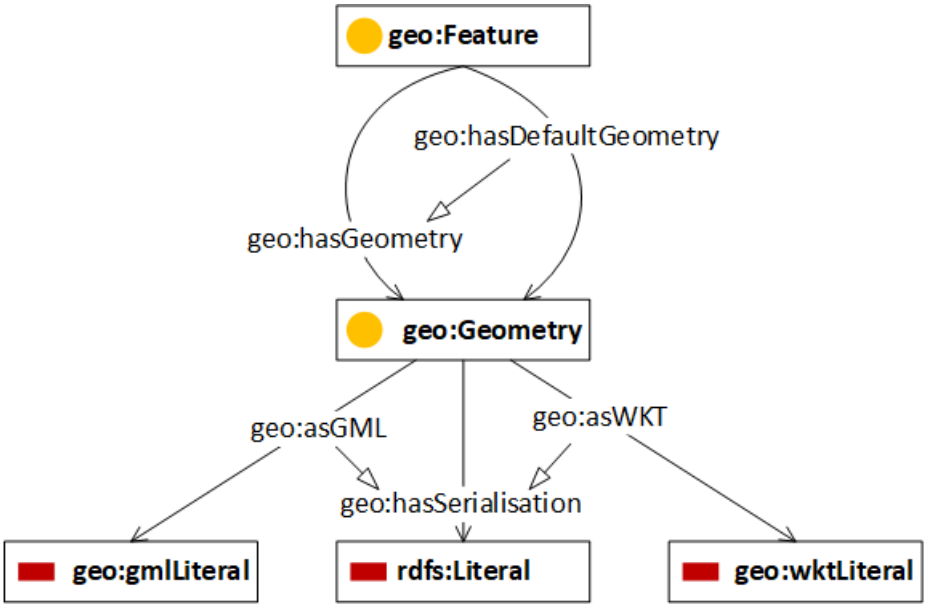


Figure 4.8: Schema for Feature and Geometry for published GeoSPARQL vocabulary. [37]

Implementations of the standard extend the SPARQL functionality to produce more sophisticated query results and reduce complexity. This functionality can include characteristics about a geometry and its spatial relation to other geometries. There exist numerous formats, or serialisations, for geometries which

provide alternative representations of the same information. Similarly, coordinate reference systems can describe geometries for use in different contexts. Therefore, implementations can provide handling of these additional requirements, but following the same data model.

The GeoSPARQL standard is influenced by the Simple Features standard [106]. This details spatial functions for use in SQL relational databases with geometry shapes represented in Well Known Text (WKT) and Well Known Binary (WKB) formats. A key feature of the Simple Features standard, and by extension GeoSPARQL, is that all geometries are treated as being in a two-dimensional plane. This provides a simplification for interpretation of spatial relations and calculation. The calculation of distance between geometries can be achieved using *Euclidean* distance derived from *Pythagorean theorem*.

However, this assumption is not always the case with many popular coordinate reference systems used for global positioning, e.g. WGS84, being geodetic. In a geodetic system the points are positioned on the surface of a sphere, which closely approximates the earth's surface. Applying Euclidean distance to these points does not measure the surface distance across the sphere but the chord. Instead the computationally more expensive *great-circle* , or *orthodomic*, distance is required for an accurate distance.

The error introduced by this assumption is accepted in the Simple Features and GeoSPARQL standards as it is offset by the computational simplification, being less significant at small scales and only applying for certain datasets. Coordinate reference systems for two-dimensional plane, i.e. Cartesian coordinates, have been developed which are highly accurate at the national level. These coordinate reference system place points relative to an origin in the plane and are only suitable for relatively small geographic areas. However, these geographic areas can cover whole countries, e.g. the United Kingdom has the single Cartesian coordinate reference system OSGB36.

The geographic area of interest in travel demand generation is typically at city or smaller scale. Therefore, the national coordinate reference systems are suitable. Datasets that are provided in a geodectic coordinate reference system can be converted to an appropriate Cartesian system using published mathematical transformations and tools.

In Figure 4.9 are shown the geometry classes of the Simple Feature standard in RDF with the primary interest being *Point*, *LineString* and *Polygon* along with their collections *MultiPoint*, *MultiLineString* and *MultiPolygon*. However, other representations of geometry classes could be utilised if the user chose.

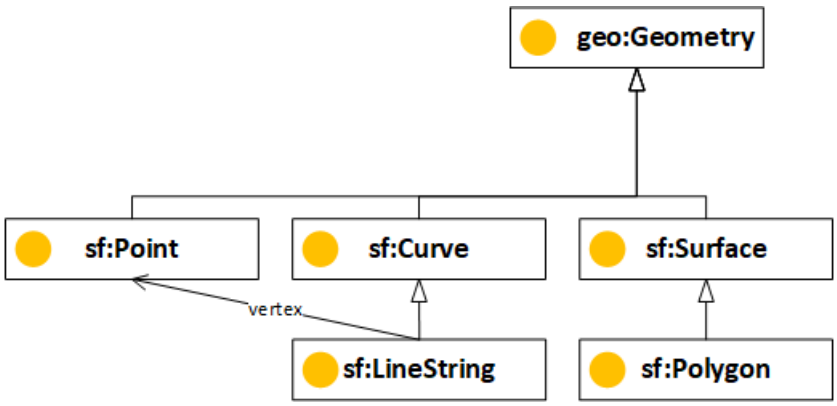


Figure 4.9: Schema for Simple Features geometries aligned to GeoSPARQL. [37]

An alternative vocabulary to GeoSPARQL in RDF is the Basic Geo Vocabulary [107]. This describes the WGS84 *latitude* and *longitude* coordinates as two properties of a *subject* and is a much simpler and directly accessible representation. However, its usage forces datasets to use a single coordinate reference system, i.e. WGS84, and would require conversion of datasets prior to adding to the knowledge-base. This is a popular global coordinate reference system for datasets, but not universal with national agencies often preferring Cartesian systems, e.g. road networks. The vocabulary can also only express one shape, i.e. points, does not describe spatial relations and there is no standardisation in the calculation of distance units of measure or method.

In conclusion, adopting the GeoSPARQL data structures provides a consistent representation of geospatial data with potential for data being readily available in this RDF format. It also allows modules and users to leverage implementations that can handle variations in format and context as well as providing more sophisticated querying. Investigation was undertaken into implementing the full GeoSPARQL standard as an extension to the Apache Jena library as the existing support for spatial query did not conform with the GeoSPARQL standard, or meet the needs of the project, and no existing alternative implementation was

identified that provided full support.

The developed implementation included features such as Semantic Web standards compliance, minimal configuration and a short initialisation period along with automatic switching between coordinate reference systems and units of measure. There was also an additionally developed novel feature of caching invariant geospatial literals and other data that is repeatedly re-used in spatial queries to produce a performance improvement of up to 20%. An extendible benchmarking framework was also implemented to enable comparison with two existing partial implementations using a published geospatial benchmark.

This benchmarking demonstrated that the developed implementation achieved comparable or faster query responses, while also providing much faster data loading and initialisation durations. This benchmarking framework is planned to be extended in future work into a conformance framework to provide an automated tool for demonstrating compliance of the developed implementation, and other GeoSPARQL implementations, with the GeoSPARQL standard. Further discussion of this investigation can be found in Appendix B.

### 4.4.2 Temporal

The representation and tracking of time is an important aspect of travel demand modelling. Human travel and activity occur over time durations and at different time points which causes variation in volume, behaviour and location. The time of day, day and season have already been discussed as having a varying influence on activity and travel (Section 1.2.3). Travel time itself is also regarded as a key metric in transport planning [108, 109] making it an important part of the analysis process. The temporal domain is therefore important in the modelling, execution and analysis of transport and travel models.

The public *Time Ontology in OWL* (OWL Time) vocabulary [38] defines temporal concepts, properties and their topological relations. The focus is upon describing resources in the world and Web pages. The representation of time concepts is achieved through literals or RDF properties. The string literal approach uses XSD *dateTimeStamp*[110] which consists of a calendar date, *wall clock* time and time-zone offset, e.g. "2000-01-01T09:00:00+0:00"^^xsd:dateTimeStamp. In

RDF, a subject will have properties representing each part of *year*, *month*, *day*, *hour*, *minute* and *second*. The recording of time is through *instants* representing a specific point in time and *intervals* representing a duration of time, which are formed from two *instants*: start and end. This robust representation of time is shown in Figure 4.10.

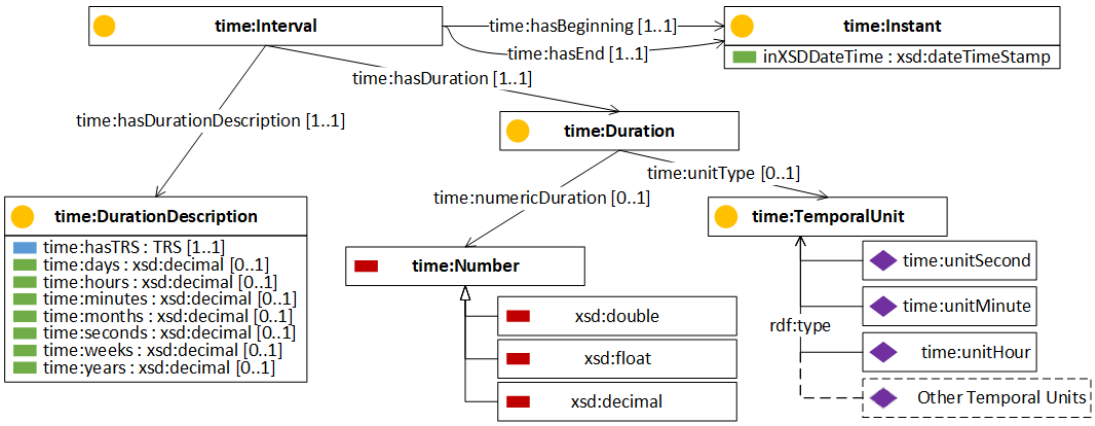


Figure 4.10: Schema for Interval, Instants and Duration from published Time Ontology in OWL (OWL Time) vocabulary. [38]

In the context of travel demand modelling this introduces a level of complexity that can be considered unnecessary, given that current travel demand models focus upon a single day period. The models are typically executed without consideration of a specific reference date. Instead time is considered from the abstract basis of a time during the day, i.e. *9am* rather than *9am on 21st March 2001*. While a base date (e.g. 01/01/1900) could be provided and the time component extracted this presents extra complexity during execution. It also introduces data which does not have relevance or correctness and could be misunderstood or misused.

The use of RDF properties, while having use for logical reasoning, presents additional complexity when storing, extracting and sharing data. A complete time value requires three triples, i.e. properties for *hours*, *minutes*, and *seconds*, compared to a single triple for the literal approach. The representation of an interval of time requires eight triples compared to two triples for the literal approach. Logical reasoning between different time values can also be performed by parsing the string literals with an appropriate library so the functionality is still

available to modules that require it. Therefore, limited or no benefit is provided by the full OWL Time vocabulary but with an increase in complexity.

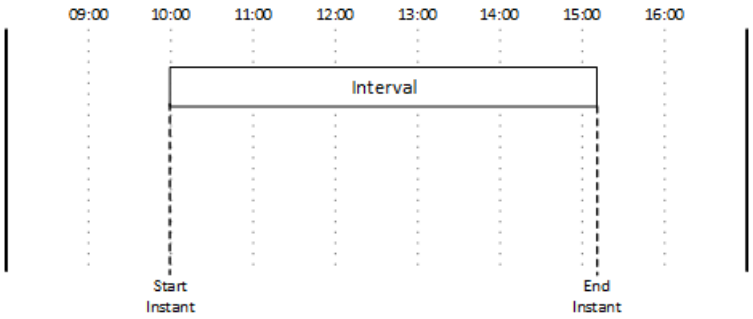


Figure 4.11: Diagram of Time Instants and Interval.

The adopted approach has been to use literals in the XSD forms of *time*, e.g. "09:00"^^xsd:time for *9am*, and *duration*, e.g. "PT10M"^^xsd:duration for *10 minutes*. These are used together with OWL Time *Day of Week* to consistently represent the seven days of the week. If a module is designed to consider specific dates, e.g. seasonal factors such as weather conditions, then the XSD *date* can be incorporated as an additional property without interference to existing data and affect upon modules which do not require it.

The time interval represents a period of time from a start to an end, as illustrated in Figure 4.11, and occurs frequently in travel demand modelling. Items may have a validity period when they are eligible for selection or utilisation. Alternatively, it could represent when an event or activity is taking place. Figure 4.12 shows the *Time Interval* class that is defined in the schema.

The class provides a simple representation of the points in time when the interval starts and ends along with the duration, i.e. the difference between the start and end times. This class can be sub-classed for those items which are themselves intervals, e.g. parts of a schedule, or referred to through the *hasTimeInterval* property when multiple or optional intervals apply, e.g. opening hours of a location. Differentiation of days is through the *has Day* property to the OWL Time class *Days of Week*, shown here with its set of instances.

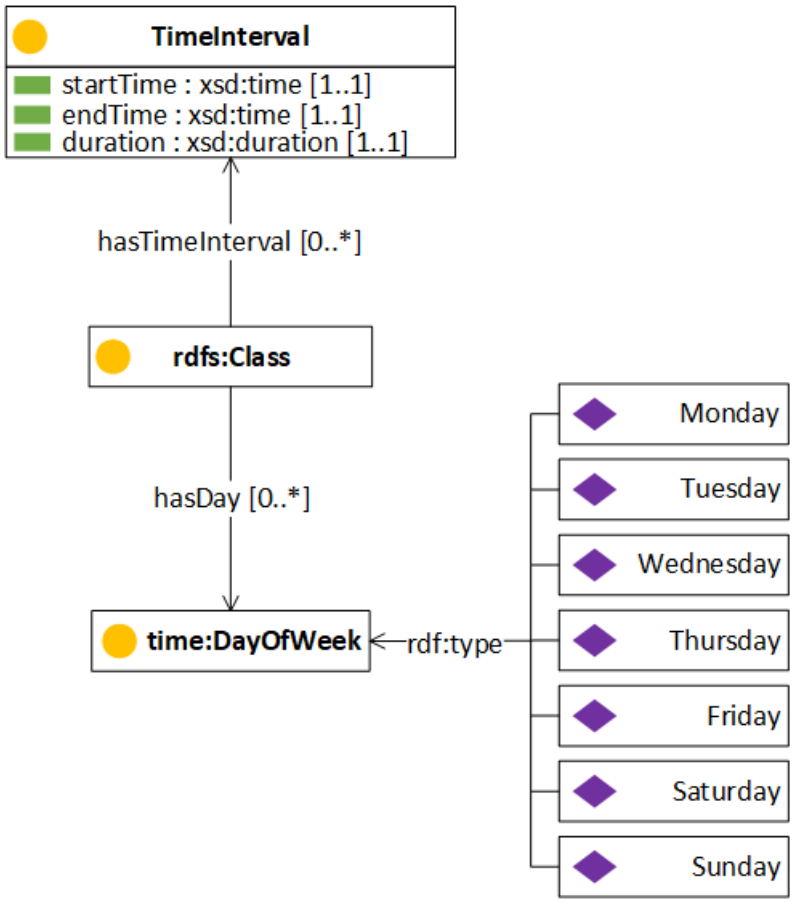


Figure 4.12: Schema for Time Interval and Days of Week.

## 4.5 Concepts from the Physical World

In this section there will be discussion of the specific data concepts that have been identified to allow the general components of the framework to consistently interact, as previously discussed (Section 3.2). Figure 4.13 shows the transmission of these data concepts between the general module components.

It is not intended that the described concepts are definitive and instead will be extended or supplemented in the knowledge-base according to the needs of the user and the selected modules, as supported by the AAA principle (Section 4.2.1). Throughout this section there are illustrative examples of how these concepts can be extended, but have not been included in the core schema. Many of these illustrative examples have been applied in the prototype scenario design discussed

in Chapter 7.

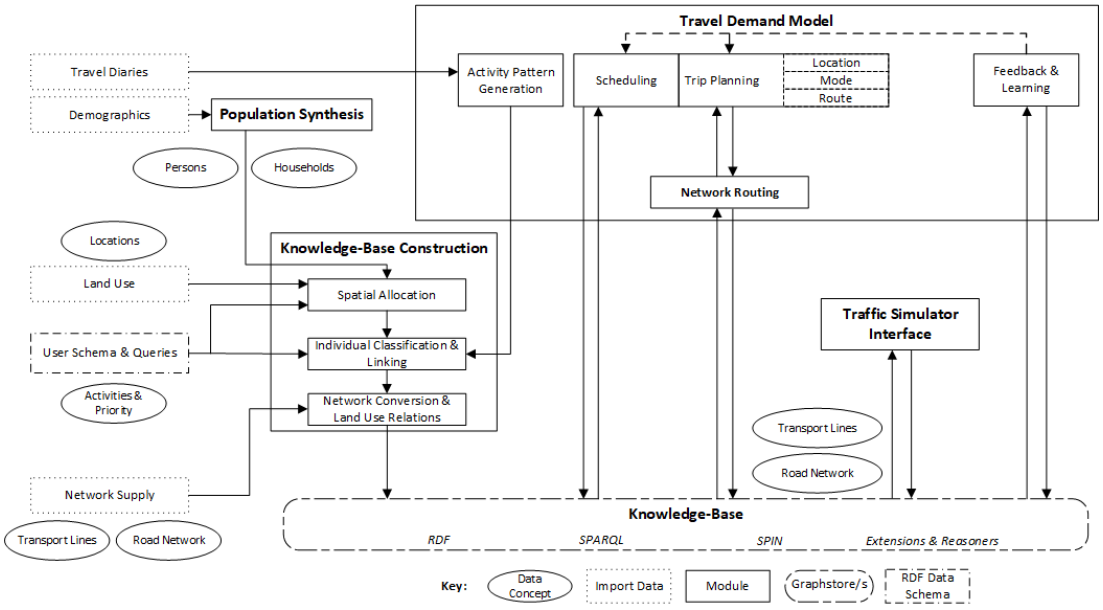


Figure 4.13: Diagram of main components and schema data concepts related to the physical world.

### 4.5.1 Person

The purpose of transportation is the movement of people and goods [11]. Currently all transportation of goods by freight and movement of people by vehicle or personal locomotion requires human oversight and control. Therefore, both the demand and supply of transportation are generated by people.

In an activity-based modelling perspective, people generate transportation demand as a consequence of undertaking activities at spatially separate locations. Activities undertaken at the same location do not require movement, or at least not a meaningful quantity to model, and therefore do not generate demand.

The transportation supply required to satisfy this demand, either through personal means or accessing a service, produces the physical manifestation of transport, i.e. traffic. Maximising the efficiency of transport supply is the primary focus of policy decisions and interventions, but are downstream actions of activity-based modelling. Therefore, the primary focus is upon the individual person as

the originator of the transportation demand.

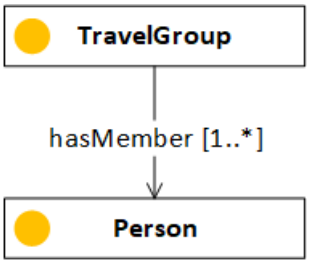


Figure 4.14: Schema of Person and Travel Group.

Figure 4.14 shows the class of *Person* and its formation into groups through the *Travel Group*. This has been placed in the *People* domain (Figure 4.7). Each person has a set of characteristics which describe them and would typically be modelled as *data properties*. However, there are no required characteristics in the schema and instead these are incorporated by the user extending the schema according to the data available and selected modules.

In a transport modelling context, a *Person* can be defined by a wide variety of characteristics, including age, sex, income, possession of driving licence and vehicle ownership [111–113], and are typically drawn from census information [114]. The census data must be converted for use in activity-based models and traffic simulators through a Population Synthesis process to convert aggregate data into a disaggregate set of persons.

The characteristics themselves can also be aggregate or disaggregate values. Figure 4.15 shows alternative representations of the same characteristics with use of aggregate groups using *object properties* (left) and disaggregate values using a mix of *object* and *data properties* (right). The disaggregate values could be represented using only *data properties*, but in some cases this would present a poor modelling approach that is prone to error, e.g. using *string* data values rather than individuals.

The selected modelling of the characteristics will be dependent on available data, techniques and models. This highlights a practical difficulty in defining a definitive schema, which itemises all object and data properties, for the travel demand and transport simulation domain. In one model an assumption that all *Persons* have only a single *employment* would be valid, but this is not the case

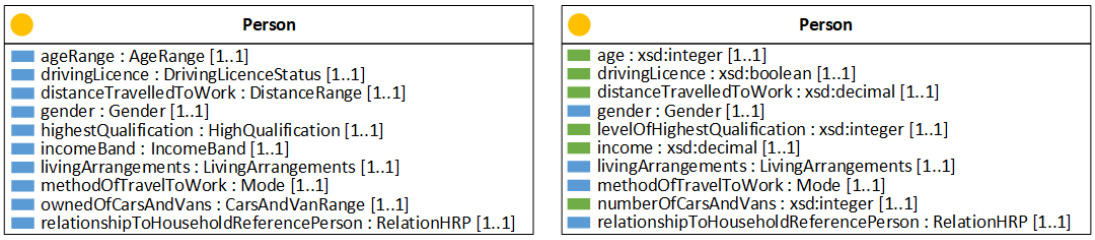


Figure 4.15: Diagram of example extension to the Person class showing alternative representations of the same characteristics.

in real-world data and so not all potential models.

Certain *person* data characteristics, e.g. vehicle ownership, home size and employment type, must be transformed and aligned with other contextual information to provide relations through *object properties* to other individuals and concepts, e.g. ownership of a car requires instantiating a car individual in the knowledge-base, typically during the Knowledge-Base Construction stage (Section 3.2). These individuals and concepts will in turn have their own characteristics and relations.

The significance and relevance of characteristics and relations will vary based upon the context and design of implemented models. For example, in the context of network routing the age of a person as a passenger in a car is not relevant. When considering driving a car then age is relevant as children should not be driving. In another context, other characteristics, e.g. holding a vehicle driving licence, can imply that a person has an appropriate age and therefore the age characteristic may not be required. When modelling using public transport the age can be relevant as pricing policies can be age based. The $CO_2$ emissions of a vehicle may not be relevant when examining the pricing structure of a toll bridge.

As outlined previously, seeking to stipulate the precise characteristics of persons in the schema as data and object properties is not practical. Therefore, explicit characteristics are not asserted in the schema and instead can be determined by the user and the particular implemented modules they are using. However, applying the proposed approach to construct a Semantic Web knowledge-base allows this wide range of data to be modelled and available for general use through SPARQL queries. The required data can be selectively retrieved according to the context and removing the need to define fixed interfaces that pass data between

stages. The concept of the *Person* can be extended with further characteristics, e.g. familial relationships, by additional domain vocabularies [115, 116] or the user's own schema.

### 4.5.2 Travel Group

Humans are social animals who live in organised groups to cooperate, share resources and coordinate their activities. This is captured broadly in Figure 4.7 by the *Organisations* domain. The typical organisational unit of persons in population and transport modelling are households [84, 114]. Scheduling of activities are coordinated between persons in the household to ensure that they can be undertaken together or that each person can undertake their activity when resources are limited [9], e.g. a parent escorting a child to an activity or families that own a single car. The notion of a household is not limited to a family, but applied more broadly to the sharing of facilities inside a home [117].

In a wider context, people organise into groups for other activities. This can include commercial organisations, but within these organisations there may be multiple sub-groups based on geographic or organisational constraints. Members of car-sharing or bike hire schemes when people can hire vehicles for short term periods are also *organisations*. Each of these groups will have different behavioural dynamics to influence scheduling; ranging from the tightly coupled relationship between parent and young child to a loose association of common interest. Therefore, there is a need to group persons together, but there is a diversity in how the groups will behave.

This is modelled in the schema by the *Travel Group* which forms a base class, from which the user can create sub-classes as required, and has members of individual *Persons*. The concept of the *Travel Group* can be enriched with further information, e.g. roles, organisational units or associated locations, by additional domain vocabularies [118, 119]. However, these concepts are not essential to travel demand modelling and so have not been included.

### 4.5.3 Mode

A person can travel between places using a variety of modes of transport and a single trip could incorporate switching between modes for different stages. These modes can include privately owned vehicles or the use of public transit, e.g. buses and trains. A set of modes not yet found frequently in travel demand modelling and traffic simulation literature are the service modes of taxi and pick-up services, which are gaining increasing prevalence with smart-phone apps able to conveniently request and pay for usage, along with car-sharing and lift-sharing schemes [57]. This concept of alternative travel methods positions *Mode* within the *Network Infrastructure* domain in Figure 4.7. The usage of a mode has implications for its availability, accessibility and utility. For example:

- Public transit only being available at a certain frequency and periods of the day.
- Public transit is accessible only at the location of connections to the service, i.e. bus stops and train stations.
- Cars accessibility is determined by its location, e.g. at home when at work, destination parking or usage by other members of the group.
- The utility of a mode can be influenced by the speed of travel, route requirements, financial cost and the weather, e.g. raining while cycling.

The mode of transport incorporates a range of factors and concepts which can be defined as data properties. In the traffic simulation stage the mode of vehicles is used to restrict access along edges and apply turning restrictions at junctions. It is also used to impose the legal speed limit that the general class of vehicles **should not** exceed, which is distinct from the speed an individual vehicle **can** achieve and is a feature of some simulators [46]. This legal speed limit informs the estimation of the best-case travel times for routes discussed later (Section 4.6.4).

It has been discussed previously (Section 4.2.4) that the OWL2 language requires that the subject of a triple must be an individual and object of a triple

must be an individual or a literal, except for the key *rdf:type* property. Therefore, a class of *Mode* is defined for which the user's choice of modes of transport is instantiated as individuals. These individuals can then have property relations with individuals from other classes to ensure consistency and re-use. The alternative design would be to define mode as the classes into which vehicles and people are placed, which would prevent the additional attribute properties being utilised and confusingly suggest *a person is a mode of transport* rather than *a person has a mode of transport*.

The additional properties provide an additional consideration during the modelling process. A user may wish to explore different characteristic values to investigate their impact and influence across different investigations. These *Mode* values would either need replacing for each investigation, have multiple variant properties with the current version selected or have the whole knowledge-base replicated.

The adopted approach is to define a *Mode Definition* concept to form an N-ary relation (Section 4.2.2) for the mode's characteristic values as illustrated in Figure 4.16. This definition can be associated with one or more scenarios (Section 4.6.1) and so re-used as required. Therefore, only the *Travel Scenario* needs to be selected to obtain the relevant parameters for the whole scenario. Multiple scenarios can be present in the knowledge-base simultaneously allowing them to be retained as a record of their defined values and also re-used.

The UK Census 2011 [13] identifies more detailed modes of transport for travelling to work by commuters as well as highlighting that 5% of the working population are based at home. These are detailed below to illustrate the breath of modes that a travel model and traffic simulator may seek to model:

- Driving a car or van
- Passenger in a car or van
- On foot (walk)
- Bus, minibus or coach
- Train (rail)

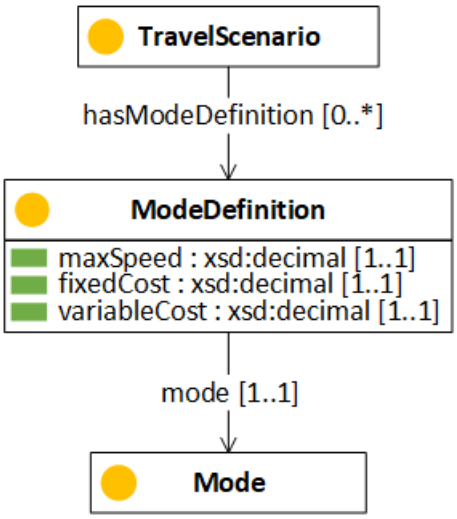


Figure 4.16: Schema for Mode and Mode Definition.

- Underground, metro, light rail or tram (rail)
- Bicycle
- Work mainly at or from home
- Taxi
- Motorcycle, scooter or moped
- Other method of travel to work

In Figure 4.17 an example *Mode* class hierarchy is shown with individual instances to show a possible organisation of the concepts discussed previously. A special individual of *AnyMode* is defined for convenience to allow access for all the defined modes when determining access through network infrastructure, e.g. generating routes or during traffic simulation.

### Personal Mode

A further area of consideration is the personal locomotion of an individual. It has previously been discussed (Section 1.2) that travel demand and traffic simulation has focused upon vehicle modes with neglect of pedestrian modes e.g. MATSim

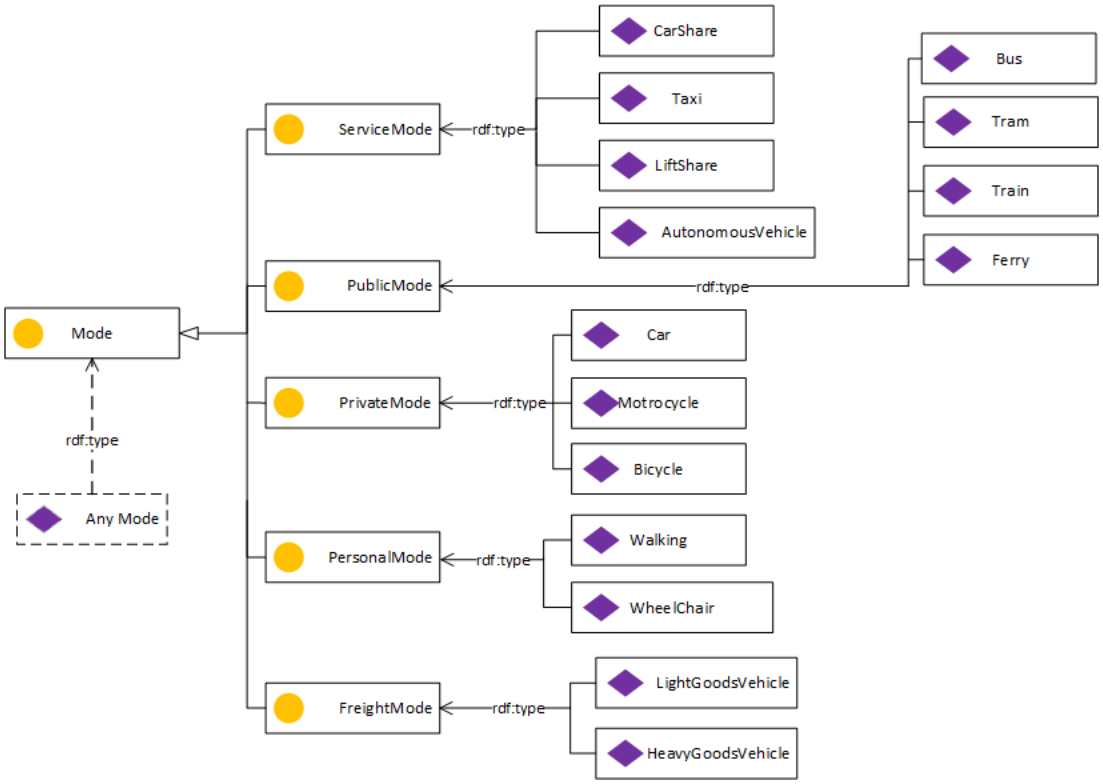


Figure 4.17: Diagram of example Mode class hierarchy with individuals.

simulator [32] does not model bidirectional routing of pedestrians and instead relies upon teleportation between places. This is also illustrated in the SUMO simulator [46] where pedestrians are treated as a type of *Vehicle*, an inconsistency with the definition of vehicles in the following section (Section 4.5.4). In both simulators the *Mode* determines the access through the transport network.

In the same way as vehicles from different manufacturers have varying characteristics so does the mobility of individual people. This mobility includes the speed at which the person travels but also the routes which are accessible. A modelling perspective will require some aggregation yet there are identifiable sub-groups that may require specific modelling. Elderly and disabled people may not be able to access flights of stairs or may use mobility aides, e.g. wheelchairs or mobility scooters, which cannot handle inclines. Therefore, modelling the pedestrian phase of a Person can lead to distinctions between personal modes. In addition, not all modelled Persons may be required to undertake pedestrian

stages, e.g. a freight delivery driver will not walk between delivery activities, but would instead be expected to move the vehicle.

It should be noted that explicit modelling for the use of mobility aides and their restricted access is not present in the SUMO [46] and MATSim [32] traffic simulators. Instead only a generic pedestrian class is defined. This shortcoming has been notified to the SUMO developers and has been adopted as a future enhancement (see Appendix A).

### 4.5.4 Vehicle

Persons can reduce travel times and improve efficiency of moving goods by utilising vehicles. The definition of a vehicle is a machine which transports people or cargo. Therefore, it includes cars, trucks, buses, trains and ferries and also bicycles [120]. Therefore, a vehicle is a broad definition that encompasses road and non-road usages that can be motorised and non-motorised. This is presented in Figure 4.7 by the domain of *Vehicles*.

Accessibility of vehicles can also vary with privately owned cars, car-sharing, car-hire or utilisation of taxi services, while public transit vehicles are only available at fixed geographic locations. Therefore, numerous sub-classes can be used to distinguish between different types of vehicles. This continues when considering the manufacturer, design and performance of a vehicle leading to further sub-classifications and distinctions.

The availability, accessibility and performance of a vehicle are influential in the modelling and simulation of the transport environment. An individual vehicle could therefore belong to multiple classes and be described by a range of characteristics, including physical dimensions, seating capacity, speed and $CO_2$ emissions. A further key characteristic is the mode to which the vehicle belongs and there is a strong relationship between *mode* and *vehicle*. The physical construction of a *Vehicle* is separate to any autonomous software or emerging technology that may control it (Section 4.7).

Following the approach described previously (Section 4.5.3) a *Vehicle Definition* is associated with the scenario and relevant individual *Vehicles* through a *Vehicle Type* as shown in Figure 4.18. This has the benefit of removing repeti-

tion of common characteristics and allowing multiple scenarios with their varying values to exist alongside each other in the knowledge-base.

These characteristics can include simulation specific parameters, e.g. vehicle max speed, acceleration and deceleration, which may be varied for different road conditions. This use of *Vehicle Type* is in keeping with the modelling approaches of SUMO and MATSim simulators where common values are re-used.

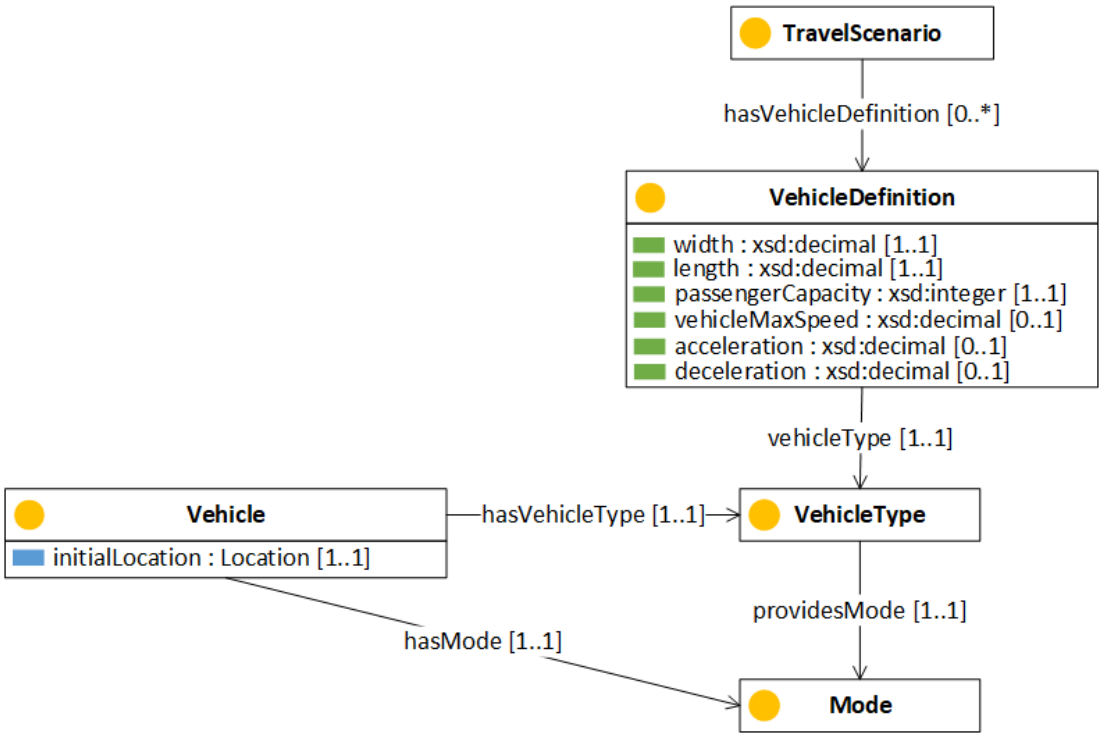


Figure 4.18: Schema for Vehicle and Vehicle Definition.

There is the potential that an input dataset has individual vehicles, numbering in the thousands or more, with each having their own characteristics. This presents an issue as there is a mismatch between the individual approach of the dataset and the schema's *Vehicle Definition* just described. Transformation, back and forth, between the characteristics being defined on individual vehicles and gathered together in the Vehicle Definition can be achieved using SPARQL queries.

An example Vehicle class hierarchy is shown in Figure 4.19 which distinguishes between cars and motorcycles. The *Car Vehicle* class has been further sub-classed

to distinguish between large and small vehicles, which each have individuals. These individuals would have their own *Vehicle Type* property, which in turn has a *Vehicle Definition* for the scenario.

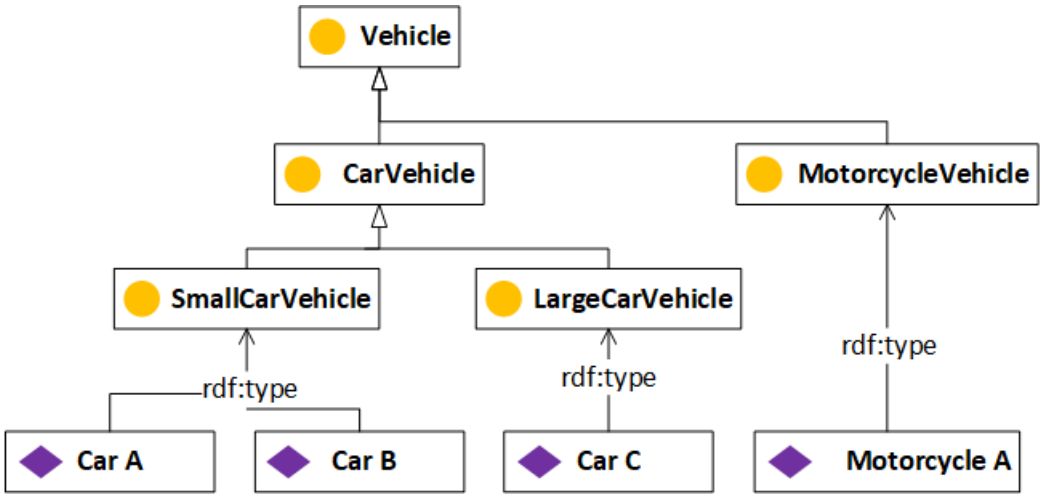


Figure 4.19: Diagram of example Vehicle class hierarchy with individuals.

### Vehicle Route

The output of the travel demand stage is a schedule of activities and the travel stages required to travel between them. The traffic simulation stage requires routing information for each of the travel stages to identify where a person, and potentially their vehicle, will travel. In the case of the SUMO simulator [46] a listing needs to be provided of each vehicle's complete route, i.e. excluding any pedestrian stages. This information is also required for vehicle's operating upon public transit lines and may also be useful for analysis of vehicle travel in a scenario, e.g. planned versus actual route.

Therefore, a *Vehicle Route* concept has been incorporated as shown in Figure 4.20 so that this information can be included in the knowledge-base during schedule construction or produced once scheduling is complete. The *Vehicle Route* is attached to the *Activity & Travel Schedule* for which it has been generated (Section 4.6.3. It identifies the vehicle with its start and end location and access points. The detail about the route is provided as a delimited string of *Road Link* and *Road Node* URIs. This provides a compact form of expressing an ordered list

that can be split when required and used for searching additional details. This removes the need to create and search the additional triples of the *Ordered List* for information that may only be required by certain modules and simplifies the process of passing the list of information between modules.

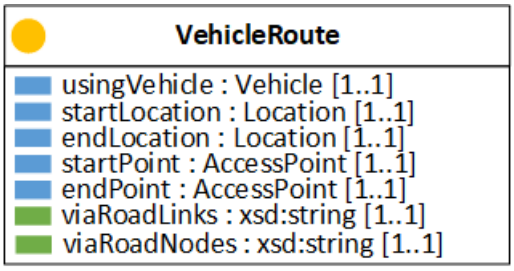


Figure 4.20: Schema for Vehicle Route.

### 4.5.5 Transit Line

The *Vehicle* concept encompasses a wide range of purposes and uses. A distinct purpose is the provision of public transit, also know as public transport, to transport passengers according to a published route and timetable. These transit lines are fulfilled by individual vehicles travelling along the route utilising the road network or dedicated infrastructure, e.g. bus lanes or railway lines. The route consists of a series of planned transfer locations, which may be on road, dedicated waiting areas or within stations, for passengers to board or alight. Adherence to the timings of the timetable, to compensate for traffic and other delays, will increase and decrease the period of waiting at each transfer point to ensure that the vehicle departs each transfer at the correct time.

Each transit line provides a specific mode of public transit and one or more transit lines may exist for a mode. This positions *Transit Line* within the *Network Infrastructure* domain in Figure 4.7 as it does not relate to the actual vehicles but the infrastructure provided. Availability of transit lines can vary by time periods and days with some services not being available in evenings or weekends. Variation in frequency according to time and day can be captured by multiple transit lines. The transit lines are provided by transit operators, which provide one or more transit lines potentially across multiple modes of transport, and

can be private or public sector organisations. Therefore, there is close linkages between *Transit Line* and the *Vehicles* and *Organisations* domains.

Figure 4.21 shows the schema developed for *Transit Lines* grouped by *Transit Operators*. Each *Transit Line* is described in more detail by a *Transit Line Timetable* and *Transit Line Route* with each being ordered lists to preserve their organisation. The *Transit Line Timetable* captures the timetabling information published for the public to plan their journeys by describing the transfer points by their spatial *Location* and departure time. The *Transit Line Route* contains the information required during travel demand generation and traffic simulation by using the stage between transfers to estimate route and travel time. This information is derived by applying the general timetable to specific road network infrastructure. The simulation of these transit lines with vehicles can be abstract or instantiated. Therefore, the *transit vehicle* property to create a relation to specific vehicles is optional.

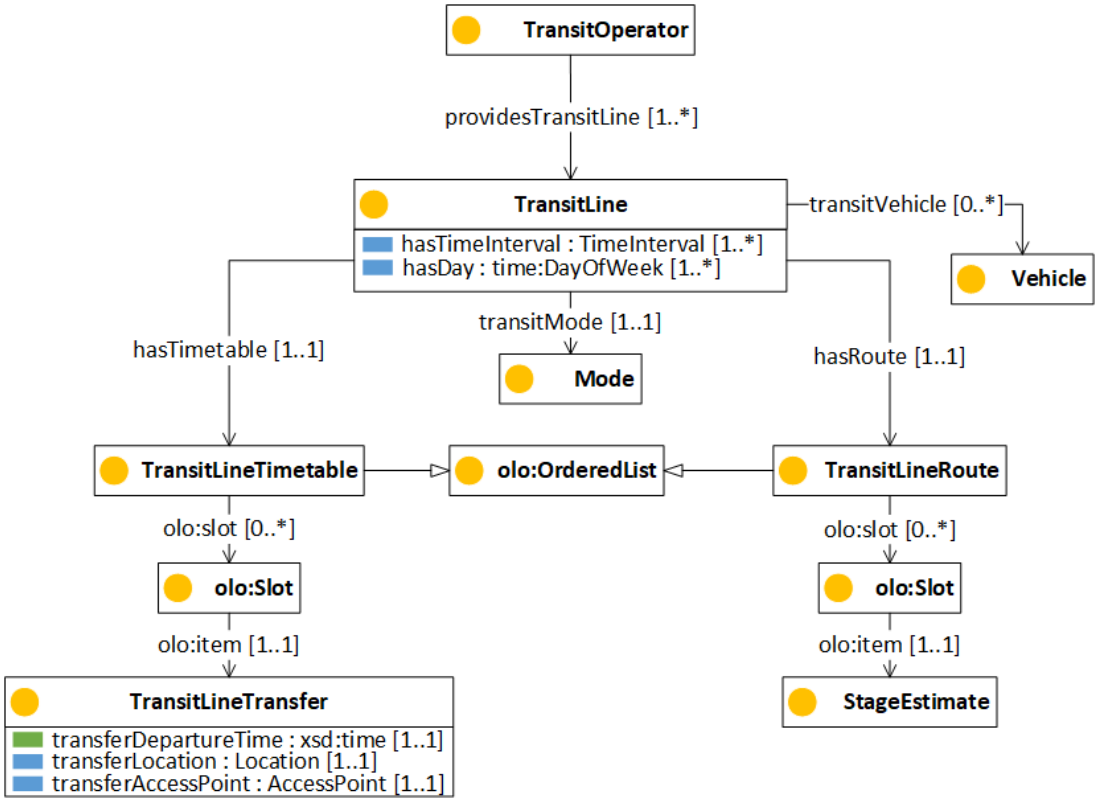


Figure 4.21: Schema for Transit Line.

In an abstract approach the point of access offers an entry and exit point into the environment without consideration of the vehicle, i.e. the vehicle always arrives as timetabled at the airport, light rail, train or ferry terminal. Persons arriving at the point of access no longer travel in the simulation and have concluded their travel while those departing from the point of access can only do so at the timings permitted by the timetable.

Alternatively, the instantiated approach has individual vehicles fulfilling the transit line by moving through the simulation and interacting with other vehicles and persons, e.g. a bus driving on the road or a train travelling through railway crossings. This can be applied to all the modes described previously and is a feature of the traffic simulator. Therefore, the detail associated with a transit line varies between the travel demand stage (concerned with the transit line timetable) and the traffic simulation stage (concerned with vehicle's physical interactions) which can be reliant upon mode, network infrastructure and simulation design.

### 4.5.6 Activity

Every activity undertaken by a person is a unique occurrence that is described by temporal and geospatial characteristics. These characteristics give the activity a time, place and duration. Here the term for an *Activity* has been applied to the more general notion of the potential to perform an activity at a specific location and during specific time periods. All activities cannot be performed at all locations and at all times and therefore differentiation is required. Each *Activity* belongs to an aggregating *Activity Type*.

The unique planned event of a *Person* is expressed in the *Activity Interval* of the *Activity & Travel Schedule* discussed later in Section 4.6.3. The *Activity*, *Activity Type* and *Activity Priority* are positioned within the *Activities* domain in Figure 4.7.

In Figure 4.22 each *Activity* is defined by one or more *effective time periods* and *effective days*. These could define narrow or wide ranging time periods, such as opening hours when a shopper can shop in a store; a student can attend a library to study; or the daylight hours when a jogger can exercise in a park. The effective times can be split into morning and afternoon time periods, or any other

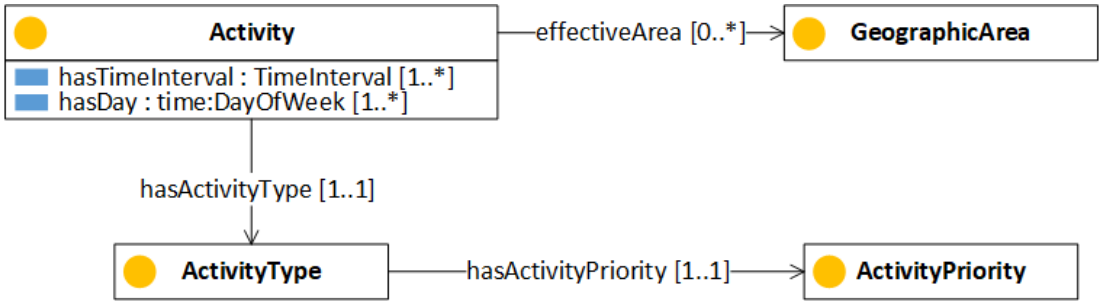


Figure 4.22: Schema of Activity, Activity Type and Activity Priority.

arbitrary division, to reflect the availability as required e.g. lunchtime closures. In the case of variation between days in effective times, e.g. weekend and weekday opening hours, then different activities would need to be created.

This allows scenarios to variate temporally and select alternative *Activities* for different outcomes. Exploration of seasonal factors, e.g. months or holiday periods, would require extension with additional properties. The spatial location of an *Activity* is defined by its relationship to *Locations* and is discussed later (Section 4.5.7).

The effectiveness of an *Activity* can also vary spatially. Attendance of children at a school may be limited by its catchment area. Large retail shopping centres have an influence on a greater area than small shops. This can be modelled in the knowledge-base by an optional relation with an abstract *Geographic Area*, discussed later in Section 4.5.8.

It has previously been discussed (Section 1.2.3) that human activity covers a wide range of undertakings that require broad classification. The specification of the entire classification is impractical for the core schema. Instead it is anticipated that users will be able to define their own categories and related characteristics through parameters and relations in the knowledge-base. However, the activities still have general groupings that need to be recognised.

The application of a class hierarchy when those classes may themselves have additional properties or used *objects* in *triple* statements can be incompatible with certain schema languages (Section 4.2.4). Therefore, the composition approach is applied to the *Activity* concept, in keeping with other concepts, where each instance has a property to an *Activity Type* that allows identification of related

instances.

This *Activity Type* is used as a generic reference to identify *Activities* as part of *Activity Patterns* discussed later (Section 4.6.2). Otherwise *Activity Patterns* would have to identify all the relevant *Activity* instances, which would be burdensome to create and manage. An extending schema could apply a class hierarchy to assist in organisation but it would not be generally suitable. Figure 4.23 provides illustrative examples of *Activity Types* that a user or model may use to extend the core schema.

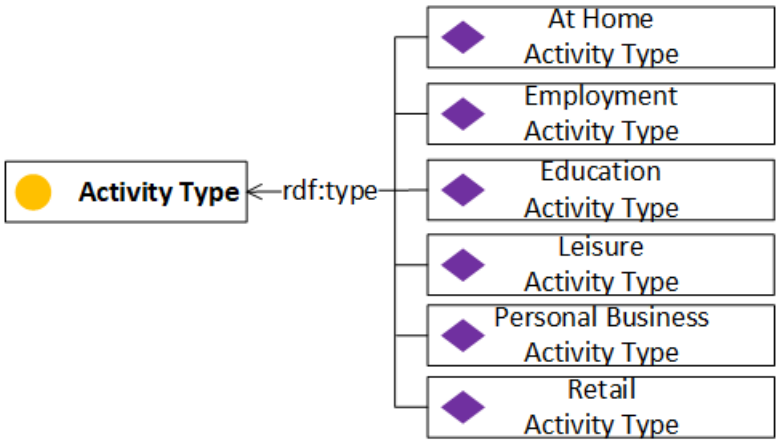


Figure 4.23: Diagram of example Activity Types.

The process of constructing a schedule reflects a series of trade off decisions. The intended activity start time, end time or both could be impacted by previous and following activities, longer travel times between activities or delays during travel. When scheduling within a household, or other travel group, the co-ordination of activities has to reconcile satisfying the activities and travel of other members. This has been modelled in travel demand models through assigning priorities to activities [9].

Those activities with high priorities may be considered inviolate while lower priority activities can be abandoned or curtailed. Activities with high priority may be placed in the schedule first with lower priority activities fitted into available gaps. The approach adopted depends upon the scheduling strategy with parameters being specified for minimum duration or tolerance for changes. This

general concept of activity priority is reflected in the Figure 4.22 schema by each *Activity Type* having an *Activity Priority*.

This places the modelling assumption that all activities of the same type have the same priority, e.g. all education is mandatory. Variation of this assumption could be achieved on an individual activity or person basis within a knowledge-base and would reflect greater individualisation in choices, but it represents a level of modelling complexity beyond current examples. The specific parameters for a particular investigative scenario or model are captured as part of the *Travel Scenario* discussed in Section 4.6.1. Therefore, data about priority parameters would also be modelled there to allow comparison between different values. Figure 4.24 shows example instances of the *Activity Priority* found in existing models [9].

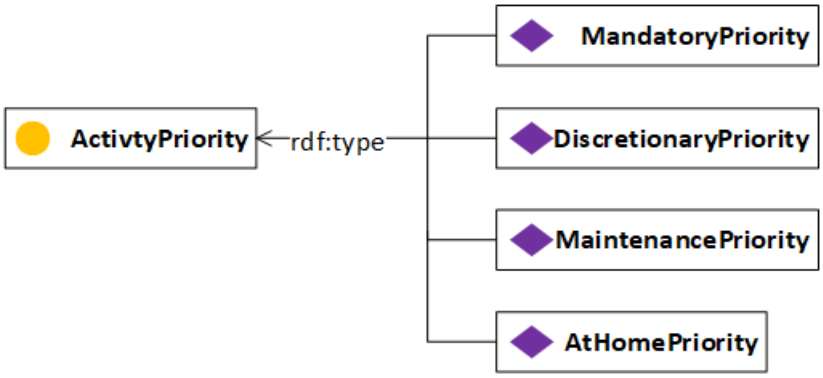


Figure 4.24: Diagram of example Activity Priorities.

The three associated classes of *Activity*, *Activity Type* and *Activity Priority* are designed to allow multiple instances to be present at the same physical *Location*. Figure 4.25 shows an example of the data within a knowledge-base for a shopping location that provides both retail and employment activities. These activities have different types and effective time periods. The scheduling of two *Persons* with interest in the different *Activity Types* would draw upon the different sets of data relating to the location to produce differing scheduling outcomes.

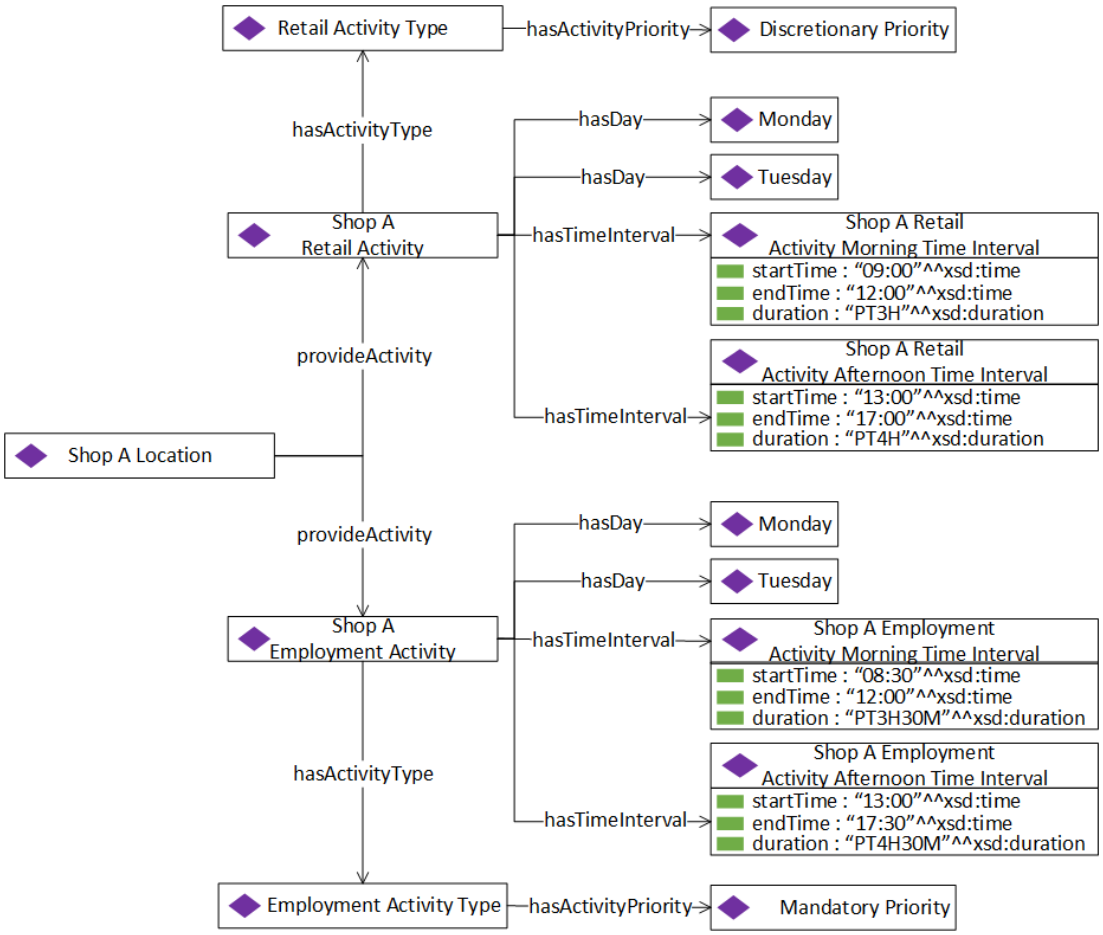


Figure 4.25: Diagram of example data model for a shop providing retail and employment activities.

### 4.5.7 Location

The activities undertaken by people take place at different physical locations. The necessity to travel between these locations is the source of travel demand. These locations can be where services are provided to undertake the types of activity, e.g. education or retail, or where an type of activity is executed by the person, e.g. employment or exercise. Therefore, locations represent any point of interest in the physical environment. This can include buildings, transport infrastructure, e.g. bus stops and motorway links, and outdoor spaces. This is represented by the *Location* concept being positioned within the *Spatial* domain in Figure 4.7.

These *Locations* can provide multiple activities and type of activities, e.g. a school provides education and employment. Figure 4.26 shows the schema for *Location* classes and their relationship to *Activity* and *Activity Type*. The relationship to *Activity Type* provides a direct relationship for convenience in retrieving data based on what can be inferred by following the relation to the *Activity* taking place at the *Location*. Therefore, the *Spatial* and *Activities* domains are connected in Figure 4.7.

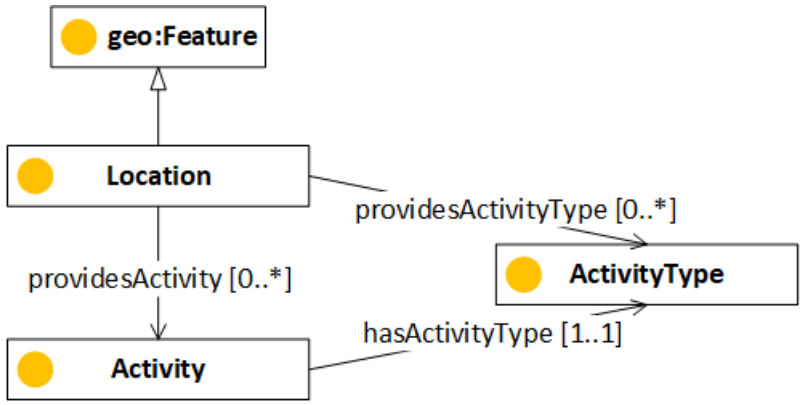


Figure 4.26: Schema for Location.

Further usage of the *Location* concept can be seen through the term *reference location* used in travel demand models and which has been defined previously (Section 1.2.2) as the location from which a person starts and ends their journey [10]. The *Location* concept is also necessary for indicating where a *Vehicle* is located at the start, during and end of the travel demand process to ensure consistency such as returning a vehicle for future use, e.g. at the reference location or hire facility, or picking-up a vehicle after performing a sub-tour by a different mode, e.g. driving to work, walking to shopping and driving home. Figure 4.27 shows an example of how a user can sub-class the general *Location* class to organise the buildings and places in the knowledge-base.

The physical dimensions of a location can vary depending upon the application context, e.g. property house prices can be referenced using Global Positioning System (GPS) coordinates, but land ownership requires highly detailed survey of the boundary. Therefore, there can be multiple descriptions of a location, which are accurate and precise in the appropriate context. This means that there is

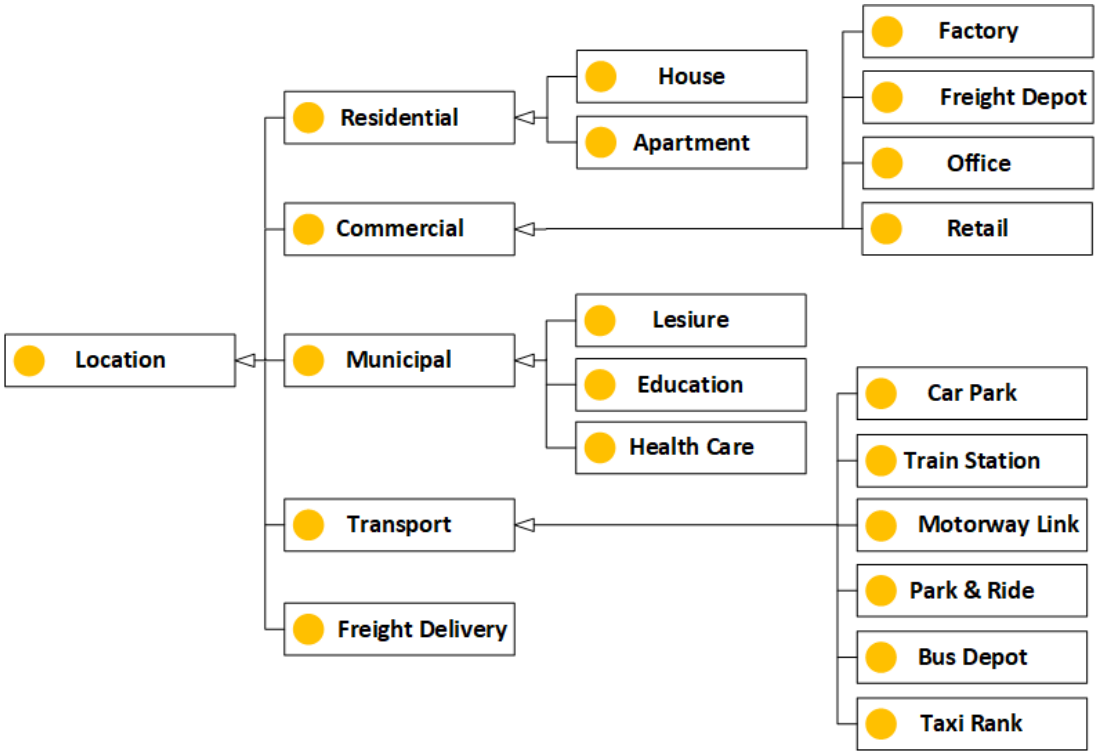


Figure 4.27: Diagram of example Location class hierarchy.

close, but distinct relationship between the *Spatial* and *Geospatial* domains in Figure 4.7.

To represent the varying application contexts, geospatial standards [37, 121] have developed the concept of a *Feature* which can be represented by multiple *Geometry*. These *Geometry* can vary in level of detail, coordinate reference system or serialisation etc. The modelling of geospatial concepts were discussed in further detail previously (Section 4.4.1).

Different stages of the modelling process may require different levels of detail about a *Location*, e.g. when constructing a schedule the general proximity of a location is required using straight line distance, but determining routing between locations needs consideration of the relative positioning of a Location to the transport infrastructure. It should also be noted that only during the traffic simulator stage, when the physical interaction of persons and vehicles is being simulated, is it required to have high fidelity tracking of spatial coordinates, e.g. second or sub-second detail.

In the general context of travel demand the focus is upon the points of access between a location and the transport network. Fine grained representation of the internal layout of a building and an individual's movements inside during an activity are not required to be modelled. The *Person* is simply deemed to be *at* or *within* the *Location* for the period of the *Activity*. However, the representation of the building's external shape may be useful for visualisation purposes.

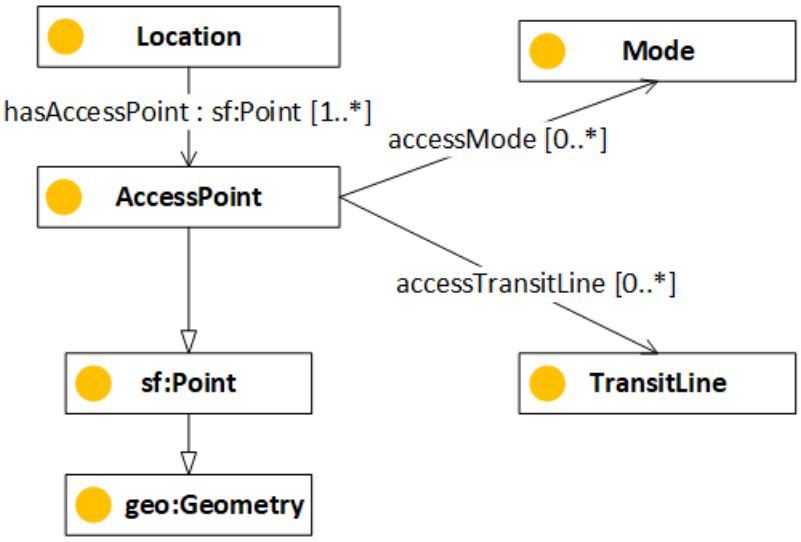


Figure 4.28: Schema for Access Point.

The points of access to enter the location, termed *Access Point* in the schema, provide a restriction on the general shape for identifying when the *Person* has arrived at the *Location*, see Figure 4.28. This permits more specific determination about the closest street or pathway when generating routes than provided by the general polygon boundary of a location. This also allows some consideration of physical barriers, e.g. fencing and walls, without having to explicitly model them.

Figure 4.29 shows an example residential area of houses. Each house is defined by an external boundary with road access. A person (black circle) travelling along the road is seeking to reach the farthest house via the shortest possible path. When only the boundary of the target location is considered (dotted line) then the path follows the shortest distance possible along the road before exiting to travel through the boundary of another house.

When an access point is provided the path follows the road to the entrance

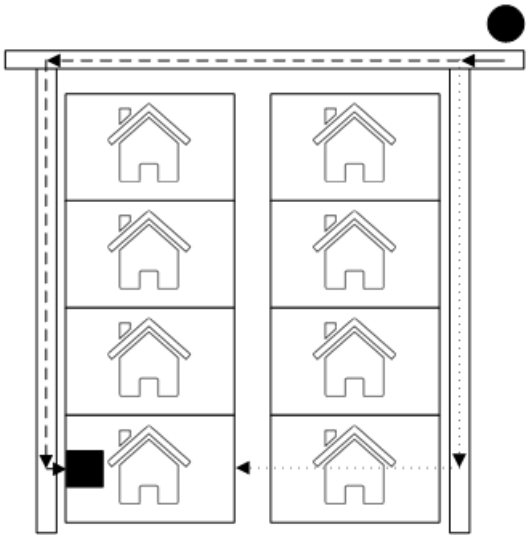

Figure 4.29: Diagram of alternative routes to reach a house location following roads with (dashed line) and without (dotted line) considering an access point (black square).

of the house and only travels a short distance outside of the road network and without crossing any other boundaries. In both cases the travel time upon exiting the road network is likely not to be modelled by a traffic simulator, but the latter case provides a more faithful representation of the desired behaviour and reduces errors.

There may be multiple access points to a location at different coordinates around, within or near its exterior boundary, e.g. shopping centres can have multiple entrances or points of access. These access points may vary in the accessibility available for different modes of transport. This variation could be one way streets for cars, which are bidirectional for pedestrians, or pedestrian only zones in city centres or residential pathways between houses. There may also be issues related to the presence of stairs or steep inclines which render the entrance inaccessible to wheelchair users. Further properties or sub-classing of the *Access Point* would allow differentiating between public and private access points to restrict the *Person* or *Vehicle* based on its characteristics, e.g. residents of a local area or employees of an organisation.

Figure 4.30 shows an example residential area of houses which have road access and are separated by a footpath between the houses. An entrance is present at the

front (black square) of the house for cars and pedestrians but the rear entrance (grey square) is pedestrian only. A person (black circle) travelling along the road is seeking to reach the farthest house via the shortest possible path.

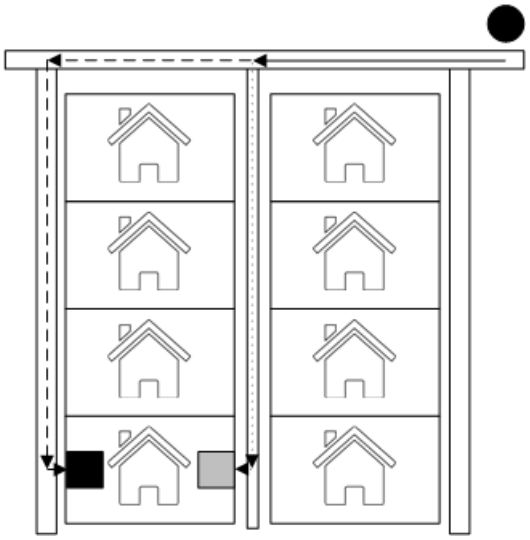

Figure 4.30: Diagram of alternative routes to reach a house location following roads and pathways using car (dashed line) and pedestrian (dotted line) modes via general (black square) and pedestrian only (grey square) access points.

When travelling by car the route must follow the road to the front of the house (dashed line), but when travelling as a pedestrian there is a shorter route along the pathway to reach the rear of the house (dotted line). The accessibility of an access point can therefore influence the selected route and mode of transport chosen by the person. The presence of multiple access points can also increase the number of potential routes to reach a destination.

This incorporation of the mode of transport also permits transfer points to be defined and identified where facilities, e.g. car parking or on-street parking, are provided for changing between modes. In urban scenarios, stopping at a location can be restricted, due to narrow streets or to prevent congestion, resulting in the unavailability of parking for vehicles even though the road network outside the location is accessible. Not explicitly modelling this access can result in cars being routed directly to the location, making the car a favourable choice, when in reality a person would have to park a distance away and travel back to the location, potentially making a nearby bus stop more favourable.

The final concept of access is the specific public transit lines that can be accessed by any person through the access point. Since mode of transport incorporates the general concept of public transit, e.g. bus, train etc., there is a need to give an indication of the specific transit service accessible at the location. Otherwise all locations accessed by a *bus mode* would provide a service to all other locations with a *bus mode*.

The physical dimensions and accessibility are not the only characteristics related to locations in the real-world, but are among the most relevant to travel demand modelling. Additional characteristics applied by a user or model could include the capacity, opening times or postal address as described by public vocabularies [122] or detailed physical building descriptions [64].

The capacity of a location to provide an activity to an individual and its attractiveness or popularity in providing those activities have also been considered in location choice models [32]. The popularity and utilisation of a location vary temporally by day and time during the day while the capacity would typically be a global value that is consistent. Figure 4.31 shows the schema for representing these factors as ordered lists for the different time slots through a day or days with an arbitrary frequency or number of slots.

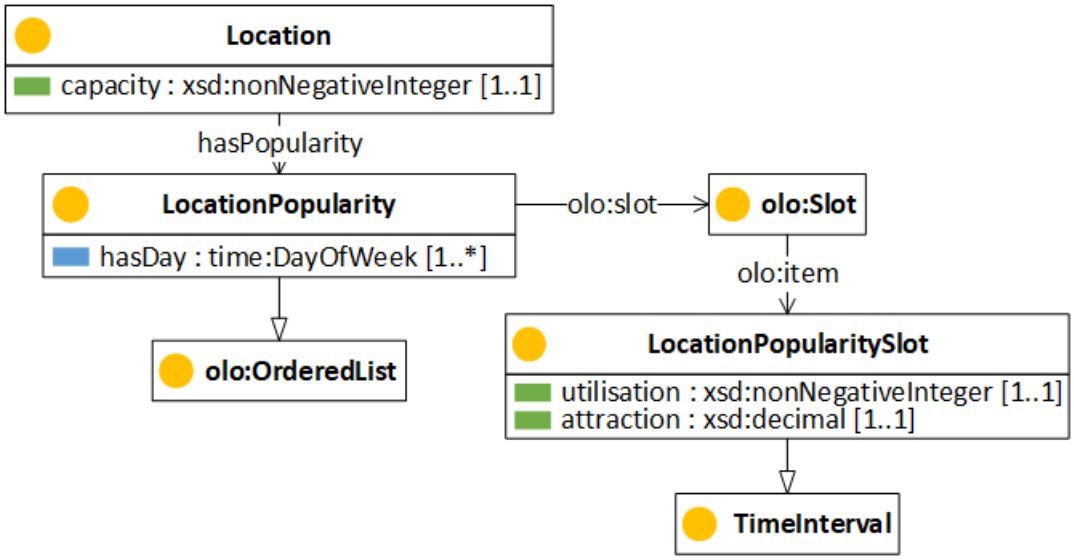


Figure 4.31: Schema for Location Popularity.

### 4.5.8 Geographic Area

A distinction has been made between *Locations*, which are physically manifest in the environment, and abstract geographic areas that are used to group locations or define geographic concepts. These *Geographic Areas*, see Figure 4.32, can include administrative concepts, e.g. local government areas, neighbourhoods, postal delivery areas, electoral wards, school catchment areas and census zones, or modelling concepts, e.g. retail attraction area, travel time boundaries, fare pricing zones.

In all cases these are polygon areas that encompass an area of geography rather than *Locations* which can be defined by points and polygons depending upon the context. These areas of geography can also vary in level of detail, coordinate reference systems and serialisation, so multiple cases may be defined. The *Geographic Area* concept is positioned within the *Spatial* domain in Figure 4.7.

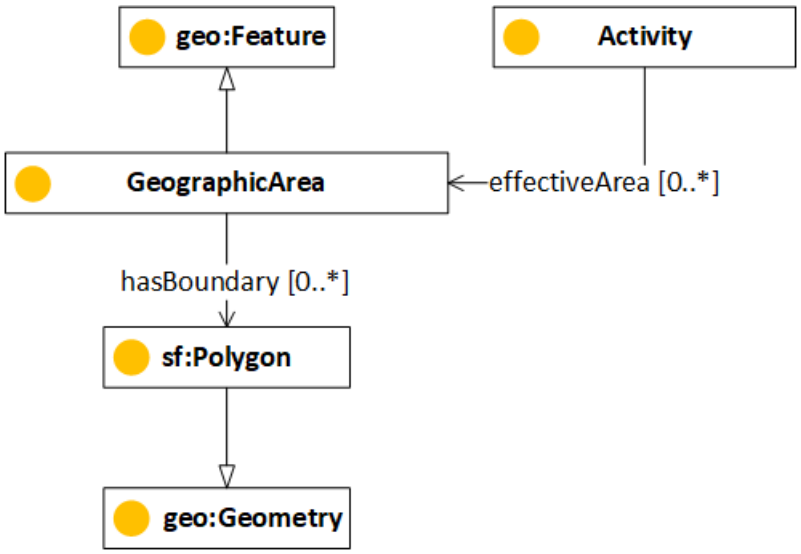


Figure 4.32: Schema for Geographic Area.

### 4.5.9 Network Infrastructure

A vitally important aspect of traffic and travel demand modelling is the network infrastructure. This defines the roadways, public transport routes and other in-

frastructure, e.g. traffic lights, that people can use to travel between activities. This is represented in Figure 4.7 by the domain of *Network Infrastructure*. Road network topology has typically been represented following a node (junction) and edge (road/link) graph structure [32, 46, 63, 83, 123]. This graph structure forms the basis for the RDF representation in Figure 4.33 based upon INSPIRE concepts [63], SUMO [46] and MATSim [32] simulator formats and the GeoSPARQL standard [37].

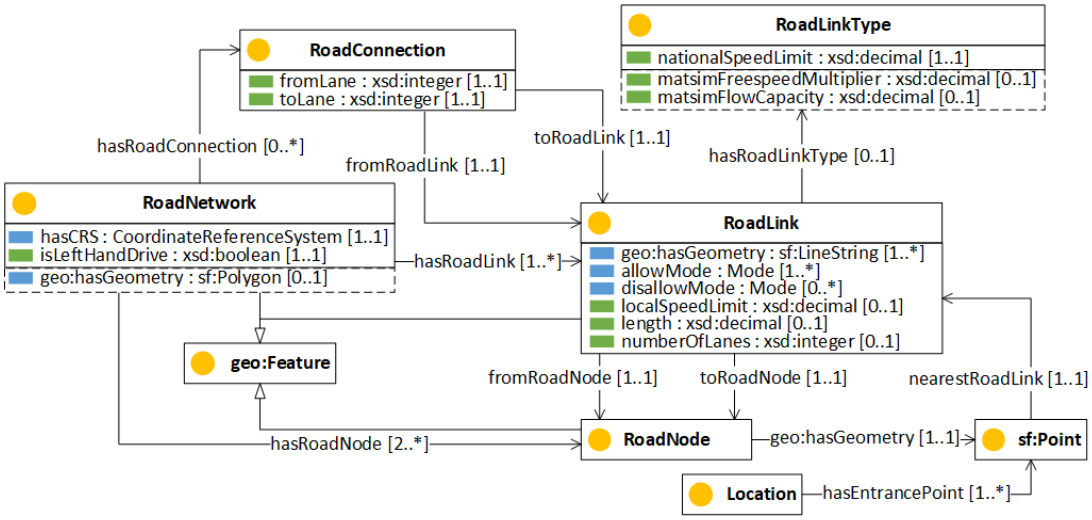


Figure 4.33: Schema for road network described as a graph structure of links/edges and nodes.

The nodes are given spatial coordinates for their placement, while the edges specify the connections between these nodes. The spatial structure is simplified with nodes only being placed when features exist, e.g. edges joining at a junction or changes in road condition. Therefore, the edge/node graph structure is an abstract representation formed from *Road Nodes* and *Road Links*.

Detailed curvature of roads used for visualisation are represented as additional geometry properties through series of segments in line strings. Further detail about the actual physical dimensions of a *Road Link* can be included, but modelling and simulation is more focused upon the number of lanes to indicate capacity than actual physical dimensions, with visualisations often applying fixed lane widths.

To support this abstract representation the length of a *Road Link* is stated

rather than being calculated as the straight line distance between nodes. The edges themselves represent the centre line of a road way, but can vary in format between being unidirectional or bidirectional. In unidirectional formats, the road extends to one side of the centre line depending upon which side of the road vehicles travel, i.e. left-hand or right-hand drive.

This means that a two-way road would be represented by two edges with reversed start and end nodes. In bidirectional formats, the road extends on both side of the centre line with a special property used to state that a road is one-way. The unidirectional approach has been adopted here as it provides a directed graph utilised in routing algorithms, e.g. Dijkstra, A-star.

Additional properties of the road link are the modes that are permitted or not permitted from travelling along the road. General characteristics applicable across *Road Links* are described in *Road Link Types*, following the design points discussed in Section 4.2.4. This can include national speed limits and other parameters, e.g. simulator specific values, that are invariant across scenarios. Values that are varying between scenarios would be placed as *Scenario Definitions* 4.6.1 and selected for the required modules (Chapter 5). Further definition, not shown here, would be lane mode restrictions, e.g. bus, cycle or pedestrian only.

The connectivity through the network can also vary with turning restrictions from certain lanes not being permitted, e.g. no left turns or bus only left turns. These are captured through explicit road connections between the lanes of roads as *Road Connections*. These *Road Node*, *Road Link* and *Road Connection* form the collection described by the *Road Network*. This *Road Network* also describes the meta-data of the geospatial coordinate reference system, the geographic bounds and the side of the road vehicles drive upon, i.e. left or right hand drive.

Further detail is also available in source datasets to describe restrictions on height, width, resident access, time period access and payload etc. However, the utilised micro-simulators, where routing functionality currently typically reside, do not incorporate this level of detail in their route planning (Section 3.3.3.4). Therefore, there are additional concepts and detail that could be considered in traffic modelling and micro-simulation, but are not currently being utilised.

Although the source datasets are highly detailed in describing the roadways

there are certain gaps. The positioning of traffic lights are not publicly available and instead modellers have relied upon partnering local transport network managers to provide this data. The corresponding phasing of these traffic lights is also not widely published for public use. Therefore, micro-simulators, such as SUMO, have provided heuristic algorithms to guess the location and phasing of traffic lights. This represents a noticeable source of error and disconnect between the simulated environment and real world conditions.

Traffic light phasing between and along priority and non-priority routes are designed to have an impact on travel times and traffic congestion, which would not be accurately represented. It would be straight forward to extend the schema to incorporate the physical traffic light systems and phasing as part of the knowledge-base using spatial concepts and the approach described for location popularity (Section 4.5.7).

Similarly, road signage for directing road users to key routes and locations is not described in the datasets or included in routing algorithms. Instead metrics for minimising cost, whether distance or travel time, are utilised. These signs can influence the routing decisions of individuals during route planning and driving.

An additional gap in the datasets is the lack of local speed limits for specific roads. Instead national speed limits must be applied based upon the type of road. However, there can be substantial variation between the type of roads and the permitted speed limits as determined by local authorities. Large multi-lane roads in urban areas can have lower speed limits than small single-lane roads in rural areas.

This again is a noticeable source of error as speed limits can double or triple in value, with consequent reduction in travel times, depending upon road conditions and local policies. Manual identification and correction of these anomalies becomes impractical in large scale scenarios; wastes user resources; and risks error. Therefore, the proposed schema has potential to be expanded with more detailed concepts, but there are practical issues in obtaining data to support the concepts.

### 4.5.10 Goods

The demand for travel does not solely originate from the need to transport people, but also the movement of goods (Section 1.2) and is included in Figure 4.7 by the *Goods* domain. These goods are moved through complex supply chains from point of manufacture or imported to commercial and retail customers. These supply chains consist of multiple participants seeking to optimise the logistics of transportation to ensure that deliveries and stock levels satisfy consumer demand. The Artificial Societies class of travel demand modelling (Chapter 2) seeks to achieve highly detailed modelling of the real-world, which could incorporate these demand and supply characteristics.

This level of detail is out of the scope for this work and therefore goods are not specifically captured by the schema. Instead the approach has been to treat the freight vehicles with their drivers and the freight companies at an abstract level of being *Persons* and *Travel Groups*, which have their own behaviour characteristics, but still seek to generate and follow schedules and routes like any other transport user. The extensible nature of the knowledge-base (Section 4.2.1) and the proposed flexibility for selecting modules (Chapter 5) means that the more detailed description of goods and their modelling could be developed for inclusion in future work.

## 4.6 Concepts for Travel Demand Modelling and Traffic Simulation

This section discusses those concepts that have been identified as necessary for the transfer of data between modules and as part of the modelling process. These are intended to be extended by modules as required by their own modelling assumptions and design. These concepts are all placed in the *Demand Modelling & Simulation* domain of Figure 4.7. The interaction of the principle concepts between modules can be seen in Figure 4.34.

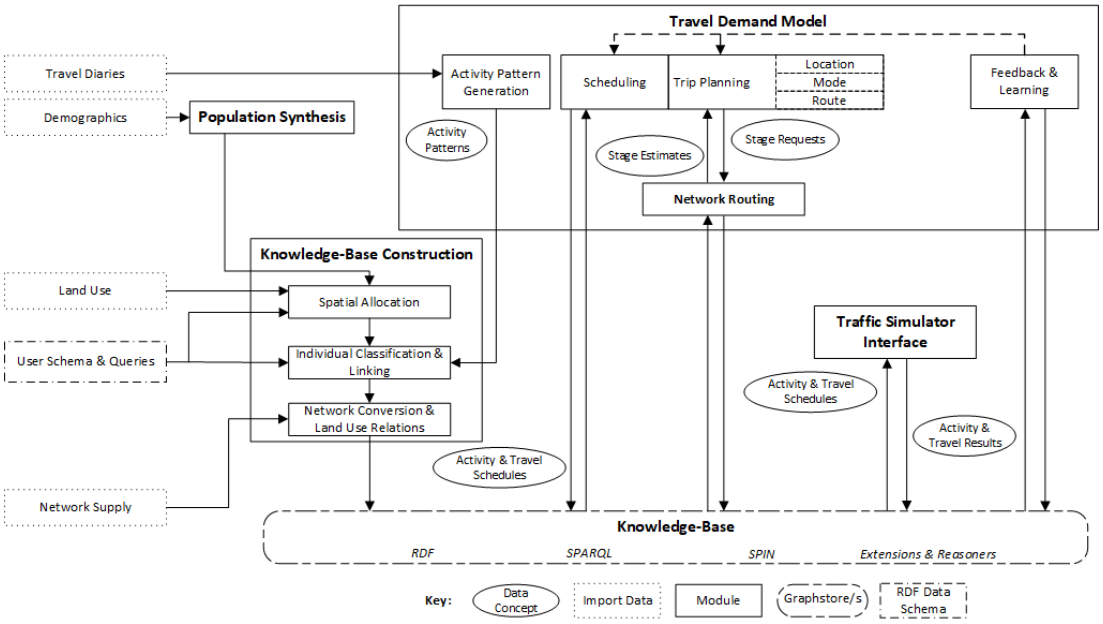


Figure 4.34: Diagram of main components and schema data concepts for module interactions.

### 4.6.1 Travel Scenario

A key objective of adopting the proposed framework is to allow greater comparison between alternative implementations and the exploration of variation within those implementations. There are a range of parameters used by each implementation, which influence the resulting outcomes, e.g. discrete-choice model coefficients. Similarly, there are parameters that vary between scenarios being explored, e.g. start time, end time and day.

In contrast, there are concepts in the knowledge-base that are invariant over the typical modelling time-frame of one-day, e.g. road network infrastructure and population demographics. Therefore, there is a distinction between the temporary data being applied for a particular scenario and the persistent data present for all scenarios using that knowledge-base. Handling of changes over the longer term or significant variations to the persistent data could be represented by the creation of multiple knowledge-bases, e.g. multi-year development of land usage.

Figure 4.35 presents the schema for the *Travel Scenario*, which is the central concept for capturing the instance data of a scenario. It requires the day and

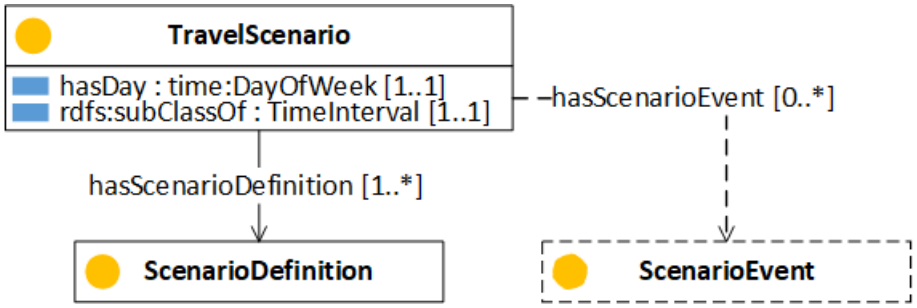


Figure 4.35: Schema for Travel Scenario.

time-period for the execution of the scenario as the focus is scheduling in the short term time-frame of a day (Section 3.3). It is then expected to be extended by additional definitions utilised by the modules of the framework for which two types have been identified.

Firstly, the *Scenario Definition* concept applies to those parameters which apply throughout the scenario. Secondly, the *Scenario Event* describes parameters that may influence behaviour or traffic dynamics for a specific time interval or geographic area of the scenario, e.g. inclement weather, sport and cultural events, road closures. Examples of these additional scenario parameters is shown in Figure 4.36 for several concepts that have been outlined previously. The main definition of interest is modes and the parameters for maximum speed, often utilised for routing, and costs, as part of trip choice selection. Other example definitions include vehicle types, activity types, activity priorities and weather events.

There is no constraint placed on the user as to the number and definition of *Modes* through the *Mode Definitions*. These can distinguish between distinct vehicle or travel types, but also variation within the types. For example, a user can define multiple personal modes, e.g. walking and wheelchair, with different characteristics to reflect varying speeds between age groups.

An example of three *Travel Scenarios* is shown in Figure 4.37. Each *Travel Scenario* is defined with varying day and time interval properties along with three *Mode Definitions*. One *Travel Scenario* utilises a different *Mode Definition* to the other two *Travel Scenarios* for one *Mode*. This demonstrates the flexibility of providing alternative parameters as data in the knowledge-base and then re-using

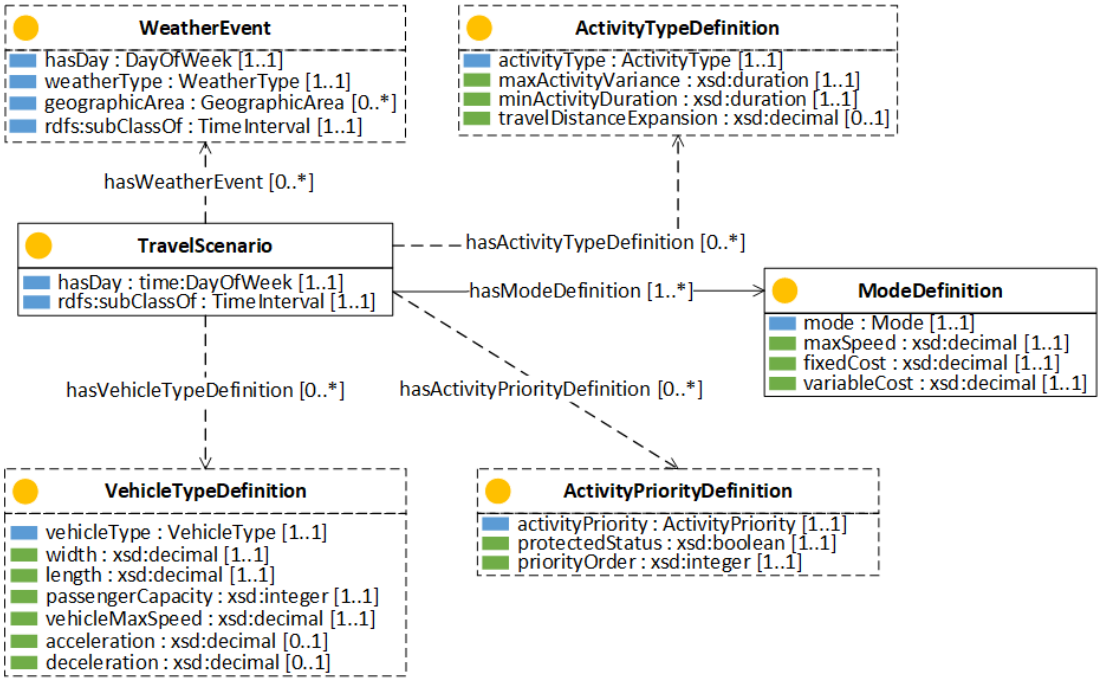


Figure 4.36: Diagram of extended Travel Scenario definitions.

them for multiple scenarios.

Consistency between scenarios can be maintained as parameters can be adjusted in a definition and then applied across multiple *Travel Scenarios*. SPARQL queries can be written with reference to retrieving *Travel Scenarios* and their parameters are then consistently selected without needing to modify any hard-coded values.

The *Travel Scenario* itself provides a convenient unique reference through its URI to describe the scenario. This allows placing the generated data and results of executing the framework with the knowledge-base into a named graph. This named graph can then be stored in the knowledge-base for later extraction, removal or re-use without interference with the persistent data or parameters. Therefore, the *Travel Scenario* provides both a reference to the configuration of the scenario and the retention of results in the knowledge-base.

However, this approach would require repeated iterations of the same parameters to have multiple instances of a *Travel Scenario*, i.e. copies of the same data. Alternatively, a different named graph for the results of each iteration can be

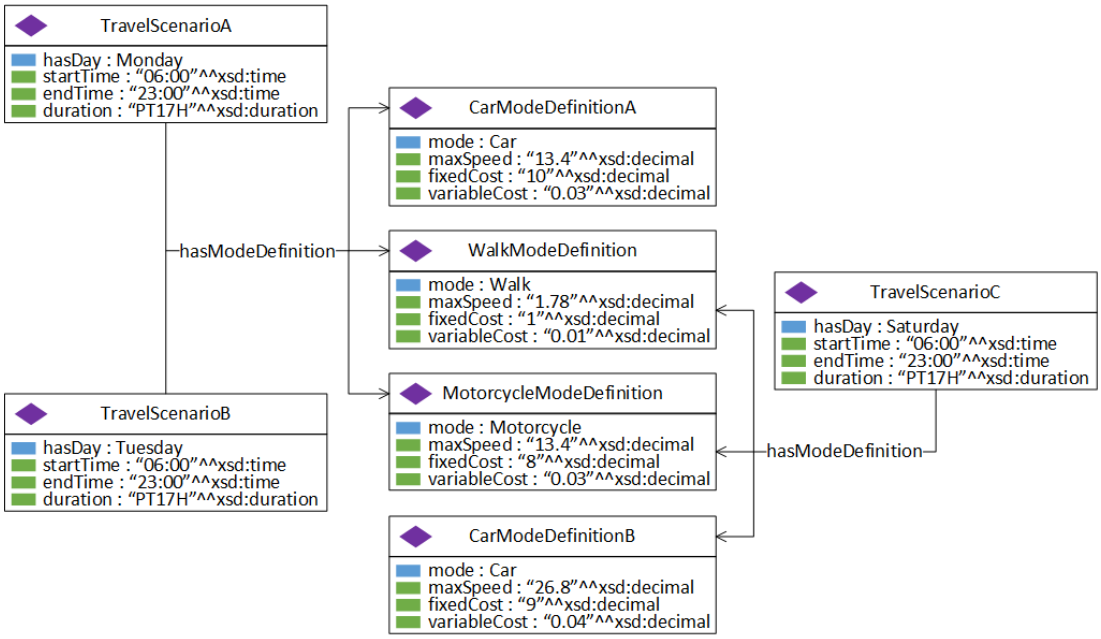


Figure 4.37: Diagram of example Travel Scenarios with Mode Definitions.

provided for the same *Travel Scenario*. This latter approach has been applied using the *Framework Configuration* discussed in Chapter 5.

The definition of *Travel Scenarios* enables multiple scenarios to be executed upon the same knowledge-base, whether with repeating or varying parameters, without interference. Repeated iterations, using the same *Travel Scenario* parameters, can also be utilised as part of learning in a feedback process.

The absence of parameters can also be used to control the scenario configuration. For example, a *Person* may have a specified *Mode* in the knowledge-base, but if the current *Travel Scenario* does not have the corresponding definition then the *Mode* would not be utilised.

### 4.6.2 Activity Pattern

In an activity pattern approach, travel diary data is used to derive context-less templates that describe a series of general activity types with non-continuous time durations, see Figure 3.5. This is modelled by an *Activity Pattern* that consists of a time ordered list of items to represent a whole day of activities, as illustrated

in Figure 4.38.

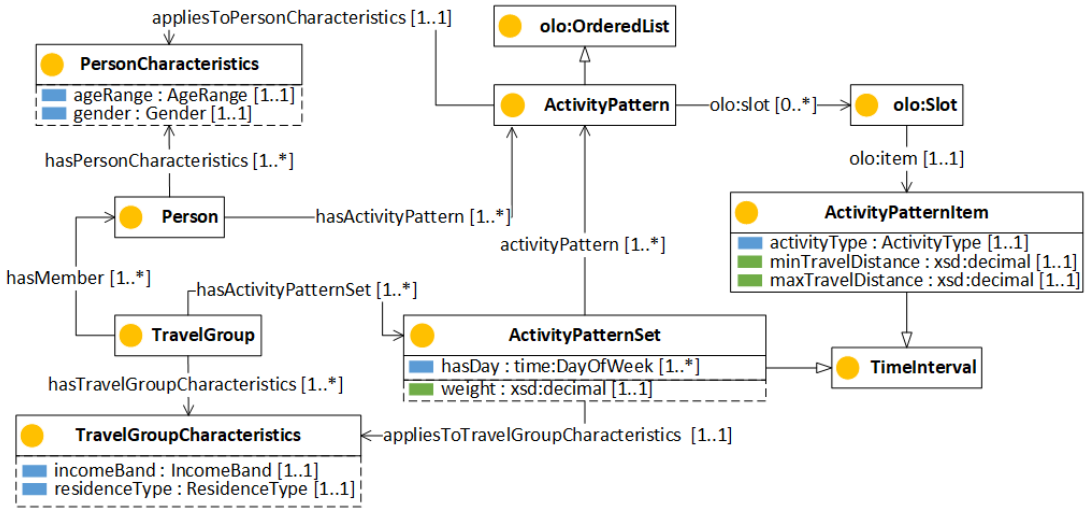


Figure 4.38: Schema for activity pattern templates.

The specific activity episodes of the template are described by an *Activity Pattern Item*, which identifies an *Activity Type*, applicable time interval and minimum and maximum travel distance. The *Activity Type* provides a relation to the *Activity* discussed earlier in Section 4.5.6 so that the skeleton *Activity Pattern* can be aligned with the contextual data of the investigative scenario.

The minimum and maximum travel distance specifies the lower and upper bounds that should be travelled to reach the activity. This can be used either during knowledge-base Construction to only assert relations with *Locations* and *Activities* that a *Person* visits or by a Scheduler module to identify potential *Locations* based on the current context. Therefore, activities which are located too close and too far away can be removed from the choice set. However, a Scheduler module could relax these bounds if no candidates are present to ensure that an activity is always selected.

The *Activity Patterns* are grouped into an *Activity Pattern Set*, so that consistent planning and cooperation for joint activities or escort travel can take place across *Activity Patterns* with aligned activity episodes. The *Activity Pattern Set* also identifies the applicable days of the week, e.g. to distinguish between weekday and weekend activity patterns.

Each *Travel Group* can be associated with one or more *Activity Pattern Sets* according to their matching characteristics, as determined by the user in the knowledge-base Construction stage. In turn, each *Person* within the *Travel Group* is also matched with an *Activity Pattern* from the corresponding *Activity Pattern Set*. Otherwise there would be incoherence in the activities assigned to each *Person*, e.g. a child assigned employment activities and an adult assigned school education.

Multiple *Activity Pattern Sets* for *Travel Group* provides the opportunity for diversity in the daily pattern between different investigative scenarios. An optional weight characteristic could be applied to control the likelihood in a stochastic process for selecting one pattern over others. The characteristics of the *Activity Pattern* and *Activity Pattern Set* provide a number of modelling assumptions that a Scheduler module may choose to employ, ignore or enhance. This highlights an advantage of a knowledge-based approach such that different Schedulers can be applied to the same knowledge-base to produce different outcomes.

### 4.6.3 Activity and Travel Schedule

The key output of a travel demand model for use in traffic simulators is each person's schedule of activities and travel, see Figure 3.5. This schedule, shown in Figure 4.39, is derived from a *Person's* selected *Activity Pattern* based upon their own and the scenario's contextual information and then according to the selected modules of the framework. These schedules are each associated with a single *Person* and *Travel Scenario* so that they are unique for each scenario instance.

The *Activity Travel Schedule* is itself a time ordered list of sequential *Travel Stages* and *Activity Intervals*. The *Activity Interval* describes the time interval, the spatial *Location* and the type of activity taking place. The one or more *Travel Stages* between each *Activity Interval* form a trip as discussed previously (Section 1.2.2).

These *Travel Stages* provide the routing detail for moving between the spatial *Locations* of the scenario. Each *Travel Stage* uses a single *Mode* with multi-mode trips consisting of separate consecutive *Travel Stages*. The *Mode* and other routing data are described in detail by the *Stage Estimate*, which the *Travel Stage*

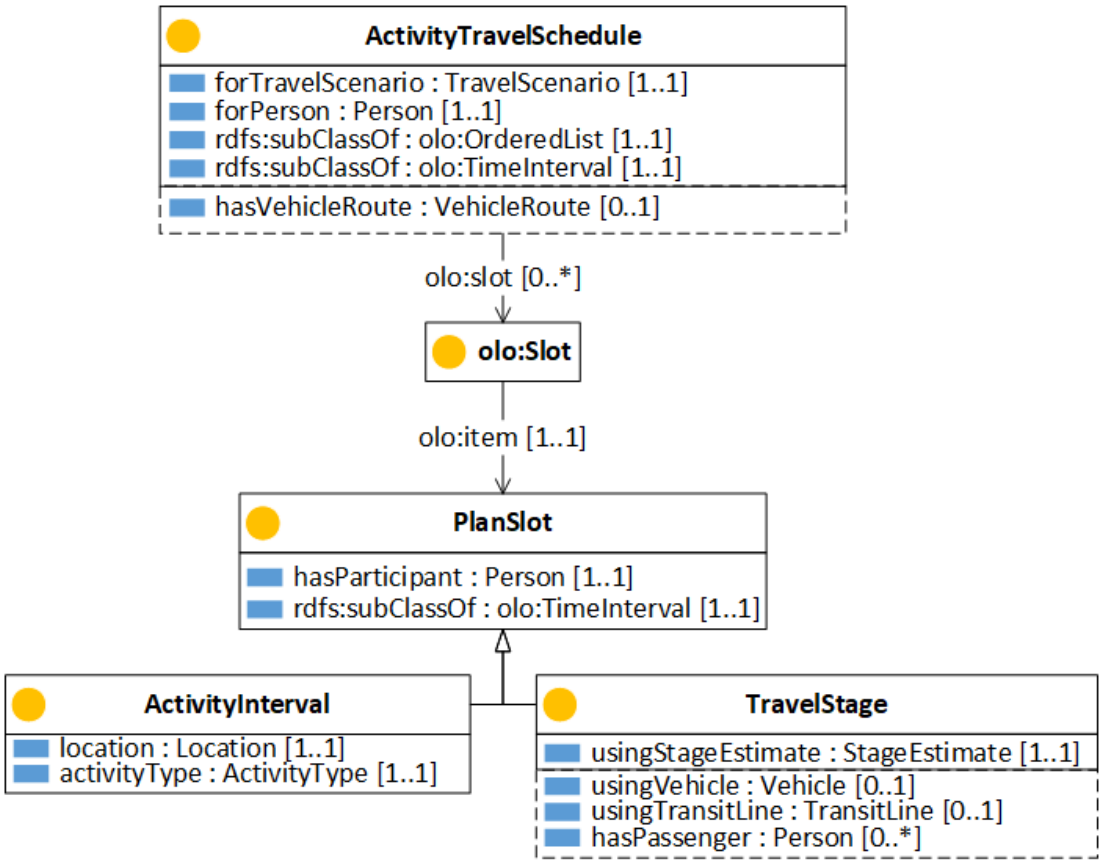


Figure 4.39: Schema for Activity & Travel Schedule.

extends with contextual information.

This re-use of the *Stage Estimate*, discussed in Section 4.6.4, reduces repetition of data in the knowledge-base. The contextual information of the *Travel Stage* is its time interval, any accompanying passengers and any private *Vehicle* (Section 4.5.4) or public *Transit Line* (Section 4.5.5) selected for travel to allow tracking of their utilisation. This tracking supports inclusion of more sophisticated scheduling and travel planning decisions in the travel demand model, such as return tours on public transport or additional stages to collect a vehicle following a sub-tour and return it to its start location [10].

There may also be zero or more associated *Vehicle Routes* for each schedule, which further describes the movement of vehicles due to the schedule, see Section 4.5.4. This describes the movement of the vehicle over the whole schedule and is

necessary input into some simulators [46].

### 4.6.4 Stage Estimate

There have been several adopted processes to decision making in activity-based models (Section 1.2.4). In constraints-based models an accurate estimate of the time required to undertake a trip is required. Discrete-choice models rely upon choosing from different trips based on their metrics. Computational process models use heuristic rules to make trip choices. These models vary in whether the route itself is an output of the decision making process. The routes themselves are split into *stages* as transitions are made between modes (Section 1.2.2).

In some traffic simulators the input data can consist of only the origin and destination of a trip, or its stages, with routes determined using the simulator's own solution. Alternatively, a detailed route can or must be specified. The former approach may be undesirable to the user as implementation differences when comparing multiple traffic simulators could introduce variation into results that would need to be considered in the analysis, i.e. variation being caused by routing solutions rather than simulated traffic dynamics.

It also limits the opportunity to incorporate and compare new routing approaches, e.g. adapting from previous experience, additional network semantics, or environment changing events. Therefore, the framework takes the routing process out of the traffic simulator stage and defines the Network Routing module (Section 3.3.3.4), which produces *Stage Estimates*. These *Stage Estimate*, as shown in Figure 4.40, provide the detailed routing between two *Locations* with additional metrics that may or may not be utilised by decision-making modules.

Each *Stage Estimate* is determined between an access point for a start location to an access point of an end location utilising a specific mode. They each contain the detailed route undertaken through the network infrastructure, typically describing roads, but can represent railway lines, footpaths or cyclepaths etc. The reciprocal journey starting at the destination to the origin would form a different *Stage Estimate* as the route may not be an exact reverse, due to one-way streets or turning restrictions etc. Similarly, a variation in *Mode* would also require a new *Stage Estimate* as vehicle access can be restricted along certain roads or

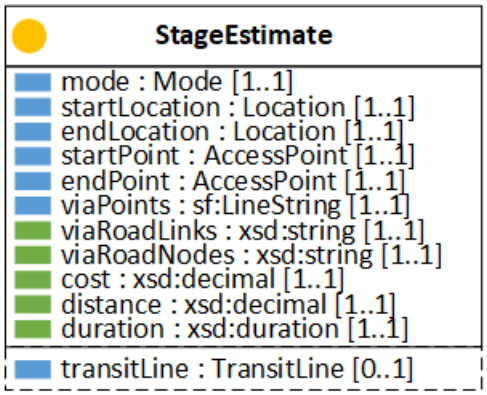


Figure 4.40: Schema for Stage Estimate.

the routing method may be distinctly different, e.g. public transit compared to private vehicles.

The terms *start location* and *end location* have been used instead of *origin* and *destination* to be analogous with the *start point* and *end point* and provide clarity as to what each describes: the general location and a specific point of access. A distinction is made between *stage* and *trip* levels of detail. At *trip* level the start location is the *origin* and the end location is the *destination* and the sub-stages within the trip use *start location* and *end location*. Therefore, the *origin* and first stage *start location* will be identical as too will be the *destination* and final stage *end location*.

To accompany the start and end locations are specific *Access Points* to enable consistency between *Travel Stages* when a *Location* has more than one for a specific *Mode* (Section 4.5.7). Otherwise a person or vehicle could enter a location through an access point on one side and immediately reappear a distance away on the other side. This can cause problems in traffic simulation as the positioning of a vehicle is disrupted. However, a trip planning model could deem that it is accurate for a person or vehicle to exit from an access point different to the one they entered using internal pathways at the location. This fine-level detail is an implementation detail, but both use cases of continuous and discontinuous access points can be represented.

In general, the *Stage Estimates* will be the best-case information derived using the shortest path through the graph structure of the network (Section 4.5.9).

These have the potential to be re-used when the *Mode* and *Access Point* parameters are identical, unless consideration is being given to temporal variations, e.g. a public transit route may not be available during the weekends or previous experience demonstrates to avoid a junction or road at a particular time of day. This re-use reduces computational time when executing the framework.

The impact of computational time becomes very important given that *Stage Estimates* will generally need to be generated as required during execution rather than the whole set being precomputed and stored in the knowledge-base. This is due to the number of routes scaling by $mn(n-1)$, where $n$ is the number of *Locations*, $m$ is the number of *Modes* and assuming a path exists between each location for each mode (Section 6.4.3). If a scenario does not exhaust the full set of travel between all *Locations* then resources have been wasted in calculating a route and the knowledge-base unnecessarily expanded.

The route used by a *Stage Estimate* would typically be generated using shortest path algorithms, e.g. A-star or Dijkstra, which can use distance or travel time as the path metric. These algorithms produce invariant results unless conditions in the network change. However, alternative routes can be derived with modifications to these algorithms, e.g. by preferring major roads over minor, and stored as separate *Stage Estimates* which can then be drawn from based on criteria other than shortest path. The details of the route are stored as *via coordinates* for map plotting and road *edges* and *nodes* for usage by different traffic simulators (see Section 4.5.9).

The *Stage Estimate* could be defined using only the *Access Points* and without the *Locations* as it is the *Access Points* and *Mode* which uniquely define them. However, the retention of the *Locations* with the *Stage Estimate* permits simpler searching and cross-referencing by giving a complete definition. The impact of this is multiple *Stage Estimates* for each case when multiple *Locations* share an *Access Point*, e.g. two shops located at the same premises using the same entrance.

The retention and re-use to avoid repeated shortest path computation can still be applied by identifying the matching *Access Point* and *Mode* characteristics and replicating the routing information, provided the storage also uses the *Locations* as identifiers. The impact on the size of the knowledge-base is no different than

if the shared premises were not co-located and therefore only optimal efficiency is being lost rather than an additional burden being created.

In addition, three metrics have been identified as being generally relevant to traffic models, i.e. *distance*, *cost* and *travel time* [124]. The *distance* of a route is the base metric that informs the other two and storing its value removes the need for repeated calculation and allows comparison between *Stage Estimates*. However, *cost* and *travel time* are likely more informative.

It has been highlighted that *travel time* provides a metric that is easily understood and communicated between travel model users, e.g. transportation engineers, planners, administrators and consumers [125]. Research has found that *cost* has an influence on travel policy and behaviour with travellers willing to pay for faster and more reliable travel [2].

The *Mode* associated with travel is important in determining these metrics. A *Mode* that is able to travel at higher speeds can cover a distance quicker than one which is constrained to lower speeds. Similarly, the cost of a route, whether the financial cost, e.g. bus fare, or a more general penalty measure, e.g. hilliness when cycling, may discourage a route being selected. In cases where a public transit *Mode* is being applied then the *Stage Estimate* will identify the specific *Transit Line* that needs to be utilised.

The *Stage Estimate* may be suited to additional metrics. Measures of quality for travel times have been developed, e.g. Misery Index; the distance between the mean travel time and the mean travel time of the 20% most unfortunate travellers; and travel time reliability at certain times of day [108, 109]. It has been shown that people prefer routes with higher mean travel time and small variability over a route with lower mean travel time but higher travel time variability [108, 109]. Therefore, a user schema and modules could include these metrics or other measures of learning or feedback for routes.

These metrics can be included as properties on *Stage Estimates*. In addition, alternative route choices between two locations could be generated to provide a greater selection of choices for travel through the network. These additional characteristics are dependant upon the modules selected for the framework rather than the core *Stage Estimate* data structure defined here.

### 4.6.5 Trip Context, Stage Request and Trip Plan

The objective of the travel demand process is to generate a schedule of *Activity Intervals* with connecting *Travel Stages* (Section 4.6.3). These *Travel Stages* are composed of contextual information and routing detail. This routing detail is provided by the re-usable *Stage Estimate* (Section 4.6.4).

This section describes how the data necessary to produce the *Travel Stages* and *Stage Estimates* are passed between the three component modules of Scheduling, Trip Planning and Network Routing as outlined in Figure 4.34. The scheduling process operates at different levels with decisions being made over the short term (Scheduling) and near term (Trip Planning) based on predicted routing information (Network Routing) as discussed in Section 3.3.

It has also previously been discussed (Section 1.2.2) that a trip is composed of one or more stages while a wide variety of options can be applied to derive a trip: mode, route, intermediate stops, duration. Therefore, the trip generation process must define the context of the necessary trip; decompose this into multiple potential trips of multiple stages; determine the route, value and viability of these stages; and select one of these trip and its constituent stages as the result. Three concepts are defined for these different stages of the process: *Trip Context*, *Stage Request* and *Trip Plan*.

These data concepts and the *Stage Estimate* concept (Section 4.6.4) are passed between the modules of the framework, as shown in Figure 4.41, starting from the high level process of Scheduling to the low level Network Routing before returning back to the Scheduling. For each step between Scheduling and Trip Planning there may occur multiple instances between the Trip Planning and Network Routing modules. Multiple alternative *Trip Plans* may be produced before one is chosen to be returned as the response to the original *Trip Context*. The whole process may be repeated many times during schedule construction as travel takes place between activities.

The associated properties of these data concepts vary in the scope of options provided, as shown in Figure 4.42, and tend from the general to the specific. The *Trip Context* is designed to express multiple alternative options that can then be explored to find the optimal response, or rather the most appropriate

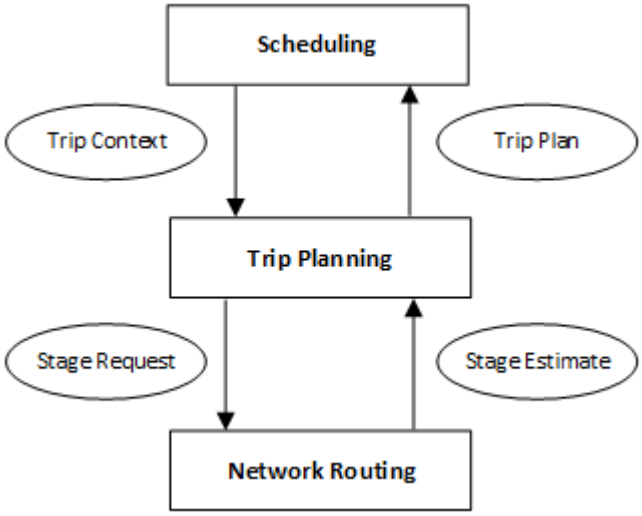


Figure 4.41: Diagram of trip generation process showing data passed between modules.

for the applied conceptual model as human behaviour is not always optimal. Therefore, multiple destinations, modes, access points, vehicles and transit lines can be included with each increasing the diversity of possible trips, stages and routes. These properties could also be asserted in the singular or excluded if a very specific outcome is required such as needing a return journey using a specific mode.

The *Stage Request* is more specific in explicitly stating the start and end of the stage. Diversity is achieved by multiple alternative *Stage Requests*. There is no *Trip Vehicle* information as it would be overseen by the Trip Planning module as to whether the stage is valid to use the vehicle before creating the *Stage Request*. An invalid stage could occur due to the vehicle not being able to access the *end location* through an *Access Point* for that *Mode*. This would require finding a transfer location to change *Mode*, discussed further in Section 6.4.2. Facilitating this change at the transfer location would require two connecting *Stage Requests*. When valid the vehicle's *Mode* is used to inform the *Stage Request*.

Usage of a public *Transit Line* and corresponding time-frame is of concern to Network Routing modules that support public transit. The generated *Stage Estimate* would be dependant on the available timetable of the *Transit Line* (Section 4.5.5). The time-frame could also be utilised by Network Routing modules

which consider changing traffic dynamics, e.g. scenario events such as road closures or weather, or utilise traffic congestion forecasts to produce varying routing solutions for the same start and end locations.

The *Trip Plan* concept represents the planned travel of an individual and therefore it is composed of an ordered list of *Travel Stages* which are directly added to the *Activity & Travel Schedule* (Section 4.6.3), but includes the overall *origin* and *destination* for ease of reference. These *Travel Stages* are formed from a reference to the underpinning *Stage Estimate* and the contextual information associated with the request, e.g. vehicle, transit line or passengers. In principle a *Trip Plan* could describe two *Locations* that are co-located and so do not require travel to move between with an empty list of *Travel Stages*.

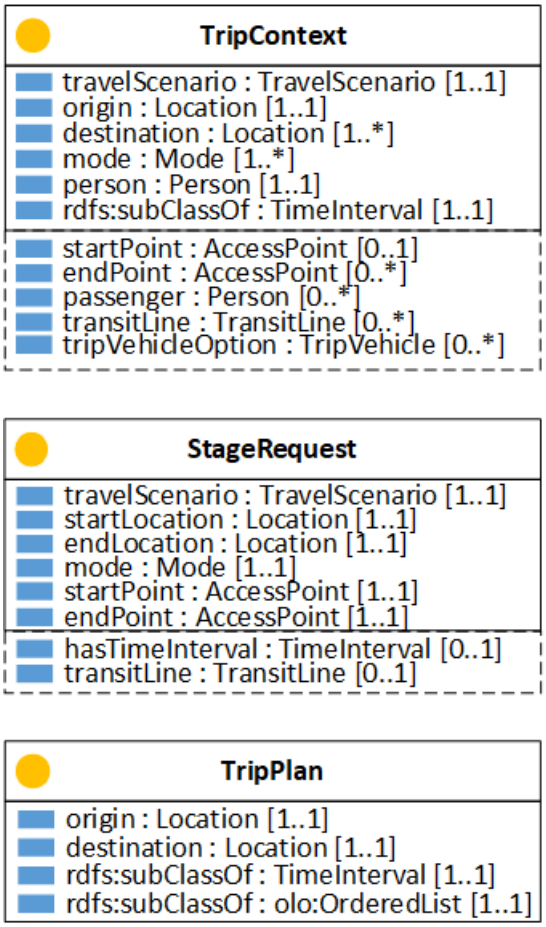


Figure 4.42: Schema for Trip Context, Stage Request and Trip Plan.

These concepts represent an inter-relationship at the data level that is illustrated in Figure 4.43. Certain properties are not necessary for the generation process to function, i.e. *based on*, *requested for* and *determined for*, but demonstrate the inter-relationship. However, their usage and storage in the knowledge-base does provide a mechanism for examining and auditing the generation process. In this view the *Stage Request*, *Stage Estimate*, *Trip Plan* and *Travel Stage* can be seen to be subordinate to the *Trip Context*, which triggers the process.

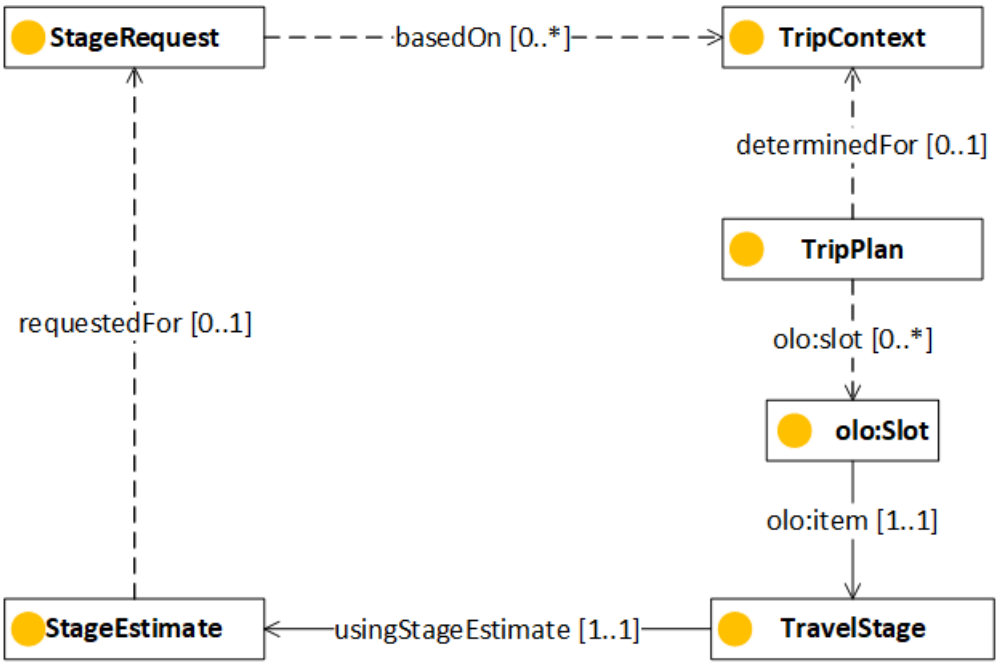


Figure 4.43: Schema for interactions between Trip Context, Stage Request and Trip Plan.

### 4.6.6 Trip Vehicle

The purpose of vehicles is to reduce the burden and travel times of transporting people and goods. These vehicles must be accessed at a spatial location that may not coincide with a person's current location over the course of the schedule or may need to be positioned at a transfer location to access a destination or other modes. Therefore, the *Trip Context* needs to provide information about the vehicle options available to the Trip Planning process and their current state.

This is achieved using the *Trip Vehicle* concept as shown in Figure 4.44 which

consists of *current location* and optional *access point*. Households and individuals may have access to multiple vehicles, e.g. car and bicycles at the home for the start of the day, and therefore these vehicles are specified as an N-ary relation (Section 4.2.2) to allow mutliple cases to be described.

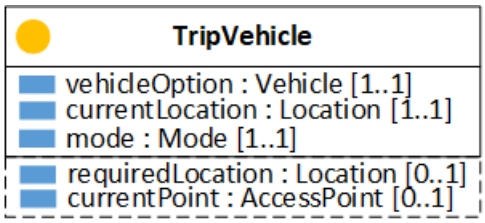


Figure 4.44: Schema for Trip Vehicle.

The *Mode* property is included to fully describe the *Trip Vehicle* for the Trip Planning process. These properties describe the current state of the vehicle within the Scheduling process. There can be situations where the use of a vehicle is mandated, e.g. completing a return journey with a car to the home, so that vehicles are not positioned inconsistently when the scenario completes.

The *required location* is used to specify that a *Trip Plan* is only valid for the *Trip Context* if it results in the vehicle being located in that position. This allows for potential chaining of vehicles during travel. For example, driving a car to a car park, walking to a cycle hire point and then cycling to the destination. This would require the process to be reversed on the return journey rather than taking a bus home and abandoning the bicycle and car.

### 4.6.7 Activity and Travel Result

It has been discussed earlier that the travel demand generation process produces the intended activity and travel timings for a person during the scenario as contained in the *Activity & Travel Schedule* (Section 4.6.3). The outcome of the Traffic Simulation stage is the activity intervals and travel stages of the schedule following experience of the simulated environment and the travel plans of the other participants.

This information can then be used for analysis or incorporated into future scheduling actions, such as influencing *Stage Estimates*, as feedback for learning.

These simulator outputs of activity and travel time intervals are captured in the *Activity & Travel Result*, which follows a similar structure to the *Activity & Travel Schedule*.

The *Activity & Travel Result* illustrated in Figure 4.45 identifies the resulting activity and travel stages. It also has properties for the contextual information of the relevant *Person*, *Traffic Simulator*, *Travel Scenario* and the proposed *Activity & Travel Schedule* to facilitate cross-reference and analysis.

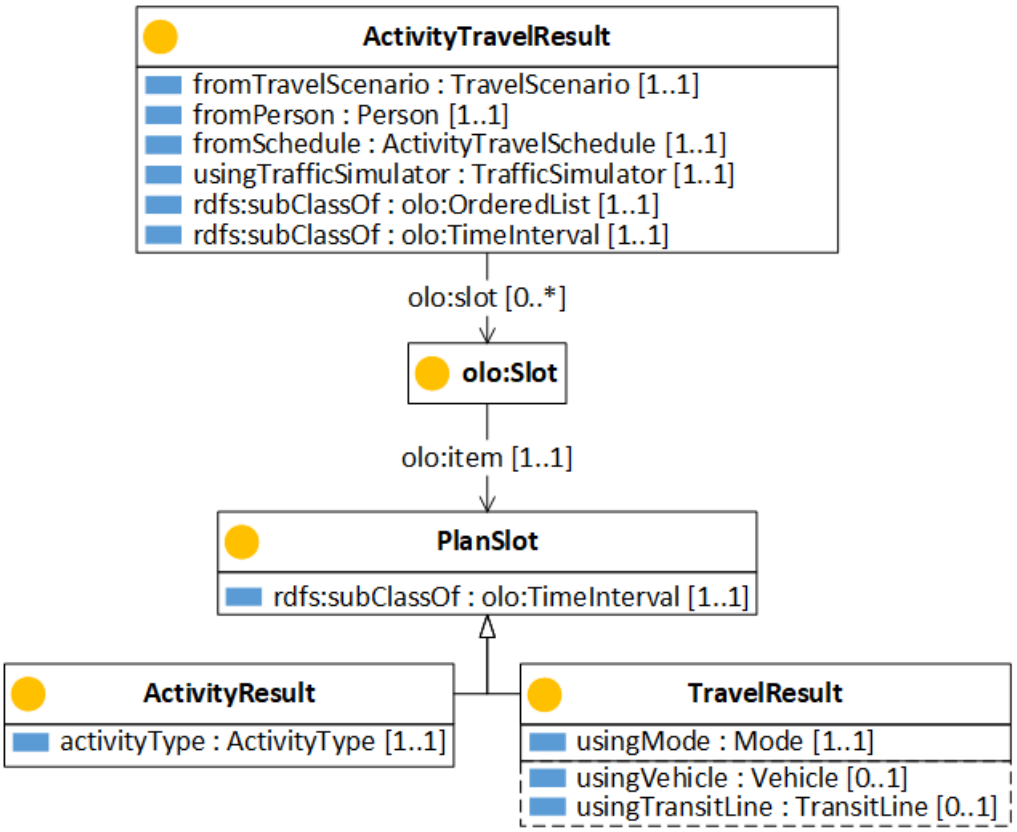


Figure 4.45: Schema for Activity & Travel Result.

## 4.7 Extension of the Person and Travel Group Concepts

In this section it will be discussed how the *Person* and *Travel Group* concepts can be extended to provide more specific modelling. These two classes are the primary

area of interest for activity-based travel demand models. They also have great potential diversity in how they are extended through sub-classes, sub-properties and additional properties. This extension approach can be applied to the other defined classes to assist in organising and contextualising the knowledge-base.

A person forms part of a population which can be sub-divided into smaller populations. The division of a population can be performed in a variety of manners, e.g. characteristic, geography and behaviour, leading to a wide, arguably infinite, set of possible sub-classes. In a Semantic Web approach these can in turn be sub-classified arbitrarily with individuals able to belong to multiple classes (Section 1.3.1).

These sub-classes were utilised in the implemented prototype (Chapter 7) and will be discussed further to illustrate alternative perspectives as shown in Figure 4.46. While the modelled behaviour can be differentiated based on the sub-class, i.e. adult behaviour being different to child, the implemented modules of the prototype instead were driven by the data characteristics of the individual person. This was to constrain the design and tractability of the prototype scenario rather than a limitation of the framework.

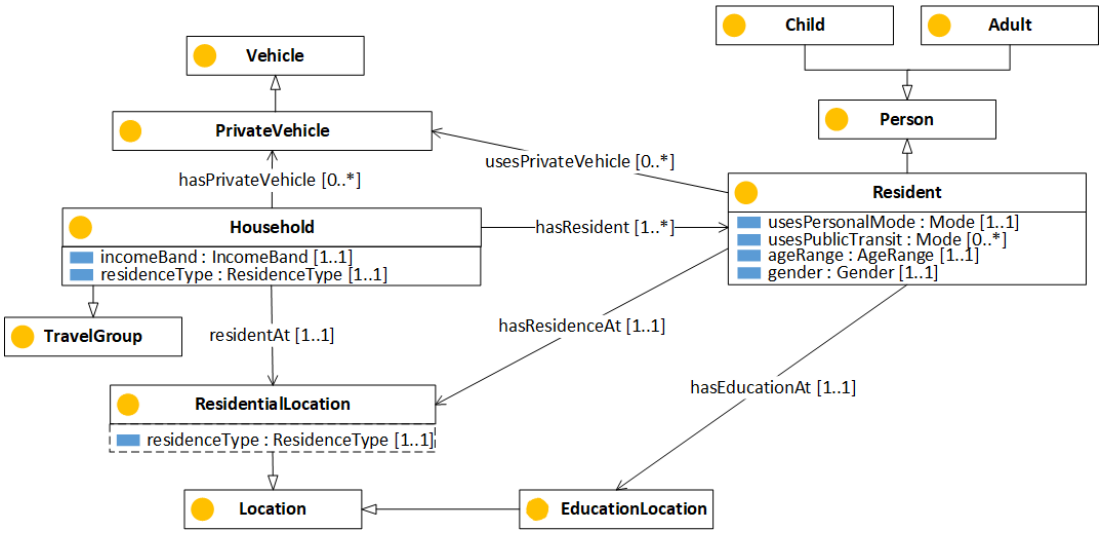


Figure 4.46: Diagram of example extension of the core concepts of Person, Travel Group, Vehicle and Location.

## Adult and Child

A clear distinction in society is made between adults and children. This differentiation influences the behaviour, activities and resources available to each. For example, adults are more likely to travel at any time of day and have greater monetary resources for travel and activities while children will be engaged primarily in education activities rather than employment. The legal distinction between adult and children varies between nations, but is distinctly defined. Further subclass distinctions can be made based partly upon age including different stages of education, e.g. school and university, and employment, e.g. retirement age.

## Resident and Non-Resident

The defined scenario for examination in modelling and simulation must be constrained to a geographic area. The population in this geographic area can be sub-divided into the residents who live within the geographic area and the non-residents. The residents begin their normal day at home and will travel to places of activity, e.g. work, education or leisure, or exit the area. At the end of the day the residents will return to their homes.

In contrast the non-residents will begin outside the geographic area and will arrive and depart over the course of the day at specific locations. These specific locations are the road links at the edge of the geographic area, motorway junctions and public transport connections, e.g. train stations, bus stops and coach depots. In Figure 4.46 characteristic values are assigned to individual Persons and Households. An alternative approach would be to use N-ary relations to reduce repetition and allow easier modification (Section 4.2.2) and applied in Section 4.5.4. The approach of placing characteristics against each individual persons is based upon two reasons.

Firstly, the potential diversity of characteristic values is much greater for persons than vehicles, both in permutations of values and number of characteristics. Vehicles are machines manufactured to a design and specification to fulfil a transport purpose. People are diverse individuals whose resources, behaviours and characteristics are influenced by society, economy and genetics etc.

Most cars have capacity for four passengers and have dimensions constrained

by the physical shape of road lanes, with occasional outlier ignored or tolerated for tractability. A person's, and potentially household's, characteristics can be much more diverse in the possible permutations. Therefore, applying the type-based approach to *Persons*, and by extension their *Household Travel Group*, could result in a large number of *Person Types* and a few *Persons* instances with each type.

Secondly, the definition of an agent for agent-based modelling (Section 1.2.4.4) includes each agent having their own set of characteristics. These characteristics could potentially be modified during modelling. However, a mix of personal and type properties could also be modelled for transient and permanent characteristics.

Transformation of the data in the knowledge-based between each approach can be performed to incorporate a new module using SPARQL queries. Also, these approaches can coherently co-exist in a knowledge-base with the only difficulty being potential confusion or error when selecting characteristics, e.g. selecting the type's characteristic rather than an instance's.

### Freight Driver

In the previous section a distinction was made between people who live in the area of interest and those who do not. Another distinction that can be made is those who regularly travel while fulfilling their employment as opposed to those who travel to reach their employment. An office worker that has arrived at work may not leave their location until the end of the working day. A freight driver will arrive at their employment and then continuously transport goods from location to location performing pick-up and drop-off activities.

This can be modelled as a *Freight Driver* class of *Person* to distinguish from *Resident* or *Non-Resident* person. This classification approach is based upon employment activity and so could be extended to include other transport services, e.g. taxi, train and bus drivers, or those routinely moving between locations during their employment, e.g. domestic plumbers. Therefore, the *Freight Driver* could be modelled as one of several sub-classes of *Employment Traveller* who undertake employment related travel.

A freight driver is an employment activity and so the individual will have travelled from their residence to place of employment. Therefore, they could also be classified as a *Resident* and *Non-Resident* with multi class membership permitted in the Semantic Web. However, the behaviour of these two classes could be considered quite distinctly different. Once at the place of employment a switch in behaviour would be needed.

This would represent a close modelling of the real world in the knowledge-base. However, the practical limits of modelling this complexity do need to be considered and whether such fine detail is of benefit, although this would fit the vision for Artificial Transportation systems [55, 126]. For example, it is questionable in the general case if there is a benefit in modelling a city-wide transport network that is reliant upon correctly simulating all its bus drivers arriving at their workplace on-time. One delayed start would have a knock-on impact to the rest of the model that would require close examination to identify. There is also an assumption that data is available to reliably produce such a fine-grained data model.

Instead a modelling simplification would be for two entities to exist in the knowledge-base for a single person. The first entity is a person that starts and ends their existence at their place of employment and exhibits the travel employment behaviour. The second entity represents the more conventional person who travels to employment and other activities but does not travel during their employment.

This approach allows the consistency of always focusing upon modelling a person with their activities and travel behaviours. Other approaches to generating travel demand for freight have applied an entirely separate modelling approach to that of other travellers [8, 32].

## Autonomous Vehicles

Travel demand has been defined as being satisfied by a countervailing travel supply [32] capable of moving goods or person between locations and encompasses public transit, taxi services, private vehicles or personal locomotion.

An emergent technology is the development of autonomous vehicles capable of

all driving activities [127], including communication and co-ordination with other vehicles and perceiving and influencing their environment. The characteristics of agency have been discussed previously (Section 1.2.4.4) and are exhibited by both humans and appropriately designed software agents, including controllers of autonomous vehicles.

Once accomplished it is envisioned that freight and personal transportation by autonomous vehicle will be another service, which people can summon to their location. This is analogous with existing taxi and courier services and therefore from a modelling perspective would no different from these existing supply methods.

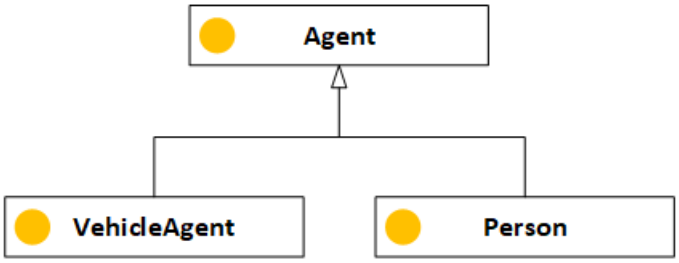


Figure 4.47: Schema of Agent, Person and Vehicle Agent.

From a travel demand perspective autonomous vehicles would still be fulfilling and travelling between activities, i.e. pick-up, drop-off, waiting and maintenance. This would follow a similar behaviour to other travel employment behaviour, e.g. freight or taxi driver, as discussed previously. Therefore, modelling of autonomous vehicles would require a separate classification from people as they form a separate population, but can be incorporated in modelling alongside existing concepts.

The incorporation of autonomous vehicles in the schema would be through inclusion of an *Agent* base class from which would be formed the *Vehicle Agent* and *Person* sub-classes. This follows the existing approach of the FOAF vocabulary [115] with *Agent* and *Person* classes. Figure 4.47 shows the extension of the schema to incorporate the distinction between persons and vehicle agents.

This extension to the schema can be incorporated without invalidating the existing schema (Section 1.3.1). The Vehicle Agent would be synonymous with the physical vehicle that provides the physical transport and could be sub-classed into freight and personal transport etc, but would be conceptually distinct.

## 4.8 Utilisation of the Schema

In the previous section there has been detailed description of the numerous concepts developed for the core schema. These have been identified with the intention of incorporating the minimum concepts and inter-relationships. Illustrative examples have been provided, but these are not mandated concepts. Instead the schema would be extended and enriched by the framework user's own schema or public vocabularies to reflect the specific models and problem under investigation. This section provides a general summary of the schema, connections between the separate concepts and how they can be utilised.

The core concept of traffic demand models is the *Person* and their relationships to *Locations* and *Activities*, illustrated in Figure 4.48. A *Person* represents any individual who travels in the scenario, so could be sub-classified by the user. Each *Person* can be a member of a grouping for organisational and travel purposes, e.g. households. Concepts can be expanded by users through sub-class and sub-property relationships to enrich the data, but retain schema validity.

A multi-dimensional relationship can be formed between *Person*, *Activity* and *Location*. In a single instance a *Person* could be linked to a single *Location* for certain *Activities*, e.g. employment, education and residence, and multiple *Locations* for other *Activities*, e.g. retail and leisure. However, it is a modelling assumption, i.e. the user's schema and selected modules, that all *Persons* have a single *Location* for certain activities and not a requirement of the core schema.

Each *Location* can also provide zero or more *Activities*. For example, a school can provide both primary and secondary education which have different effective time periods and eligible school ages. A school is also a place of employment for teaching and administrative staff. Similarly, homes are the residence of individuals but also a place to visit for social interactions between friends and relatives.

The *Activity* itself may be modelled as unique to a *Location* or shared between multiple *Locations*. Each *Activity* has an effective time and days to reflect availability, such as morning and afternoon opening times. Therefore, a *Location* can have multiple *Activities* with different characteristics but the same *Activity Type*. The *Activities* can be sub-classed according to their characteristics while retaining a grouping through the enumerating of *Activity Types*.

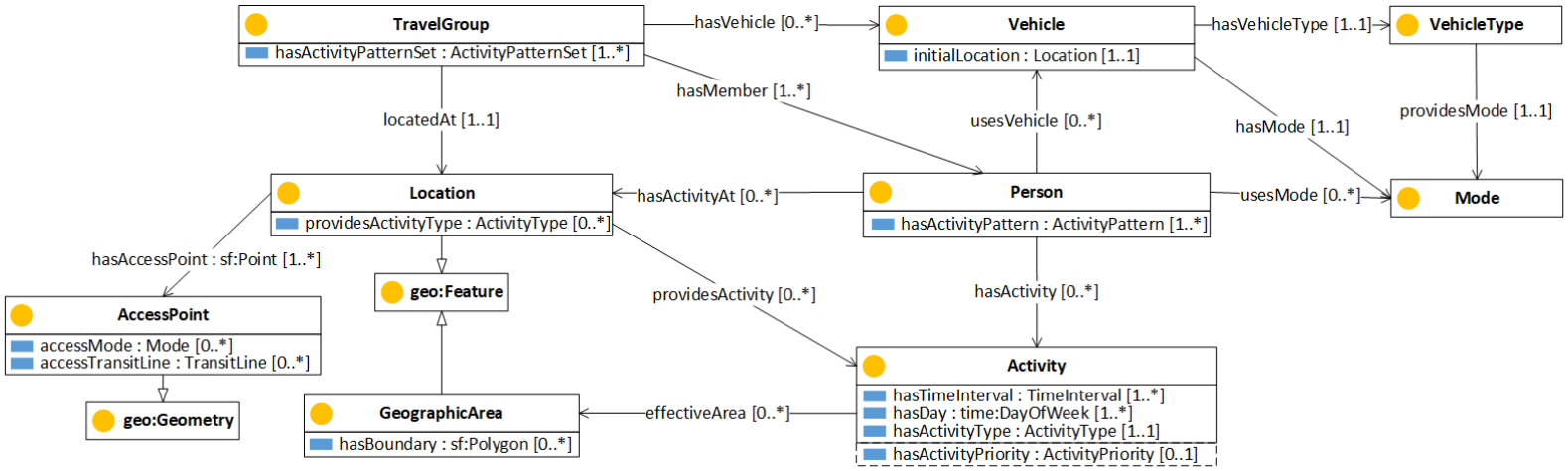


Figure 4.48: Schema for Travel Group, Persons, Locations and Activities.

The value set of *Activity Types* ensures OWL 2 compliance as Object Property relationships must be between individuals and not classes [93] as previously discussed (Section 4.2.4). This approach means that a single *Activity Type* can form a relationship that links many *Activities* to *Activity Patterns*. Each component *Activity* and *Activity Pattern* item identifies a single *Activity Type*, which together forms a multiplicitous relationship, e.g. employment would be an *Activity Type* while employment at a specific office would be an *Activity*. Otherwise a user would have to identify and link every relevant individual *Activity* to an *Activity Pattern* and so impose a modelling burden.

A similar approach is taken to express a *Person's* travel modes as defined by the *Mode* class. These are either the personal or public transport modes a Person uses or those of their *Vehicles*. *Locations* can also identify *Modes* which have access and in turn those that do not. For example, city centre locations with no parking facilities would not be the direct destination for people using a car. Similarly, locations without wheelchair access would not be selected as viable for those people with that mode.

The relationships can be formed using the class, characteristics (data properties), geographic relation or arbitrarily asserted. An example SPARQL query is shown in Listing 4.1 to both classify *Persons* as school age and link them to their local school according to its geographic *catchment* or effective area. The OPTIONAL clause ensures that all are classified, even when not in a school catchment area. Given a *Location* may provide multiple *Activities* then effective areas are applied according to *Activity* rather than *Location*. Extending properties for the *Locations* would be to apply comparative weighting for popularity based on the time and day of the week.

Further detail in terms of sub classes, relationships and characteristics can be included in the knowledge-base by the user as required. For example, a highly detailed set of land use data could distinguish between several types of buildings, their occupants and the types of activities they provide in a hierarchy and structure required by the user.

```
PREFIX ex: <http://example.org/myKnowledgeBase#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>

INSERT{
        ?person a ex:PrimarySchoolChild .
        ?person ex:hasEducationAt ?school .

}WHERE{
        ?person a ex:Person ; ex:age ?age.
        FILTER( 5 <= ?age && ?age <= 11)

        OPTIONAL{
                ?person ex:hasResidenceAt ?house .
                ?school a ex:PrimarySchool ; ex:providesActivity .
                ?education a ex:PrimaryEducation ; ex:effectiveArea
                   ↪ ?catchmentArea .
                ?house geo:sfWithin ?catchmentArea .
        }
}
```

Listing 4.1: Example SPARQL query to classify Persons.

## 4.9 Organisation of the Knowledge-Base

The data of the knowledge-base can be organised into logically distinct components, termed *graphs*. These graphs can be referenced in SPARQL queries either individually by its own URI or as a composite *union* of graphs. The *union* graph represents a merging of the triples so that different query responses would be produced according to the graphs forming the *union*. Therefore, no structural changes would be needed to a query except for the graph references to form the *union* graph. This allows for the combining of different experimental data and parameters simply by introducing a new graph and merging it with a base graph

or graphs.

Several domain concepts have been previously described (Section 4.3). These domains can be applied to assist in the organisation of the knowledge-base as distinct graphs, e.g. all person data contained in a person graph and all vehicle data in a vehicle graph. However, the inter-relationship between these domains can present problems to inferencing.

The boundary between concepts may not be clear cut or convenient for frequent query operations. Inferences that rely on data which spread between graphs would have to be performed on the merged graph. Retaining any inferred triples would require extraction and asserted in an appropriate graph, if trying to retain a purist organisational approach, which would be complicated if a schema had been applied to numerous merged graphs or a wide range of inferences were made.

Similarly, queries that use data from multiple graphs would need to have the references to all the constituent graphs for that query or be performed on the merged graph. In some cases, the contents of a particular graph may be very small, e.g. only having a single vehicle type, so provides limited benefit to separation. In contrast having additional data in a single graph for the whole knowledge-base does not interfere with its operations.

Operations upon a single graph knowledge-base would be simpler to apply in all cases. However, large scale graphs may be slow to query due to size or sub-optimal queries. Datasets that are selectively organised into small graphs of key information would be quick to search. This can have implications if a knowledge-base is being used in a parallel environment where a prolonged search by one execution can block the search or writing progress of other executions.

The division of the knowledge-base does present an advantage given the varying permanence of the different data concepts. Certain data may persist across all executions, e.g. person characteristics and road network infrastructure, while other data may be re-used multiple times, e.g. scenario parameters and activity templates, or only be required for the duration of execution, e.g. intermediate data generated by modules.

The time-frame of the scheduling and planning processes are typically focused on the short term, i.e. days, weeks or months. A longer term view across years would incorporate changes to land use, resources and demographics and is a

feature of some models [9, 55]. Therefore, the notion of persistent or permanence depends upon the context.

Finally, adopting separate graphs would allow the addition and removal of datasets in their entirety. This would allow quick disposal of incorrectly configured or no longer required executions and the inclusion of new parameters. Also, alternative versions of parameters, data and results can reside alongside each other in separate graphs. Therefore, the results of different scenarios or the outcome of repeated executions could exist in the same knowledge-base as a comparable set.

The division of the knowledge-base into graphs can provide benefits, but can introduce complications to its usage. Figure 4.49 shows the named graphs used in the prototype implementation (Chapter 6) and is provided for illustration. The *People*, *Organisation* and *Vehicle* concepts have been brought together due to their close inter-relationship and placed into the *Travel Group* graph. The *Framework Configuration* graph provides the information for locating datasets within the knowledge-base and modifying the execution of the framework (Chapter 5). The *Demand Modelling & Simulation* concept has been split into *Scenario* and *Results* so that the execution output can be easily modified and removed without impacting the scenario set-up.


Figure 4.49: Diagram of named graphs applied to the prototype knowledge-base.

There is no requirement by the framework for a user to adopt this approach. A single graph could be used for all the data in the knowledge-base and referenced

in the configuration of the framework. Adjustments can be made according to the user's choices and to meet the requirements of the selected modules. The utilisation of the separate graphs and the configuration of the framework is discussed further in Chapter 5.

## 4.10 Chapter Summary

This chapter has discussed the main data concepts of the core schema for the framework for travel demand generation. It has outlined the design principles which have been applied to the schema to enable interoperability with Semantic Web technologies. There has been identification and use of public vocabularies to integrate fundamental concepts so that links with other datasets can be established and to facilitate re-use.

The extendible approach of the Semantic Web enables additional connections to vocabularies and new concepts to be incorporated into the schema by the user. This has been discussed through alternative examples to extending the *Person* and *Travel Group* concepts, which is applied further in Chapter 7.

The top-level concepts of the schema have been identified and then applied as the basis for more detailed consideration of data concepts. These have generally been identified as concepts relating to the physical world and those relating specifically to travel demand generation processes. The described concepts have sought to be a minimum representation with the user and module implementations able to determine more specific definitions as they require.

The investigation of these concepts has led to the identification of several gaps in road network semantics and datasets, specifically relating to traffic signalling; implemented routing algorithms; and local speed limits, that have an influence on the modelling behaviour and outcomes. The incorporation of these concepts can be accommodated in the proposed knowledge-base approach, but are not currently supported by published datasets. Further development of routing algorithms can be supported by data concepts being added to the knowledge-base in the People, Vehicles and Network Infrastructure domain. In addition, applying the modular design for Network Routing discussed in Chapter 3 would remove dependency on traffic simulator implementations.

The varying permanence of certain properties within the knowledge-base has also been identified. The objective of comparing modules and scenarios can lead to different versions of parameters and content that would be problematic to manage and retrieve if directly associated with an entity. This has been resolved by the *Travel Scenario* and associated *Scenario Definitions* to provide N-ary relations that describe the set of parameters to be used in an execution instance. These are associated but distance from the permanent data concepts of the knowledge-base.

There has also been discussion of the logical organisation of the knowledge-base around named graphs in an arrangement to suit the user and their investigative scenario. This arrangement can align with the top-level concepts, but also take into consideration the permanence of the data so that transient data generated during execution can be easily removed from the knowledge-base. This also provides the opportunity for users to select and switch between different datasets and sources of data within the same knowledge-base and is one of the objectives of the framework. The proposed mechanism to support this is discussed further in Chapter 5.

i

# Chapter 5

# Framework Configuration for the Selection of Alternative Behaviour, Techniques and Data

## 5.1 Introduction

In this chapter there will be exploration of the framework's design to deliver the objectives of easing the burden on users in assembling, controlling and comparing their investigative scenarios with multiple implementations of travel demand generation models and traffic simulators. The previous chapters identified the different considerations in the design of the proposed framework (Chapter 3) and discussed the data concepts necessary for the core schema (Chapter 4). This core schema provides a basis for modules and the knowledge-base to be aligned to minimise data transformation. However, it may be necessary for the user to have facility to specify modifications and transformations of data between modules.

A user may investigate multiple alternative versions of modules and datasets, representing different modelling assumptions, parameter values, implementation detail or scope. Therefore, investigations need to be able to compare across multiple configurations of the travel demand modelling process. The framework seeks to provide an environment to assist the user in managing and configuring the differences between these alternative choices. These differences in approach

may be inconvenient when setting up one configuration and then become highly burdensome or inhibitive when applied to multiple configurations.

This chapter will seek to address research question RQ3. It will discuss how the knowledge-base can be constructed from local and remote data sources. There is also examination of the mechanisms available and developed to specify modules; transform data; select alternative modules based on the data; and control the execution of the framework. There will also be inclusion of optimisation options to reduce execution run-times through using local file system knowledge-bases and modules caching invariant data. The facility for users to transform data and make selections introduces potential for integration errors. Therefore, there is discussion of validation steps to ensure that errors are identified early and communicated to the user for correction.

## 5.2 Constructing the Knowledge-Base of the Framework

The previous discussion of the proposed framework in Chapter 3 outlined that current demand modelling is a multi-stage process. The conventional process is reliant upon file conversion between the data sources and the main stages of the travel demand process to present and obtain data in the correct formats. This is illustrated in Figure 5.1 where multiple file conversion processes are required to align the available files of data with the module interface.

Each of these conversion processes may require format conversion, e.g. CSV to XML, or reformulation, e.g. different XML schemas. These conversion processes may form an input or output option for a stage implementation to target another stage implementation, i.e. activity-demand model X produces output in traffic simulator Y's format. However, if this is not the case then the user must implement their own process.

These conversion processes can also require alignment of the data for later stages, e.g. land use indicating relevant zones or population allocated to groupings. In this example, the activity pattern generation process is incorporated into the demand model as it is a tightly integrated component of the overall demand

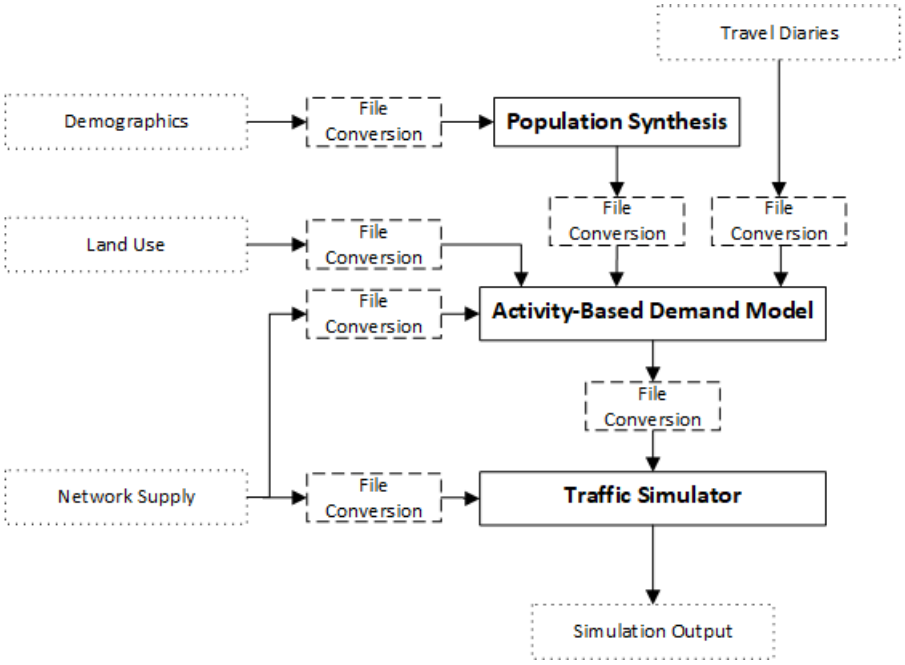


Figure 5.1: Diagram of construction process for conventional activity-based travel demand process.

model. Therefore, only the implemented approach can be utilised and alternative forms of activity patterns or methods for their generation cannot be incorporated into the model.

The first stage of the framework process is constructing a knowledge-base. The required datasets are prepared and added to the knowledge-base. However, datasets are typically published by different authorities and therefore require reconciliation to align. This section outlines the different options that are possible with the framework to construct the knowledge-base and how misaligned data can be transformed or utilised.

### 5.2.1 Constructing a Local Knowledge-Base from Local Sources

The adoption of a knowledge based approach provides a central repository for the data of the travel demand process. The separate input, output and configuration files for the different stages of the process can be stored in the knowledge-base us-

ing a single format: RDF (Section 1.3.1). Therefore, the operation of the process can be reconfigured as shown in Figure 5.2, which is an alternative representation of Figure 3.3.

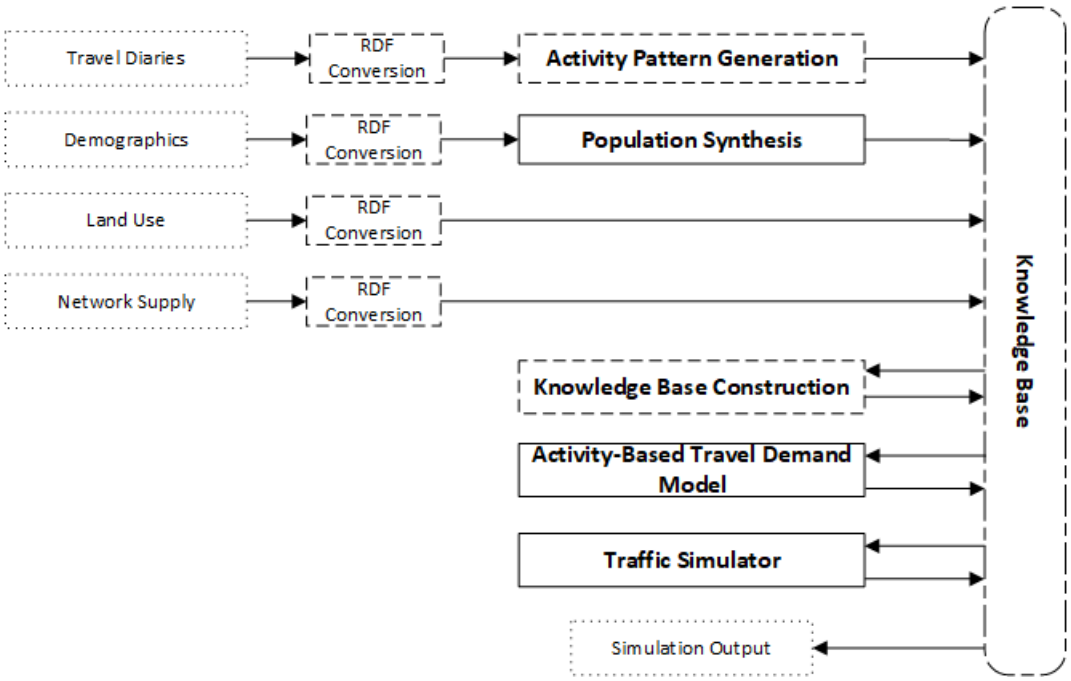


Figure 5.2: Diagram of construction process for knowledge based activity-based travel demand model.

The RDF conversion process is still required to take the published datasets, typically XML or CSV but not exclusively, and prepare them for the knowledge-base. However, published datasets have also included several RDF serialisations (e.g. RDF/XML, json-ld, Turtle etc.) which could be directly added to the knowledge-base. When conversion is required the benefit is that rather than many different file format conversions processes, as illustrated in Figure 5.1, there is the single target format of RDF.

The Activity Pattern Generation and Population Synthesis stages shown in the diagram take the input data and generate new data for the knowledge-base. This new data could be a disaggregate representation or result of a statistical process. The Activity Pattern Generation module is shown as distinct from the Travel Demand Model to indicate its potential for replacement and to feed into

the Knowledge-Base Construction process.

Overall, this approach has potential to reduce the number of interfaces required between modules. Any conversion to wrap an existing implementation is between the knowledge-base RDF and the modules desired input format on a one-to-one basis, where previously it was directly between each module in the process. Therefore, modules are now more interchangeable once they adhere to the knowledge-base format. The benefit of this increases as more alternative modules are considered at each stage.

However, conversion is still required to obtain datasets as RDF or to align existing RDF with the knowledge-base schema. This alignment process is incorporated into the Knowledge-Base Construction stage where the separate data concepts of the proceeding stages and datasets are brought together. Similar steps were required in the previously described process (Figure 5.1), but would take place as part of file conversion or manual adjustment.

### 5.2.2 Constructing a Local Knowledge-Base from Remote Sources

The online design of the Semantic Web is intended to facilitate the sharing and reuse of data. A mechanism for achieving this in the SPARQL protocol is *Federated Queries* [43]. This allows RDF graphstores to be made accessible for SPARQL querying by specifying a URL address. The relevant part of a query is executed on the graphstore with results returned to the originator. The retrieved data can be utilised within the query to interact with local or other remote data. Graphstores made accessible in this manner are termed SPARQL endpoints.

This enables pre-prepared datasets to be made available without needing to process local files as shown in Figure 5.3. The file and RDF cleaning and conversion processes are removed as the data is already readily available in RDF. This approach has been applied by a growing number of dataset publishers as part of the Linked Data initiative [61, 73, 74].

The data being retrieved can be filtered and selected using standard SPARQL syntax without needing to remove it from input files or during knowledge-base construction, e.g. a retrieval query only selects data relating to a specific ge-

ographic area or time period. The data sources described can be intermixed so that a knowledge-base is constructed from both SPARQL endpoints and file datasets on the basis of data availability or user preference, e.g. the user has a more detailed dataset or has generated a dataset for a specific topic of interest.

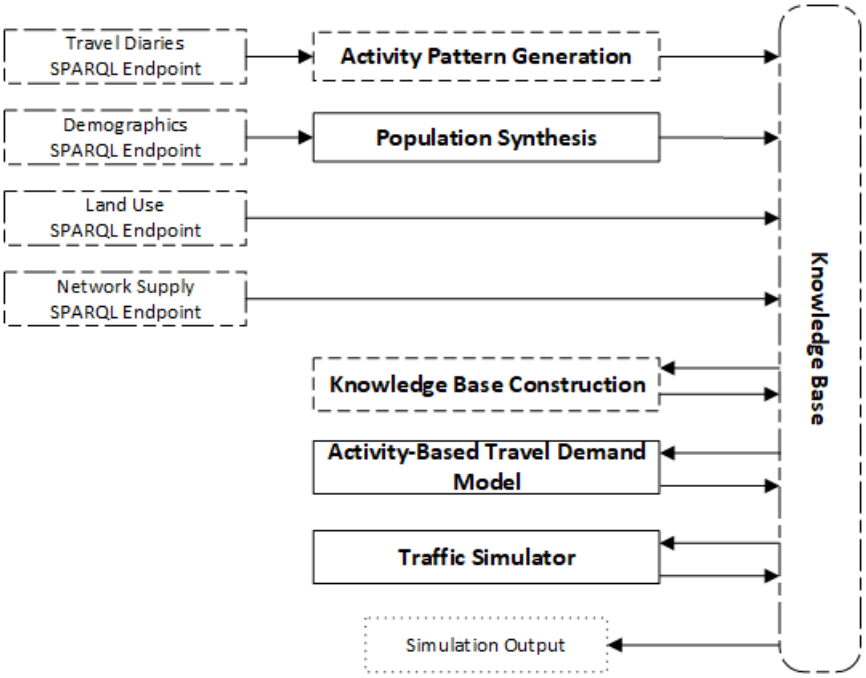


Figure 5.3: Diagram of construction process for knowledge-based activity-based travel demand model using SPARQL endpoints.

An example process of retrieving data from a remote endpoint is shown in Listing 5.1. The geographic area is identified using the GeoSPARQL vocabulary (Section 4.4.1), which could be described using a geospatial shape, to find only locations of interest. These locations are then used to find the households and persons along with their related triples.

```
PREFIX popData: <http://example.org/populationData#>
PREFIX trav: <http://example.org/travelDemandSchema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>

CONSTRUCT{
       ?household ?householdProp ?householdObj .
       ?person ?personProp ?personObj .
}WHERE{

       #Choose an area to target.
       BIND(popData:AreaA AS ?targetArea)

       SERVICE popData:Service {
              #Find locations in the area.
              ?location geo:sfWithin ?targetArea .

              #Households at the locations.
              ?household locatedAt ?location .
              ?household a trav:Household .
              ?household ?householdProp ?householdObj .

              #Person at the locations.
              ?person hasActivityAt ?location .
              ?person a trav:Person .
              ?person ?personProp ?personObj .
       }
}
```

Listing 5.1: Example SPARQL query to retrieve household and person data based on geographic area from a remote endpoint.

An area of technical difficulty with this approach is the retrieval of the necessary graph relating to a URI. In the example listing, the related triples where the households and persons are the subject are retrieved. However, there is no additional data about the properties or objects of those triples. If a module later tries to discover the property of an object, there would be no data in the local knowledge-base. There is no inherent mechanism to record where supporting data can be looked up. It may be that the URI of the resource contains a hostname or domain component that refers to the service, but there is not a requirement for this. Therefore, one of several options would be required.

Firstly, the retrieval query would need to extract the triples to the depth required by the executing modules. This could be an arbitrary number of layers and there is no explicit SPARQL syntax to express retrieval from all sub-branches of the graph. A user would need to be aware of the depth of data available in the endpoint or required by the module. This would require either publication of information about the graph and module structure or investigative effort by the user.

Secondly, the query could retrieve the entire graph into the local knowledge-base. Ideally the endpoint would have an organised structure so that only relevant data is contained in each graph, e.g. one graph per geographic area. Therefore, this may require performing a filtering process to exclude data not of interest, e.g. households outside of the target area, and may leave some orphaned data in the knowledge-base that would need to be ignored during execution or risk disrupting execution.

Finally, modules could be provided with the URL so that they can query the remote endpoint for the supplemental data as required. The local knowledge-base would only contain the URI to initiate the module, i.e. household URI but no composition or member data. This would provide a versatile approach whereby the user would only need to specify the minimum information.

The final scenario to consider in this process is when datasets have been prepared that align in schema and context, i.e. structure and frame of reference. This situation is illustrated in 5.4 by the removal of the Knowledge-Base Construction stage, as transformation and reconciliation of the data is no longer required.

These aligned endpoints could represent national or canonical scenario datasets

that have been prepared for large geographic areas or across multiple time-frames. Users would be able to select and retrieve a subset, e.g. by geography, administrative area or time-frame, through the SPARQL query mechanism for local usage. Transformation could also be applied during this retrieval process, see 5.2.3, to satisfy any investigative or usage requirements.

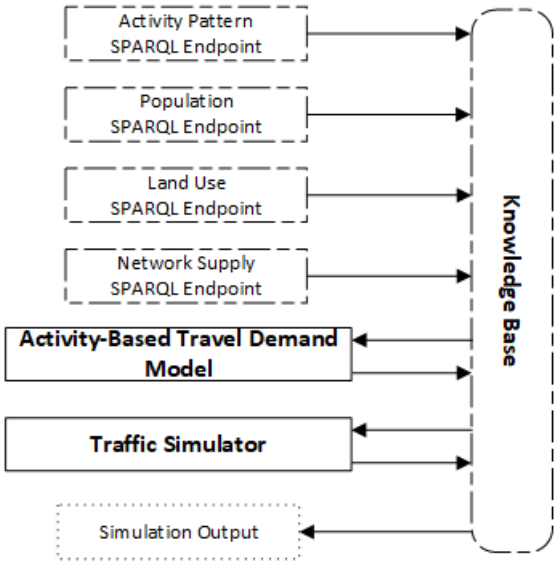


Figure 5.4: Diagram of construction process for knowledge-based activity-based travel demand model using schema and context aligned SPARQL endpoints.

Publication of aligned datasets as flat files is achievable without using Semantic Web technologies. The likely lack of such publications are a reflection of the fragmented approach to travel demand modelling as exhibited by the diverse range of implementations, lack of a overarching schema and no authoritative or centralising organisation. Therefore, certain barriers to this approach are not technical in nature.

An RDF approach would allow the core concepts to be published in an accessible and structured format along with additional characteristics and relations that may only be relevant to specific modules. The outputs of alternative modules, e.g. population synthesis or activity generation, could also be published across multiple endpoints with interchange between them only requiring a change in URL address. The benefit of this approach is that data retrieval can be undertaken quickly and accurately allowing focus to be placed upon the investigative

stage.

The hosting of these remote endpoints does not have to be in a full online environment but can also take place across an internal network. This would allow large scale or computationally intensive investigations to be distributed across multiple computers. One set of resources could host the data endpoints, while another set undertakes the execution of different experimental scenarios. The computers executing the scenarios can retrieve the required data to construct new knowledge-base instances facilitating the potential for automating experimentation across numerous scenarios.

### 5.2.3 Retrieving and Transforming Data for the Local Knowledge-Base

The process of constructing a local knowledge-base requires the importing of RDF datasets or their retrieval from remote online sources. These datasets may then require transformation to align them with the user's or module's schema. The testing of this alignment can be performed automatically on RDF datasets using the schema (Section 5.3.6). The greater the alignment of the schema in the knowledge-base(s) the less need for modification during the execution phase as discussed later (Section 5.3.9). This section will briefly outlines the use of SPARQL to retrieve and transform datasets.

The first identified use case was obtaining data from local file sources (Section 5.2.1). These local files may contain data that is not required or must be transformed to the schema. In Listing 5.2 a source file has been loaded into a graph, but has mislabelled properties and changing datatypes (in this case a string to an integer). These are deleted and replaced with the correct values.

```
PREFIX sch: <http://example.org/schema#>
PREFIX src: <http://example.org/source#>

WITH <http://example/final-graph>
DELETE{
    ?subj src:prop ?obj .
    ?subj src:value "18" .
}INSERT{
    ?subj sch:prop ?obj .
    ?subj sch:value 18 .
}WHERE{
    ?subj src:prop ?obj .
    ?subj src:value "18" .
}
```

Listing 5.2: Example SPARQL query to select and transform data within a graph.

When the source files contain a lot of extraneous data then triples can be selectively extracted. The files can be loaded into a graph of the knowledge-base and then the cleaned data transferred to a new graph, which will actually be used in the framework.

This is shown in Listing 5.3 where a temporary graph is searched for the required data. This data is transformed, in this case using the schema's properties and classes, and then inserted into the final graph. Once completed the entire temporary graph can be dropped from the knowledge-base.

```
PREFIX sch: <http://example.org/schema#>
PREFIX src: <http://example.org/source#>

INSERT{
    GRAPH <http://example/final-graph>{
        ?subj sch:prop ?obj .
        ?subj a sch:CorrectClass .
    }
}WHERE{
    GRAPH <http://example/temp-graph>{
        ?subj src:prop ?obj .
        ?subj a src:WrongClass .
    }
}
```

Listing 5.3: Example SPARQL query to select and transform data between graphs.

The second use case is obtaining data from remote SPARQL endpoints (Section 5.2.2). The endpoint services are queried over HTTP using the *Federated Query* mechanism. The results can then be inserted into the local knowledge-base. This is shown in Listing 5.4 by the inclusion of a service URI to signify the remote SPARQL endpoint.

```
PREFIX sch: <http://example.org/schema#>
PREFIX src: <http://example.org/source#>

INSERT{
    GRAPH <http://example/final-graph>{
        ?subj sch:prop ?obj .
        ?subj a sch:CorrectClass .
    }
}WHERE{
    SERVICE <http://example/remote-endpoint>{
        GRAPH <http://example/temp-graph>{
            ?subj src:prop ?obj .
            ?subj a src:WrongClass .
        }
    }
}
```

Listing 5.4: Example SPARQL query to select and transform remotely held data from a service.

In conclusion, datasets retrieved or loaded into the local knowledge-base can be transformed using SPARQL queries. This allows the preparation of the dataset in advance of execution and so enables usage of a common schema by modules (Chapter 4). These SPARQL queries can be distributed for re-use to support the transformation of published datasets into a schema, e.g a module developer could publish the transformations required to prepare a well-known dataset for their module.

It has previously been identified that a configuration of the framework is to use only remote data sources without a local knowledge-base. Since the schema of modules and data sources could vary there needs to be a method for transforming or reconciling the data. The proposed solution to this is discussed in the next section.

## 5.3 Controlling and Executing the Modules of the Framework

An objective of the framework is to allow the platform-independent interchange of modules. The framework has been designed to fulfil this objective without requiring that the modules all adhere to a strict interface, but instead interact through the data of the knowledge-base. Further, the modules can be implemented in different programming languages, e.g. Java and Python, and executed on different computing platforms or physical computers. Therefore, it is intended that a diverse range of modules can be utilised with minimal barriers to implementation and access.

The framework achieves these features through an RDF structure and leveraging the SPARQL protocol [43]. The RDF structure, termed the *Framework Configuration*, provides a directory of services, graphs, modules and queries. This structure identifies where the modules can retrieve data for their execution and replacement queries supplied by the user. Therefore, multiple versions of the *Framework Configuration* can re-use the same set-up of parameters for multiple iterations. Alternatively, the same scenario parameters can be applied to different knowledge-bases of the contextual data, e.g. geography, demography and infrastructure, as the user chooses.

The SPARQL query protocol is used to allow the user to retrieve data using standard querying syntax or modify the execution of modules. The *Federated Query* [43] component of the protocol provides access to remote and local datasets over HTTP using standardised syntax. Therefore, the framework does not variate the SPARQL standard, but extends it through a small set of requirements for modules to implement.

The conceptual structure of the framework is shown in Figure 5.5. This abstract view does not take into consideration the physical configurations that the framework facilitates as discussed previously (Section 3.4). The *Framework Configuration* interacts with the modules that provide the functionality of the travel demand process. It provides the modules with the *Service Definitions*, *Module Definitions* and *Query Definitions* that are used to locate data sources, select data and call other modules to perform discrete functionality.

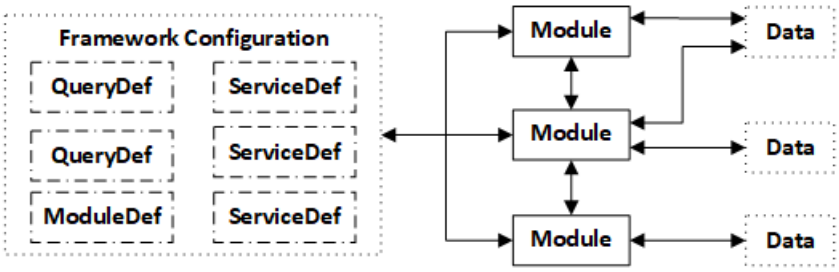


Figure 5.5: Diagram of the framework structure using the RDF data model of the Framework Configuration to exchange information with modules and data graph.

This section describes the *Framework Configuration* by examining the data structures and processes required to support it. The premise of the *Framework Configuration* is providing the user with control over the configuration by allowing them to select modules and mediating any schema misalignments that may occur between modules. This creates the prospect of users providing incorrect information through badly formed queries. Therefore, there is also consideration of the selected mechanisms for ensuring the validity of data and queries which are passed between modules at the direction of the user.

### 5.3.1 Framework Configuration

The *Framework Configuration* schema is shown in Figure 5.6. Each instance is described by properties to the *Service Definition*, *Query Definition* and *Module Definition*. This configuration information can be stored in the knowledge-base as a separate graph or kept in a single graph together with the other parameter data for the *Travel Scenario* and the execution results (Sections 4.5 and 4.9).

In addition to controlling the configuration during execution, this information may be of use in post-execution analysis or the reconstruction of the investigation by other users, e.g. as part of reproduction studies. The *Framework Configuration* also provides a central reference for associating any global data that a particular module may require, e.g. additional configuration data for a traffic simulator. An optional *Framework Service* property allows the URL HTTP service on which the *Framework Configuration* can be retrieved. This allows the whole graph of configuration to be passed from service to service using only two references: the

*Framework Configuration* URI and its service.

### 5.3.2 Service Definition

The *Service Definition* describes where data can be retrieved relating to a particular part of the core schema. Each *Service Definition* includes a *Service Type* that identifies which parts of the schema are satisfied by the *Service Definition*. Additional *Service Types* may be defined by modules if their data requirements are broader than the core schema. The user would fulfil these requirements in the knowledge-base and then signpost to them using the *Service Definition* of the *Framework Configuration*.

It was discussed previously (Section 4.9) that the knowledge-base can be divided into multiple logical graphs to separate concepts or alternative sets of data. The *service URI* indicates the address of the SPARQL endpoint where the data is located. The *graph URI* indicates which *graph* within the endpoint holds the required data. This allows two *Framework Configurations* to point to the same service and retrieve different versions of data, e.g. Year 1 and Year 2. Alternatively, the two *Framework Configurations* could point to different services and retrieve their alternative versions of the same data, e.g. Year 1 from Service A and Year 1 from Service B.

The user may decide to organise their knowledge-base following their own graph structure or physically separate the graphs onto different computers each operating a different endpoint. The *Service Definition* permits this to take place without restriction. The *Service Type* identifies which sets of data are satisfied by each *service URI* and *graph URI* pair. The module will seek the *Service Type* it requires without concern for the underlying organisation.

In the most simple configuration a single knowledge-base would have a single graph containing configuration, scenario and results data. Similarly, while a module may distinguish two areas of the data as being separate, e.g. person and vehicle data, the user may decided to place them in the same graph, as they will not interfere with each other. Therefore, the *Service Definition* may refer to multiple *Service Types*, which are using the same *service URI* and *graph URI*.

This is shown in Figure 5.7 by the "Data" *Service Definition*. In this example,

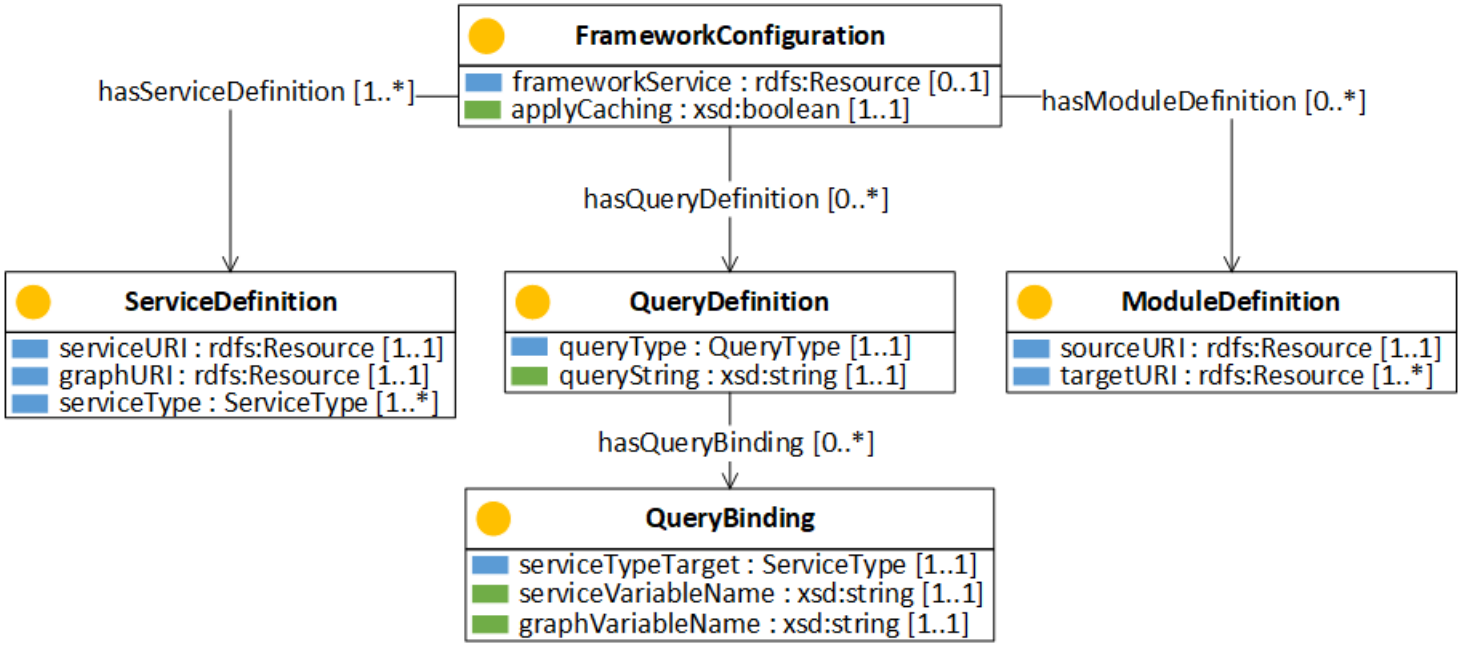


Figure 5.6: Schema of Framework Configuration.

two *Framework Configurations* are defined, which use the same knowledge-base for the persisting data, i.e. data that is not influenced by parameters of the scenario, and then identify different "Scenario" and "Results" *Service Definitions*. Therefore, the parameters of the scenario are being varied, but not the demographic, network infrastructure or land use data.

This separation into four parts is recommended as the minimum division, although a single *Service Definition* would be a valid configuration. This recommendation is based upon partitioning the scenario parameter from the persisting data, so that multiple iterations and variations can be executed, and placing the results of the scenario in a separate graph, so that they can be easily removed, e.g. if an error occurred during set-up or execution.

It was previously outlined (Section 4.9) that six graphs were used to organise the prototype knowledge-base and modules (Chapter 6). Therefore, six *Service Types* were defined in the *Framework Configurations*. However, these were design decisions and not a mandatory requirement. Implementing modules are able to state the *Service Types* they require and the user would then satisfy them through the knowledge-base and the *Framework Configuration*. The lack of mandatory requirements relating to Travel Demand Modelling means that this approach could be used in other contexts outside of travel demand generation.

#### 5.3.2.1 Service and Graph Query Manipulation

The *Service Definition* enables the user to identify the *services* and *graphs* that they wish to use in the framework. These definitions are utilised by applying the parameters to template SPARQL queries. This section discusses the mechanism developed for manipulating these template queries. This mechanism is executed by modules but is designed to be a generic re-usable component that modules can access as a library. Each module may perform multiple queries and retrieve the *Service Definitions* it requires based on the *Service Type*.

The *Service Definition* identifies the *service* and *graph* URIs available for each *Service Type*. These services and graphs contain the datasets necessary to execute queries and obtain results. The initial use case would have all the information contained in a single graph on a single service for the query to retrieve.

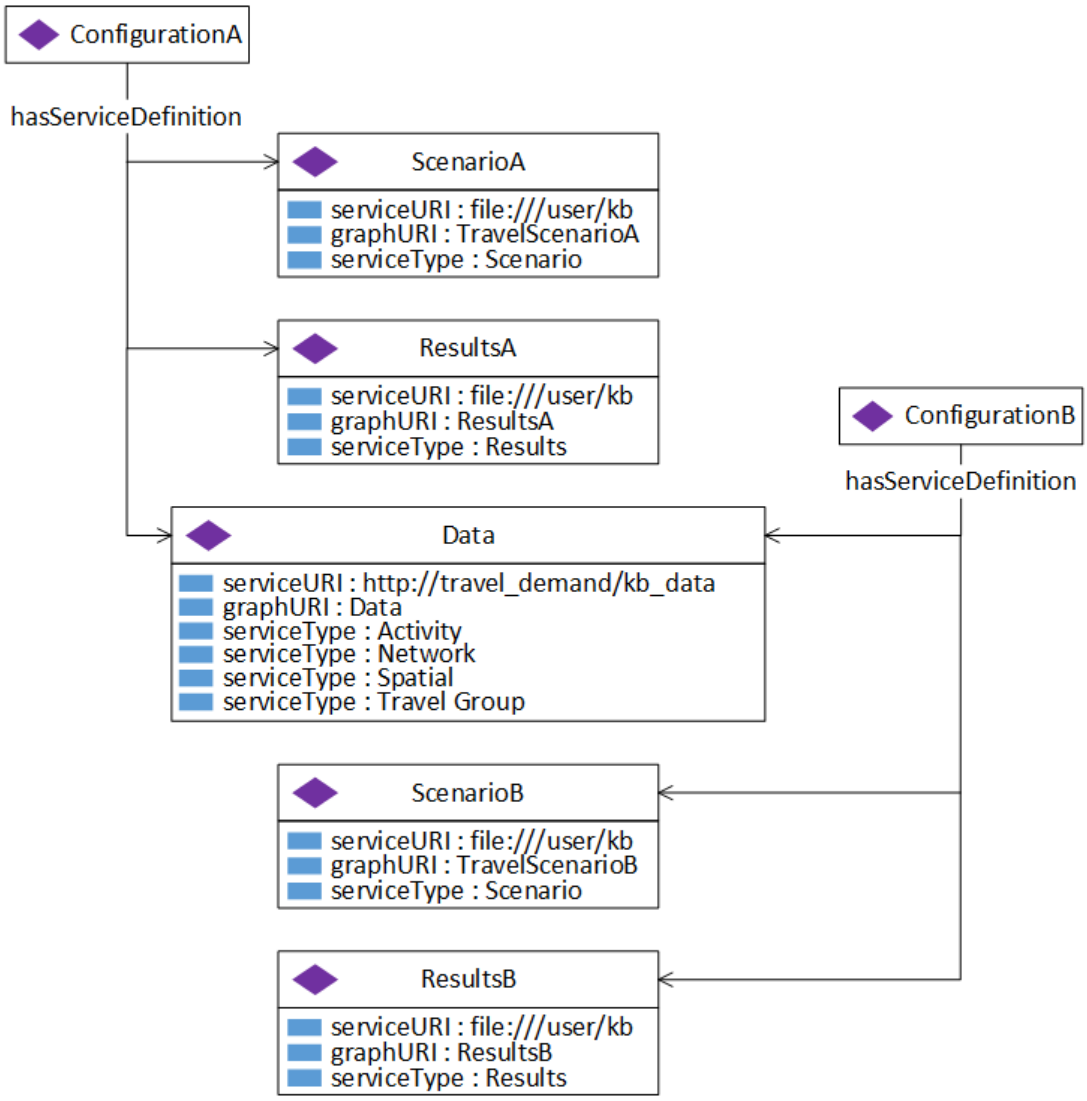


Figure 5.7: Diagram of two example Framework Configurations for services using local scenario and results with remote knowledge-base.

A broader scenario would find data spread across multiple services and multiple graphs. The graph structure of RDF and the SPARQL protocol *Federated Query* enable the drawing together of this data without needing to execute multiple separate queries. This removes the need for implementation specific programming code to handle the multiple stages of obtaining and passing the results between separate queries.

The need for multiple stages would increase complexity and reduce the flexibility for users in specifying replacement queries to suit their knowledge-base or configuration as discussed later (Section 5.3.3). The *Framework Configuration* enables the user to specify alternative configurations so that different services and graphs, i.e. alternative datasets, can be used for each execution of the framework.

The mechanism is based upon queries being written as text strings which are interpreted at run-time and therefore can be manipulated prior to execution. The Listing 5.5 shows an example query template. A *target* variable has data retrieved for three properties: *hasName*, *hasLabel* and *hasValue*. As *target* is unbound the results will be for every *subject* that has these three properties.

Each property is expected to be retrieved from a different remote source specified by enclosing within *SERVICE* and *GRAPH* clauses. These clauses have an accompanying identifier, shown in this case within square brackets *[...]*. The identifiers are specified by the module and are substituted with the service or graph URI from a *Service Definition* according to the *Service Type*.

```
PREFIX sch: <http://example.org/schema#>

SELECT ?target ?name ?label ?value
WHERE{

    ?target sch:hasName ?name .

    SERVICE [labelService]{
        GRAPH [labelGraph]{
            ?target sch:hasLabel ?label .
        }
    }

    SERVICE [valueService]{
        GRAPH [valueGraph]{
            ?target sch:hasValue ?value .
        }
    }
}
```

Listing 5.5: Example SPARQL query template with identifiers for service and graph.

Figure 5.8 shows an example *Framework Configuration* that could be applied to this template query. Each definition specifies a different service URI and graph URI and *Service Type*. These are applied to the template to produce the query as shown in Listing 5.6.

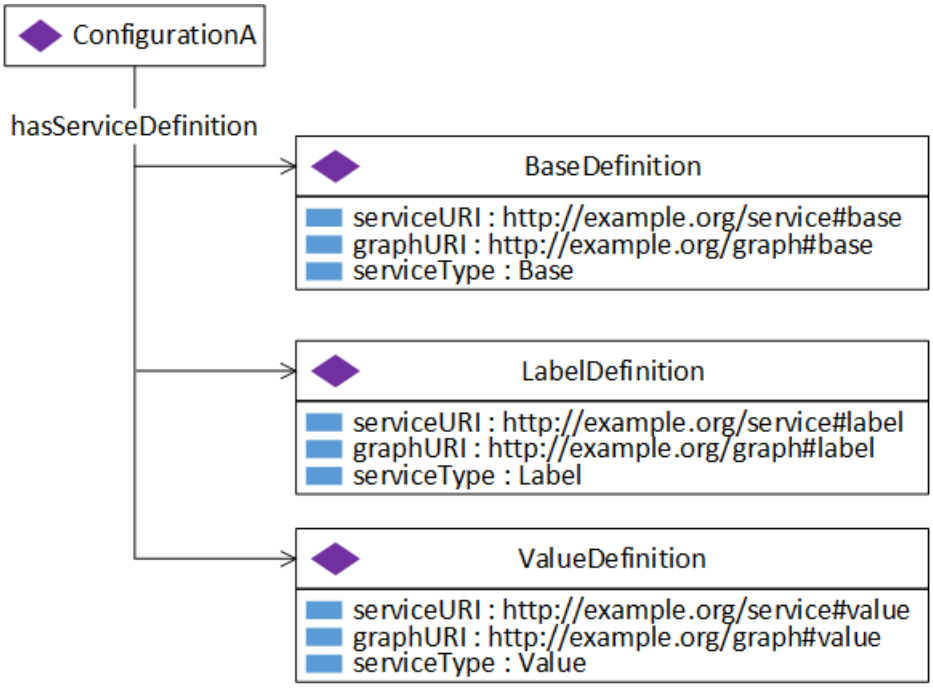


Figure 5.8: Diagram of example Framework Configuration for use in query manipulation.

The SPARQL protocol specifies additional requirements for valid queries. Each query is executed in the context of a base service and graph. The base service is not included as part of the SPARQL query. The base graph is identified by the *FROM* clause which specifies the default graph. Any additional named graphs are stated by *FROM NAMED* clauses. These statements must be inserted between the *SELECT/CONSTRUCT/ASK/DESCRIBE* and the *WHERE* clause. Therefore, these clauses must also be inserted when named graphs are being used.

```
PREFIX sch: <http://example.org/schema#>

SELECT ?target ?name ?label ?value
FROM <http://example.org/graph#base>
FROM NAMED <http://example.org/graph#label>
FROM NAMED <http://example.org/graph#value>
WHERE{

    ?target sch:hasName ?name .

    SERVICE <http://example.org/service#label>{
        GRAPH <http://example.org/graph#label>{
            ?target sch:hasLabel ?label .
        }
    }

    SERVICE <http://example.org/service#value>{
        GRAPH <http://example.org/graph#value>{
            ?target sch:hasValue ?value .
        }
    }
}
```

Listing 5.6: Example SPARQL query prepared for execution with substituted service and graph URIs.

It is not a requirement that the graphs of a knowledge-base are separated over multiple services. In a simpler configuration a single service may provide all or the majority of the required graphs. Therefore, comparison is made between the service URI of the base *Service Type* and the service URI of additional *Service Types*. When the two service URIs are identical the *SERVICE* clause, including surrounding braces {...}, is removed. Listing 5.7 demonstrates this scenario by only specifying a single *SERVICE* clause.

```
PREFIX sch: <http://example.org/schema#>

SELECT ?target ?name ?label ?value
FROM <http://example.org/graph#base>
FROM NAMED <http://example.org/graph#label>
FROM NAMED <http://example.org/graph#value>
WHERE{

    ?target sch:hasName ?name .

    GRAPH <http://example.org/graph#label>{
        ?target sch:hasLabel ?label .
    }

    SERVICE <http://example.org/service#value>{
        GRAPH <http://example.org/graph#value>{
            ?target sch:hasValue ?value .
        }
    }
}
```

Listing 5.7: Example SPARQL query prepared for execution on a single service.

Another scenario is that the knowledge-base has not been separated into the same number of graph as the module originally defined. Instead several graphs have been merged together, i.e. a *Service Defintion* with multiple *Service Types*. The simplest framework configuration would be a single graph on a single service.

In the case of merged graphs, the *graph* URI of the additional *Service Type* is checked against the *graph* of the base *Service Type*. If they are identical then the *GRAPH* clause, including surrounding braces {...}, is removed. This is illustrated in Listing 5.8 where the first service and graph clauses have been removed to retrieve the data from the base service and graph. The second service and graph clauses have been substituted with the relevant URIs.

```
PREFIX sch: <http://example.org/schema#>

SELECT ?target ?name ?label ?value
FROM <http://example.org/graph#base>
WHERE{

    ?target sch:hasName ?name .

    ?target sch:hasLabel ?label .

    SERVICE <http://example.org/service#value>{
        GRAPH <http://example.org/graph#value>{
            ?target sch:hasValue ?value .
        }
    }
}
```

Listing 5.8: Example SPARQL query prepared for execution on single service and graph.

The procedure for the query manipulation mechanism is described in Algorithm 1. The string *query* is modified during the process with clauses being added and removed. The *Service Definitions* are retrieved from the *Framework Configuration* and stored in an associative array termed *serviceDefs*. This information may be re-used across multiple queries within each execution of a module but also across multiple executions, as discussed later (Section 5.3.5).

The query specific information are provided by the *identifiers* and *base type* variables. The *identifiers* associate specific service or graph clauses with a *Service Type*. The *base type* identifies the contextual *Service Type* that the query is being applied within.

Applying this mechanism gives the user control over the organisation and provision of data whether from local or remote sources. The modules are able to use the SPARQL query protocol to obtain data in a decoupled manner from

the underlying knowledge-base organisation. These queries can also be made available for replacement by the user, as discussed later (Section 5.3.3), to give further flexibility in the data retrieval and processing.

**Algorithm 1** Service and Graph Query Manipulation

```
function QUERY MANIPULATION(query, serviceDefs, identifiers, baseType)
    baseServiceDef ← serviceDefs.get(baseType)
    baseService ← baseServiceDef.serviceURI
    baseGraph ← baseServiceDef.graphURI

    query ← INSERTFROMCLAUSE(query, baseGraph)

    for identifier in identifiers do
        serviceID ← identifier.serviceID
        graphID ← identifier.graphID
        type ← identifier.serviceType

        serviceDef ← serviceDefs.get(type)
        service ← serviceDef.service
        graph ← serviceDef.graph

        if service equals baseService then
            query ← REMOVESERVICECLAUSE(query, serviceID)
        else
            query ← INSERTFROMNAMEDCLAUSE(query, service)
            query ← REPLACESERVICECLAUSE(query, serviceID, service)

        if graph equals baseGraph then
            query ← REMOVEGRAPHCLAUSE(query, graphID)
        else
            query ← INSERTFROMNAMEDCLAUSE(query, graph)
            query ← REPLACEGRAPHCLAUSE(query, graphID, graph)
    return query
```

#### 5.3.2.2 File and HTTP Service URIs

A final consideration of the *Service Definition* is the use of *file* URIs. The SPARQL *Federated Query* protocol is based upon using the HTTP protocol to query a SPARQL endpoint, either located locally within a network or remotely online. When using the framework this may present an unnecessary burden both in configuration and processing overhead.

The URI definition permits the use of HTTP and File as schemes [128, 129]. These schemes identify the method to access the resource the URI describes. This has been incorporated into the framework design and is shown in Figure 5.7 by the *service URI* for the "Scenario" and "Results" *Service Definitions* of both configurations.

The simplest use case of the framework is executing a programme against a knowledge-base stored on the local file system. Requiring the use of the HTTP protocol would mean always establishing a HTTP endpoint for the file system knowledge-base. Semantic Web libraries can support this process, but it is an additional configuration step.

The User Application would be required to separate execution of modules from the knowledge-base and run the HTTP SPARQL server. This would impose requirements upon the user and could delay the investigative stage for no benefit. There is also a computational overhead introduced by a local HTTP route that can be avoided using a local file system knowledge-base. An illustrative outline of the steps needed for the local HTTP route is shown below with the equivalent step for a local file system shown in **bold**:

1. SPARQL query is converted by the User Application into a HTTP request.
2. HTTP request routed through the network adapter to the SPARQL endpoint.
3. HTTP request converted back into SPARQL query.
4. **SPARQL query is executed by the SPARQL endpoint.**
5. SPARQL query results converted into HTTP response.

6. HTTP response routed through the network adapter to the User Application.

7. HTTP response converted back into SPARQL query results.

Handling of File URI is not a feature of the SPARQL standard which is based upon HTTP URIs for *Federated Queries*. Therefore, File URI cannot be used in SPARQL queries. This is due to the interpretation of the File URI, i.e. the contents of the indicated file or folder, being implementation dependent and reliant upon the Semantic Web library of the User Application (Section 3.4).

The framework accepts the use of File URI by applying several restrictions. First, that the folder or file pointed to by the File URI can be accessed by the Semantic Web library executing the query. In the context of a local closed system this is a reasonable assumption as most applications will use a single Semantic Web library. Second, when the base service is a File URI then the query is not executed as a remote request, but executed on the local knowledge-base using the Semantic Web library. Third, that a sub-service using a File URI can only be used when the base service is a File URI. Fourth, that a sub-service using a File URI must align with the base service.

In more complex configurations there may be a mix of local file and remote HTTP services. A Semantic Web library compliant with SPARQL *Federated Query* can be executed on a local knowledge-base and will retrieve the data from the remote sub-services within the query. However, any sub-services also using a File URI must have the SERVICE clause removed as the SPARQL protocol does not interpret the File URI. Therefore, the data for these File URI services is retrieved from the local context, i.e. the base service File URI. This means that all the File URIs used in a single query must match and a File URI cannot be used for a sub-service if the base service is a HTTP remote.

When considered for compatibility with the query manipulation mechanism discussed previously (Section 5.3.2.1), the File URI is being used as an identifier rather than a resource locator, i.e. the data is **not** located on a HTTP service. The user application would already have used the File URI to locate the knowledge-base and access it using the Semantic Web library. This means the approach to File URIs is compatible with the query manipulation mechanism and

enables the same queries to be used on different configurations of the framework (local and remote) by only changing the configuration parameters.

The restrictions established by applying this process can be summarised as:

- File URI support is dependent upon the Semantic Web library of the user.
- Only a single File URI can feature in a single query.
- A sub-service can only use a File URI if the base service uses a File URI.

Overcoming these design restrictions to support multiple File URI would require variation to the SPARQL standard or enhanced features of Semantic Web libraries. This kind of variation to the SPARQL standard is highly unlikely as a Semantic Web design principle is platform independence, which file system knowledge-bases introduces. The enhancement of Semantic Web libraries is also unlikely as the underlying SPARQL standard has not been changed and different libraries could adopt incompatible approaches. In both cases support of multiple File URIs represents a move away from the benefits of using standards based technologies of the Semantic Web for a subset of use cases.

The previously outlined query manipulation (Section 5.3.2.1) can be implemented by a module without modifying the Semantic Web library it utilises. The mechanism is applied to the text of the query and should be straightforward for modules to implement, or use a library developed for the proposed framework. Therefore, providing additional implementation details for the framework to address the multiple File URI is undesirable.

This means that some framework configurations using multiple File URI, i.e. multiple local knowledge bases, are not supported. However, these configurations can be supported by all but one one of the local knowledge-bases being set-up as a local HTTP server. Their *service* URIs will then be HTTP URIs and the stated restrictions would be met.

This may have the consequence of slower execution run-times in this specific use case. However, travel demand generation and traffic simulation are not real-time processes and so faster execution run-times are desirable rather than critical. The user would also still have the choice of simplifying their configuration by consolidating the multiple local knowledge-bases into a single instance and applying

named graphs to ensure data separation (Section 4.9). The benefit of re-using the same queries across multiple configurations can therefore be fully realised in the proposed mechanism.

### 5.3.3 Query Definition

The objective of the framework is to provide the user with flexibility in the configuration of modules, retrieval of data and execution of the modules. This section discusses how the latter two objectives can be facilitated by enabling users to rewrite the queries executed by modules. The queries are written in SPARQL syntax which conforms to the published standard [43]. Therefore, the semantics and vocabulary of the queries are clearly defined. The queries are interpreted at runtime as text strings meaning that modifications can be applied or transmitted without requiring modification or recompilation of modules.

Users with understanding of the underlying knowledge-base can re-write the queries to retrieve alternative pieces of data for a module. Similarly, sub-modules can be called to perform additional or alternative processing of the data within the module as *property functions* using standard SPARQL syntax. The only requirement is that the *SELECT* and *CONSTRUCT* variables are unaltered and bound so that the executing modules can retrieve the expected data from them.

The *Query Definition* provides for these replacement queries to be defined as part of the *Framework Configuration*. This allows the replacement queries to be accessed by local or remote modules; ensures the full configuration is stored within the knowledge-base; and facilitates re-use across multiple configurations. Figure 5.9 provides an example of two *Framework Configurations* which each utilise two *Query Definitions*.

In this example there is one common *Query Definition* and one specific to each configuration. When modules are executing a query then a look up is performed to check if the required *query type* is defined in the *Framework Configuration*. If present then the replacement query is used and if absent then the default query is executed.

The modules will only need to state the *query type* URI to identify relevant *Query Definitions* and provide an example query for users to modify, i.e. the

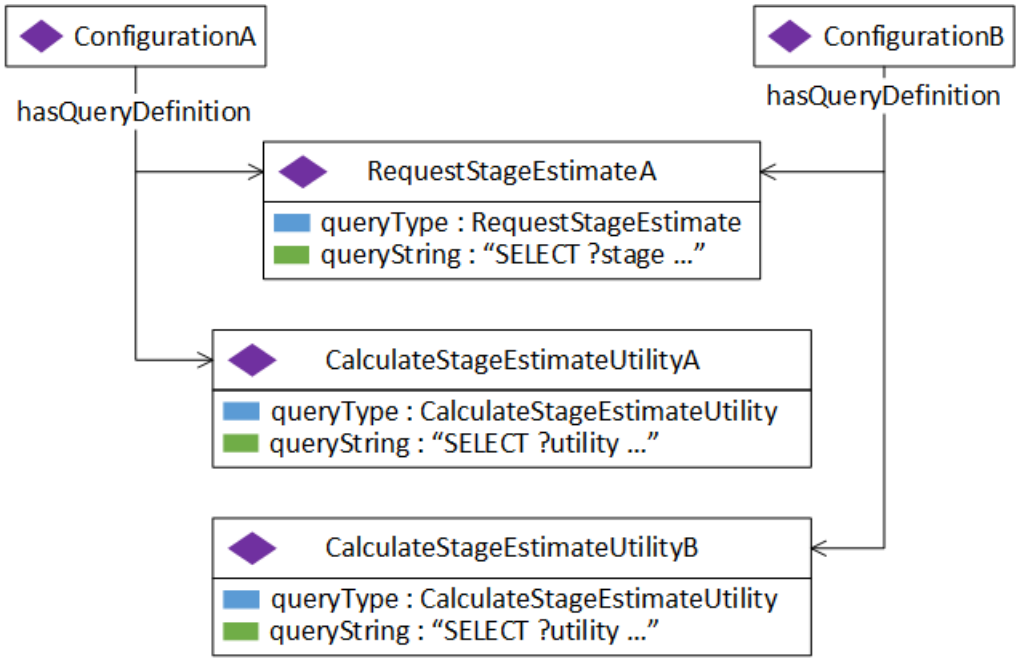


Figure 5.9: Diagram of two example Framework Configurations for query using the same module to request stages but different calculations of utility.

default query that the module already executes. The queries used by modules can be considered to serve the purposes of data retrieval and sub-module execution, e.g. requesting a stage estimate during the trip planning phase. By publishing either of these types of query a module will be providing greater control to the user.

The former type of queries will allow a module to be adapted to a new schema in the knowledge-base by the user rather than the alternate case of transforming the data in the Knowledge-Base Construction phase (Section 5.2.3). Another use case is the modification of calculations and equations, e.g. utility in discrete choice models (Section 1.2.4.2).

Publishing these queries by the module developer would be recommended, but not mandatory as the user can still utilise the module by complying with the published schema. There may also be a large quantity of these queries used by a module which are trivial in nature and a burden to publish. In addition, design decisions and assumptions for a module could make modification of certain queries problematic, although compliance with the core schema should assist

in preventing this (Chapter 4). However, by not publishing these queries, and exposing the module's data retrieval process, a requirement is placed on the user to conform to the module's expected schema.

The latter type of queries provide control over the configuration of the framework. It is through these queries that modules are accessed and so how alternatives can be selected, e.g. a replacement sub-module for the default sub-module. Therefore, these types of queries should be mandatory for publication so that users can have control over the framework configuration to select alternative functionality.

In both cases the user may wish to redirect part of the query to one or more services and/or graphs. The user could place *SERVICE* and *GRAPH* clauses with explicit URIs within the query. However, this would remove the flexibility for changes to be applied during execution according to the *Framework Configuration* when different datasets or modules may be explored.

The user would have to manually edit each change in the *Query Definition* for each *Framework Configuration*; both time consuming and prone to error. Therefore, the *Query Binding* structure provides a cross-reference from the query defined in the *Query Definition* to the existing *Service Definitions* in the *Framework Configuration* (Section 5.3.2.1).

The user can then include additional service and graph placeholder variable names in the query. Alternatively, they can overwrite the default bindings for existing placeholders used by a module. When the query is processed these variable names are substituted with the corresponding *service* or *graph* from the *Service Definition* with the matching *service type*. Any variable names can be provided by the user allowing as many cases as they require within a query and across multiple queries. Additional *Service Definitions* can be included in the *Framework Configuration* beyond those required by the executed modules allowing further flexibility to the user in how data is retrieved.

The approaches described enable the externalisation of the data retrieval, calculation and sub-module selection performed by modules. These processes can then be adapted and modified by the user to their configuration of data and modules as opposed to being restricted to configurations with compliant interfaces and data structures.

### 5.3.4 Module Definition

A feature of the framework is that the user is able to provide alternative queries using the *Query Definition* as described previously (Section 5.3.3). This introduces the need for validating the query against the schema of the knowledge-base as discussed later (Section 5.3.7). A scenario this introduces is the user providing a query that replaces the URI of a sub-module in a query with another module's URI in order to use its alternative implementation or functionality.

A step of the query validation process is checking whether the *property* URIs of a query are contained in the knowledge-base's schema. Otherwise the defined triple will never yield a result, if the knowledge-base schema fully describes the contained data, and making attempts to execute the query redundant. The URI of modules are used as *properties* in queries but do not have to be included in the schema, which describes the shape of the data.

A module can know its own URI and those of the default sub-modules that it utilises and so these can be included and recognised during the query validation process. However, replacement module URIs included by the user as part of a *Query Definition* will be unknown and cause a validation error. The *Module Definition* provides a mechanism for the user to declare the module which is having its default sub-modules replaced and the replacement sub-module URIs.

These replacement sub-module URIs will feature in the new queries defined in the *Query Definitions*. When the query is validated the *Module Definition* can be checked for the sub-module URI. The *Module Definitions* are only required if the user changes the sub-module within a query from the default expected by the module. Since a query could potentially include multiple sub-modules the *Module Definition* can define multiple target sub-modules URIs.

### 5.3.5 Caching of Invariant Data

The storage and processing of large quantities of data has led to the development of databases. The RDF graphstore, that would be expected to store any knowledge-base of size, and other non-SQL databases have been developed to provide this functionality as alternatives to the traditional relational database. These databases can provide persistent storage on the file system with features

including querying and parallelism.

Once data has been loaded and prepared the persistent file storage allows very large datasets to be accessed. Optimisations have been developed, e.g. indexing and query caching, to assist in the retrieval of data. However, database retrieval is still slower than an application retaining data in-memory.

The in-memory data is already in the required form for the application and does not require retrieval from the file system and processing. While computer memory capacity has continued to develop, they can still be relatively small in size compared to the scale of datasets. This can inhibit an entire dataset being retained in-memory by an application. Therefore, there is a compromise between large scale, but slow access, databases versus the small scale, but quick access, of in-memory storage.

A similar issue is found with the HTTP requests over a network used by the proposed framework to access remote datasets. These HTTP requests themselves are relatively slow to perform as discussed previously(Section 5.3.2.2) and their reduction or removal would lead to improved execution run-times. The discussed approach with supporting local file system references can remove some HTTP requests when modules and data are set-up locally. However, the removal and reduction of computation will enhance the efficiency of any system.

Another consideration is the manner in which SPARQL queries and property functions, which execute and wrap the modules, are resolved by SPARQL query engines. A property function is called for each potential solution to a query with the parameters for each case passed for processing in isolation. The differences between these cases may be a single parameter value. This can mean that a query with many thousands of cases could be retrieving from the graphstore the same or very similar data for every case.

Finally, graphstores can experience reductions in query performance as the dataset size increases and with sub-optimal queries. The greater the number of triples present in the dataset and the less specific the query then the greater the cross-product of cases that could be legal query solutions. Separation of knowledge-bases into multiple graphs can reduce the number of triples being searched and query optimisation by ordering of steps to reduce candidate cases early can improve response times. However, not performing redundant querying

will always be quicker. In each of the outlined situations the caching of invariant data in-memory can provide part of the solution. The cached data can be repeatedly accessed to remove the need to access the graphstore, process HTTP requests and obtain data for usage between cases.

It has already been discussed (Section 4.9) that the data in the knowledge-base belongs to a variety of contexts. These contexts include invariant data that will not be changed by the state of the modelling process, e.g. the location of a building, while other data may undergo change, e.g. person and vehicle positioning. This does not mean that the state of objects in the knowledge-base will not change over the long term, but that they will not be adjusted in the time-frame of the modelling process, i.e. a building on a site may be demolished between different dataset years, but not over the course of the day or week of traffic demand generation.

Another context to consider is the data that changes between modelling executions, e.g. framework and scenario parameters. These parameters distinguish one execution from another. However, they are invariant within the modelling process. Therefore, the values do not require retrieval and updating once they are already known by a module. These types of data can be identified by modules, extracted from the knowledge-base and retained for use from case to case during an execution.

Overall, the user is able to construct a large knowledge-base containing contextual data for the concepts, while the modules are able to extract and retain targeted data for faster execution. This means that the user does not need to extract and format a subset of the data for the module to utilise or compromise the richness of the knowledge-base.

The lack of variability in framework and scenario parameters means that *Framework Configuration* instances can be used as a reference to allow multiple sets of cached data to be retained, identified when needed and discarded when no longer used. A number of configurations for the framework have been discussed (Section 3.4) including local and remote execution of modules.

In both cases it would be expected that the in-memory capacity will be limited as with any resource. In the local execution this is not a concern as modules will likely be executed for short periods of time and on a single *Framework Configu-*

*ration* instance. Therefore, cached data would be discarded once the execution has completed, i.e. the User Application exits and releases its resources.

In remotely executed modules there would be continuous availability as a service. The service would be available for long periods of time and expected to execute multiple *Framework Configuration* instances. Each instance could have different data associated and so must be logically discreet from other instances, e.g. scenario parameters may differ. Therefore, there is potential for accumulating large quantities of data in the cache.

The physical limit to in-memory caching can be increased by operating system funcationality, e.g. paging to disk, and caching to disk libraries, [130, 131]. However, retaining cached data indefinitely is a waste of resources as once the execution has completed a *Framework Configuration* instance will not be used again, or if it is can incur the initial pre-caching cost, while periodic restart of a service is not ideal.

The proposed solution is for the cached data to expire if it has not been utilised for a period of time. The travel demand generation typically occurs as a continuous process with households and persons iterated through in rapid succession. Therefore, a period of seconds or minutes would indicate that the process is completed and the cached data is no longer required.

The framework design is based upon modules being provided with a *Framework Configuration* instance in order to retrieve the relevant service and query information. Each *Framework Configuration* instance identifies the scenario, knowledge-bases and current results. Therefore, modules can check the cache for the *Framework Configuration* instance, retrieve the data and update the most recent request log.

Once cached data is no longer being requested it can be discarded. A service under heavy load would be able to limit cache sizes to not exhaust in-memory capacity, stop caching when full, but expect cache space to released as the load is completed.

The *apply caching* property has been included in Figure 5.6 to enable users to specify a choice as part of the global configuration. However, it would be a module implementation decision whether to honour or ignore the parameter. There may not be relevant data which is deemed invariant or there is insufficient

resources for caching.

This caching solution allows the execution costs of database access, HTTP requests and redundant SPARQL querying to be mitigated without wasteful use of resources or requiring additional data to be passed to modules. The effectiveness of this solution is evaluated in Chapter 7.

### 5.3.6 Ensuring Validation and Conformance of Data to the Schema

The effectiveness and accuracy of any model or system is heavily influenced by the quality of the input data. The proposed framework is developed upon the provision of data from publishers which is then acted upon by the consuming modules. Both parties must ensure that the data conforms to the schema that they publish or utilise. This section outlines several areas that this is particularly important in the Semantic Web and identifies an available solution.

The technical details of this solution are not discussed here but it provides an existing mechanism for providing assurance in data conformance. Low quality and non-compliant data will lead to errors in the produced results and can prevent successful completion of any computational process. This is especially true of the Semantic Web due to several factors:

- The SPARQL query process matches triples to the template graph patterns in the query. When a set of triples do not match a graph pattern it does not result in an error but instead that the particular set of triples should be ignored. Data which is malformed, such as incorrect property URIs or missing properties, will not cause a warning or error, but instead be silently ignored to return less or no results. The inverse also applies where a SPARQL query is not correctly aligned to the underlying data (Section 5.3.7). Therefore, minor typographical errors in either stage will produce unexpected results. This in contrast to SQL where a mismatching key/column during data loading or querying will produce an error.

- The open and online retrieval of data from multiple sources also presents a problem. The user does not control these data sources or the data they

publish. Therefore, the quality and compliance is beyond their control. There will inevitably be errors, omissions and variations between different versions. The AAA principle (Section 4.2.1) also allows publishers to include properties of their own choice, which could conflict with a user's schema or vary away from the standard schema a user was expecting.

- The graph structure of RDF makes human inspection and validation of data difficult. In a tabular structure, data is held in rows and columns that can be inspected for gaps or malformed entries. Additional columns can easily be identified. In RDF, the triples relating to a single subject may be dispersed throughout a file and the description of a single concept may be spread over a chain of triples. There may be multiple occurrences of a *subject-property* when only one should be present or none when at least one is required.

A user needs to establish confidence in the knowledge-base and modules prior to utilising them. Otherwise, there can be little confidence in the resulting output until it has been thoroughly validated. Therefore, an automated solution is necessary to ensure that a dataset complies with the intended schema. The user will then be informed of the non-compliance and can take steps to resolve them.

The Shapes Constraint Language (SHACL) [95] is a Semantic Web standard technology that has been developed to provide automated validation and reporting. The schema is composed of RDF triples that express the expected shape and constraints of the data. This shape can be described with a variety of characteristics, including:

- the presence of properties;
- the frequency of properties;
- the datatypes of literals;
- class membership;
- and the values and ranges of data.

Rules can also be expressed to apply inferences to the data to produce new triples. The approach applies Closed World Assumption (Section 4.2.1), i.e. that no new data exists that would invalidate the inferences or outcome, and does not apply the Non-unique Name Assumption (Section 4.2.1), i.e. entities with different names cannot be the same entity.

This is in contrast to RDFS [101] and OWL [93] schema languages. These have been primarily designed to derive inferences from the data according to the schema, rather than enforce data conformance. If the data and those inferences are logically inconsistent then it is reported as being invalid for the schema. However, the cause of the invalidity may not be apparent to the user.

The available terms in RDFS are very focussed and so the expressiveness is narrow. The OWL language uses Open World Assumption (Section 4.2.1), applies the Non-unique Name Assumption and is primarily used for classification. This means that OWL may not produce the outcomes which a user may expect and is not broad enough for all data validation purposes. For example, a missing property when the schema states a minimum cardinality of one is not invalid as the property *may* exist in the open world. The absence of a statement is typically assumed to mean that the statement is false, but in OWL it is concluded that the statement *may* be true or false.

The SHACL triples are encoded as RDF and therefore can be shared as part of a schema. The schema can then be applied to the data when constructing the knowledge-base. When the schema and data of the knowledge-base are processed by a SHACL engine any violations can be identified for rectification. This can also be applied to models retrieved from remote data sources using SPARQL *CONSTRUCT* queries to ensure that the correct structure is being obtained.

In summary, there is a technological solution available for automated data validation within the Semantic Web. This solution can be incorporated into User Applications, datasets and modules to ensure that the data produced is compliant with any required schema. This schema can be customised according to the needs of the user and modules using a standardised set of properties, which provide both data validation and inferencing.

### 5.3.7 Ensuring Validation and Conformance of SPARQL Queries to the Schema

The functionality of permitting users to supply their own queries through the *Query Definition* raises the issue of query validation in SPARQL. A closed system has queries written by developers with detailed understanding of the schema and opportunity for thorough testing. An open system can accept queries from users who lack detailed understanding and do not have time to thoroughly test queries. Errors originating in user input should be discovered as early as possible [132] to prevent queries being processed that ultimately fail, after wasting resources undertaking the processing, or can return unexpected and incorrect results.

An advantage of SPARQL querying is the flexibility in allowing the user to structure queries and retrieve data of their own choice. Therefore, a technical solution to address a user's lack of understanding about the schema of the knowledge-base, i.e. predicting the user's true intention from a malformed query, is difficult. However, identifying typographic and logical errors within queries would be a useful assistance and save resources for both users and developers of the framework and implementations of SPARQL generally.

In an SQL database the schema is stipulated by the designer through the structure of table fields/columns and no additional fields/columns can be defined. This means that queries can be validated to identify those fields which are incorrect at the outset, either through user mistyping or misunderstanding of the schema, i.e. a column in the table must exist for it to contain any data. The SPARQL standard is based upon the graph of data in the knowledge-base being schema-free and does not contain such a checking mechanism, i.e. the absence of a triple match could be an incorrect query or the lack of matching data in the knowledge-base.

In SPARQL, the *property* of a triple is analogous to a field/column in SQL. The multiple statements in the graph patterns of the query are matched to the data. The graph structure of the data is walked along for each case until a mismatch to the graph pattern is found. A case which completes the graph pattern is returned as a result.

Following the AAA principle (Section 4.2.1) and no schema, a *property* which

has been mistyped in a query is considered to be different to the correctly typed *property* present in the data. This can also be applied to *class* names. It is instead considered that the user (*Anyone*) has made their own statement (*About Anything*).

There will not be a binding to the statement in the graph pattern so it is likely all cases will fail and the query will return no results. This lack of results is interpreted that there are no matches for the query in **this** dataset, but there could be in another dataset, rather than an error.

Similarly, SPARQL queries can contain named variables which are bound to the data and re-used in later parts of the graph pattern and/or returned as results. These variable names are only defined within the context of the query. Mistyping a variable name will result in two variables when only one was intended.

When the data is being walked in the graph pattern these variables are bound to all possible cases for the triple and then later rejected by other statements. A mismatch between two variable names will result in cases not being rejected and more of the dataset being explored than was required or intended.

It has also been identified that SPARQL implementations have focussed upon grammatical checking of queries [133]. These grammatical checks identify when keywords are mistyped or functional requirements cannot be fulfilled, e.g. variables named as a result, but are not included in sorting or grouping statements. They do not pro-actively ensure that queries are valid for logical or schema constraints and instead reactively error and fail during query processing.

These logical and schema constraints have been categorised into *syntactic* and *semantic* validation [133]. The identified syntactic rules cover several cases including positioning errors, e.g. a *literal* being used as a *subject* or *property*, and filter conditions using *literals* of different data-types. The semantic rules are formed into an OWL ontology to use inferencing to check for logical consistency in the query.

It has previously been discussed that the execution of SPARQL queries can lead to the inefficient repetition of actions on invariant data (Section 5.3.5). This situation also applies to the queries defined by a *Query Definition* in a *Framework Configuration*.

Once a query has been validated its content will not be changing and repeated

validation would be a waste of resources. Instead the outcome can be cached for re-use according to the *Framework Configuration* and *Query Type* URIs. Therefore, a successful query can be repeatedly executed, while a previously rejected query can signal a swift termination.

The following sections will examine the validation of the query URIs and variable names for incorporation into the framework. This validation seeks to ensure that the potential exists for meaningful results to be returned by a query rather than the actual results. The inclusion of more general SPARQL validation errors [133] has been partially implemented on internal module queries to assist with module development, but is an area of future work.

#### 5.3.7.1 Validation of Query Unique Resource Identifiers

The previously identified issue of mistyped *property* and *class* URI is included in the category of *syntactic* errors and can be resolved by checking to ensure queries only contain URIs explicitly contained in the schema. However, in a *Federated Query* the schema of the target service would need to be known for a check to be carried out.

This is possible if the target service, i.e. modules and knowledge-bases, are following the same core schema (Chapter 4). However, the framework has been developed to tolerate local variations in modules and knowledge-bases by the user providing their own *Query Definitions* (Section 5.3.3). These *Query Definitions* may be the source of such errors through typographical errors or misunderstanding variations in schema between modules and knowledge-bases.

To manage such errors a module could execute the query and then check the outcome of the remote service's validation. This is undesirable as it requires reacting to failures after the execution has taken place. A single execution of a query could contain multiple cases which will all result in errors. Therefore, resources are wasted in reaching an outcome that could be predicted by validating the query.

Alternatively, the module could request the schema from the remote service and perform local query validation before executing the query. However, SPARQL is designed to be schema free and there is no defined mechanism to request a

schema. An enhancing feature of the framework could perform this request, but it would create an extra burden on implementing the framework. Knowledge-bases complying with the SPARQL standard, but not set-up for the framework, would not be able to perform validation. This would either preclude them from being used, and so not achieve the framework objective of remote data retrieval, or limit the application of query validation.

A third option is to use the SPARQL *ASK* or *DESCRIBE* keywords to query the remote service for the relevant graph pattern within the *SERVICE* clause. The explicit *class* and *property* URIs can be extracted from the graph pattern and then queried against the module or dataset. An *ASK* query provides a boolean response to whether a solution exists to the query graph pattern. A *DESCRIBE* query provides an RDF graph response about one or more resources, which include *property* URIs, with the precise response being implementation dependent. Obtaining confirmation that the *property* URI is recognised and is an instance of RDF *property* would meet the above requirement for error identification.

This is illustrated in Listing 5.9 which shows the specific targeting of a *property* URI and the retrieval of all. These can be targeted at a specific graph. Therefore, either keyword could be applied with the *ASK* providing the more direct check, but potentially requiring multiple HTTP requests, while *DESCRIBE* allows the checking of multiple instances or characteristics.

These checks will identify whether the explicit *class* or *property* is contained in the remote service, either in the data or schema. It does not resolve variables or SPARQL syntax as this requires execution of the query.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

ASK{
    <http://example.org/schema#propA> a rdf:Property .
}

DESCRIBE ?prop ?class
WHERE{
    ?prop a rdf:Property .
    ?class a rdfs:Class .
}
```

Listing 5.9: Example SPARQL query for ASK and DESCRIBE keywords to confirm execute a remote module through its Property Function.

#### 5.3.7.2 Validation of Query Variable Names

The matching of variable names can have a range of implications for a query. The connection between statements in the graph pattern will not be made and so extraneous results can be included. A newly encountered variable name will obtain all instances that match the triple. If the variable name is mistyped then numerous instances will be obtained when the true variable name would have resulted in one or a few instances. This error can also affect the optimisation of queries, performed by Semantic Web libraries, which seek to arrive at results quicker by targeting those triples that limit the result set most effectively.

A mismatch in variable names between the *CONSTRUCT* template and *WHERE* body will also cause those triples to not appear in the returned graph, implying no data for those triples. There can also be a mismatch between the variable names in the body of a query and the specific bindings of *BIND* and *VALUES* clauses, which would result in the whole dataset being queried rather than a targeted subset.

The incorrect naming of variables can also cause grammatical errors as key-

word clauses, such as *GROUP BY* and *ORDER BY*, must include all variable names that are included in the *SELECT* clause. While the variable names in these keyword clauses may align, and so pass grammatical validation, there can still be a misalignment to the variable names in the *WHERE* body, which would not be detected.

These variable names only exist in the context of the query and therefore variations in schema are not an issue. However, there is a challenge in identifying variable names which have some similarity, and so are potentially intended to be the same, but not so dissimilar that every pair of variable names is reported.

The variable names themselves can be groups of characters, words, parts of words and compounds formed from multiple words or parts of words. The variable names are case sensitive and so mistyping using name conventions, such as *Camel Case*, can result in multiple variables , e.g. "myVar" is different to "myvar". The inclusion of a numeric character is also acceptable, e.g. "var1" and "var2". A variable name is permitted to only be placed and used once in the *WHERE* clause and not in other query clauses to indicate that any value is permitted, i.e. a test of another part of the triple.

These factors mean that any candidates for correction can only be highlighted as a *warning* for the user, rather than an error, as similar variable names may be the user's intention, e.g. an alphanumeric suffix for variable names. Since variable names do not have to be words the application of spell checking using an approved list is inappropriate.

The grammatical error of a variable name in the *GROUP BY* or *ORDER BY* clause not being matched in the *WHERE* body would be an *error*. Similarly, applying the assumption that a user would only provide in-line data in the *VALUES* clause for use in the query means any unmatched *VALUES* variable names would be an *error*. An exception to this is when a variable name is used in the *SELECT* clause and the *VALUES* clause, but not the *WHERE* body, to directly bind data for the results.

A number of distance metrics have been proposed for measuring the similarity between character strings [134], including Levenshtein distance, Jaro-Winkler metric, Jaccard similarity and Hamming Distance. These metrics examine the strings from a variety of perspectives and produce varying numerical values to

quantify the level of similarity. The Levenshtein distance measures the edit distance, i.e. number of insertions, deletions or substitutions, required for two strings to match with a zero being an exact match.

This metric has been used in the implemented Algorithm 2 to identify variable names that are similar, i.e. less than three edits, but not exact, i.e. greater than zero edits. The threshold of three was selected to permit the insertion or deletion of up to two characters or the transposing of a single pair.

A minimum threshold is applied to ensure that only strings which are long enough to have a greater similarity than difference are checked, i.e. four characters. Otherwise short words or single character variable names, which are commonly used in SPARQL queries, would be flagged. Similarly an edit distance of two is only reported for strings of equal lengths, to identify transposing. The two edit distance cases of two insertions or an insertion and a substitution are treated as being dissimilar strings.

The query is provided as a parameter from which are obtained the variable names in the *WHERE* keyword clause and candidate names from the result keyword clauses, i.e. *SELECT*, *CONSTRUCT* or *DESCRIBE*. An exception to this are any variable names that are bound in the *SELECT* clause due to aggregation of results, e.g. the result of a *COUNT* function. These specifically bound variable names will not feature in the remainder of the query and so will never be matched.

The variable names of the query *WHERE* clause are iterated through and compared to the set of variable names. The equality of names is checked first to provide an early exit for those strings with a distance of zero.

A check is then performed to ensure the length of both strings is sufficient for the edit distance checking. Strings which are too short are still added to the list of candidates as they may provide future matches for variable names. The result of the *Levenshtein Distance* is checked for the one and two edit distance cases, as illustrated in Table 5.1.

In the one edit case a further check is made into whether the initial or final characters of equal length strings are matching. When not true the edit is in the remainder of the strings and can be reported. An exception is made for characters that only differ by case, e.g. "varA" and "vara" are reported. Otherwise the edit

distance is ignored as being an enumeration with a prefix or suffix.

This could potentially under report errors as there is no checking to ensure a consistent sequence of enumeration, but it could also be considered restrictive to enforce sequential numbering. The enumeration is also limited to a single character, i.e 1-9, a-z and A-Z, and therefore would report two variables using more than two characters as similar, e.g. "var1" and "var10". Resolving these reporting edge cases is an area of future work to investigate potential naming conventions and the appropriate rules for enforcement.

In the two edit case a check is made to ensure the strings are equal in length and whether the edits are the transposing of adjacent characters through two substitutions. The two edit cases which result in two insertions, two deletions, non-adjacent substitutions and non-transposing substitutions, i.e. more than two character values, are ignored as being too dissimilar.

When an exact match is found the variable name is added to a set of matched names. A name without an exact match is added to the set of candidate variable names to allow comparison between variable names inside the *WHERE* clause.

Once all the names have been checked the other keyword clauses of the query, i.e. *VALUES*, *ORDER BY* and *GROUP BY* and the original result clause, are checked to ensure all of their names have a matching name in the *WHERE* clause. Otherwise these variable names would never be bound to values. The *VALUES* clause is checked for matches against the other clauses' variable names for the case when in-line data is being directly bound to them.

**Algorithm 2** Check Variable Name

```
procedure CHECK VARIABLE NAME(query)
    list reports ← empty
    editThreshold ← 3
    set names ← query.where.varNames
    set candidateNames ← query.result.varNames
    set matchedNames ← empty

    for name in names do
        isMinLength ← name.length > editThreshold
        isMatched ← false
        for candidateName in candidateNames do
            if name = candidateName then
                isMatched ← true
                break loop
            if isMinLength and candidateName.length > editThreshold then
                dist ← LEVENSHTEINDISTANCE(name, candidateName)
                isReport ← false
                if dist = 1 then
                    pos ← FINDEDITPOSITION(name, candidateName)
                    if pos = −1 or(pos ≠ 0 and pos ≠ name.length − 1) then
                        isReport ← true
                else if dist = 2 then
                    isReport ← CHECKTRANSPOSE(name, candidateName)
                if isReport then
                    report ← CREATEREPORT(name, candidateName, dist)
                    reports.add(report)
        if isMatched then
            matchedNames.add(name)
        else
            candidateNames.add(name)

    unusedReports ← REPORTUNUSEDNAMES(query, matchedNames)
    reports.add(unusedReports)
    return reports
```

| Variable Name | Candidate Name | Edit Distance | Reported | Comment |
|---|---|---|---|---|
| x | x | 0 | No | Variable and candidate names are identical. |
| x | y | 1 | No | Variable and candidate names are too short. |
| x | xx | 1 | No | Variable and candidate names are too short. |
| x | xxx | 2 | No | Variable and candidate names are too short. |
| x | xxxx | 3 | No | Variable name is too short. Edit distance is too great. |
| sub | subj | 1 | No | Variable name is too short. |
| subj | sub | 1 | No | Candidate name is too short. |
| subja | subjA | 1 | Yes | Final character only varies by case. |
| asubj | Asubj | 1 | Yes | Initial character only varies by case. |
| subjarea | subjArea | 1 | Yes | Single character substitution. |
| subjBrea | subjArea | 1 | Yes | Single character substitution. |
| subjA | subj | 1 | Yes | Single character deletion. |
| subj | subjA | 1 | Yes | Single character insertion. |
| subjA | subjB | 1 | No | Only final character varies, i.e. enumeration. |
| subj1 | subj2 | 1 | No | Only final character varies, i.e. enumeration. |
| aSubj | bSubj | 1 | No | Only first character varies, i.e. enumeration. |
| asubj | bSubj | 2 | No | Two character substitution, not transposed. |
| subjA | subjAA | 1 | Yes | Single character insertion. |
| subj1 | subj10 | 1 | Yes | Single character insertion. |
| objA | subjA | 2 | No | Single character insertion and single character substitution. |
| subj | sujb | 2 | Yes | Transposed adjacent characters. |
| jubs | subj | 2 | No | Transposed non-adjacent characters. |
| suja | subj | 2 | No | Non-transposed adjacent characters. |
| juba | subj | 2 | No | Non-transposed non-adjacent characters. |

Table 5.1: Table of edit distance and validation outcome between variable and candidate names.

### 5.3.8 Reporting the Schema Data and Query Validation

The framework is developed upon Semantic Web design principles of an open network. Information is transferred between modules and knowledge-bases of the framework with customisation by the user. The previous sections have outlined the mechanisms available and proposed for ensuring that the data being produced and consumed is valid and to ensure that queries have the potential to produce meaningful results.

Once these validation steps have been performed it is necessary to report back to the user the outcome so that remedial action can be taken. If the validation is performed by the User Application then this reporting could be made directly available to the user. However, if the validation has been undertaken by a module then it cannot be directly reported. There may be multiple levels of modules between the validating module and the User Application.

The outcome of the validation process can instead be sent to an additional *Service Definition* defined in the *Framework Configuration* with the express purpose of receiving these reports. The User Application can then check for these reports at each stage of the execution and convey them to the user.

The modules positioned between the reporting module and the User Application can also check for errors reported by their sub-modules and abort their execution. The inclusion of this validation reporting has more general usage as a means for modules to also report other information that may assist the user, e.g. policy, execution errors or additional meta-data, without it being included with the results of executing the framework.

The data structure for capturing the data and query validation reports is shown in Figure 5.10. The structure has properties for a text summary of the validation results, e.g. the variable names or URI found to be invalid in a query, and whether the result constitutes an advisory warning or a critical error.

Additional properties are defined for the identified subclasses of query and data validation errors. *Data Validation Results* provide specific references to the data source through service and graph URIs. *Query Validation Results* identify the URI of the invalid query used in the *Framework Configuration*. Each *Query Validation Result* also indicates the result type so that further background information into the cause can be found.

Examples of these types as applied in the implementation are shown in Figure 5.11. Further subclasses and properties could be included to provide greater detail or coverage for other validation errors.

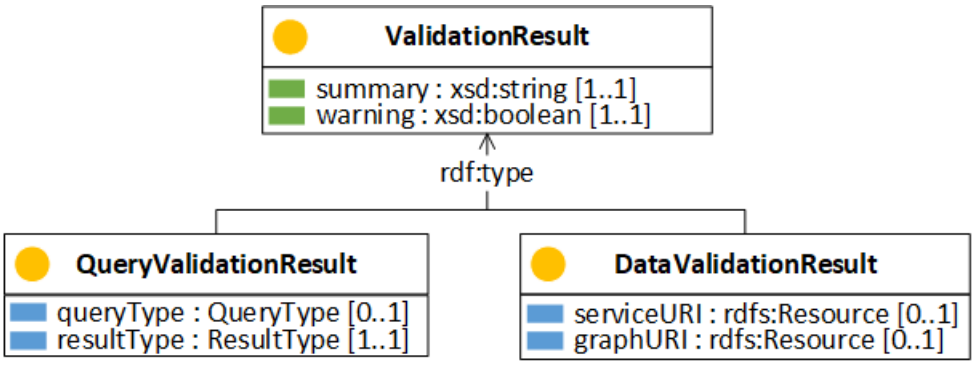


Figure 5.10: Schema for Validation Result.

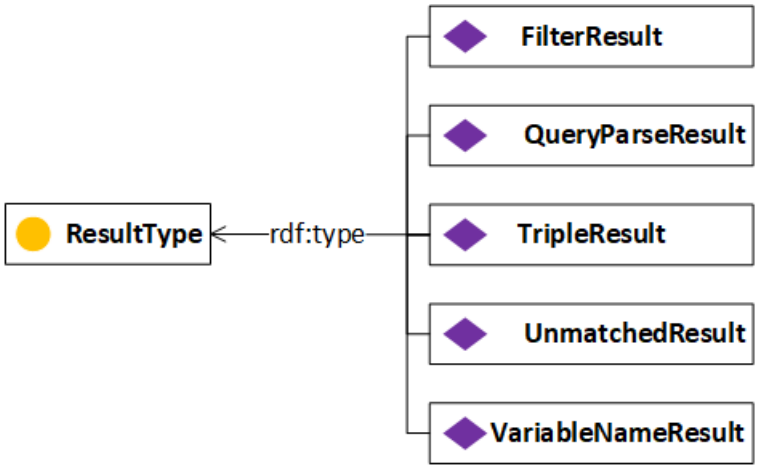


Figure 5.11: Diagram of example schema for validation Result Types.

### 5.3.9 Executing the Framework in Local and Remote Configurations

It has been discussed in the previous sections that the framework can be organised into several configurations. These configurations represent the local and remote access of data endpoints, which form the knowledge-base, and modules. The identity and access of these endpoints and modules is controlled through referencing *service* and *graph* URLs stored in the *Framework Configuration*.

This presents an issue in how the content of the *Framework Configuration* is accessed. The data endpoints are unidirectional and do not require any information contained in the *Framework Configuration*. They simply respond to HTTP requests for data. However, the modules of the framework require access both to the *Service Definitions* and *Query Definitions* to operate. In a local configuration the local knowledge-base can be searched. Yet in a remote configuration the executing modules would not be able to locate the content of the *Framework Configuration*.

The *Framework Configuration* URI provides a unique identifying reference that could also be used as service reference to publish online, or on the local network, the knowledge-base storing the *Framework Configurations* content. However, this would require setting up a different URL for each execution of the framework and would quickly introduce an administration and technical burden.

An alternative is to provide each module with a service reference URL that can be re-used across multiple *Framework Configurations*. Therefore, the published knowledge-base would contain multiple *Framework Configurations*. This removes the technical burden of configuring multiple URLs. This published knowledge-base could also contain other scenario information which the *Framework Configurations* reference and can be used as the destination for the results of the modelling process.

The modules of the framework are defined in SPARQL queries through *Property Functions*. These *Property Functions* accept a variable number of positional arguments. The *Framework Configuration* URI is required by all modules in either local or remote configuration. Therefore, this should be the first parameter of each module *Property Function* as shown in Listing 5.10. The example query is being executed with the module and the knowledge-base containing the *Framework Configuration* in the same local context and so there are no service URLs in the query.

```
PREFIX ex: <http://example.org/example#>
PREFIX mod: <http://example.org/module#>

SELECT ?result
WHERE{
    ?result mod:propFunc(ex:FrameworkConfigA ?arg1 ?arg2).
}
```

Listing 5.10: Example SPARQL query to execute a local module through its Property Function.

The execution of modules which are not in the same context as the knowledge-base containing the *Framework Configuration* requires the service URL. This is shown in Listing 5.11. The service clause to access the remote module through a *Federated Query* is specified while the service URL is passed as the final argument of the module's *Property Function*.

The local knowledge-base, containing the *Framework Configuration*, would need to be accessible as an endpoint to respond to the HTTP requests that the module will send. The other services defined in the *Framework Configuration* could optionally point to other knowledge-bases set-up remotely as endpoints. These queries could be expanded to obtain data for the module's parameters or to act upon the results it provides.

```
PREFIX ex: <http://example.org/example#>
PREFIX mod: <http://example.org/module#>
PREFIX ser: <http://example.org/service#>

SELECT ?result
WHERE{
    SERVICE ser:remote-endpoint{
        ?result mod:propFunc(ex:FrameworkConfigA ?arg1 ?arg2 ex:
            ↪ serviceURL).
    }
}
```

Listing 5.11: Example SPARQL query to execute a remote module through its Property Function.

### 5.3.10 Altering the Execution Flow of Modules

The previous sections have addressed how the user can control the modules and knowledge-bases which are accessed by modifying the *Service Definitions*. The user also has control over the data that is retrieved and its manipulation through the *Query Definitions*. This section outlines how the execution flow of the framework can be modified.

The modules of the framework can call other modules to perform sub-tasks. The implementers of the main module may intend that only a single sub-module is interfaced, but the user has determined that they wish to use multiple modules, as shown in Figure 5.12. This could be due to each providing different functionality e.g. alternative behaviour models, or to investigate a specific sub-set of individuals in the data.

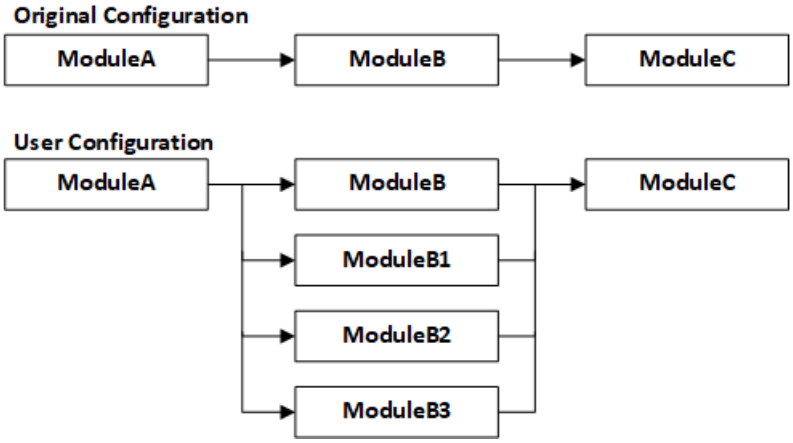


Figure 5.12: Diagram of alternative configurations of modules during framework execution.

In a conventional modular system, such as MATSim [32], the user would need to rely on the main module providing a mechanism for substituting the sub-module. The user would then need to develop, or rely on a developer to produce, a wrapping module that integrates with the main module's interface and then performs the selection logic to their choice of multiple sub-modules.

Unless a developer was prepared to produce a generic and configurable wrapping module then each user would have to perform this process for their own investigation. This development would need to take place using the system's platform and design approach, which may vary between systems and so require the user to develop multiple skill sets. In each case an investment of time and resources is required from users that delays the investigation, introduces potential error and is likely duplicating the efforts of other users.

In the framework, this re-configuration can be achieved by modification of the SPARQL query, through the *Query Definition*, which is used at the boundary

between modules. The main module developer would publish the default query and schema for the data that the sub-module is expected to produce in the knowledge-base for the main module. The modified query would include a *union* clause to select alternative choices based upon the data as shown in Listing 5.12.

```
PREFIX mod: <http://example.org/module#>
PREFIX ex: <http://example.org/local#>

SELECT ?person ?route ?value
WHERE{
    {
        ?person a ex:Quartile4Income .
        ?route mod:routingMethodA (?person ?start ?end).
    }UNION{
        ?person a ex:Employee .
        ?person ex:income ?income .
        FILTER(?income > 50000)
        ?route mod:routingMethodB (?person ?start ?end).
    }UNION{
        ?person a ?quartileIncome .
        FILTER(?quartileIncome IN (ex:Quartile3Income ex:
            ↪ Quartile2Income))
        (?route ?value) mod:routingMethodC (?person ?start ?end).
    }UNION{
        ?person ex:maxPrice ?maxPrice .
        ?route mod:routingDefault (?person ?start ?end ?maxPrice).
    }
}
```

Listing 5.12: Example SPARQL query to select different routing modules based on class, data property filtering, list of classes and default option.

This example shows alternative routing modules being selected according to the income of the individual. The user can apply a variety of selection methods such as classification, value filtering, list of values and a default option as shown.

Each module can be defined according to its own required parameters without regard for the other modules in the query, as shown in the final option which has four arguments or the third option which returns two arguments.

In this example, a URI is returned that can be used to retrieve additional properties related to the process, either in this or a separate query. A convention is applied here for modules using *Property Functions* to use the triple's subject as the output and the object for input, but this can be reversed as required by the module design. The modules can apply different design paradigms and implementation choices. Data transformations can be included by the user in the query if necessary to assist integration. The user is able to control the execution flow using only the SPARQL language, which they used to set-up the knowledge-base and framework, and pass data to modules implemented on different platforms.

The graph pattern in each clause are tested to ensure the statements are true or a match is found in the data; if no match is found that pattern and clause is closed without result. The *union* will only produce a single result from the multiple clauses. Therefore, the clauses are evaluated in the order they are defined and the first which contains true statements will be returned as the result. The remaining clauses will not be evaluated and the user will need to determine an order of precedence if the query statements do not select the cases on a mutually exclusive basis.

A module that has identified multiple branches to sub-modules within its design would be able to use the *Query Type* mechanism to allow the user to supply a different query to retrieve data for each branch. These alternatives would be encoded as any other *Query Definition* in the *Framework Configuration* and retrieved by the module during execution. Therefore, both the user and implementer can explore options for alternatives and introduce more diverse modelling of behaviour. The singular approach to behaviour modelling has been a criticism of travel demand models as discussed previously (Chapter 1).

## 5.4 Requirements of the Framework

In this chapter and previously in Section 3.4 there has been detailed discussion of the framework's design and operation. This has included several design points

and requirements to facilitate and improve operation. These requirements are summarised below:

- *Modules* shall define *property functions* that accept a *Framework Configuration* URI as the first parameter.
- *Modules* shall define *property functions* that accept an **optional** final parameter as a service URL. This service URL will identify the service providing a named graph containing the *Framework Configuration* URI and its properties.
- When a service URL is provided, the service URL will be queried for the *Framework Configuration* and its properties using the *Framework Configuration* URI as the graph name.
- When a service URL is **not** provided, the local *knowledge-base* shall be queried for the *Framework Configuration* and its properties using the *Framework Configuration* URI as the graph name.
- *Modules* shall be configured by the properties of the *Framework Configuration* URI. These properties shall define *Service Definitions* and *Query Definitions*.
- *Service Definitions* will identify the *service* and *graph* properties where data shall be retrieved by *Modules* according to a *Service Type*.
- *Modules* **must** publish the *Service Type* URIs it requires to function and identify the expected schema for those *Service Types*.
- *Service Definition service* property may consist of either File or HTTP URI. All *Service Definitions* for a *Framework Configuration* must use the same File URI.
- *Service Definition graph* property must be a URI for a named graph.
- *Framework Configurations* shall only refer to one *Service Definition* of each *Service Type*.

- *Query Definitions* shall provide alternative SPARQL queries to replace the *Module's* default queries according to a *Query Type*.
- *Modules* **must** publish the *Query Type* URI and SPARQL queries that interface with other *Modules*.
- *Modules* **may** publish other *Query Type* URI and SPARQL queries that it utilises.
- *Framework Configurations* shall only refer to one *Query Definition* for each *Query Type*.

## 5.5 Security of the Framework

A final important consideration of the framework is its security. It has been discussed that the framework can be utilised in a local and remote environment. In a local environment, there are limited security concerns as the user has control over the data being utilised, all processes are run upon the local hardware and the framework is not accessible externally through online services. Therefore, users can check the data and processes being applied and control the burden upon hardware resources.

The remote environment approach can also be utilised across an internal network to distribute the resource requirements across multiple computers, but again security concerns are minimal. Accessibility to services can be constrained to only local network addresses. Therefore, a security concern arises if the local network is not secured and has been compromised, which represent wider issues than the framework.

In a remote environment a number of security concerns can be identified. Private and personal data should not be freely accessible. Hostile users may seek to perform *denial of service* attacks by overburdening services through high volumes of requests or requests designed to take excessively long to resolve. These hostile users may also seek to corrupt the data by adding, modifying or deleting triples.

The data being utilised by the framework, i.e. the travel demand generation process, is generally derived from public or synthesised data. Therefore, there should not be private data in the dataset, either due to it being anonymised or engineered for public usage. However, the extendible nature of the knowledge-base would permit a user to add a dataset that contains the core schema that is further enriched with personal information, e.g. names, addresses, dates of birth.

Mitigating the risk of this private data accidentally being exposed online can be achieved by placing the sensitive data in a separate graph, as discussed in Section 4.9, as part of a separate knowledge-base instance that is not externally accessible. The current SPARQL protocol controls access on a knowledge-base basis rather than individual graph basis.

This approach creates a logical and physical separation while still allowing cross referencing through the URI resource of the individuals. The user would be able to access the additional characteristics for analysis or other uses by using federated SPARQL queries to the private knowledge-base. When the configuration uses only a local knowledge-base then all the data can be contained in the same knowledge-base. Therefore, the user can ensure data protection without needing to introduce new security concepts or mechanisms to the process.

The SPARQL protocol [43] utilises the Hypertext Transport Protocol (HTTP). The HTTP protocol has access control and authentication mechanisms [135] so that only those identified as safe parties will receive responses to their requests. The HTTP protocol also provides secure communication through cryptography [136] to prevent interception and modification by third parties.

These mechanisms ensure that both participants (requester and responder) can have confidence they are communicating with the intended trusted party. The SPARQL protocol also permits services to place restrictions upon size, number and frequency of requests that it receives so that resources are not dominated by a hostile or naive user, e.g. during a *denial of service* attack.

The integrity of a user's data can be assured by the separation of read-only query services from read-write update services that allow insertion and deletion of data. Therefore, the input data of the process, e.g. population data, can be published by a user or public data-source as a read-only SPARQL endpoint to prevent a hostile user from modifying its contents.

Finally, the framework also proposes passing SPARQL queries between modules and services. These queries may have been rewritten by the user. This presents an attack vector where malicious steps are included in the query. However, this can be prevented by reviewing the query prior to processing to ensure it is syntactically correct and does not perform malicious behaviour, e.g. additional insert or delete commands.

Each query is a text file interpreted at run-time rather than a piece of executable code. Therefore, its contents can be automatically scrutinised for areas of concern prior to processing upon the knowledge-base. The available operations and their usage are restricted by the SPARQL protocol which limits opportunity for abuse. The need for sanitising SPARQL queries exists for any SPARQL endpoint and is not an issue introduced by the framework. Therefore, supporting Semantic Web libraries may already include features to prevent this attack vector.

This issue of query manipulation is also not unique to SPARQL endpoints and needs to be addressed for any system that utilises database querying [137]. The advantage of the SPARQL protocol is that it has been designed for use in an online open environment, rather than SQL protocols which were established on the basis of a controlled closed environment. In conclusion, there are no identified security concerns arising from the framework which cannot be handled using existing mechanism of the SPARQL protocol.

## 5.6 Chapter Summary

This chapter has discussed the different configurations which can be achieved by utilising the proposed framework. There has been examination of the different methods for obtaining source data and how establishing a knowledge-base can assist in retrieving and integrating this data. This includes direct importation from local sources, transformation of the data and retrieval from remote sources to reduce the need in developing data converters and module interfaces.

By developing a core schema for the travel demand model issues of misaligned data can be reduced. Published datasets or SPARQL endpoints which orientate themselves around an agreed schema can be more readily consumed and utilised. However, this does not have to be enforced and the framework supports adapta-

tion by the user through the modification of module queries.

There has been consideration of issues arising from user and developer error or misalignment in these queries. It is proposed that these can be resolved by performing data and query validation. This validation can utilise existing techniques but there has been the need to develop solutions to specific issues. By applying these solutions the risk of incorrect results or wasted execution effort can be reduced. A mechanism is also established to provide feedback to the user when problems do occur.

The proposed framework can support local, remote and mixed configurations. The accessing of these configurations is achieved using the HTTP support provided by SPARQL federated queries, but also allows direct access to file system knowledge-bases to provide efficiency and simplify set-up. It has been identified that a restriction exists in configurations with multiple local knowledge-bases, but users have a number of options to mitigate this issue and still meet their requirements.

There has also been consideration of the security of the framework and mitigating steps that can be taken if necessary. The leveraging of the SPARQL technology, which is designed for online utilisation, provides some protection from known threats and reduces the risk of issues developing.

The use of SPARQL is applied throughout the framework to construct, transform, redirect and execute scenarios. This provides a single language that is platform independent so that users do not need to learn multiple programming languages and can apply their developed skills repeatedly. The framework does not introduce any variation to the SPARQL standard and is instead an extension of its language and principles. Therefore, the barrier to using the framework is lowered and potentially requires a narrower skill set than a conventional solution for the travel demand generation process.

The requirements of the framework have been established with no domain specific requirements identified. It is put forward that the framework provides a general solution for accessing and configuring modular solutions for other problems. The further development of the framework would seek to develop, or expand upon existing, mechanisms for the discovery and negotiation of remote services of modules and datasets to assist the user in the configuration process.

It is proposed that the developed framework forms a contribution to the field. This approach to supporting the travel demand generation process has not been identified in conventional frameworks and differs from other examined Semantic Web approaches for accessing remote services by providing control of the selection, alignment and redirection of services and data through RDF data and SPARQL query. This provides the potential for the travel demand generation process to readily access online datasets; align and link those datasets around coherent and consistent concepts; permit access to modules independently of their platform; and produce platform-agnostic travel demand data.

# Chapter 6

# Implementing the Travel Demand Generation Framework

## 6.1 Introduction

This chapter discusses the prototype implemented to investigate the travel demand generation and traffic simulation concepts discussed in Chapter 4 and the design of the framework for controlling the execution of modules discussed in Chapter 3. It seeks to address research question RQ4 by examining the technical design choices for implementing the framework modules; the configuration of the framework and knowledge-base; and the design features of the prototype. There is discussion of the technical details upon which the implementation is built. This is followed by description of the organisation and configuration of the framework and knowledge-base. Finally, there is design explanation of the implemented modules.

## 6.2 Implementation of Prototype Framework Modules

The prototype was implemented using the Apache Jena Semantic Web API [138] in a Java environment. In the course of developing the prototype a number of contributions were made to the Apache Jena Semantic Web API, Java API and

SUMO traffic simulator projects (see Appendix A).

A Semantic Web API provides a library for utilising Semantic Web technologies. Utilising a library provide a robust foundation to allow development resources to focus upon the prototype. Several APIs are available, across several programming languages, for module implementations to select with interoperability being achieved through compliance with the Semantic Web standards. The Apache Jena project was selected due to its compliance with the Semantic Web standards and providing extensions for persistent data storage, HTTP server and extendible SPARQL query engine. The project also actively contributes to the development of the Semantic Web standards.

An extension framework complying with the GeoSPARQL standard [37] was also developed for geospatial querying (see Appendix B). This was necessary as Apache Jena currently has limited support for spatial querying and alternative implementations required persistent graph database with varying GeoSPARQL compliance [139, 140]. The implemented extension allows flexible deployment of in-memory or persistent graph databases to enable the prototype to be a *pure Java* solution. Additional functionality was implemented to determine geospatial relationships useful when interpreting road networks such as the side a point is positioned along a directed edge and the distance a point is placed along an edge.

The usage of a Java environment provides a high performing programming language which is supported across multiple operating system platforms. Java is widely used in enterprise applications, is popular in Semantic Web applications and frameworks [138, 141–143] and has also been used in the development of traffic simulators [32]. The Gradle build tool was used to access published on-line libraries and provides a straight-forward build process for the developed code to be re-used by the community.

The modules of the framework are implemented as *property functions* which are registered with the SPARQL query engine of the Semantic Web API. These plug-in *property functions* are recognised during query execution and processing is handed over to execute their functionality. A custom *property function* could represent a single discrete function that processes only the arguments or retrieves additional data from the knowledge-base. They can also be an entry point to a cascade of multiple functions that in turn call other *property functions*,

execute additional queries and create or modify triples. The incorporation of existing models and algorithmic implementations as framework modules requires the wrapping of the model in the *property function* interface and the identification of the data concepts necessary for its operation.

## 6.3 Configuration of the Framework and Knowledge-Base

The knowledge-base provides the underpinning dataset for the execution of the framework. In the prototype the dataset has been divided into multiple graphs based upon identified domains, i.e. road network, spatial locations, and travel groups, as discussed in Section 4.9 and illustrated in Figure 6.1, as previously presented in Figure 4.49.


Figure 6.1: Diagram of named graphs applied to the prototype knowledge-base.

It has been discussed previously (Chapters 3 and 5) that the Semantic Web is designed to allow components to be physically distributed and communicate through HTTP requests. Access configuration is controlled through the *Framework Configuration* RDF graph which specifies the service type, graph URI and service URI. The service URIs can be a single local file URI or multiple HTTP URLs. This means that the knowledge-base and modules can be physically distributed over multiple datasets accessed using SPARQL's Federated Query stan-

dard. Executing the prototype can also be separated into multiple batches for multi-thread and multi-computer execution.

The domain data has been divided according to the requirements of the three implemented modules across four inter-related graphs. The parameters of the *Travel Scenario* are stored in a single graph to allow repeated re-use of the parameters through multiple executions of the *Framework Configuration*. The data generated by all the prototype modules is stored in a named graph specified by the *Framework Configuration*, so that all generated data can be easily exported or removed.

## 6.4 Design Features of the Prototype

This section will examine the design features of the prototype. The complete process of travel demand generation incorporates multiple stages including the gathering and preparation of data; generation of travel demand; and the simulation of the traffic environment (Chapter 3). The prototype implementation has focussed upon the second and third of these three phases as shown in Figure 6.2.

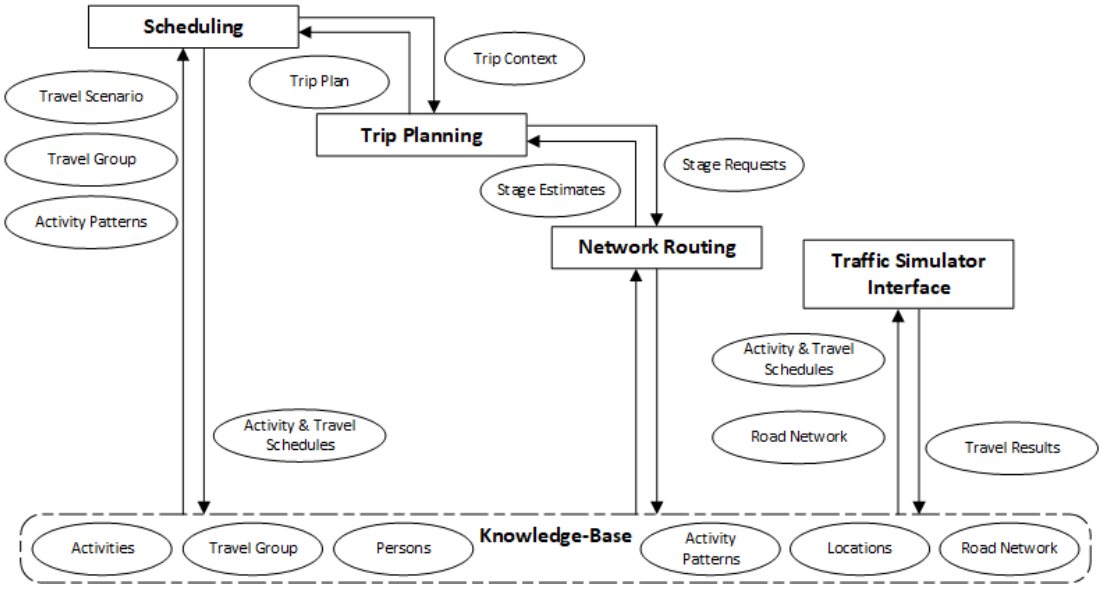


Figure 6.2: Diagram of the modules implemented for the prototype.

This focus has been determined based upon the second phase being the pri-

mary phase for the whole process. The initial phase of knowledge-base construction has been excluded as an ideal dataset can be constructed and refined as an input for the second and third phases. The final phase was included to ensure that integration of the third party traffic simulators could be achieved with the framework and the proposed data concepts (Chapter 4).

The data structures passed between the three implemented modules have been discussed previously in further detail (Section 4.6.5). It should be noted that these data structures and implemented modules provide one approach for the generation of travel demand. Alternative implementations, e.g. modules which exchange different data structures, and configurations, e.g. a single travel demand module, can be incorporated by conforming to the *Activity & Travel Schedule* data structure, which are ultimately produced by the *Scheduling* module.

### 6.4.1 Scheduling Module

The implemented *Scheduling* module is the entry point of travel demand generation phase, i.e. the second phase. It finds an *Activity Pattern Set* for each *Travel Group* and then builds the schedule for each *Person* within those *Travel Groups*. The specific parameters of the scenario are provided in the *Travel Scenario* for that execution. The schedule is constructed starting with the earliest *Activity Pattern* item and building forwards in a single pass. Each *Activity & Travel Schedule* always contains at least one activity and must start and end with an activity.

The *Activity Pattern* for a *Person* consists of items in an ordered list. The activities are handled in order according to their position in the list with first and last representing special cases (Algorithm 3). The conclusion of the previous activity is used as the basis for planning the travel of the next activity.

The first activity in an *Activity Pattern* is not preceded by any travel and therefore can be placed directly on the schedule (Algorithm 4). The scheduling process will not produce a continuous time sequence of activity and travel events. There will be time intervals gaps caused by mismatches between travel duration and time allowed between activities and the removal of activities. These gaps need to be closed or filled, since conceptually a person is either performing some

**Algorithm 3** Build Schedule Part 1

```
procedure BUILD SCHEDULE(person, travelScenario)
    maxVar ← travelScenario.maxActivityVariance
    minDur ← travelScenario.minActivityDuration

    for activity in person.activityPattern do
        if activity is first then
            priorActivity, priorEnd ← FIRSTACTIVITY(activity,
                                      travelScenario)
        else if activity is not last then
            priorActivity, priorEnd ← MIDDLEACTIVITY(activity,
                                      travelScenario, priorActivity, priorEnd)
        else
            LASTACTIVITY(activity, travelScenario, priorActivity, priorEnd)
```

form of activity or travelling.

**Algorithm 4** Build Schedule Part 2

```
function FIRSTACTIVITY(activity, travelScenario)
    start ← travelScenario.start
    maxVar ← travelScenario.maxActivityVariance
    minDur ← travelScenario.minActivityDuration

    earliestEnd ←PROTECTTIME(start, activity.end, maxVar, minDur)
    latestEnd ← activity.end + maxVar
    end ←VARIATETIME(activity.end, earliestEnd, latestEnd)

    SCHEDULEACTIVITY(activity, start, end)
     return activity, end
```

There are multiple potential approaches to handling these time gaps, including allocating additional activities and increasing the duration of existing activities [9]. The adopted approach has been to extend the activity durations as their definition includes any non-travel waiting periods (Section 1.2.2).

A case of these time interval gaps is caused by the mismatch between the *Travel Scenario* and the *Activity Pattern*. The *Travel Scenario* defines the start and end times of the scenario. These extend beyond or truncate the activities defined in the *Activity Pattern*. The duration of the first and final activity are

extended so that the schedule covers the full time interval of the scenario as shown in Figure 6.3 (Algorithm 4 and 8).

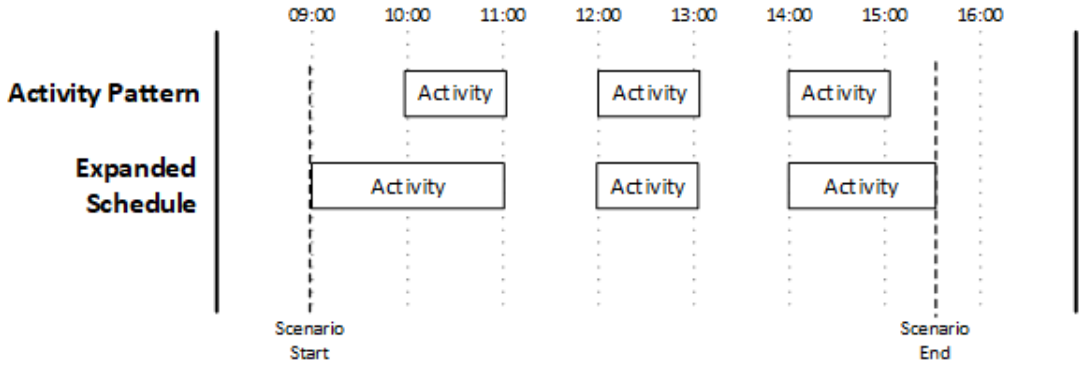


Figure 6.3: Diagram of initial and final activity duration expansion to fill the scenario time interval.

The middle activities require travel to reach the activity which is requested from the *Trip Planning* module (Algorithm 5). This module responds to the request with a *Trip Plan* consisting of one or more *Travel Stages*. These *Travel Stages* are added to the schedule and used to inform the start time of the next activity. The start and end times are varied to apply a stochastic component to the template *Activity Patterns*. The failure to obtain a *Trip Plan* results in a second attempt with the activity delayed as last as possible and reduced to the minimum duration.

The request for a *Trip Plan* is obtained using a *Trip Context* that specifies the *Modes*, *Vehicles*, *Transit Lines* and destination *Locations* that can be used (Section 4.6.5). Each item in the *Activity Pattern* has an *Activity Type*. This *Activity Type* is intersected with a *Person's* related *Activities*, provided by one or more *Locations* in the scenario, to produce a shortlist of potential destination *Locations*. Therefore, a *Person* with an *Activity Type* that only has an *Activity* at a single Location, e.g. education, employment and residence, will always be respected and the *Person* will return to them during a tour.

If no *Locations* with the current *Activity Type* are asserted for a *Person*, e.g. leisure and retail activities, then the *Scheduling* module searches for potential *Locations*. These *Locations* are selected based on the *Activity Type* and distance

**Algorithm 5** Build Schedule Part 3

```
function MiddleActivity(activity, travelScenario,
                                priorActivity, priorEnd)
    start ← travelScenario.start
    maxVar ← travelScenario.maxActivityVariance
    minDur ← travelScenario.minActivityDuration

    travelStart ← priorEnd

    earliestStart ← activity.start − maxVar
    latestStart ← activity.start + maxVar
    start ←VariateTime(activity.start, earliestStart, latestStart)

    earliestEnd ←ProtectTime(start, activity.end, maxVar,
                                minDur)
    latestEnd ← activity.end + maxVar
    end ←VariateTime(activity.end, earliestEnd, latestEnd)

    tripPlan ←RequestTripPlan(travelStart, start)
    if tripPlan is not success then
        start ← latestEnd − minDur
        end ← latestEnd
        tripPlan ←RequestTripPlan(travelStart, start)
    HandleTripPlan(tripPlan, activity, priorActivity, start, end)
     return activity, end
```

from the current *Location* within the minimum and maximum travel range radius of the *Activity Pattern* item.

The restriction of the travel range is relaxed by iteratively expanding the search, by a distance specified by a *Travel Scenario* definition (Section 4.6.1), until at least one potential *Location* is found. These *Locations* may later be rejected if there is insufficient travel time to reach them but an attempt to travel is always made. Therefore, a *Location* can be selected based on the current context of previous decision making as well as being asserted during Knowledge-Base Construction (Section 3.3.2). Alternative options for identification of candidate destination *Locations* could be *Person* being within a geographic area (Section 4.5.8) of the *Location*, e.g. retail attraction or service catchment area, or the

popularity of a *Location* considering its current usage (Section 4.5.7).

A second cause of time interval gaps is between proposed *Travel Stages* and *Activities*. The scheduling of travel between activities is timed for arrival/trip end at the next activity's start time. The duration of the travel will either be equal or less than the available time slot. The time interval gap between a previous activity ending and the travel starting is closed by extending the duration of the previous activity as shown in Figure 6.4. An alternative implementation could have travel always fitted into the time slot and the activity duration reduced or start time delayed to allow the journey, although this presents issues with activities being excessively shortened or delayed.

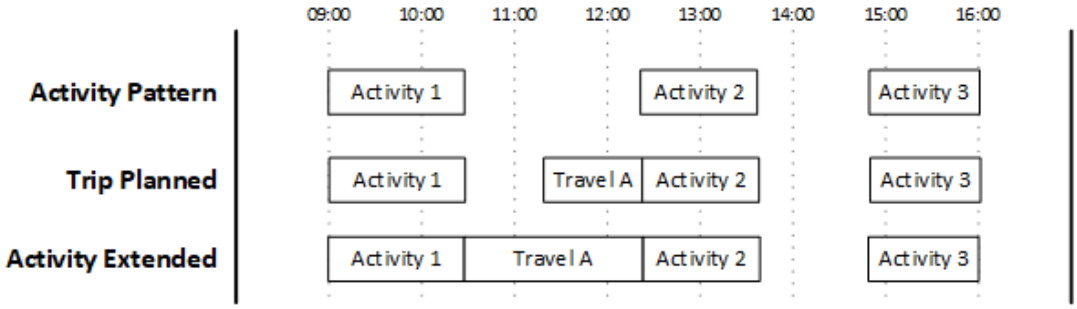


Figure 6.4: Diagram of activity duration extended following selection of travel.

The *Activity Patterns* represent a template that are shared by multiple individual *Persons* in a scenario. A direct replication of the *Activity Pattern* timing into the schedule would result in large numbers of identical schedules. This would cause large spikes in travel demand with waves of persons travelling at the same time instances. To provide diversity the start and end times of activities are randomly varied.

This random variation is specified by *Scenario Definitions* (Section 4.6.1) for maximum variation and minimum duration to ensure that excessively large or small activity durations are not proposed, e.g. entire day or instantaneous durations. The minimum duration of the activity is protected by identifying the earliest time the activity can end (Algorithm 6). A random variation is then applied to increase or decrease the initial start time while still respecting the upper and lower boundaries of the activity (Algorithm 7).

This variation to activity duration assumes that there is always sufficient time

**Algorithm 6** Protect Time

```
function PROTECT TIME(activityStart, plannedEnd, maxVar, minDur)
    minimumEnd ← activityStart + minDur
    proposedEnd ← plannedEnd − maxVar

    if minimumEnd after proposedEnd then
        return minimumEnd
    else
        return proposedEnd
```

**Algorithm 7** Variate Time

```
function VARIATE TIME(initialTime, earliestTime, latestTime)
    variation ← (latestTime − earliestTime)∗RANDOMGENERATE(0, 1)

    if RANDOMGENERATE(0, 1) < 0.5 then
        variedTime ← initialTime + variation
    else
        variedTime ← initialTime − variation

    if variedTime before earliestTime then
        return earliestTime
    else if variedTime after latestTime then
        return latestTime
    else
        return variedTime
```

interval for travel to take place. However, this may not always be the case as the destination locations may be too far or available modes too slow to reach in the time interval between activities. It may also be the case that the *Activity Pattern* templates have been provided with very limited durations between activities.

When travel is not possible for the proposed time period then the process is repeated but travel time is maximised by the activity having the minimum duration and ending as late as permitted by the scenario parameters (Algorithm 5). This may have a knock on effect to later travel time intervals when insufficient gaps are provided between activities. However, when sufficient gap is provided the delay may be absorbed by the next travel interval with the next activity being varied as normal.

The last activity in the schedule also requires travel to reach it (Algorithm 8), unless there is a single activity which would be expanded to fill the scenario time period. Therefore, a request is made of the *Trip Planning* module to provide a *Trip Plan*. The time slot for travel begins when the previous activity is finished and concludes when the scenario ends. This provides the largest time slot for the travel to be incorporated.

It is assumed that any minimum duration for the final activity can be fulfilled after the scenario has ended. This assumption is based on prioritising that travel schedules are complete with all individuals returning to their start *Locations*, e.g. homes or commuting access point, if the *Activity Pattern* is designed to achieve that. This behaviour of return journeys would be assumed to be typical over a full day schedule.

**Algorithm 8** Build Schedule Part 4

**function** LastActivity($activity, travelScenario,$
$priorActivity, priorEnd$)
$travelStart \leftarrow priorEnd$
$end \leftarrow travelScenario.end$
$tripPlan \leftarrow$RequestTripPlan($travelStart, end$)
$start \leftarrow tripPlan.travelEnd$
HandleTripPlan($tripPlan, activity, priorActivity, start, end$)

The *Travel Stages* obtained from the *Trip Plan* for the final activity are placed on the schedule but using a different approach to the middle activities. The final activity is positioned as late as possible in the scenario. Extending the previous activity to fill any time gap until the travel commences could create an excessively long duration between the previous activity and short duration final activity.

Instead the *Travel Stages* are moved to immediately after the previous activity and the final activity is extended as shown in Figure 6.5. Therefore, there is still potential that the last activity will satisfy some or all of its minimum duration once the necessary travel has been planned. However, this approach assumes that the *Travel Stages* can be moved in the schedule without consideration of time constraints, e.g. public transit or route planning using traffic forecasts or previous experience.

In the case of public transit, a journey may take longer at an earlier time

of day, e.g. services are less frequent, or may not be possible, e.g. services are discontinued late at night. This situation can be handled through an option specifying that Trip Plans are built either from the start of the travel interval forwards or the end of the travel interval backwards. In all cases except for travel to the final activity the latter approach would be used, as the previous activity duration would be extended. In the final activity the former approach would be used and the final activity duration extended to the end of the planned trip. Public transit has not been implemented in the prototype.

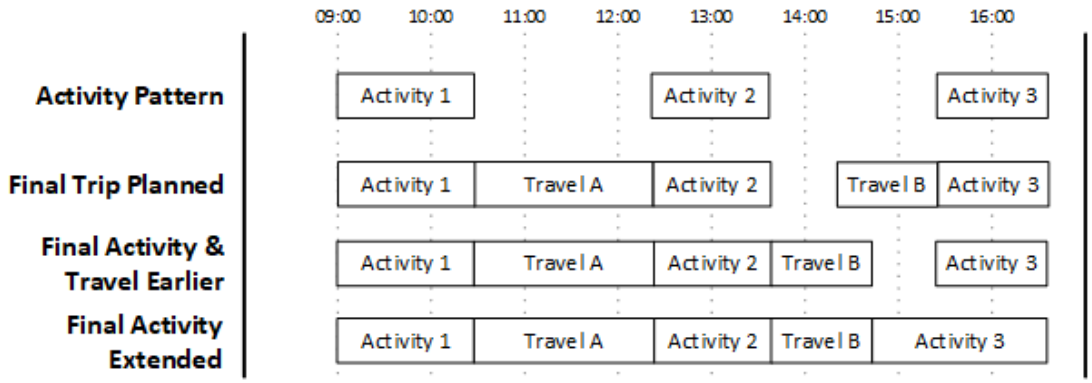


Figure 6.5: Diagram of travel moved forward and final activity duration extended.

There are several other additional actions that may be required once *Activities* and *Travel Stages* have been added to the schedule (Algorithm 9). The final activity is scheduled as late as possible in the scenario. Therefore, rather than extending the previous activity the travel stage and final activity are brought forwards. The final activity is then extended to fill the scenario as shown in Figure 6.5. Activities earlier in the schedule are extended to fill the gap between the activity end and the travel start as shown in Figure 6.4.

Other actions include an activity being skipped if there is insufficient time to travel to it at a destination location, once the attempt to delay and reduce the activity has been attempted. This can result in two activities with identical *Activity Types* occurring at the current *Location*. When this occurs the activities are merged into a single activity. Activities with different *Activity Types* can also take place at the same *Location* and therefore can follow on without travel.

These processes for travel and activity adjustment are focussed upon a single

**Algorithm 9** Handle Trip Plan

```
function HANDLE TRIP PLAN(tripPlan, activity, priorActivity, start,
                          end)
    if tripPlan is success then
        if activity is last then
            start ←MOVETRAVELSTAGESEARLIER(priorActivity, tripPlan)
        else
            EXTENDPREVIOUSACTIVITY(priorActivity, tripPlan)
        SCHEDULEACTIVITY(activity, start, end)
    else
        if tripPlan.location equals priorActivity.location then
            if activity.type equals priorActivity.type then
                MERGEACTIVITY(activity, priorActivity, end)
            else
                FOLLOWONACTIVITY(activity, priorActivity, end)
        else
            SKIPACTIVITY(activity)
```

person and the schedule being constructed in a iterative manner based on time order. This can mean that essential activities, e.g. employment and education, are unfulfilled while non-essential activities are undertaken, e.g. retail and leisure. It has also been discussed previously that individuals co-ordinate activity and travel within household groups to share and access resources, e.g. sharing cars and escorting children (Section 1.2.3). An area of future work is utilising activity priority and travel group co-operation as part of the scheduling process.

Consistent vehicle usage between travel stages is ensured so that a vehicle used for travel is returned at some point in the schedule to its start location. Therefore, commuters do not abandon their vehicles after travelling to other locations but also do not insist on a single mode for an entire schedule. The *Trip Context* provides information on available vehicles and their current location through the *trip vehicle option* property. This also specifies when the *Trip Planning* module must provide a *Trip Plan* that places all vehicles at their *required location*, e.g. when a vehicle has been moved and the last activity is being scheduled. Alternative approaches could include taxi services where a return journey is not required and vehicles being carried by other vehicles, e.g. bicycles in cars and on public

transport.

The prototype *Scheduling* module has implemented a single approach to the closing of time interval gaps, scheduling process, location search and vehicle usage. It has been outlined that multiple alternatives exist for each of these choices. These alternatives could be implemented by defining them as sub-modules and incorporated into the framework using the query overwriting mechanism (Section 5.3.3).

The selection of these alternatives can be dependant upon contextual information which the framework also facilitates the user selecting. Therefore, modules can be implemented either as a wide set of completely defined alternatives or a deep set of sub-components. Further development of the *Scheduling* module would investigate these discrete pieces of functionality being formed into sub-modules. This area of future work would enable narrowly defined modules to be implemented which will reduce the implementation burden and provide more variety of behaviour for investigations.

The sub-module approach has been applied in the prototype between the *Scheduling* and the *Trip Planning* and *Network Routing* modules. The *Trip Planning* sub-module has self-contained functionality that can be replaced with an alternative implementation while the *Scheduling* module is only concerned with the outcome, i.e. a trip plan, and not the method, e.g. Random Utility Model, Computational Process Model or Agent Based (Section 1.2.4), of the sub-module. The *Trip Planning* module applies the same approach in turn to the *Network Routing* module for resolving specific functionality.

The selection of these sub-modules is controlled through SPARQL query and can be controlled through the query overwriting mechanism. This demonstrates the framework supporting the modular approach to constructing travel demand models and allowing the user control over how these modules are brought together.

### 6.4.2 Trip Planning Module

The *Trip Planning* module is provided with a *Trip Context* by the *Scheduling* module and responds with a *Trip Plan* consisting of multiple *Travel Stages*. The *Trip Planning* module constructs a choice set of potential *Trip Plans* from which

a single plan is selected using a Random Utility Model (RUM).

The choice set of *Trip Plans* is constructed using a recursive algorithm to produce multi-stage and multi-mode *Trip Requests* for each destination *Location* and *Mode* provided in the *Trip Context* (Algorithm 10). A *Trip Request* is formed from one or more *Stage Requests* while the corresponding *Trip Plan* is formed from one or more *Travel Stages*.

The initial condition is an empty list of stages that is passed into the recursing function (Algorithm 11) which adds additional stages to move from the origin to destination locations. The resulting *Trip Requests* are checked to ensure that any proposed usage of vehicles ensures the plan moves the vehicle to its optional *required location*. Any *Trip Requests* that do not satisfy this condition are removed.

The final set of *Trip Requests*, are later converted into *Trip Plans* using the *Network Routing* module to provide contextual detail. This means an initial set of general *Trip Requests* based on spatial information is produced to which the contextual detail of the network infrastructure and scoring is applied to produce *Trip Plans*.

**Algorithm 10** Trip Planning Part 1

```
procedure BuildTripRequests(tripContext, limit)
    set trips ← empty
    origin ← tripContext.origin
    for dest in tripContext.destinations do
        for mode in tripContext.modes do
            list stages ← empty
            trips addAll PlanTrip(stages, origin, mode, dest,
                                  tripContext, limit)
    if any vehicle.isRequired in tripContext.vehicles then
        trips ← CheckVehicleUsage(trips)
    return trips
```

The recursing function plans a single stage of the trip using one *Mode* which may or may not be able to reach the destination (Algorithm 11). Each *Location* specifies the *Modes* for which it is accessible (Section 4.5.7). Ideally a *Mode* is

accessible at both the origin and destination. Therefore, the first check is made for a single stage from the origin to the destination. When both are accessible this stage is built and appended to any other prior stages from early recursions of the function. These lists of one or more stages are then converted into *Trip Requests*.

**Algorithm 11** Trip Planning Part 2

```
function PlanTrip(priorStages, origin, mode, dest, tripContext, limit)
    set trips ← empty
    originHasAccess ← LocationAccess(origin, mode)
    destHasAccess ← LocationAccess(dest, mode)

    if originHasAccess and destHasAccess then
        list of lists stagesLists ← BuildStages(priorStages, origin,
                                    mode, dest, tripContext)
        for stages in stagesList do
            trips add new TripRequest(stages)

    set transfers ← FindTransfers(dest, mode, tripContext, limit)
    for transfer in transfers do
        tLoc ← transfer.location
        tMode ← transfer.mode
        list of lists transStagesList ← BuildStages(priorStages,
                                    origin, mode, tLoc, tripContext)
        for stages in transStagesList do
            trips add PlanTrip(stages, tLoc, tMode, dest, tripContext,
                                    limit)
    return trips
```

The direct access of the destination using a mode does not terminate the recursion. However, this point is where the current progress of stages are finalised (Algorithm 14) and added to the list of proposed *Trip Requests*. It is therefore the termination condition of the recursion. Alternative trip requests containing more stages may be more viable, e.g. the current mode provides access to locations with alternative faster modes, and therefore the process of searching is continued.

Transfer locations that are accessible to the current mode and other modes permitted in the *Trip Context* are sought to allow a *Mode* change (Algorithm 12).

Each change of *Mode* forms a *Travel Stage* of the *Trip Plan* and so a *Stage Request* of a *Trip Request*. These transfer locations are chosen by closest proximity to the overall destination to achieve the greatest progress using the current mode to travel from the current location via the transfer location.

This assumption of maximising progress is applied to minimise the number of mode transfers performed by an individual. The transfer stages are added to the previous stages and these form the progress of *prior stages* for the next iteration of the recursion in a *depth first* manner. The next iteration also progresses by using the transfer location as the origin and the transfer mode as the mode while the destination remains the same.

**Algorithm 12** Trip Planning Part 3

```
1:  function FindTransfers(dest, mode, tripContext, limit)
2:      set transfers ← empty
3:      transferModes ← tripContext.modes

4:      for tMode in transferModes do
5:          if tMode.isVehicle then
6:              vTransfers ← FindVehicleTransfers(mode, tMode,
                                        tripContext)
7:              transfer addAll vTransfers
8:          else if tMode.isTransit then
9:              tTransfers ← FindTransitTransfers(dest, tMode,
                                        tripContext, limit)
10:             transfer addAll tTransfers
11:         else if tMode.isPersonal then
12:             pTransfers ← FindPersonTransfers(dest, mode,
                                        tMode, limit)
13:             transfer addAll pTransfers
14:     return transfers
```

The identification of transfer locations is undertaken by iterating through all potential modes permitted by the *Trip Context* (Algorithm 12). Therefore, the *Scheduling* module can control the use of modes, and so the transfer stages, through defined properties of the *Trip Context*, i.e. modes can be excluded from trip planning. The current implementation identifies three categories of mode (vehicle, public transit and personal) with each having different behaviours for

identifying transfers (Algorithm 13).

The vehicle *Modes*, i.e. not public transit or personal, are constrained by the need to utilise a physical vehicle. The only possible location for a transfer to take place is at the vehicle's current location. A check is made to ensure that this location can be accessed by both the transfer mode and the current mode. Otherwise the traveller would not be able to reach the transfer location, i.e. where the vehicle is located.

The public transit *Modes* are restricted to locations serviced by specific *Transit Line* of the mode. Public transit vehicles servicing a *Transit Line* travel between the pre-determined locations of the line (Section 4.5.5). Therefore, not all locations that are accessible by a mode are serviced by all *Transit Lines* , i.e. all buses **do not** visit all locations accessible by buses, except for the exceptional case of a single *Transit Line* in a scenario.

A *Trip Request* may need to use one *Transit Line* to travel to a transfer location then switch to another *Transit Line* to reach the destination. There may need to an intermediate walking stage to access the second *Transit Line* as the same locations may not be visited by both *Transit Lines*, e.g. two bus stops on different streets. Therefore, the public transit *Mode* may be repeated in multiple stages of the *Trip Request* but each with a different *Transfer Line*.

The start and end location of the stage must be accessible to the public transit *Mode* and be visited by the *Transit Line*. The available locations for a transfer to take place are limited to those serviced by the *Transit Line* with the closest to the destination being selected. The vehicles and transit lines available for consideration are defined in the *Trip Context* and so can be controlled by the *Scheduling* module.

**Algorithm 13** Trip Planning Part 4

```
1: function FindVehicleTransfers(mode, transferMode, tripContext)
2:     set transfers ← empty
3:     vehicles ← tripContext.vehicles
4:     for vehicle in vehicles do
5:         if vehicle.mode equals transferMode then
6:             vLoc ← vehicle.location
7:             if JointAccess(vLoc, mode, transferMode) then
8:                 transfers add new Transfer(vLoc, mode, transferMode)
       return transfers

9: function FindTransitTransfers(dest, transferMode, tripContext,
                                 limit)
10:    set transfers ← empty
11:    transitLines ← tripContext.transitLines
12:    for transitLine in transitLines do
13:        if transitLine.mode equals transferMode then
14:            locations ← transitLine.locations
15:            list transferLocations ← FindNearest(dest,
                                 locations, limit)
16:            for tLocation in transferLocations do
17:                transfers add new Transfer(tLocation, transferMode)
       return transfers

18: function FindPersonTransfers(dest, mode, transferMode, limit)
19:    set transfers ← empty
20:    list transferLocations ← FindNearest(dest, mode,
                                 transferMode, limit)
21:    for tLocation in transferLocations do
22:        transfers add new Transfer(tLocation, transferMode)
       return transfers
```

The personal *Modes*, i.e. not used by a vehicle or public transit, are only constrained by the joint accessibility of locations to make a transfer between current and transfer mode. It is assumed that in principle if a location is accessible by a personal or vehicle mode then there is always a path to the location from any other similarly accessible location. This assumption is based on personal modes representing human mobility and road networks being continuous in traffic investigations where by isolated locations are not of general interest or use. Considering such a scenario would increase complexity for an unusual edge case, e.g. two islands not connected by a road bridge or vehicle ferry.

A *Mode* is permitted to be re-used in later stages so that public transit modes can be repeated in a *Trip Request*. This allows repeated switching of personal and public transit modes by an individual, e.g. walk, bus, walk, bus, walk. There is no constraint on the number of stages that can be added to the *Trip Request*. Instead all combinations of modes are sought and no assumption is made at this point over which will be the most optimal.

The search for transfer locations in the public transport and personal category of *Modes* is based on proximity to the destination and shared access. This is to make the most progress with the minimum number of transfer stages. However, the proximity of locations does not always equal the shortest path between locations. The road network infrastructure, mode and geography, e.g. routing via a river bridge, can constrain the route travelled as shown in Figure 6.6.

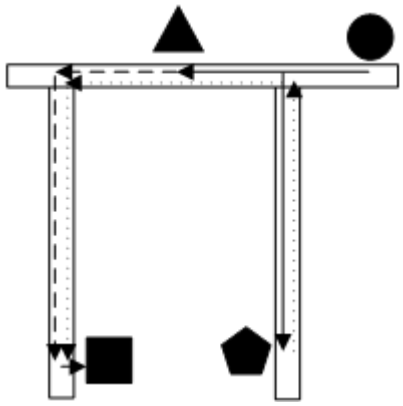

Figure 6.6: Diagram of showing proximity of transfer locations to destination not providing shortest path route.

Travel from the origin (circle) to destination (square) can be achieved via two transfer locations (triangle and pentagon). The pentagon transfer location is most proximal to the destination but there is not a direct path between the two that does not pass the alternative triangle transfer location. Similarly, the most proximal location could be beyond the destination and so require travel past the destination when an intermediate transfer location would be closer to both origin and destination.

This can mean a less proximal transfer location can provide a more optimal routing path. Therefore, the closest **n** transfer locations are selected for candidate routes. The value of **n** transfer locations is controlled by a scenario parameter (default value of three chosen for the prototype scenario). The accessibility of a destination location encompasses concepts such as pedestrian access, motor vehicle parking, public transit links, multiple building entrances and freight delivery (Section 4.5.7).

The recursive function (Algorithm 11) tracks the multiple stages of the trip by adding the current stage and the next transfer stage to the previous list of stages. The new stages are built (Algorithm 14) according to the category of mode, i.e. personal, vehicle and public transit. An exit condition of not revisiting *Locations* in the trip is made to ensure that acyclic graphs of travel stages are produced, i.e. it is redundant for a trip to return to a location as this implies a change of mode and the minimal number of mode switches should be made. When a *Location* is revisited the whole chain of stages is discarded as being unviable.

The current stage is then built according to the category of mode so that further conditions for its viability can be checked (Algorithm 15). In the case of a vehicle *Mode* a search is made of available vehicles which match the stage's mode and are located at the origin. Otherwise use of the vehicle would imply teleportation, self-driving or driving by another individual, which the prototype does not support.

An additional check is then made to ensure that the vehicle is not required to reach a specific location. A vehicle will only be used once in a trip and therefore if it is required to reach the stated location then it must do so when used. An alternative trip may achieve reaching the required location but a candidate trip is not viable if it moves a vehicle somewhere other than its intended destination.

**Algorithm 14** Trip Planning Part 5

```
1: function BuildStages(priorStages, origin, mode, dest, tripContext)
2:     list of lists stagesList ← empty

3:     if origin equals dest or CheckVisited(dest, priorStages) then
4:         return stagesList

5:     if mode.isVehicle then
6:         stagesList ← BuildVehicleStages(priorStages, origin, mode,
                        dest, tripContext)
7:     else if mode.isTransit then
8:         stagesList ← BuildTransitStages(priorStages, origin, mode,
                        dest, tripContext)
9:     else if mode.isPersonal then
10:        stagesList ← BuildPersonalStages(priorStages, origin, mode,
                        dest)
11:    return stagesList
```

In the case of the public transit *Modes* a check is made to ensure that the current stage is serviced at both the origin and destination by all relevant *Transit Lines* for the mode. There are no specific conditions attached in the case of a personal category *Mode* as neither personal or public transit vehicle is required.

**Algorithm 15** Trip Planning Part 6

```
1: function BuildVehicleStages(priorStages, origin, mode, dest,
                                                    tripContext)
2:     list of lists stagesList ← empty
3:     vehicles ← tripContext.vehicles
4:     for vehicle in vehicles do
5:         if vehicle.mode equals mode and vehicle.location equals origin
   then
6:             vReqLocation ← vehicle.requiredLocation
7:             if not vehicle.isRequired or dest equals vReqLocation then
8:                 stages copy of priorStages
9:                 stages add new StageRequest(origin, dest, mode, vehicle)
10:                stagesList add stages
       return stagesList

11: function BuildTransitStages(priorStages, origin, mode, dest,
                                                    tripContext)
12:    list of lists stagesList ← empty
13:    transitLines ← tripContext.transitLines
14:    for transitLine in transitLines do
15:        if transitLine.mode equals mode then
16:            originIsTransitLocation ← HasLocation(transitLine, origin)
17:            destIsTransitLocation ← HasLocation(transitLine, dest)
18:            if originIsTransitLocation and destIsTransitLocation then
19:                stages copy of priorStages
20:                stages add new StageRequest(origin, dest, mode, transitLine)
21:                stagesList add stages
       return stagesList

22: function BuildPersonalStages(priorStages, origin, mode, dest)
23:    list of lists stagesList ← empty
24:    list stages copy of priorStages
25:    stages add new StageRequest(origin, dest, mode)
26:    stagesList add stages return stagesList
```

The completion of the recursive algorithm produces a set of multi-stage and multi-mode *Trip Requests*. These only describe the origin, destination, mode and any vehicle or transit line usage for each of the stages in the candidate plan. There is no detailed route in the plan to travel the network infrastructure between the origin and destination of each stage. A contextual score is also required to provide a weighting to select a single a plan for return to the *Scheduling* module.

The detailed route and information for the context scoring is provided by the *Network Routing* module. The *Stage Requests* of the *Trip Request* are each passed to the *Network Routing* module from which are obtained *Stage Estimates*. These *Stage Estimates* do not have any temporal context and are converted into *Travel Stages*, which have temporal context of a start and end time, by using target end time and subtracting the *Stage Estimates* duration.

Multiple stages are formed by working backwards from the final stage and using the activity start time as the initial target end time. The start time of a stage forms the end time of the proceeding stage. These *Travel Stages* are combined to form the *Trip Plan*. The total duration available for travel in the schedule is used as an upper limit on the *Trip Plan*. Those *Trip Plans* that take too long are rejected to ensure the selected choice will always fit into the schedule. This avoids selecting a choice that will later be rejected when viable alternatives had been found. An extension to the *Trip Planning* module would be applying further contextual information, such as heavily penalising or excluding walking during night time or cycling in the rain, which can be applied to the *Stage Estimate* metrics.

A Random Utility Model has been implemented to select a *Trip Plan* from the choice set based on a utility score. The utility score is calculated for each *Travel Stage* and then summed for the *Trip Plan*. The utility score is derived from coefficient weightings applied to metrics of the *Travel Stage*. The selected metrics are the commonly used trip cost, duration and distance [124]. Each *Travel Stage* has a detailed route that consists of multiple roads, i.e. edges, through the network (Section 4.5.9).

- Distance: the physical distance travelled through the network to complete the *Travel Stage*. A fixed value determined by the route a *Mode* can take

through the network from origin to destination, i.e. not straight line from origin to destination.

- Duration: the time taken to complete the *Travel Stage*. Determined by the maximum speed (metres per second) of the *Mode* or road (edge), whichever value is lower. The total duration is the time taken to travel along each leg of the *Travel Stage*.

- Cost: the penalty, without unit or currency, for using a mode. Proportional to distance and formed from a fixed and variable component such that $cost = fixed\_cost + (distance * variable\_cost)$. The values for the *Mode* related metrics are defined by the *Mode Definition* of the *Travel Scenario*.

The coefficient weightings are applied to each of these metrics as defined in Equation (3). This was derived to provide differentiation between modes based upon varying trip distances as illustrated in Figure 6.7. The prototype's Random Utility Model, as used in the evaluation scenario (Chapter 7), has been tuned using the mode parameters specified in Table 6.1 and the person weightings of cost (-2.7), distance (0.0), and duration (-0.022).

$$U_J^i = \beta_{cost}^i x_J^i + \beta_{distance}^i x_J^i + \beta_{duration}^i x_J^i \tag{3}$$

| Mode | Max Speed (m/s) | Fixed Cost | Variable Cost |
|---|---|---|---|
| Car | 31.27 | 6.0 | 0.001 |
| Walking | 1.79 | 0.0 | 0.0 |
| Bicycle | 8.93 | 3.5 | 0.0011 |
| Bus | 26.9 | 6.0 | 0.001 |

Table 6.1: Table of mode definition parameters for maximum speed (m/s), fixed cost and variable cost.

The coefficient weighting values have been tuned to give a dominating preference to *walking* for trips under 1.5 kilometres that then transfers to *bicycles* and then *car* and *bus* at distances greater than 3.5 kilometre. The Simulated Annealing parameter optimisation technique was applied to derive approximate gobal optimum values for these thresholds [144].

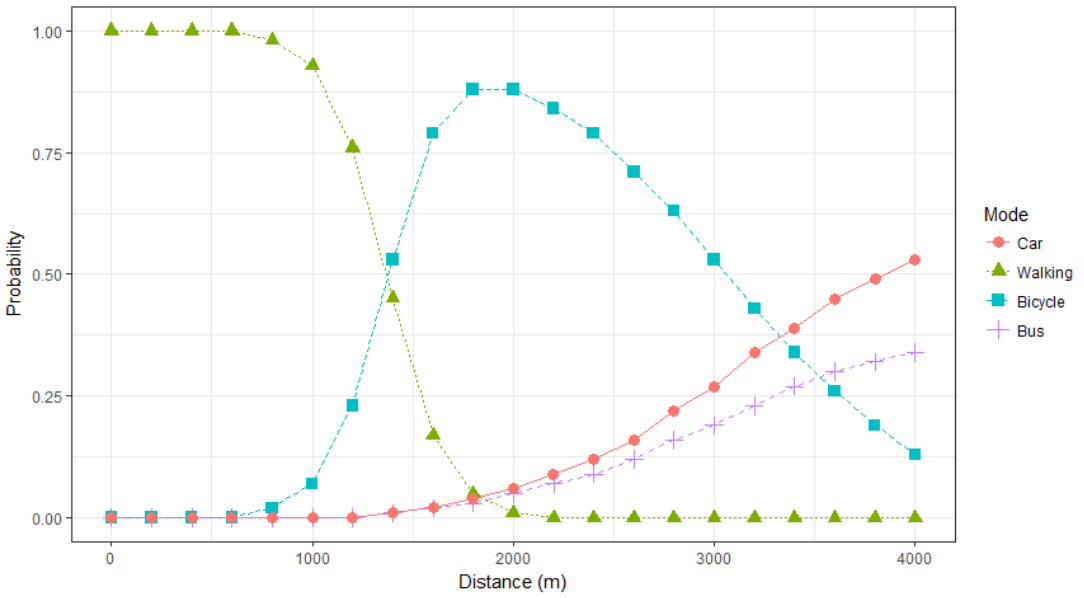


Figure 6.7: Graph of probability change over distance by mode for Random Utility Model.

The thresholds are intended to provide a mix of mode usage and were based upon indicative distance of traveller walking [14]. The prototype does not support public transit modes but the graph includes this mode for illustration of the choice set that can be constructed. It should be noted that *Trip Plans* can be formed of one or more *Travel Stages* which each have a different *Mode*. Therefore, the graph shows the case of single stage *Trip Plans* following the same route, i.e. equal distance. In multi-stage *Trip Plans* the utility contribution from each *Travel Stage* will vary according to its *Mode* and distance.

The value of these weightings are obtained during execution from properties of the individual *Person*. Therefore, each individual may have their own weightings to determine their own choice behaviour. These weightings could also be positioned on a *Person Type* to provide consistency across groups or as global parameters through a definition in the *Travel Scenario*. Definition in the *Travel Scenario* would also allow easier comparison between alternative sets of values if that was a user's investigative focus.

The utilities are calculated through query of the knowledge-base (Listing 6.1) and not a hard coded equation in the *Trip Planning* module. Therefore, the

values can be directly modified within the knowledge-base to produce alternative probabilities. The prototype also permits the overwriting of the query, and so the equation and data utilised, using the query overwriting mechanism of the framework (Section 5.3.3). The only requirement is satisfying the result variables of the *SELECT* query, i.e. *duration*, *utility* and *destination*.

The use of the query overwriting mechanism allows alternative utility equations and weightings to be investigated without modifying the *Trip Planning* module. The values can also be repositioned, as the previous examples outlined, by the user to suit their needs. The user has control over both the data and the use of the data to explore the impact of alternative utility calculations. This demonstrates the functionality of the framework in enabling greater variety of behaviour within the modelling process and giving control over that behaviour directly to the user.

```
PREFIX rou: <http://example.org/tom/schema/route#>
PREFIX util: <http://example.org/tom/schema/utility#>
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>

SELECT ?duration ?utility ?destination
WHERE{

    BIND(?stageEstimateVar AS ?stageEstimate) .
    BIND(?personVar AS ?person) .

    #Retrieve required data.
    ?stageEstimate rou:cost ?cost; rou:distance ?distance; rou:
        ↪ duration ?duration; rou:endLocation ?destination .

    #Retrieve utility values from the Travel Group domain graph.
    SERVICE ?travelGroupService{
        GRAPH ?travelGroupGraph{
            ?person util:tripCostWeight ?costWeight; util:
                ↪ tripDistanceWeight ?distanceWeight; util:
                ↪ tripDurationWeight ?durationWeight .
        }
    }

    #Duration converted to seconds for multiplication.
    BIND(((fn:hours-from-duration(?duration)* 3600) + (fn:minutes-
        ↪ from-duration(?duration)*60) + fn:seconds-from-duration
        ↪ (?duration) ) AS ?durationSecs)
    #Calculate the utility for the stage estimate.
    BIND( ((?cost * ?costWeight) + (?distance * ?distanceWeight) +
        ↪ (?durationSecs * ?durationWeight)) AS ?utility) .
}
```

Listing 6.1: SPARQL query implemented for calculation of trip utility.

### 6.4.3 Network Routing Module

The *Network Routing* module provides the detailed route through the network infrastructure. The module accepts a *Stage Request* and responds with a *Stage Estimate* of the route and corresponding metrics of *cost*, *distance* and *duration*. These metrics are widely used in travel forecasting and measuring location accessibility [124] and represent the best case estimate for travel time between the two locations. The route is determined using the A* shortest path algorithm [145] between an origin and destination location. The shortest path metric used is the travel time, i.e. duration, rather than distance as this has been found to be more important to travellers [108].

The output of the module serves two purposes. First, the metrics form the basis for scoring the different route choices so that a single route can be selected. Second, the *Travel Stages* explicitly state the route to be followed so that all traffic simulators are simulating the same set of journeys, rather than substituting their own routing implementation which may introduce minor or major changes, to give consistency of results. This would also allow the analysis of differences between planned and simulated travel, if dynamic re-routing is being applied during simulation.

The module provides a best case scenario of travel, i.e. travelling at maximum speed and by the shortest path. It does not take into account dynamic factors, such as traffic congestion, road closures, weather, traffic signalling or service restrictions for public transport, as previously highlighted (Section 3.3.3.4) which could form alternative module implementations. These features have not been identified in existing routing components of traffic simulators and forms an area of future work to improve modelling realism. The implementation of network routing for public transit is more concerned with the temporal context than the routing between locations as the *Transit Line* restricts the available *Locations* (section 4.5.5).

The exclusion of dynamic factors and temporal context ensures that separate requests for an origin-destination pair using the same mode will receive the same response. The *cost* and *maximum speed* values defined for each mode can result in different routes and metrics. This means that the *Stage Estimate* responses,

between two *Locations* for a single *Mode*, can be stored and retrieved for re-use in the knowledge-base to reduce computational processing. However, the potential characteristics of the network infrastructure do not guarantee that the reverse route will be the shortest path, e.g. one-way streets and turning restrictions, and so reciprocal routes must be calculated.

These *Stage Estimates* can be calculated dynamically on-demand or pre-computed for every *origin-destination-mode* tuple. However, the pre-computation approach can require calculating a large number of routes that will not be used. Each route is formed between two locations for a specific mode and so the number of routes required can be calculated based upon the *k-permutations of n* equation [146, 147] as shown in Equation (4), where $x$ is the number of routes, $n$ is the number of locations, $m$ is the number of modes, and $k$ is number of locations selected, i.e. 2. This ignores the trivial routing case of the same location for origin and destination.

$$\begin{aligned} x &= \frac{n!}{(n-k)!} \times m \\ &= n(n-k+1) \times m \\ &= n(n-1) \times m, \ when \ k = 2 \end{aligned} \tag{4}$$

It can be seen in Table 6.2 that the number of routes generated increases dramatically as the number of locations increases. In scenarios with thousands of buildings and thousands of people the number of routing requests is still likely to fall short of the millions of routes produced by pre-computation.

As an illustrative example, a typical individual may be expected to perform a few trips during a day long schedule. Each trip may have as many stages as there are modes with each stage having a few possible destinations. An estimate of the routes required is shown in Equation (5) where $r$ is the number of routes, $t$ is the number of trips, $s$ is the number of stages per trip, and $d$ is the number of destinations per stage.

| Locations | Modes | Routes |
|---|---|---|
| 100 | 1 | 9,900 |
| 100 | 2 | 19,800 |
| 100 | 3 | 29,700 |
| 1,000 | 1 | 999,000 |
| 1,000 | 2 | 1,998,000 |
| 1,000 | 3 | 2,997,000 |
| 10,000 | 1 | 99,990,000 |
| 10,000 | 2 | 199,980,000 |
| 10,000 | 3 | 299,970,000 |

Table 6.2: Table of number of routes generated in an exhaustive set of origin and destination locations for each mode.

$$\begin{aligned} r &= t \times s \times d \\ &= 5 \times 2 \times 5 \\ &= 50 \end{aligned} \tag{5}$$

Considering a two mode scenario and defining a *few* as 5, each individual in the scenario would need up to 50 routes. Therefore, when 1,000 locations are in the scenario there would need to be a minimum of 39,960 individuals to request every calculated route. A typical household could be estimated to average 3 individuals. Therefore, if every location in the scenario was a dwelling, and so no locations for employment, eduction or retail activities etc., then 92.49% of the individuals would need to be commuters from outside the scenario geography.

The individuals of the scenario would also need to not visit any locations already visited by other individuals or more individuals would be required. Assuming each route took 100ms to calculate then it would take 55.5 hours to calculate the full set of routes for this set, although parallel or distributed computing can be applied. In conclusion, the pre-computation approach very quickly generates large numbers of routes that it is improbable will be utilised unless a scenario is repeatedly executed and even then certain combinations of locations are likely to be improbable. The number of routes generated also assumes a routing algorithm

only generates a single optimal shortest path and does not propose alternative routes, e.g. branching at key junctions.

The storage and searching of this large quantity of data will also have implications for the execution times and resource usage when performing demand modelling. The exhaustive set of routes is also very vulnerable to changes in the knowledge-base. The set will become invalid or incomplete when any locations are added or removed; any changes are made to the road network, e.g. maximum speeds or turning restrictions; any modes were added or removed; or if any of the scenario parameters were adjusted in the *Mode Definition*. Therefore, the pre-computation approach produces a lot of data that is unlikely to be used; takes a long period of time to generate at scale; is vulnerable to changes; and can have an implications for execution times. In most use cases a dynamic approach during execution will produce the required data in the most efficient manner. The use case where pre-computation may be appropriate is the provision of exemplar remote datasets which could be re-used repeatedly by a large number of users (Section 3.2 and 3.4.2).

The road network data structure is defined as a directed graph of nodes and edges (Section 4.5.9). A modelling simplification used in traffic simulators, such as SUMO, for vehicles is that changes of direction, i.e. turning around, can only occur at nodes, i.e. start and end of edges, and not mid-edge. The directed graph represents the flow direction of vehicles along roads, i.e. bidirectional on two-way roads and unidirectional on one-way roads. However, this distinction does not apply to pedestrian, or similar, modes of transport which always treat footpaths as bidirectional and may cross edges at designated crossings. Consideration of unidirectional and bidirectional routing has an impact on the shortest path as illustrated in Figure 6.8.

The origin (circle) is positioned on an adjacent edge to the destination (square). However, the edge direction for the road/edge is away from the destination. A vehicle following the road must travel (dashed line) away from the destination (edge A) before turning around and heading back in the intended direction (edge B). The vehicle must then travel beyond the destination (edge C) in order to turn around and reach its closest edge (edge D). The pedestrian using the footway, running alongside the roadway, is able to travel in the reverse edge direction

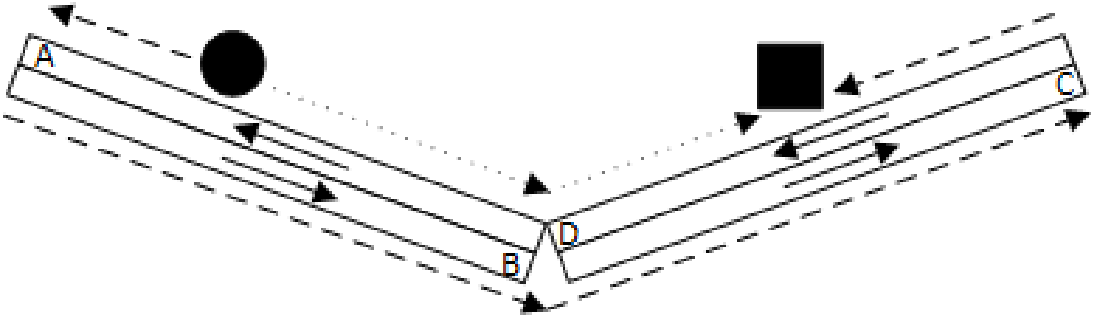


Figure 6.8: Diagram of unidirectional (dashed line) and bidirectional (dotted line) routing through directed graph of edges.

(edge A) and directly to the nearest edge (edge D). Non-consecutive travel by a vehicle or person, e.g. arriving at edge A and departing on edge B, and turning mid-edge is not permitted by traffic simulators, such as SUMO and MATSim. The creation of *virtual* vehicles which start on the alternative edge would complicate the analysis of vehicle data. Therefore, the module applies unidirectional routing for vehicle *Modes* and bidirectional routing for personal *Modes*.

The module does not assume or enforce any specific distance units so can be applied to any road network with consistent units. During routing a check is made between the current edge's maximum speed and the modes maximum speed, defined by the *Mode Definition* of the *Travel Scenario* (Section 4.6.1). The lower value is selected to determine the travel time, i.e. duration, of the edge. It is assumed that the maximum speed is honoured and applied universally as physical acceleration/deceleration and human behaviour of exceeding speed limits are traffic simulator concepts. The parameters from the *Mode Definition* are also used to calculate a non-denominational cost for the stage based upon an upfront fixed cost and distance based variable cost.

### 6.4.4 Traffic Simulator Interfaces

The third stage of the travel demand modelling process is the execution of the schedules using a traffic simulator. The traffic simulator seeks to simulate the physical environment and the interactions between road users and network infrastructure. There are multiple approaches and implementations of the physical

behaviour just as has been found with the human behaviour and decision making of the second stage. Therefore, simulation should ideally take place across multiple simulators with comparison of results to assist verification and validation.

The incorporation of traffic simulators into the framework is achieved by considering them as Modules. Ideally simulator interfaces would be developed to accept RDF input and produce RDF output but existing implementations have been designed as standalone tools with their own interfaces and represent complex software artefacts. Therefore, a Module wrapper was developed to provide conversion between the simulator input and output formats and enabling them to be used in the framework. This approach can also be applied to other existing implementations from earlier stages, e.g. population synthesis, activity generation, travel demand generation, to convert the input and output between RDF.

The development of a wrapper interface presents a problem when considering the framework objective of incorporating a flexible schema that the user and other modules can extend. In a fixed schema the interface can be designed to the specific data items and conversion can be performed using a specific programming language. However, changes or additions to the schema, e.g. new properties or alternative property names, would require the interface to be redeveloped and published. Alternatively, the user, or developer of the module changing the schema, would need to understand, modify and re-compile the interface. This requires an investment of resources and reduces the flexibility for modules to inter-operate. The further development of a traffic simulator may also make an interface obsolete and prevent a user from accessing new features. Therefore, there would be a need for interface developers to continually update revised interfaces.

The proposed solution to these issues is for wrapper Modules to be developed using Extensible Stylesheet Language Transformations (XLST) templates and processors [67]. This standards based technology provides for the conversion between different dialects and schema of Extensible Markup Language (XML) documents and also supports other formats, e.g. Comma Separated Values (CSV) and JavaScript Object Notation (JSON). The XML format is a widely used format and is the format adopted by both traffic simulators utilised in the prototype, i.e. MATSim and SUMO.

The file based XLST templates are read by the XSLT processor alongside an input file/s to produce an output file/s. The template content describes the source and target structure and changes to the template are reflected in each execution of the processor. Therefore, modifications can be performed at runtime by a user. The templates are agnostic to the underlying platform, unlike a programmed interface, and so can be transferred and executed by any compliant processor. There is also full access to the contents of the knowledge-base so that additional data can be incorporated as it is available for a particular simulator, which a programmed interface may not have been designed to include.

The Module wrapper has to perform two stages of conversion. The initial stage is conversion from the framework's schema in XML to simulator input schema in XML. The simulator would be executed and the output produced. The conversion process is then reversed to convert the simulator output schema in XML to framework schema in XML.

Obtaining the initial data of the framework schema in XML can be achieved using SPARQL *CONSTRUCT* query and then outputting the resulting graph as an RDF/XML serialisation. The *CONSTRUCT* query permits transformation of data, e.g. property name changes, and so certain schema changes can be managed by only modifying the query. The standardised RDF/XML serialisation [100] is the original format for writing RDF graphs to file and so is widely supported by Semantic Web libraries. The use of SPARQL queries again enables the user to adapt the data extraction process at runtime so that changes to the schema from the core schema (Chapter 4) can be incorporated.

The process for conversion is illustrated in Figure 6.9. The output of the second stage travel demand model is the *Activity & Travel Schedules*. These are utilised along with network infrastructure, and other simulator relevant data in the knowledge-base, as input to the traffic simulator. The traffic simulator interface extracts one or more graph files using SPARQL queries. The resulting RDF/XML is then converted into the required format and schema of the simulator. The traffic simulator is executed and then the output converted into RDF/XML and directly added back into the knowledge-base following the configuration provided by the *Framework Configuration* (Chapter 5). Provided the number of input and output files of the simulator does not change between ver-

sions then adaptations can be made to the schema and simulator input without modification to the wrapper Module. However, this restriction could be accommodated through a more sophisticated wrapper module.

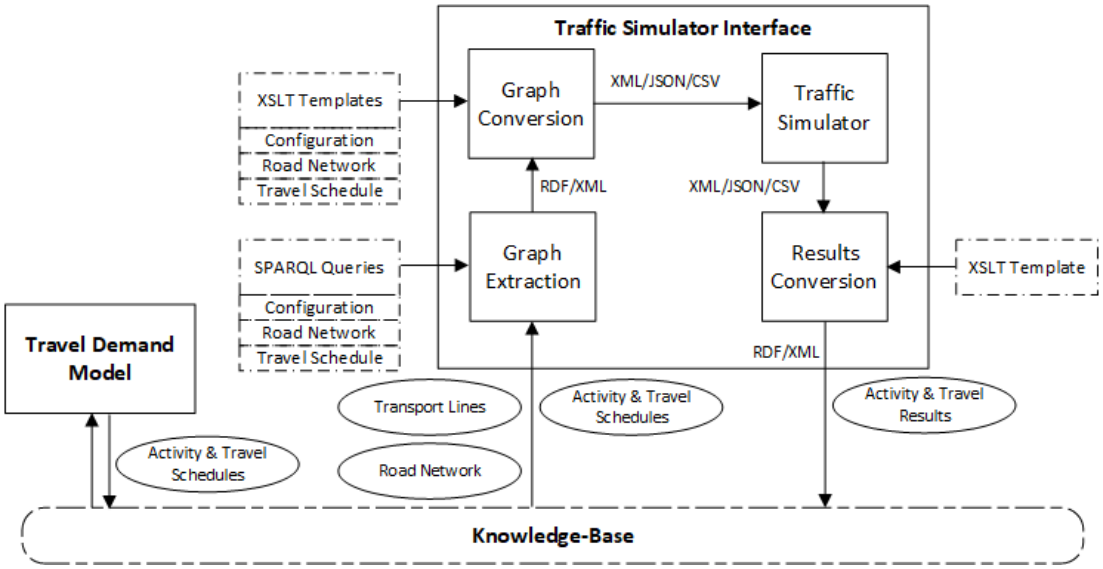


Figure 6.9: Diagram of SPARQL query and XSLT template process for Traffic Simulator Interface.

Interfaces were implemented in the prototype for MATSim and SUMO simulators. The *Activity & Travel Schedules* and *Road Network* data, along with other relevant information, e.g. locations as points of interest, were extracted from the knowledge-base. These were then converted into the multiple input files for execution. SUMO is a micro-simulator that simulates individual vehicle interactions while MATSim is a meso-simulator which takes a higher level approach with a more abstract queuing system. The parameters for this queuing system are not clearly specified in the MATSim documentation [32] and were selected from alternative research [148].

The data requirements between simulators were very similar with some differences in emphasis. For example, SUMO uses road edges for routes while MATSim uses nodes but both are based on a graph structure. SUMO, with its original emphasis on vehicle only simulation, requires complete and consistent vehicle routing as a separate input to person plans while MATSim obtains vehicle routing from the person plans. The proposed core schema was able to satisfy the

data requirements of both simulators to execute the *Activity & Travel Schedules*. In both cases the wrapper Modules interfaces were implemented to reformulate the knowledge-base data in the required formats using only XSLT and SPARQL query, invoke the simulator and then convert the output into *Activity & Travel Results*.

During the development of these interfaces it was found that the multiple queries and serialisation of very narrowly defined sets of data were noticeably quicker than the more general approach of a single query that retrieved all required data. Therefore, a one-to-many relationship may be necessary between a simulator input file and the knowledge-base queries required to construct it. Multiple simple queries are generally easier to maintain and more accessible for users to modify but require orchestration by the Module wrapper interface.

The proposed approach does not remove the need to develop a wrapper Module interface, unless traffic simulators are adapted to directly utilise RDF, but it does reduce the burden and increase the flexibility and longevity in providing them. It does require the user, or module developers, having skills in SPARQL and XSLT languages. These are non-trivial requirements but not excessive and only apply when changes are needed from the core schema.

Ideally modules and datasets would conform to the core schema and no adjustments would be needed. However, it should be considered that integration between components in current travel demand processes may require multiple interfaces in multiple programming languages with requirements changing between different configurations. In this approach the requirement is consistent and therefore developed skills will be re-used across configurations. In addition, the SPARQL language is the primary language for operation of the framework. SPARQL is widely used for Semantic Web applications and so is a core skill for users in the domain.

## 6.5 Chapter Summary

The prototype provides the user with control over the activity patterns, schema, module parameters, module selection and discrete choice calculation. These can be applied based on the class and properties present in the data with minimal

design assumptions, e.g. modes are defined in the data and not an imposed hierarchy beyond personal, vehicle and public transit.

The approach allows the user to include their own schema of concepts; select alternative modules based on those concepts; access and modify both local and remote datasets; and apply the generated demand to multiple traffic simulators. The implemented modules are intended to be generic representations with minimal design assumptions that cannot be modified through query. However, the overall modular architecture is intended to allow the substitution and selection of modules for the user's modelling approach. These can help address the current shortcomings of singular behavioural models and burden of comparing between travel demand frameworks.

The implemented scheduler produces full day schedules and is discrete from the trip planning and network routing stages. The scheduler builds the schedule in a single forward pass and there have been identified opportunities for variation, including potential sub-modules to enable variation within this module. Additional features that have not been implemented include co-operation within travel groups and the prioritisation of activities in the scheduling process.

The trip planning module can produce trips where the number of stages is dependent on the modes and available transport resources, rather than those pre-defined by the design. The construction of trips considers spatial access constraints to locations and satisfy any requirement to return vehicles to a starting location at the end of the schedule. The destination for trips can be formed from asserted options in the knowledge-base or searched from viable options according to the activity pattern.

The selection of trips uses a discrete choice calculation which is performed through a query of the knowledge-base. This query can be substituted for alternative formulations or expanded to select formulations according to traveller class or other characteristics present in the knowledge-base by using the proposed framework. Therefore, the user is not limited to the implemented calculation but can adapt and explore according to their investigation.

The routing module produces best estimates for travel between locations through the network infrastructure for personal and vehicle modes using the A* algorithm. Areas of future work includes the consideration of temporality in

public transport timetables, learning from past travel experience and alternative routing, e.g. trunk road preference. It has also been identified that some road network datasets contain semantic information that could be used to influence routing choices, but are not currently considered in routing tools provided by the examined simulators.

The integration of two third-party simulators has been achieved through interfaces that combine SPARQL querying of the knowledge-base with the XLST language for XML transformations. This allows change, variation or expansion of concepts in the knowledge-base or traffic simulators to be compensated for by the user without requiring re-development of the interface. Both of these techniques use platform independent and text based templates which facilitates their inspection and distribution. The use of a unifying knowledge-base also provides potential for users to develop new interfaces for alternative simulators using the same techniques. Chapter 7 considers the implementation of the prototype when utilised with a knowledge-base of scenario data and the performance of the framework in alternative configurations.

# Chapter 7

# Evaluation of Prototype Travel Demand Generation Framework

## 7.1 Introduction

The previous chapters have established the design of the framework and the operational modules; the schema of the knowledge-base on which the modules will function; the design of the framework to control the module selection and operation; and the implementation of the prototype to demonstrate the framework design and function. This chapter will seek to address research question RQ4 of whether a Semantic Web framework can be implemented for the generation of travel demand by examining the prototype developed based on the concepts covered in the previous chapters.

The discussion is formed into four parts. The first part discusses the scenario constructed as the basis of the evaluation. The second part considers the prototype developed using the framework schema and the organisation of the implemented modules to produce travel demand for traffic simulation with two third-party traffic simulators. The third part uses the prototype to consider the alternative configurations of the framework and their performance. The final part will outline the challenges encountered during the development of the prototype and applying the described Semantic Web based approach to travel demand generation.

The objective of this chapter is to consider the progress and issues encountered in applying these approaches rather than seeking to establish their effectiveness in replicating traffic and transport behaviour. Performing an analysis into their effectiveness would require a dataset to provide a ground truth, with no published dataset identified and is acknowledged as a challenge in the validation of traffic models and microsimulation [46, 113].

Investigation was undertaken into identifying supporting data for the evaluation of real-world scenarios. National traffic flow data is routinely published [149], but sensor locations are sparsely distributed along major national roads and are predominantly situated between or around urban environments. The use of local traffic flow data from the Nottingham SCOOT system was also investigated, but encountered issues with fragile data collection, inactive sensors and the need for extensive manual data preparation to assign geographic coordinates to sensors. Performing primary research to gather a traffic flow dataset for a target area would have significantly expanded the scope and resource requirements of the project.

Investigation was also undertaken into the population synthesis process to use aggregate census data as the basis for constructing a scenario and utilising travel surveys for generating activity patterns. These datasets were successfully converted into RDF algining with the core schema for the knowledge-base (Chapter 4). However, the available published spatial datasets contained either limited detail [123] or suffered from data completeness issues [123]. Therefore, further datasets were required along with additional research into techniques to align the population, activities and spatial locations to complete the Knowledge-Base Construction process, which is also acknowledged as an area of ongoing research [84].

An additional approach for evaluation would be comparison to existing implementations, but the resource requirements and model availability made this impractical in the project timescale. Conversely, both of these issues form part of the overall objective that this work is seeking to address by improving the accessibility of both models and datasets. Therefore, the evaluation considers the prototype and framework implementation in the context of a constructed scenario to consider the schema, prototype and framework rather than the efficacy of the

generated travel demand.

## 7.2 Construction of Travel Demand Generation Prototype Scenario

The outcome of travel demand generation is influenced by the modelling and implementation choices of its components and the scenario data to which the process is applied. This section will describe the scenario produced of five thousand individuals that is utilised in the following sections. These individuals were assigned activities and locations to visit during the course of the scenario period of one day.

The scenario was generated using a developed application rather than being derived from published data. The application applied random processes to produce consistent RDF datasets that follow the previously discussed schema. The properties of the generated dataset and schema were derived from data fields present in published census [114], road network [123], travel diary [14] and travel survey [2] sources.

This approach was selected to allow the scenario data to be quickly modified and developed as the project progressed. The utilisation of real-world datasets would have required further investigation into additional techniques for retrieving and reconciling across datasets. This would have expanded the scope of the project and potentially introduced issues in aligning the various data sources. An area of future work is the investigation of these processes and their incorporation into the framework as part of the overall travel demand generation process discussed previously (Chapter 3).

An implication of this approach is that there is no contextual influence of land usage. Locations, people and activities are distributed around the road network in random positions. Therefore, there is no clustering that may be expected in typical land-use, such as housing estates, industrial zones and retail districts.

The scenario knowledge-base was constructed using a randomly generated road network of 14km by 8km produced by SUMO simulator's NETGENERATE application [46] and converted into RDF road network schema using an XSLT

template (see Fig. 4.33). This road network can then be used by the travel demand generation and both traffic simulators. The generated roads are all single lane with pedestrian pavements running alongside. Pedestrian crossings were situated at road junctions, when required by the simulator, i.e. SUMO.

This means that there is a high level of pedestrian access throughout the network rather than pedestrians incurring additional travel durations they may experience due to accessing foot bridges and crossings or the lack of pavements along busy roads. The pedestrian model of SUMO gives pedestrians priority at crossings with no delay waiting for traffic light phasing to control vehicles. MATSim does not currently simulate this level of detail for pedestrian modes. Therefore, pedestrian modes are highly favoured in terms of access and delaying factors through the road network at simulation.

The phasing of traffic signals has also not been incorporated into the knowledge-base. There are no identified datasets providing this data and only SUMO utilises the data in its simulation. This again highlights the limited benefit of trying to utilise a road network from a public data-source. Although the topology of the road network will reflect a real-world location the behaviour of traffic signals at junctions will not and so weaken the simulation outcomes. Instead the default SUMO approach to traffic signalling has been applied. The quality and appropriateness of published road networks for traffic simulation also limits their use without substantial reconciliation due to a lack of data on the number of lanes, presence of traffic lights, maximum speeds and presence of footpaths.

All roads in the network were allocated the same maximum speed. This means that although routes are based on travel time they will also be the shortest distance. Therefore, there are no routing effects from higher speed trunk roads, which often have multiple lanes and so higher capacity. The load on specific roads will be dependent upon the generated people and locations rather than also incorporating factors from the road network topology.

RDFS inferencing was applied to the knowledge-base following the RDF schema and published public schemas described previously in Chapter 4. This provided automatic inferencing and data validation, e.g. datatype checking, cardinalities and inferred sub-class membership and sub-property relationships. Applying OWL2 inferencing and additional property relationships would enable more

diverse inferencing, e.g. relationships between locations and persons based on common activity types and person mode usage based upon vehicle usage.

A dataset was produced based on a road network containing one thousand *residence* locations, five *education* locations, one hundred *employment* locations, five *freight depot* locations and thirty locations each for *retail*, *leisure*, *personal business*, and *freight delivery*. Each location was assigned geospatial coordinates randomly selected from a set of evenly spaced points running alongside road links.

Vehicles were permitted to access all locations except for *leisure*, while pedestrians had access to all locations. Therefore, no differentiation is made in the scenario between large delivery vehicles, car sized vehicles and bicycles. This was a simplification of the scenario data generation and would be supported by the travel demand modelling.

Locations were also selected at the road links near to the cardinal points and a central train station to provide starting points for external non-resident travellers using *transport link* activities. The road network and locations are shown in Figure 7.1.

Each *residence* location contains a single *household* Travel Group consisting of four persons to simulate four thousand *resident* individuals. *Households* were assigned one of the ten Activity Pattern Sets with each person in the group being allocated a single Activity Pattern. Ten Activity Pattern Sets were manually created with each consisting of four Activity Patterns. The Activity Patterns started and ended with *home* activities and consist of one or more activity blocks ranging from half an hour to nine and a half hours. The start and end activities could be any Location or Activity Type but all were assigned to *residence* Locations.

The activity pattern's start and end times were chosen from the four quarters of the hour, with later random variation of plus or minus fifteen minutes. Lunch time and evening activities were included around core day time *education* and *employment* activities, but interrupted by lunch time, with a *home* activity prior to evening activities.

Each *resident* Person was randomly allocated activities at locations according to activity types with one *employment* and *education* location and ten each for *retail*, *leisure* and *personal business* locations. Locations were assigned multiple

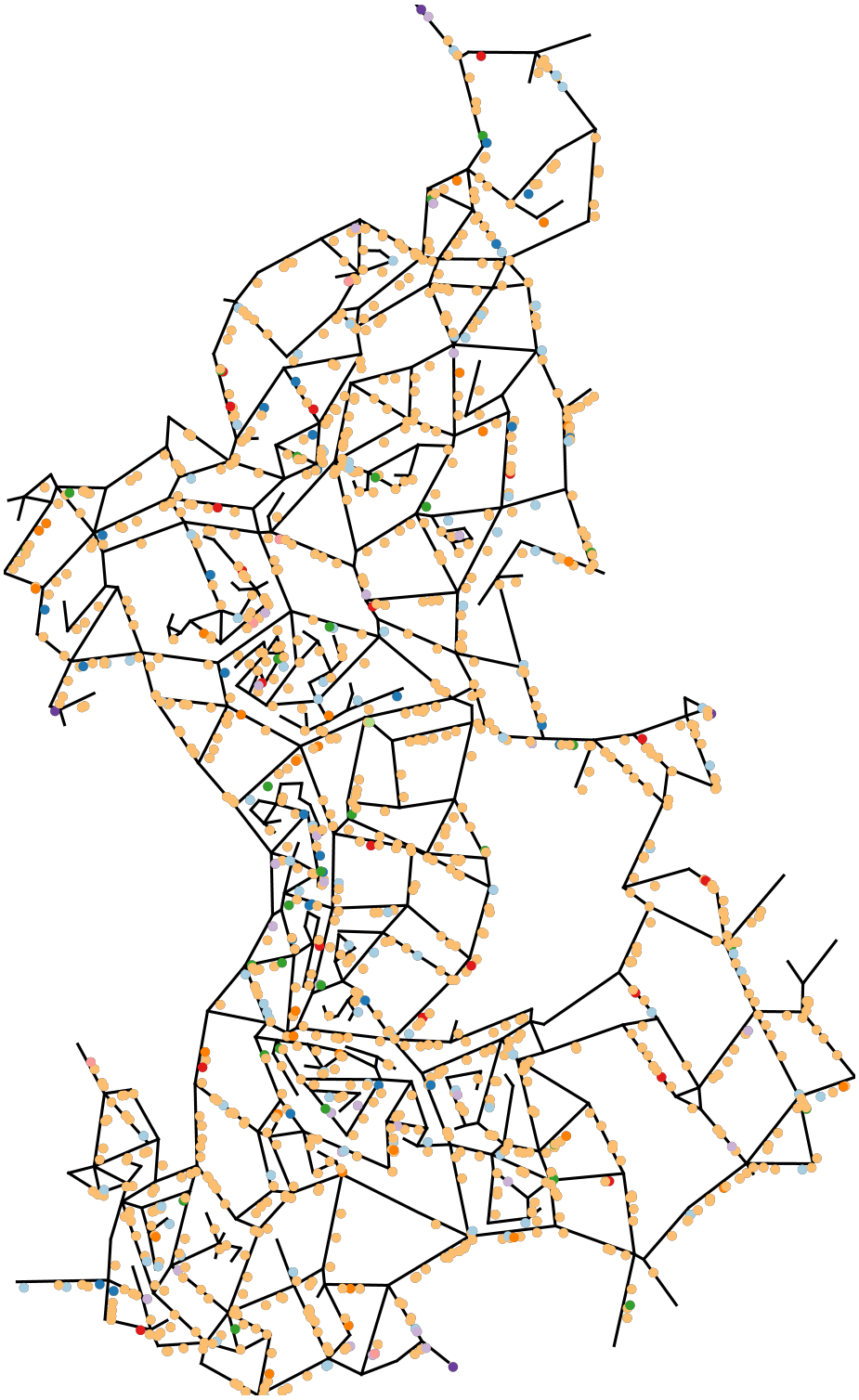

Figure 7.1: Map of road network and locations.

activity types. House residences provide *home* and *leisure* activity types due to *leisure* activities also including socialising with friends and family. Other locations provide *employment* and other related Activity Types. This demonstrates the potential for multiple activities and alternative activity types to take place at a single geographic location.

*Non-resident* persons were similarly assigned locations for activities but were not assigned *residences*. Instead these were allocated to *transport link* activities at edge of network Gateway Links and Train Station locations. Two hundred Travel Groups were split evenly between the five *transport link* locations with four Persons per group. Activity Pattern Sets following those of the *resident* persons were produced but with *residence* activities replaced by *transport link*.

*Freight driver* persons were allocated an activity at a *freight depot* location to start and end the schedule. Each *freight depot* was allocated a Travel Group consisting of ten *freight drivers*. All *freight drivers* were assigned the same Activity Pattern of deliveries every thirty minutes throughout the day but with varying travel range.

No *freight delivery* locations were asserted for the *freight drivers*. Instead potential locations were searched dynamically according to proximity of the current location and travel range of the Activity Pattern item demonstrating contextual selection. Destination selection was equalised through the *freight driver* utility coefficients and freight vehicle mode parameters. All the described Locations and Activity Types could be intermixed so that *residents* and *freight drivers* may travel out to *gateway links* and *freight delivery* activities can take place at *residences*.

The described person-types, activities and locations were applied in the user schema, rather than prototype design, and can therefore be modified by the user. These have been selected to illustrate typical domain concepts a user may wish to model. The implemented prototype is able to operate upon these in a generic manner while the user can still apply selection to use alternative modules, e.g. trip planning for freight. This is in contrast to some travel demand models, e.g. CEMDAP which divides the population into workers and non-workers with fixed activity travel patterns [47].

The *resident* group were split into *adult* and *child* groupings. Each *adult*

*resident* was allocated a single private Vehicle from a distribution covering *car* (1:2), *motorcycle* (1:6) and *bicycle* (1:6) or no vehicle (1:6) with children only allocated *bicycles* (1:4). All *non-residents* arriving via *gateway link* were assigned either *car* (3:4) or *motorcycle* (1:4) Vehicles. Those *non-residents* arriving at the train station were not assigned vehicles. All *freight drivers* were assigned *Heavy Goods Vehicles*. All *residents* and *non-residents* were assigned personal *walking* Modes, while *freight drivers* were not assigned a *personal* Mode to enforce continuous usage of their vehicles.

Personal utility coefficient weightings for the Random Utility Model (RUM) were specified according to the three person-types as a model simplification rather than technical requirement. Each Mode was assigned max speed, fixed cost and variable cost definition for the Travel Scenario. The RUM was tuned, except *freight drivers*, to provide a walking preference for stages shorter than 1.4km and using vehicles for longer trips as can be noticed in Figure 6.7. This threshold is intended to provide a mix of mode usage and was based upon indicative traveller walking distances [14].

It can be seen from the description in this section that the data requirements of the travel demand generation process are not trivial. Several items of data have a strong influence over the behaviour of the travellers including the activity patterns, assigned locations and accessible modes. An area of future work is the generation of public transport data for the usage in the scenario as this would broaden the mode choices available to all travellers.

## 7.3 Evaluation of Travel Demand Generation Prototype

This section examines the generated results from executing the previously described scenario. It considers the schedules generated by the travel demand stage of the process. There is then examination of the outcome from simulating the generated schedules with two integrated traffic simulators. Finally, there is identification of issues in varying the number of participants in the prototype scenario and summary of the evaluation.

### 7.3.1 Activity Intervals and Travel Stages of Generated Schedules

The travel demand generation process was executed over the period of one day for all five thousand individuals. These five thousand individuals were split across three person-types of 4,000 Residents in 1,000 Travel Groups, 800 Non-Resident in 200 Travel Groups and 200 Freight in 20 Travel Groups. The number of activity intervals and travel stages produced varied across the person-types according to the activity patterns for each as shown in Table 7.1. The Freight group followed the same regular activity pattern while the other two groups picked from a choice of ten prepared patterns.

| | Person Type | Mean | Std. Dev. | Min | Max |
|---|---|---|---|---|---|
| Activity Interval | Freight Driver | 18 | 0 | 18 | 18 |
| | Non-Resident | 4.703 | 1.418 | 3 | 8 |
| | Resident | 4.611 | 1.395 | 3 | 8 |
| Travel Stage | Freight Driver | 17 | 0 | 17 | 17 |
| | Non-Resident | 5.295 | 2.697 | 2 | 14 |
| | Resident | 4.782 | 2.632 | 2 | 19 |

Table 7.1: Table of scenario activity intervals and travel stages by person type.

The distribution of activity intervals in progress across the entire scenario day is shown in Figure 7.2. The scheduling process retained 25,806 out of 26,280 (98.2%) activity pattern items as activity intervals with 26,765 travel stages planned. The figure shows the switch from *home*, *delivery* and *transport link* activities at night to day time activities. The structure and hierarchy of these activities is determined by the schema and data, so the user is able to expand and modify as required.

The inverse of the activity intervals is the scheduled travel stages as individuals who are not performing activities would be travelling. The travel stages can be identified by the declining number of activities corresponding with the peaks seen in Figure 7.3. There is a clear domination by walking as shown by Table 7.2 where over half of the travel stages do not use a vehicle. This is not surprising as this mode is always an option for travellers to utilise and is also used by them to reach vehicular modes if the vehicle has been positioned at another location.

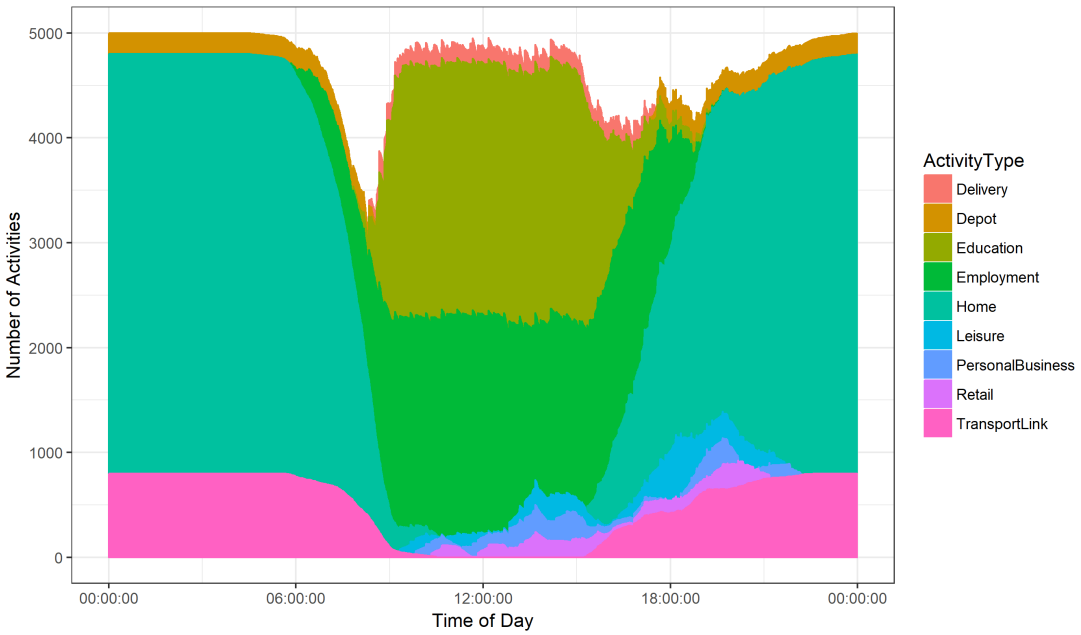


Figure 7.2: Number of activities by activity type per one-minute interval.

| Mode | Count | Share |
|---|---|---|
| Bicycle | 2,484 | 9.28% |
| Car | 4,341 | 16.21% |
| Heavy Goods Vehicle | 3,400 | 12.70% |
| Motorcycle | 1,291 | 4.82% |
| Walk | 15,249 | 56.97% |

Table 7.2: Table of mode share for travel stages.

The prevalence of walking travel is influenced by the positioning of locations, activity patterns, mode access, vehicle availability and the trip selection process, in this case a Random Utility Model. The trip selection process only favours walking in trips less than 1.5 kilometres in a road network covering 112km$^2$ with only a single type of location inaccessible to vehicles. Yet, the modelling of pedestrians in SUMO simulator has only recently developed a pedestrian model that influences vehicle travel and MATSim handles vehicle and pedestrian modes separately with no interaction.

The aggregated values of the travel stages by mode for distance travelled and duration are shown in Table 7.3 and Table 7.4 respectively. It can be seen

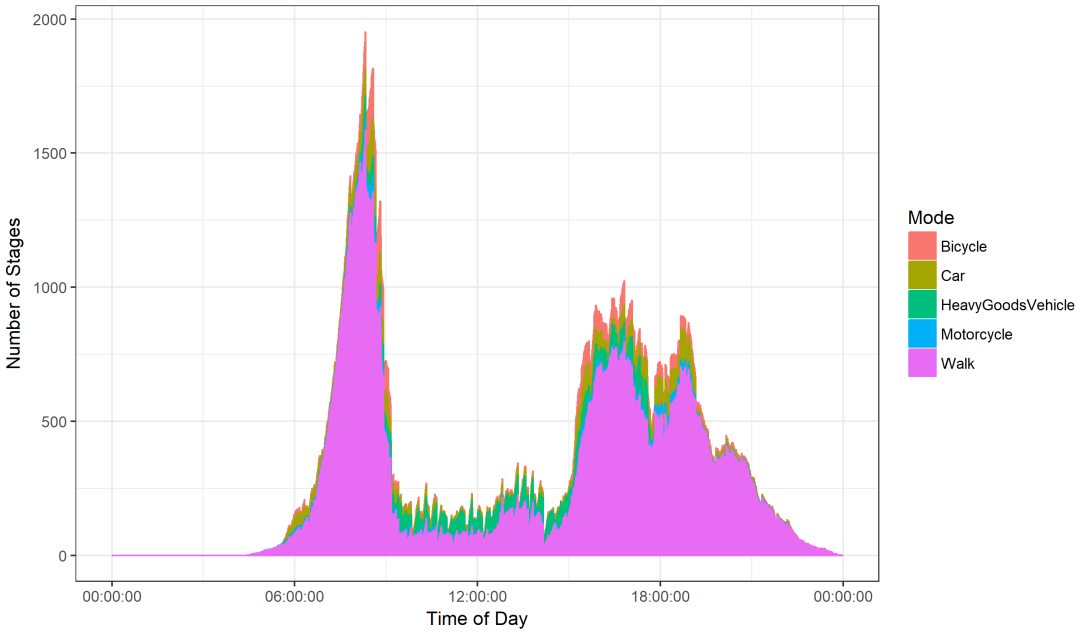


Figure 7.3: Number of travel stages by simulator per one-minute interval.

| Mode | Mean | Std. Dev. | Median | Min | Max |
|---|---|---|---|---|---|
| Bicycle | 6,279.18 | 3,555.556 | 5,300.227 | 12.435 | 18,922.75 |
| Car | 6,530.543 | 3,468.799 | 5,693.75 | 16.073 | 19,057.45 |
| Heavy Goods Vehicle | 7,084.732 | 1,990.131 | 6,750.848 | 2,149.135 | 15,110.62 |
| Motorcycle | 6,762.546 | 3,632.69 | 5,798.207 | 20.883 | 17,053.04 |
| Walk | 2,597.206 | 3,376.757 | 1,225.354 | 0.115 | 16,421.44 |

Table 7.3: Table of travel stage distance (metres) by mode.

that the walk mode is used for shorter stages despite taking longer to complete. The distribution in distance travelled is wide as indicated by the large standard deviation and range between minimum and maximum distances. The trading off between modes, other than vehicles to walking, is not taking place due to the single allocation of vehicles and lack of public transport.

The maximum walking distances, and corresponding durations, reflect that 3 in 4 of the *child* and 1 in 6 of the *adult* Residents were not allocated any vehicle and therefore were forced to walk, so contributing to the walking prevalence in Figure 7.3. Therefore, long walking journeys up to 2.5 hours are produced due to the demographic data that has been used, and the absence of public transport,

| Mode | Mean | Std. Dev. | Median | Min | Max |
|---|---|---|---|---|---|
| Bicycle | 702.884 | 397.998 | 593 | 1 | 2,118 |
| Car | 469.862 | 249.460 | 410 | 1 | 1,370 |
| Heavy Goods Vehicle | 509.664 | 143.108 | 486 | 154 | 1,084 |
| Motorcycle | 486.511 | 261.168 | 418 | 2 | 1,227 |
| Walk | 1,451.057 | 1,886.587 | 684 | 1 | 9,174 |

Table 7.4: Table of travel stage duration (seconds) by mode.

rather than the activity-based model.

The Heavy Goods Vehicle mode can also be seen to be searching for locations above the minimum threshold for each trip. This threshold was not enforced for other modes and so very short trips are demonstrated. Enforcing a minimum travel distance would produce a greater volume of travelling, but does not fit with the utilitarian view that humans select the most efficient option available. Therefore, other factors, such as location preference or popularity, should be considered to design out, or give greater substance, to choices than proximity.

These mode choices for short trips can also reflect the mandatory requirement to return a vehicle to the starting location at the end of the day and walking transfer stages to collect or drop-off a vehicle to access a location. Therefore, enforcing one type of behaviour can introduce complexities in reflecting other desirable behaviours. Overall, a diversity of trip distances and durations have been produced across the modelled modes.

The distribution of travel stages being completed consecutively can be seen in Table 7.5 with the majority of trips only requiring a single stage. The overall ratio of travel stages per trip was 1.286 ($26,765\ travel\ stages/(25,806\ activity\ intervals - 5,000\ initial\ activity\ intervals)$). The low number of three stage trips can be attributable to only a single location type not providing vehicle access and there being no provision of public transport. Therefore, these trips will be of a walk-vehicle-walk pattern. However, this does demonstrate the implemented multi-stage, multi-mode trip planning algorithm being applied (Section 6.4.2).

A final consideration of the travel demand generation is the distribution of travel around the road network. This is visualised in Figure 7.4 which shows the travel along the road network as planned by the schedules. The frequency of link

| Stages | Count | Share |
|---|---|---|
| 0 | 0 | 0% |
| 1 | 15,422 | 57.62% |
| 2 | 4,809 | 17.97% |
| 3 | 575 | 2.15% |

Table 7.5: Table of travel stages between activity intervals.

usage, in either direction, has been split into five groups across the range from dark green for low usage to dark red for high usage, with black for no usage. It can be seen that a wide area of the road network has routes planned along it and travel throughout. It can be seen that the many roads have low usage by the prevalence of dark and light green. The distribution of road links in trips is illustrated in Figure 7.5 and is left skewed with many links being used a small number of times and a long tail of low number but high frequency cases.

As discussed in the previous section, the positioning of locations was applied using random distribution methods and so there is no spatial context to land-use from historic, social and economic developments which may be found in a real world scenario. All roads in the road network have the same lane capacity and speed so travel is determined by shortest path rather than variation in infrastructure. These are characteristics of the data supplied to the travel demand process, and not an implementation constraint. However, Figures 7.4 and 7.5 do illustrate that travel is not isolated to a small section of the road network and that certain areas are more frequently travelled.

Figure 7.4: Map of road network link usage density across five groupings.

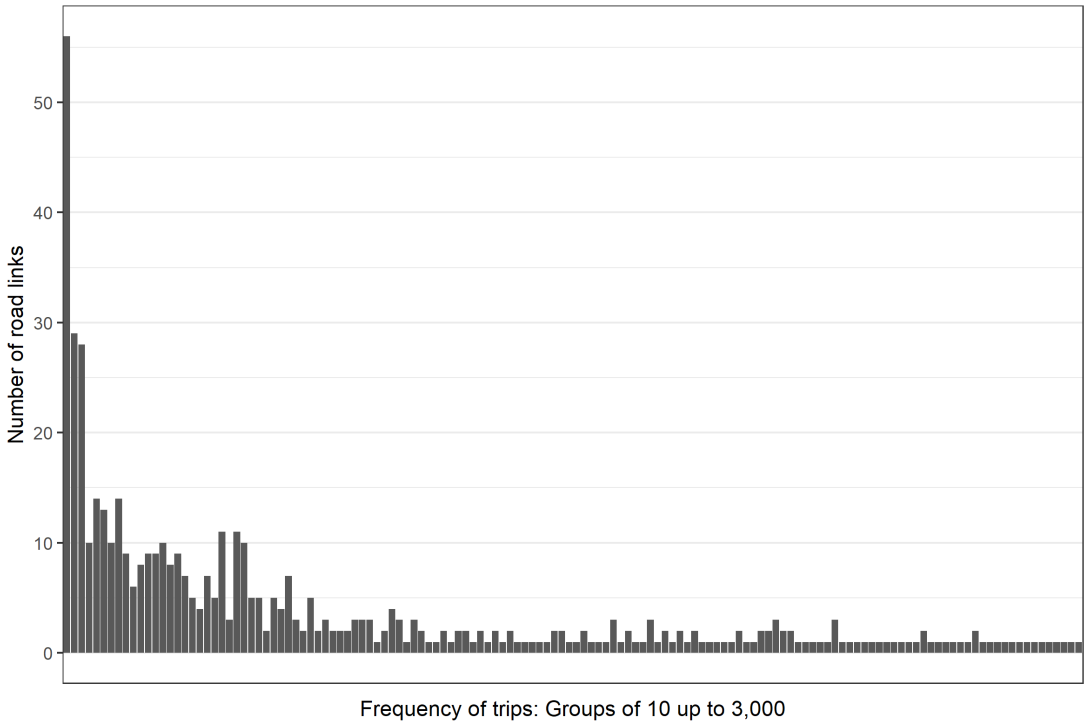


Figure 7.5: Distribution of trip frequency for road links from schedules.

### 7.3.2 Variation of Traffic Simulation Results to Generated Schedules

The generated schedules of the travel demand process represent the ideal planned activities and travel of the individuals. The traffic simulation stage of the process models the interaction of these individuals with each other and the network infrastructure environment. Therefore, it is expected that the generated schedules and simulation results will not exactly align while variations in modelling, design and implementation choices introduce the potential for differences in outcomes between simulators.

The trips in progress for the schedules by mode have shown the general M-shaped curve of weekday commuting in the morning and afternoons [2, 16] in Figure 7.3. There are additional lunch time and evening travel periods with the general pattern directly influenced by the activity pattern templates. Figure 7.6 shows the difference between the planned schedule for travel and the trips in

progress during traffic simulation.

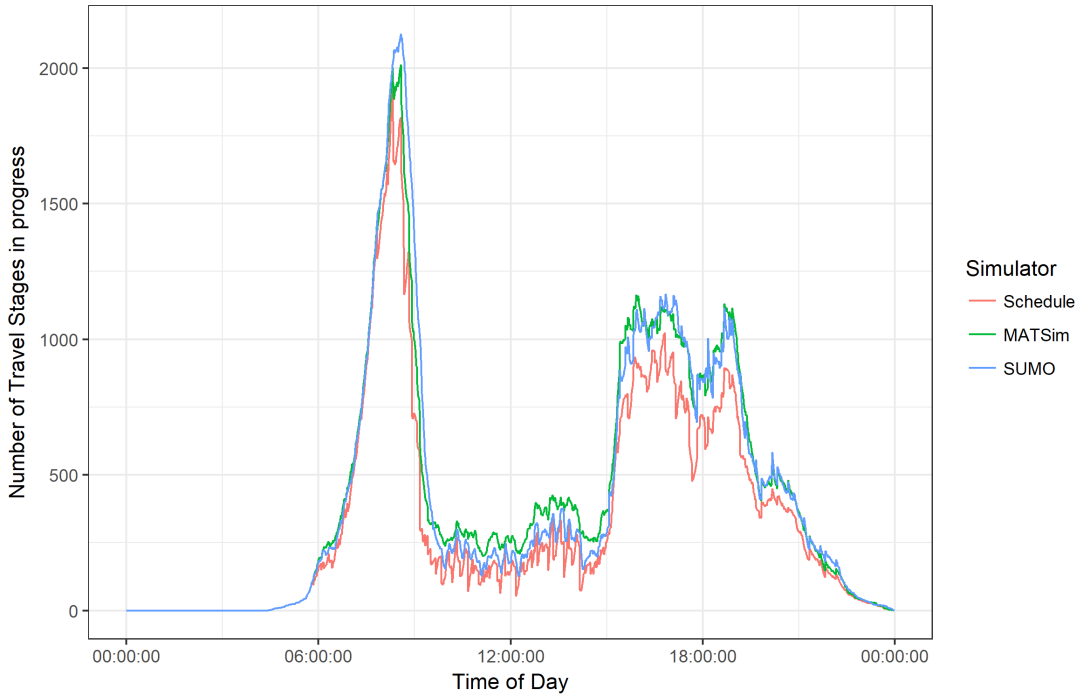


Figure 7.6: Number of travel stages by simulator per one-minute interval.

A greater number of trips in progress would be expected and both simulators can be seen to have higher numbers. This is an indication of modelling factors, e.g. vehicle acceleration and deceleration, and interactions, e.g. traffic queuing, which delay the travellers in their journeys. This is shown in Table 7.6 by the higher mean and median values for the simulators compared to the schedule. The high standard deviations shown that the data is widely distributed indicating trips are being undertaken with a range of durations. The MATSim simulator has the same maximum value as the schedule while SUMO has some form of delay.

| Simulator | Mean | SD | Median | Max | Min | Count |
|---|---|---|---|---|---|---|
| Schedule | 1056.37 | 1506.058 | 511 | 9174 | 1 | 26765 |
| MATSim | 1347.628 | 1487.298 | 911 | 9174 | 0 | 26765 |
| SUMO | 1327.307 | 1765.498 | 743 | 10744 | 1 | 26765 |

Table 7.6: Table of simulator and schedule duration for travel stages.

The graph illustrates that in the peak morning period the SUMO simula-

tor has a higher volume, but during the midday period MATSim is consistently greater. This illustrates the need for comparison to be made between simulators as although the same travel demand input is provided there is a difference in the simulated output. There is also a general indication of SUMO tracking the underlying schedule pattern more closely, although not exactly, than MATSim through less smoothing of the peaks and troughs. A closer examination of the difference between schedule and simulation is illustrated in Figure 7.7. This graph shows the mean delay between the ideal schedule and the simulated travel of the traffic simulators.

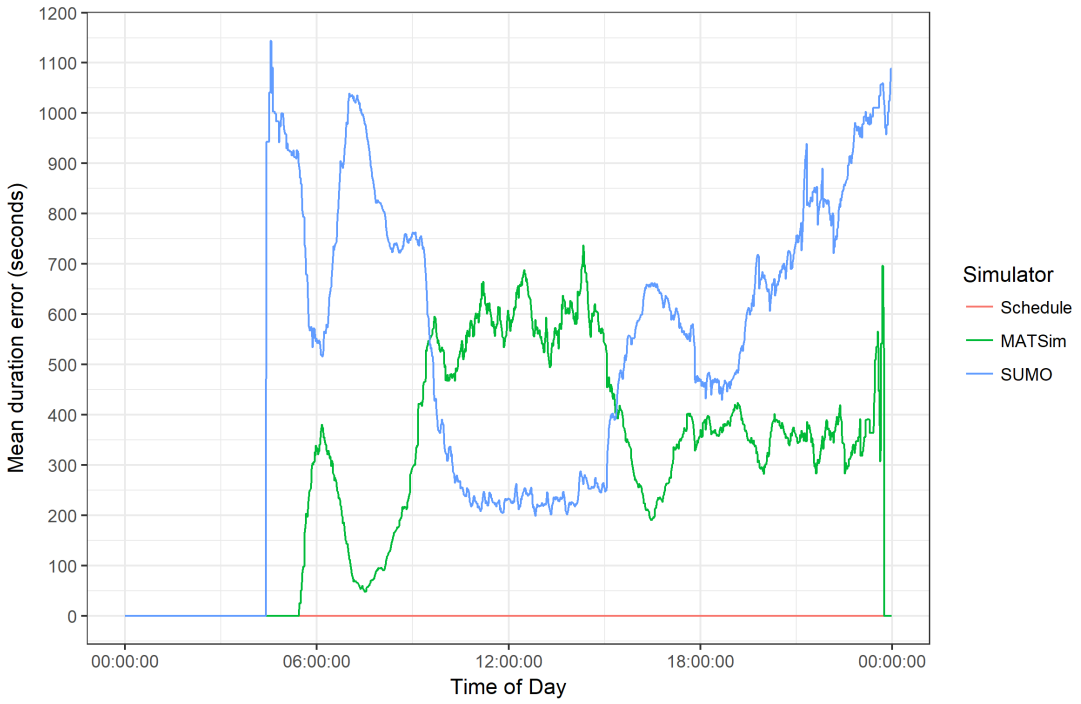


Figure 7.7: Mean error of travel stages by simulator per one-minute interval.

In SUMO, extreme values can be seen at both ends of the day when travel volumes are lowest, indicating severe disruption for a few travellers. The large spike prior to 06:00 in particular raises further questions as the volume of travellers is very low and it is not until about 09:00 that peak volume is reached. In a low volume situation, the delay, or error, should be minimal as queuing and vehicle interactions should be minimal. Examining MATSim, the mean delay peak occurs during the lower volume midday period, excluding an end scenario

spike, suggesting more generalised delays are being experienced.

The importance of a delay can be considered in both relative and absolute terms. Figure 7.8 illustrates the mean delay as a percentage of the scheduled duration. In the case of SUMO the large errors at the start of the travel period are relatively less than encountered in the rest of the period. There is a substantial difference in the scale between the relative percentage delays in SUMO and MATSim, while SUMO peaks at 40% it can be seen that MATSim reaches 100% delays.

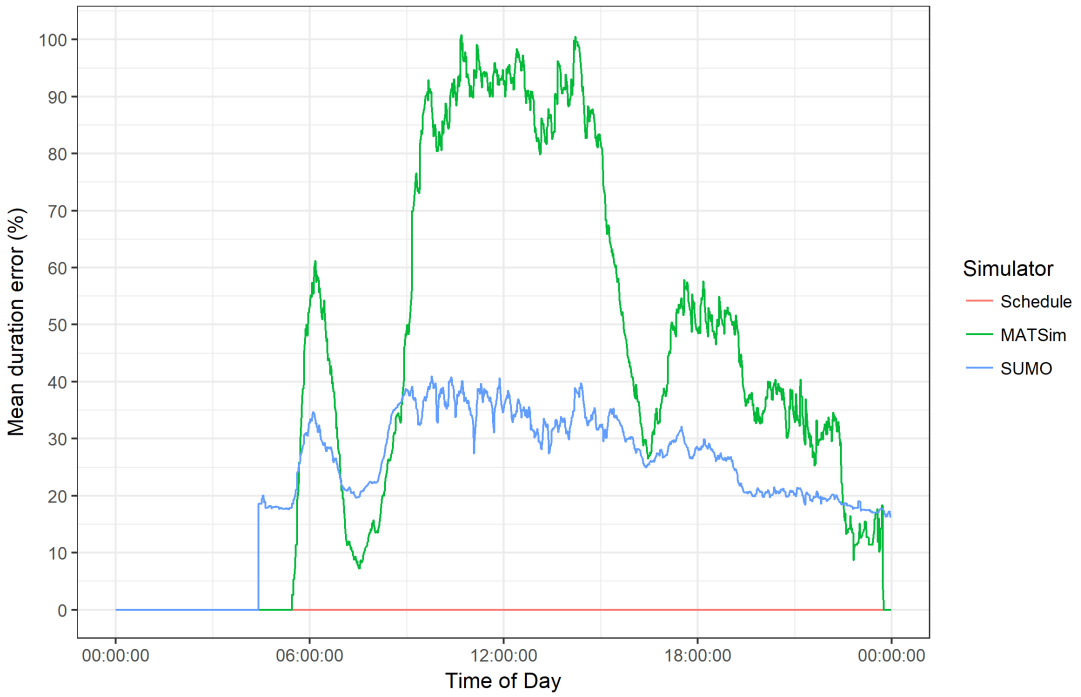


Figure 7.8: Mean percentage error of travel stages by simulator per one-minute interval.

These are aggregated mean values and so the worst case will be higher with outliers being hidden in high volume situations. Figure 7.9 shows the maximum error experienced for each minute segment by an individual for the whole trip, i.e. not the delay at that point in the trip. The same individual will report the maximum value from segment to segment until they complete their trip or are exceeded by another individual. The graph illustrates that while SUMO has a high initial error at the beginning of the travel period the maximum error is

typically simulated in MATSim. This maximum error is at a much higher level and peaks at 5,041 seconds, or 83 minutes, delay as shown in Table 7.7.

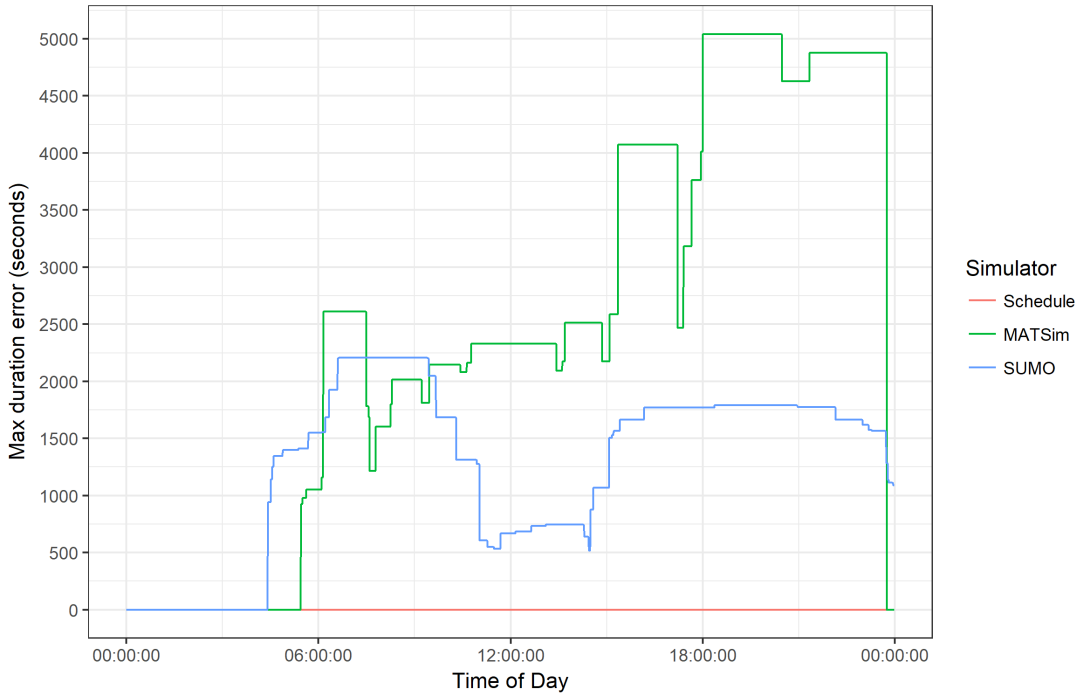


Figure 7.9: Maximum error of travel stages by simulator per one-minute interval.

| Simulator | Mean | SD | Median | Max | Min | Count | Negative |
|---|---|---|---|---|---|---|---|
| MATSim | 291.2577 | 372.8335 | 165 | 5041 | -2559 | 26765 | 707 |
| SUMO | 270.937 | 293.4432 | 191 | 2209 | -37 | 26765 | 186 |

Table 7.7: Table of difference between simulator and schedule duration for travel stages.

In both simulators the means are at similar levels of a 4 to 5 minute delay compared to the schedule, but have lower median values showing lower centre to the distribution. MATSim has a much greater spread than SUMO between the maximum and minimum values. The schedule duration represents the best case scenario for travel of maximum road or mode speed and no interactions for queuing or crossings. However, both simulators report travel stages that are quicker than this best case scenario, as indicated by the *Min* and *Negative* count columns.

In SUMO the values are relatively small and occur for a small percentage of travel stages (0.69%). All the values occur for the walking mode and are less than 8 seconds. The one exception is a heavy goods vehicle which is responsible for the 37 second minimum difference. In all cases the individual's difference to the schedule is smaller than the duration. The individuals are being simulated as travelling and the difference could be due to a simulator error or misalignment between travel demand and simulator road networks. The internal representation of road networks used by SUMO applies areas for junctions that are not included in edge and node representations, even that used by SUMO as an input. These areas truncate the lengths of edges and so may modify the distances travelled. However, the exact cause has not been identified.

In MATSim the values occur more frequently, but still represent a relatively small percentage of travel stages (2.64%). However, the difference values are much larger and frequently exceed the simulated travel stage duration, i.e. the MATSim duration is less than half the best-case duration of the schedule. The majority of cases apply to the walking mode but cases of all other modes are found.

In these non-walking modes the duration is often 0 to 2 seconds indicating that the simulator is teleporting the individual. Teleportation is a feature of both simulators to overcome issues with gridlock or individuals expecting vehicles that are out of position. However, the former only applies after a minimum waiting duration and neither situation was reported by the simulator.

In the walking modes a large range of values are found in MATSim's simulator results, and all have non-zero durations. This suggests that the simulator is not teleporting the individuals, but instead simulating shorter travel durations. However, an overestimation in the duration of the schedule's travel stages for walking by the travel demand generation process should also be reflected in the SUMO simulator results. However, there is not the volume or duration of mismatches in the SUMO simulator results to suggest this is the case.

The difference in duration for the first 186 cases reaches upto 41 seconds compared to SUMO's 8 seconds. The spread of MATSim duration differences continues until 849 seconds. There is then a jump to a single outlier with the maximum value of 2,559 seconds. This individual is reported in the logs as

being teleported due to being *stuck* without a vehicle. However, the vehicle was positioned in this location and so the cause of the individual being *stuck* is unknown and an anomaly.

These negative differences are illustrated in Figure 7.10 which shows the minimum error experienced for each minute segment by an individual for the whole trip, i.e. not the delay at that point in the trip. The same individual will be shown until their value is exceeded or the trip is completed. The SUMO simulator spike can again be seen at the start and end of the simulation when volumes are at their lowest. The minimum and maximum of these trips are both high resulting in a high mean value.

This supports the suggestion from Figure 7.8 that these are long duration trips that are taking longer in the simulated environment rather than an unexpected delay. There can then be seen the persistent occurrences of shorter durations than the schedule which occur even during the peak morning and afternoon periods. The MATSim outlier mentioned previously can also be seen around 22:00 where the individual becomes *stuck* without a vehicle rather than due to travel.

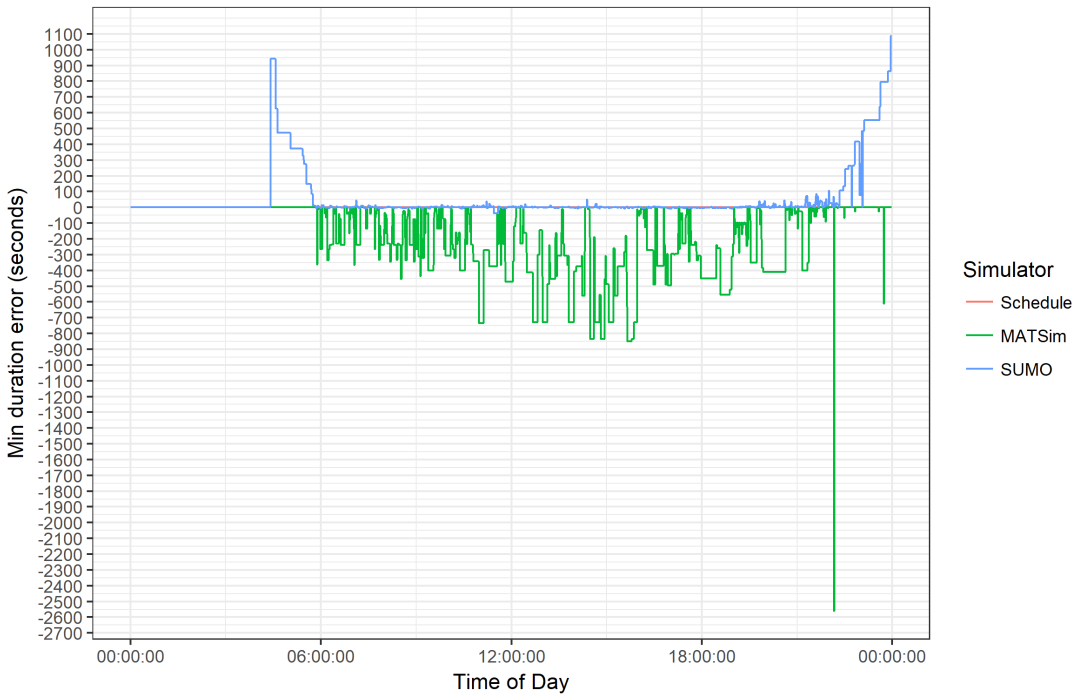


Figure 7.10: Minimum error of travel stages by simulator per one-minute interval.

### 7.3.3 Issues and Summary of the Prototype Scenario

The construction of the prototype scenario encountered several issues that have been discussed in the previous section. However, there were several issues encountered in varying the number of participants in the scenario.

The examined scenario schedules the travel for 5,000 individuals in a road network contained in 112km$^2$. This represents a population density of 44.64 people per km$^2$ compared to a United Kingdom density of 272.28 people per km$^2$ [150]. A further experimental scenario was performed for 12,500 individuals and simulated for a density of 111.61 people per km$^2$. There were no issues encountered in the travel demand generation process or MATSim simulation. However, the execution of the SUMO simulator did introduce two points that halted further investigation with larger populations.

Firstly, the output of the simulation, i.e. the simulation completed successfully, caused an error when being parsed by the Java XML library. Initial investigation did not identify an immediate cause of this error or whether the issue resided in the SUMO simulator XML output or the XML parsing library and there was not opportunity to resolve the matter.

Secondly, the SUMO simulator logging reported large numbers of vehicles being stuck and requiring teleportation through the road network. This is a known issue in the design, for which teleportation is the solution, caused by the non-reactive design of entities in the SUMO simulation. In high volume situations the entities can become stuck crossing junctions. This causes tailbacks along their edges that impacts on other edges and can trigger gridlock. While gridlock occurs in real-world traffic scenarios the situation ultimately alleviates but in the simulator will continue until termination with no further travel.

The entities are not designed as agents to consider re-planning their route to continue their onwards journey and there is no strategy of swapping positions of different modes, e.g. bicycles past larger vehicles, or allowing vehicles in the queue but travelling to a none grid-locked edge to make progress.

The first vehicle in the queue waits a set amount of time and then is teleported to the next free edge on their route at a cost of the average speed of the edges it would have driven along. The vehicle behind then begins their waiting period

and so an increasing amount of delay is experienced. This has implications for the results of the simulation as a low waiting time will see these high traffic conditions being quickly resolved through teleportation and so affecting what would have been normal traffic dynamics. A high waiting time will introduce an arbitrary amount of delay that varies according to the waiting time rather than the ability of the road network to manage traffic volumes and ease congestion.

The proposed solution to these grid-lock conditions is modification of the travel demand, junction priorities and traffic light timings. However, modification of this input data could have unintended consequences that undermine the veracity and reliability of the results obtained. Particularly when the scenario's travel demand and road network environment has been configured to model the real-world as closely as possible.

In summary, the travel demand process is able to generate travel demand across for a variety of persons, modes and activities provided by the user through the knowledge-base. The activity patterns provided give a structure to the timings of travel that are undertaken across trips of varying stages and destinations. The produced schedules can then be simulated through multiple simulators with the results captured for further analysis.

There are identifiable issues when comparing the simulation results with each other and the original schedule. The cause of these issues may originate in the integration between travel demand generation and traffic simulator. However, there are indications that they are a consequence of design and implementation choices of the simulator. Overall, this highlights the importance of reliable integration between travel demand models and multiple simulations that the proposed framework provides so that the results can be analysed and compared.

## 7.4 Evaluation of Framework Configuration

This section considers the framework configuration design from Chapter 5 to support the selection of data and redirection to alternative modules. Several of the benefits of the framework configuration approach are qualitative or difficult to consistently measure, such as the use of RDF and SPARQL for configuration and control. This provides a control method that does require some technical

knowledge but with appropriate tools would be comparable to that of passing parameters to a command line interface for a traditional application.

This RDF and SPARQL approach provides greater flexibility in the retrieval of data and execution through the provision of replacement queries and key parameters. However, it risks introducing user error to the process, which the design of query validation and feedback processes described previously (Chapter 5) and implemented in the prototype seeks to address. These processes can assist in identifying error but are not a definitive solution for the reasons previously discussed. However, they have proven to be useful in quickly identifying the cause of issues in the configuration of modules and execution of queries. The specific identification of the query, variable name or URI assists in focussing upon the likely cause of queries not providing any results or taking excessively long to execute.

Another area of importance in the proposed approach, but difficult to quantitatively assess, is the construction of the knowledge-base. An approach that reduces the time spent gathering datasets, preparing a scenario, assembling tools and interfacing the tools allows more time for investigating alternative scenarios or tool-sets. Therefore, any approach that can complete this set-up process quickly would be at an advantage even if the travel demand generation process was slower. This is a task that varies in duration depending upon complexity, user skill and available tools and datasets for both the proposed framework and traditional approaches. Therefore, measuring the total time required to undertake this task would be difficult, while producing generalisable results that could be compared to traditional approaches would be very problematic.

A quantitative element of the process that can be reviewed is the variation in execution duration from the different configurations of the framework. These include the alternative search patterns of the person type; persistent storage of data versus in-memory data during execution; caching versus non-caching of data and comparison between local and remote configurations of modules and knowledge-bases.

The previous sections discussed the evaluation of a scenario with 5,000 individuals and considered both the generated travel demand and the results of traffic simulation. In this section the same scenario set-up is considered but with 1,000 individuals. The change in the number of individuals was made to balance

between scenarios of a reasonable size against the execution duration period so that repeated cases could be performed.

The evaluation focuses on comparing different configurations of the travel demand generation stage and without executing the traffic simulators of the final stage. The execution of traffic simulators forms a discrete task rather than the iterative retrieval between modules and knowledge-base of the travel demand generation process. Therefore, changing from a local to remote configuration would incur a one time overhead for transmission and receipt.

All experiments were performed on the same desktop computer which was a x64 Windows 10 operating system with Intel Core i7-4820K with 16GB and Samsung 850 EVO 500GB. Each configuration was executed ten times to provide consistency in the range of durations. The knowledge-base used in each iteration was copied from an archetype version of the knowledge-base to ensure there was no influence between one iteration and the next. Each iteration was executed in its own Java virtual machine through scripting rather than within the application. Investigation was made into using a Java benchmarking harness [151], but configuration and initialisation issues were found with Apache Jena and the execution throughput could not be gathered without influencing the benchmarks.

This step was necessary due to an identified issue with how the Semantic Web library, Apache Jena, handled the deletion of data for its persistent storage. In testing, instructions to remove previous graphs were issued at the start of each iteration. However, the file size of the knowledge-base did not reduce even when the knowledge-base was no longer being used, i.e. the expectation of a clean-up process at shut-down or start-up.

Further testing found that a knowledge-base repeatedly loaded with exactly the same data that was then deleted continued to grow in size, i.e. a large file size but empty content dataset. The deleted statements were no longer present in the dataset, but it is unclear if the increasing size has any influence on performance. Therefore, each iteration commences with a fresh version of the knowledge-base.

The initial area of analysis is considering the three types of person included in the scenario: *resident*, *non-resident* and *freight*. These groups varied in the number of travel stages as shown previously in Table 7.1. The *resident* and *non-resident* utilise the same set of activity patterns while the *freight* type use a more

frequent pattern and search for their next destination rather than choosing from an asserted list.

A configuration choice available is the use of in-memory and persistent file storage. The in-memory option would be expected to be quicker, but not suitable for very large datasets. The results for both of these choices across three scenarios each containing only one person type of 1,000 individuals across 10 iterations is shown in Table 7.8. The in-memory option shows itself as expected to be faster for all three scenarios, but the advantage varies depending upon the person type.

| Storage | Person Type | Mean | Std. Dev. | Min | Max |
|---|---|---|---|---|---|
| Memory | Freight | 372.693 | 4.848 | 364.924 | 378.955 |
| | Non-Residents | 180.517 | 1.800 | 177.892 | 183.391 |
| | Residents | 270.469 | 4.587 | 264.932 | 278.806 |
| Persistent | Freight | 3,192.122 | 24.358 | 3,164.94 | 3,239.36 |
| | Non-Residents | 850.388 | 9.569 | 836.810 | 864.182 |
| | Residents | 1,491.176 | 122.815 | 1,142.696 | 1,543.365 |

Table 7.8: Table of mean completion durations (seconds) across person-types and storage options (10 iterations).

The Freight scenario is 8.5 times faster in-memory versus persistent storage compared to 4.7 for Non-Residents and 5.5 for Residents. The Freight scenario has a greater workload of approximately 3.4 times the number of travel stages than Residents and Non-Residents. In addition, there is the additional search and routing for potential destination locations when the other person-types may only have a single or few destinations to consider. Therefore, it would be expected that the Freight scenario would be slower to complete. However, the consistent difference between Residents and Non-Resident completions is surprising given the similarities in their configuration.

These figures do not show any additional time spent writing the in-memory generated data to file once the execution has completed, which the persistent option has already achieved. This could be into persistent storage or a file on disk. Serialisation of RDF to file has been found to be very slow during development of the traffic simulator interfaces (Section 6.4.4) with only portions of the knowledge-base taking several minutes to output.

Examination of the pace of progress in generating schedules as shown in Figure

7.11 shows the initial caching period of the first batch is slow to undertake. This first batch also includes query and dataset optimisation by the Semantic Web library that can cause delays in initial query usage. The graph shows the duration required to complete each batch of 20 schedules and shows the minimum and maximum iterations as error bars.

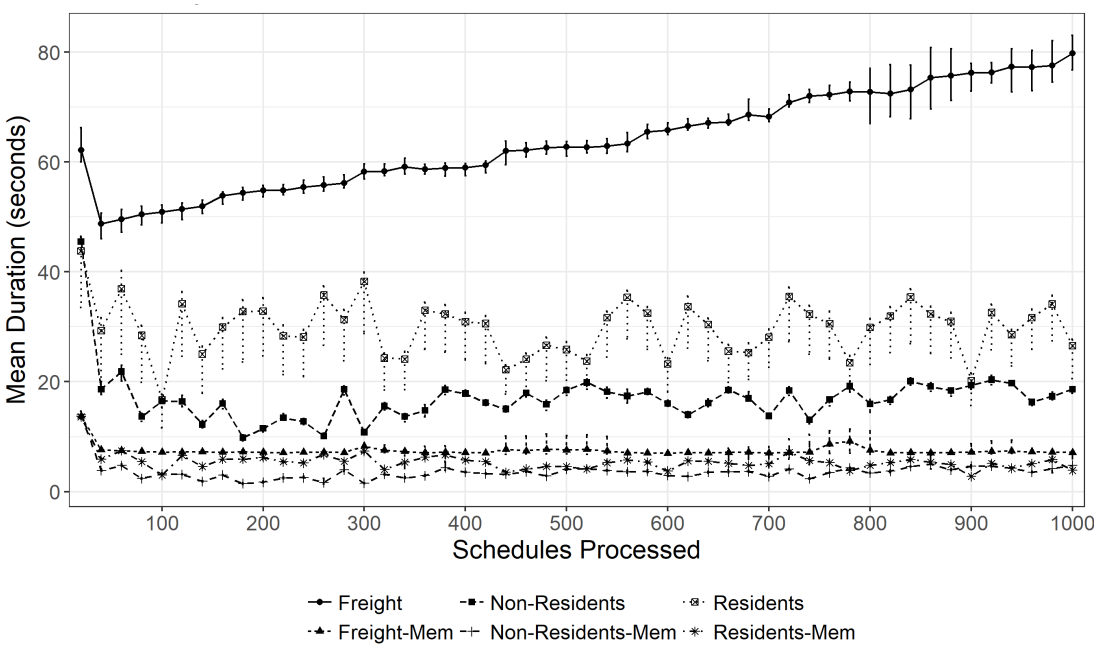


Figure 7.11: Mean duration for completion of person type scenarios (10 iterations).

The in-memory scenarios show some variation between batches, but are generally stable throughout. In contrast, the persistent storage scenarios of Freight and Non-Residents show an upwards trend as the batches of schedules are processed. This suggests that data being written to the knowledge-base is slowing the progress or some form of consumption of resources. The Freight scenario produces approximately 3.4 times the number of travel stages to complete and so more data is written to the knowledge-base. However, the slowing of completions begins immediately for the Freight scenario, but is not reflected in a similar pattern and shallower gradient for the other two persistent storage scenarios.

The Residents scenario is much more erratic in the mean completion duration and the variation as shown by the wide error bars and the range of 400 seconds

shown in Table 7.8. The scenarios were executed in consistent conditions with no other applications to cause interference, but further iterations, hardware set-ups and alternative Semantic Web libraries are likely required to investigate the precise cause.

The exact cause of this upward trend as schedules are completed is difficult to identify without detailed examination of the Semantic Web library, but it indicates that although the persistent storage can hold large knowledge-bases there will be a non-linear increase in completion durations. The in-memory approach should be preferred with potentially writing the generated data to file upon completion or as progress is made.

Another option that has been included in the Framework Configuration is the use of caching to store invariant data rather than retrieve it from the knowledge-base. All results reported are with this option switched on. Testing of a single iteration of the Resident scenario with in-memory storage saw performance drop from 4 minutes 30 seconds to 5 hours 25 minutes 27 seconds, an increase of 72.3 times. Therefore, the application of caching is not really an optional implementation decision. There cannot be a reliance upon retrieving data from the knowledge-base on-demand and instead opportunities to retain data for future use must be identified. These performance figures can be expected to become worse for persistent storage and remote configurations.

The use of caching does not necessarily require large quantities of in-memory data, but can be effective by storing small amounts of information that no longer have to be retrieved from the knowledge-base, whether in-memory or persistent storage. For example, the Travel Scenario information does not change between schedules and caching saves 999 retrieval operations in these person-type scenarios. If each retrieval took 50ms then approximately 49.95 seconds are saved for an object with about a dozen variables. The retrieval of this data can require execution and resolution of SPARQL queries and so can incur additional processing than a look-up of a specific value.

Another implementation choice that can improve performance is execution in parallel across multiple threads. In the implemented *Scheduler* module each Travel Group does not interact or are influenced by any other group. The schedules generated for that group therefore form a discrete batch. This means that

the generation process should be suitable for spreading across execution threads. The outcome of applying this approach can be seen in Figure 7.12 and Table 7.9 where the same Resident scenario is re-used, but with a varying number of threads available. These threads each process a single Travel Group from a queue before proceeding to the next group. Only a single knowledge-base instance is used in an in-memory configuration.

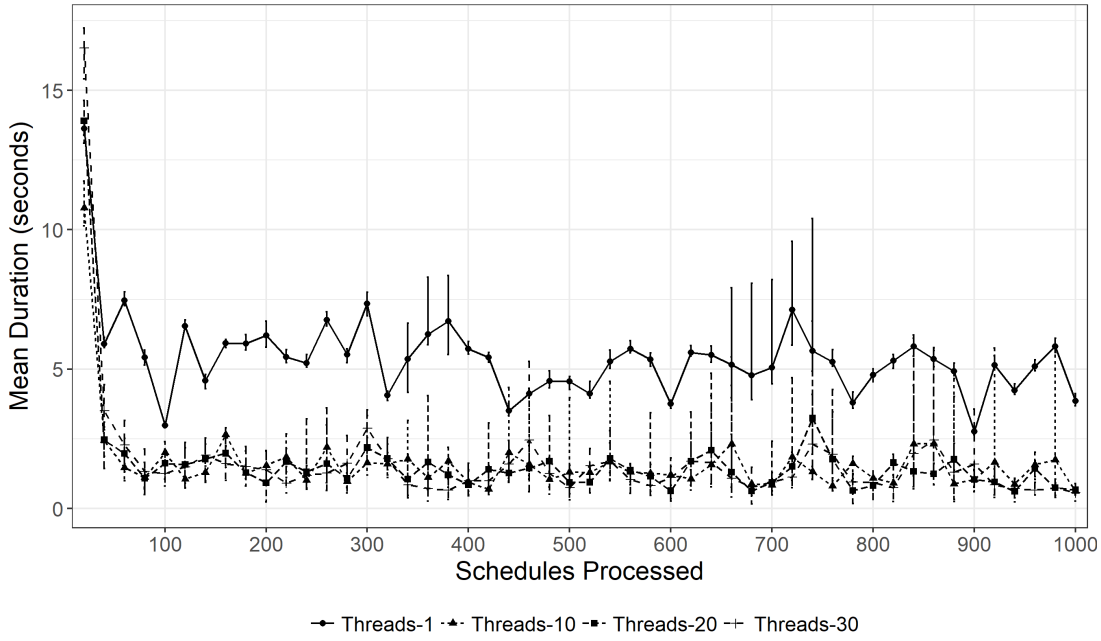


Figure 7.12: Mean duration for completion of in-memory Resident scenario with varying execution threads (10 iterations).

It can be seen that increasing the thread count does improve the completion duration by 3.3 times, but there is a not continuous progression as more threads are used. The use of a single knowledge-base means that write operations can block other write and read operations from taking place. Therefore, an upper limit of improvement is reached as threads must interact with the knowledge-base and are blocked in their progress.

Replication of the knowledge-base so that each thread has their own copy would alleviate this issue. The graphs containing the generated data could then be consolidated back together. This would also allow the potential for the whole problem to be distributed over multiple computers which are then coordinated

| Threads | Mean | Std. Dev. | Min | Max |
|---|---|---|---|---|
| 1 | 270.469 | 4.587 | 264.932 | 278.806 |
| 10 | 81.097 | 2.291 | 77.479 | 83.321 |
| 20 | 81.732 | 1.629 | 78.494 | 83.806 |
| 30 | 83.872 | 2.971 | 79.635 | 88.462 |

Table 7.9: Table of mean completion durations (seconds) of in-memory Resident scenario with varying execution threads (10 iterations).

by the users application to overcome the issue of increasing computational complexity as models and datasets develop, but this represents an area of future work.

This graph also shows the variability in performance of the Resident scenario in the single thread configuration, as shown in Figure 7.11. Several batches have a wide spread of several seconds between minimum and maximum results. These variations could be caused by changes in the order that groups are processed, although there is no obvious reason for this to occur.

These multi-thread scenarios were all performed with the in-memory storage option. The use of the persistent storage should be seamlessly possible with the Semantic Web library by only changing the dataset upon which the travel demand generation is performed. However, issues were encountered when more than one thread was used. These issues revolved around checksum errors for read operations with the dataset being modified.

This is despite the library having a transactional model, which was followed in the prototype implementation, for read and write operations across multi-thread applications to prevent data loss and corruption. These issues were encountered intermittently and so only became apparent when multiple iterations were performed.

The large scenario of 5,000 individuals was successfully executed multiple times with 20 threads over several hours with no issues. However, performing repeated iterations of the different person-types in the 1,000 individual scenarios caused these errors to occur several times. Therefore, results for the persistent storage with multi-threading could not be obtained.

The final area for consideration is the alternative configurations that can be

achieved using the Framework Configuration as described previously (Section 3.4). These configurations would allow the user to directly access remote endpoints for data that forms part of the knowledge-base and to execute implementations of modules. The coordination of these different remote services has been developed and implemented in the prototype (Chapter 5). However, there is network communication cost associated with accessing these remote services.

The previous scenarios have all utilised the local configuration shown in Figure 3.9. In this configuration the file system can be directly accessed and there is no overhead from the HTTP communication (Section 5.3.2.2). Three additional configurations have been explored as described below:

- Local: local configuration with knowledge-base, modules and results all local to the user application.
- Data: remote data configuration with the entire knowledge-base on a remote server while modules and results are local to the user application (Figure 3.10).
- Joined: remote data and module configuration with both on the same remote server and results are returned to the user application.
- Split: remote data and module configuration with each on a separate remote server and results returned to the user application (Figure 3.12).

These configuration scenarios have been executed as in-memory storage and using local services on the same computer, i.e. *http://localhost* addresses through the loop-back of the network adapter. Therefore, there is no consideration of network latency, configuration, throughput or capacity. These results are intended to be indicative of the impact of applying these alternative configurations and represent best-case scenarios. Establishing a complete benchmarking process to evaluate a real-world networking environment was deemed beyond the resources and scope of this project.

The results of these different scenarios can be seen in Figure 7.13 and Table 7.10. The Local configuration is quickest to complete as would be expected. This configuration does not incur the HTTP overhead as it can directly access the

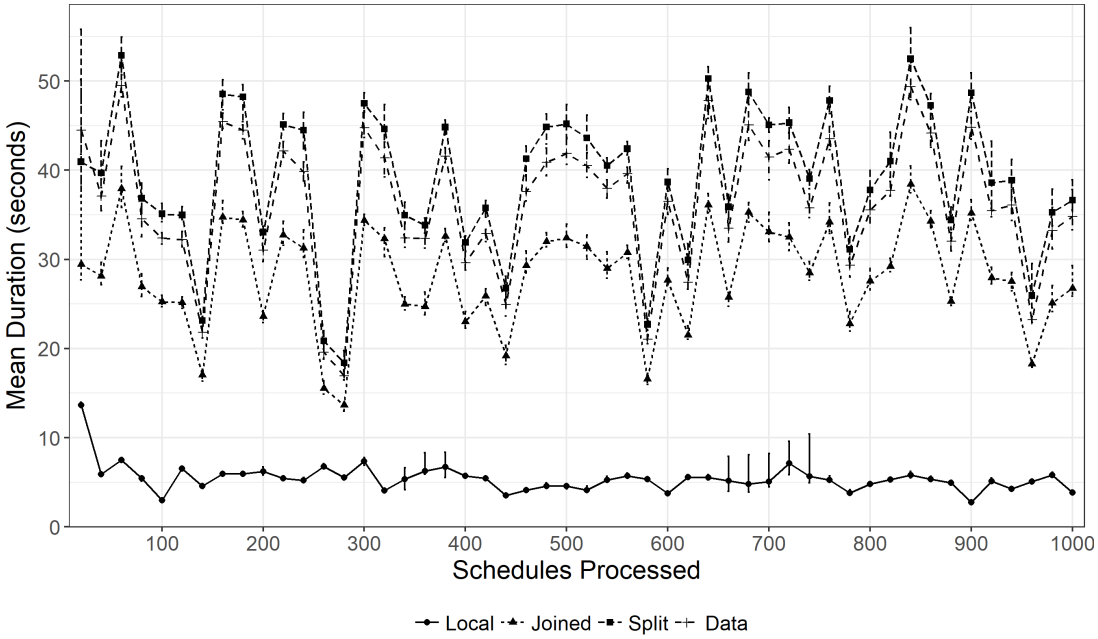


Figure 7.13: Mean duration for completion of alternative in-memory configurations (10 iterations).

knowledge-base and results for all operations. The Joined configuration is next quickest as network communication only incurs when commencing the generation of each Travel Group and when returning the generated results back to the user application. The Data configuration is then closely followed by the Split configuration. These incur the most communication as the modules have to retrieve data about the Travel Group being processed. The advantage of the Data configuration is that it has local access for the storage of results once generated.

| Configuration | Mean | Std. Dev. | Min | Max |
|---|---|---|---|---|
| Data | 1,820.389 | 12.741 | 1,802.013 | 1,838.979 |
| Joined | 1,407.687 | 10.102 | 1,392.702 | 1,421.278 |
| Local | 270.469 | 4.587 | 264.932 | 278.806 |
| Split | 1,952.167 | 16.220 | 1,931.723 | 1,981.499 |

Table 7.10: Table of mean completion durations (seconds) of alternative in-memory configurations (10 iterations).

The Split configuration represents the idealised scenario where the user only

needs to set-up the configuration before all the required data of the knowledge-base is retrieved and processed by remote module. It is found to take 7.2 times longer to execute on the same scenario data as the Local configuration even in this best-case scenario of local HTTP connection. However, there would be the potential for these remote services to employ greater computational resources or distributed computing as mentioned earlier to offset the communication costs.

It can also be seen that each of three alternative configurations follows the same general pattern with peaking and troughs in the same schedule batches. However, these do not follow the fluctuations shown in the Local configuration suggesting that the difference between the remote and local configurations can be attributable to the cost of HTTP communication.

The same issue with using persistent storage and multi-thread execution was encountered in these separated configurations. This included the configurations where knowledge-base and generated data were completely separate from the application logic developed for the prototype. In these configurations the knowledge-base and generated data were placed behind HTTP server applications provided by the Semantic Web library. These server applications would therefore handle the transactional operations of reading and writing, but issues were still encountered suggesting there may be an underlying issue with the Semantic Web library.

In summary, the prototype can be seen to have successfully implemented the Framework Configuration to allow the selection of alternative data and services. This has been done in a generalised manner so that the functionality of query validation and service selection can be applied in other use cases by using the developed library. The prototype implements the three travel demand generation modules using the functionality of this library to manage the retrieval of data and select services through the described Semantic Web based mechanisms.

The impact of utilising these remote configurations has been shown to be noticeable, but not prohibitive. The travel demand generation process does not require real-time responses and so once in progress can proceed without user involvement over a prolonged period, while preparing data and creating integrating interfaces requires a direct investment of resources and time by the user. The processes of caching and multi-threading have also been found to improve execution and an online approach creates the potential for accessing greater computational

resources. In principle, continued progress in network communication technology and speeds should also reduce the difference between local and remote configurations, but would be offset by computational performance improvements and a difference will always exist in some form.

The retrieval of all necessary data to construct a local knowledge-base as the initial process would still be considered the most efficient approach. This also presents the opportunity for customisation and adaptation by users. These are still activities that the Semantic Web can support in achieving by simplifying and standardising the process, whether for local, remote or hybrid execution, as has been discussed in previous chapters. The proposed Framework Configuration represents a further development to try and minimise the burden for users and improve investigations.

## 7.5 Challenges in utilising Semantic Web Technologies for Implementing Travel Demand Generation

The previous sections discussed the results from using the implemented prototype. This section discusses the challenges encountered and how they were addressed in the development of the prototype. There is consideration of the usage of Semantic Web technologies, application of the framework approach and the process of integration with two third-party traffic simulators.

### 7.5.1 SPARQL Language Expressivity and Query Optimisation

The SPARQL query language is a powerful tool for searching, extracting and transforming the data contained in the knowledge-base. The language supports numerous built-in functions and the potential for performing complex operations solely within SPARQL queries. However, its execution approach does not support iteration over a set of data with actions dependent upon earlier iterations, e.g. scheduling of later activities and travel cannot be based on earlier scheduling

decisions in a single query. Instead multiple queries would need to be executed, requiring management of this process, or a property function implemented, as in the prototype.

This also presents issues when performing calculations that require the total value of the set, e.g. mean or normalisation. The total is re-calculated for each individual case in the set and not re-used for the next case. For example, in SPARQL the calculation of the cumulative probability of a choice set requires:

1. determining the utility of a case;
2. taking the exponential of the case utility;
3. determining the utility of all cases;
4. taking the exponential of all case utilities;
5. summing the exponentials of all case utilities;
6. dividing the case exponential by the sum of exponentials;
7. and then repeating the whole process for next case.

Therefore, considerably more work is required than performing steps 1. and 2. for each case; then finding the total; and finally normalising each case by the total. This can be achieved through a separate query, or sub-query, but this still requires calculating the utility and exponential twice for each case. Instead, a property function, that is called in the SPARQL query, can achieve this more efficiently using imperative programming.

Another area of inconvenience is the inability to express SPARQL queries as re-usable functions. The retrieval and transformation of data may repeatedly require several steps. SPARQL supports sub-queries, written out in full so increasing query complexity, or extending the query engine with filter and property functions requiring additional configuration. The SPIN and SHACL technologies use SPARQL syntax to describe queries encoded into the schema and executed by their engines. However, these are extensions and not standard SPARQL. Expressing re-usable SPARQL queries as functions within a core schema would make distribution easier, reduce repetition and improve maintenance.

Further, the SPARQL language enables the writing of sophisticated queries which can perform calculations and generate new values and triples. However, the complexity of expressing these operations and the query optimisations currently possible can make performing these queries computationally expensive. Operations may be performed on data that is later discarded or the same values calculated repeatedly for alternative results in the dataset. Therefore, while sophisticated functionality, e.g. calculating Discrete Choice probabilities and constructing Travel Stages from Stage Estimates, can be achieved within SPARQL queries the use of property functions to access an imperative programming language can improve maintenance and performance.

In addition, the optimisation of SPARQL queries through subtle changes can have a dramatic effect upon execution duration. In several cases, the reordering of statements or the replacement of binding data was found to improve query execution from 30 to 45 minutes down to 10 to 15 seconds. This can be due to the earlier binding of variables, so reducing the number of attempts to store potential variable bindings, and query optimisation by the underlying Semantic Web library.

The implications of this are that complex queries provided by users could have a dramatic impact on the performance of the travel demand generation process. Inadvertent replacement of triples or sub-optimal ordering could make the execution of a configuration intractable. The issue of query optimisation is not unique to graphstores, but the RDF graph structure does introduce additional considerations with the algebraic and semantic optimisation of SPARQL queries being an area of active research [152].

Another approach to improving query performance is the separation of queries into their constituent parts. Therefore, rather than a single query retrieving several pieces of related data a faster result can potentially be achieved by extracting each component as a separate query. The outcome of one query is then fed into the next query. This targeted approach requires more complex programming outside of the queries to handle the extraction and handover between queries. This reduces the flexibility for users in adjusting the content of the queries and increases the likelihood of maintenance and adaptation being required of the programmed code.

Finally, SPARQL queries are checked for syntactic correctness but not consistency between variable names or the existence of predicates in the dataset. This can result in minor typographic errors or schema changes causing unexpected and unnoticed outputs as discussed previously (Section 5.3.7). These errors can be easily made by users with the consequences of zero results, additional unnecessary computation or the inclusion of extraneous values.

The identification of such minor mistakes can be a time consuming process in a large and complex system, although the modular approach and the use of discrete queries can help alleviate this. This process can be more challenging in the absence of feedback on the specific issue, since a SPARQL query engine do not have to identify these mistakes as errors. This issue has sought to be addressed in the design of the Framework Configuration (Section 5.3.8). Users can directly access the knowledge-base using a SPARQL endpoint to test, explore and revise their queries prior to their inclusion in the *Framework Configuration*. An impediment to this is if queries contain *property functions*, i.e. modules, that have not been registered with the SPARQL endpoint, although this can also can be worked around when testing queries.

### 7.5.2 SPARQL Extension Property Function Arguments

The processing of arguments passed into filter and property functions in SPARQL queries follows the Functional Programming paradigm. A function is called for each combination of parameter values without visibility of previous or following calls. Therefore, processing of encoded, invariant or externally retrieved data would happen repeatedly leading to an avoidable increase in execution time. This can be overcome by using a caching strategy within the property function to retain re-usable data (Section 5.3.5). A caching strategy was applied in the prototype for invariant data, e.g. location coordinates, to reduce execution duration by 72.3 times.

The caching of arguments was also applied in the implementation of the GeoSPARQL standard, discussed in Appendix B, where geospatial geometry literals were repeatedly deserialised. The deserialisation of these geometry literals was a relatively expensive computational task with invariant result. It was also

found that each instance of a property function resulted in seven deserialisations of the literal during processing by the SPARQL engine of the Semantic Web library. Applying caching of the deserialised geometry literal improved performance by up to 20%. Therefore, consideration always needs to be given to both the computational weight of literals as well as the frequency of retrieval operations.

The Functional paradigm also affects related items that might expected to be treated as a collected list and are instead separated into multiple discrete function calls. These repeated function calls can be performing the same processing steps with only a minor variation for one list variable. In some cases, the processing could be performed once and then the list variable applied to the outcome.

This issue can be mitigated by using the SPARQL *GROUP_CONCAT* aggregation term to produce a delimited string. The delimited string can be passed as a single argument to the property function which can then iteratively work through the items it contains. Alternatively, an N-ary relationship (Section 4.2.2) grouping structure or more formal collection structure must be utilised. The root URI is passed into the function, which then retrieves the collection of related items.

### 7.5.3 RDF/XML Serialisation for Traffic Simulator Interfaces

The Traffic Simulator Interface stage involves the conversion of the *Activity & Travel Schedules* to the simulator input format, XML for both SUMO and MATSim simulators, and return of the results to the knowledge-base. The XSLT technology is proposed as a means to convert between XML schemas. The output of the knowledge-base can be serialised into several RDF formats including RDF/XML [100].

There are several versions of RDF/XML, but it is necessary to use the *plain* serialisation with XSLT due to template complexity and efficiency. The *plain* serialisation forces the use of a consistent *rdf:Description* label for all resources and places all classes as *rdf:type* properties. In other serialisations an arbitrary class name is selected for the initial label with the remaining classes identified through the *rdf:type* property.

In RDF, it is common for individuals to belong to multiple classes. These classes may be shared across multiple sub-classes. In non-*plain* serialisation approaches, two individuals with identical class names can be described by different initial labels while two individuals with only one common class name could have the same initial label. Handling this structure in XSLT templates would require checking both the initial label and the *rdf:type* property for a class name, which increases template complexity. Forcing the use of *rdf:Description* alleviates this requirement and removes the risk of error when changes are made.

A second issue that the *plain* serialisation alleviates is applying a flattened structure. Every element is described by its properties at the top level with cross-referencing from other elements through an identifier. In other RDF serialisations an element can be described as a nested child element of another element. Later occurrences of the element use a cross-reference to the nested child element.

This use of nested child elements complicates the XSLT template structure as a check is required of whether a nested child element has been encountered or is a cross-reference. In a flattened structure a cross-reference can only be encountered in the first level. Therefore, the RDF/XML *plain* format provides several advantages over other RDF/XML formats for use with XSLT templates. A drawback of the *plain* format is it being relatively slow to produce the serialised output. Both XSLT and RDF support alternative serialisation formats that offer flattened structures, e.g. JSON, which may be quicker to output and is an area for future work.

### 7.5.4 Traffic Simulator Integration

An area of significant challenge in the development of the prototype was the integration of the traffic simulators MATSim and SUMO. This challenge does not come from the selected technical approach (Section 6.4.4), but rather the implementation and documentation of the simulators. Both simulators have been utilised in multiple pieces of research, but are in active development of new features with SUMO only recently achieving its first official release.

The development and testing of the prototype with these simulators encountered numerous issues. These issues included features not actually being im-

plemented; behaviour not being as stated; data parameters being incorrectly stated; suggested data parameters not being stated; configuration syntax not being stated; input data not being validated; software errors not being gracefully managed and exhaustive debug logging in normal operation rather providing information to assist the user.

These issues are not unique among software applications, but the impact is wasted resources, effort and potential error that could be avoided for frustrated users. These problems were particularly acute for MATSim where the published 620 page documentation [32] is not in sync with the application development and so provides little assistance for users in using the application and resolving issues. The MATSim simulator also does not include a readily accessible visualisation tool preventing the review and tracking of entities during simulation to assist in identifying errors, unlike SUMO. These issues may be contributory factors to the points identified with MATSim's results in the previous section, but substantial effort, technical knowledge and skill is required in determining where they originate.

This experience highlights the need for the proposed framework in reducing the technical burden and resource demands of integrating multiple traffic simulators. The investigation, development, and deployment tasks of the travel demand generation process are cumbersome and draw on expertise in a number of scientific and software engineering disciplines. Moreover, there has not been a determined effort to think-through and document efforts to develop similar systems with a view of facilitating their exploitation in various simulation domains.

There is collectively wasted effort and error introduced by researchers and users trying to gain an understanding of individual traffic simulator interfaces to utilise them. The proposed approach seeks to provide more than a convenient way to access multiple simulators and instead remove this troublesome phase from investigations. Instead, traffic simulator developers would be able to provide the necessary interfaces for integration of the knowledge-base and schema based on their own detailed understanding of the simulator. Users would then have a starting point for identifying extensions or making adjustments.

## 7.6 Chapter Summary

This chapter has examined the implemented prototype and considered its effectiveness in the generation of travel demand and in the alternative configurations offered by the developed Semantic Web-based framework. There has been discussion of the developed scenario and the limitations of the data it provides. There has also been consideration of the challenges encountered during this process and where possible overcome.

The prototype has produced travel demand that varies by mode, destination and time according to local and global concepts and parameters. The generated travel demand is platform agnostic and can be transformed and configured according to the target application, e.g. traffic simulators, or analysis, e.g. aggregation by type or geographic area.

The instances of concepts, e.g. activities, modes and travel users, and modification of parameters can be controlled and expanded through changes to the knowledge-base rather than being explicitly designed into the modules. More generally it is the quality of the constructed knowledge-base from collected road network, population, land-use and activity patterns that is the limiting factor to further investigation.

The absence of public transport as a mode choice is noticeable in the prevalence of long distance and duration walking stages in the generated schedules. This area of future work would require the development of an additional module to account for the temporal variation of public transport timetables, but could be incorporated as an alternative to the existing prototype module. This alternative can then be selected in query or through development of the trip planning module based on the properties of modes described in the core schema, rather than explicitly stated mode instances.

The design of the travel demand process as three discrete sub-modules has been successful with them each fulfilling a clearly defined role. There is a need to expand these modules with support for public transport as a transport mode or further sub-modules so that greater comparison can be made between them.

The framework for supporting these variations to the modules and sub-modules has also been successfully developed. It allows the modification of queries to ob-

tain alternative data, execute alternative modules and redirect to other sources. There has been consideration of several of these configurations and their implications in producing travel demand.

It was found that the remote configurations introduced a noticeable increase in execution duration compared to the local configuration. However, the travel demand process does not require real-time responses and the remote configuration offers the most opportunity for minimising the preparation and setup of investigations through the deployment of SPARQL endpoint data-sources to accelerate the knowledge-base construction. The utilisation of caching and multi-threading was found to be beneficial to execution performance, while the remote configuration presents the opportunity for accessing additional computational resources through distributed computing. Further work is needed on applying the framework in a local and wide area network context to more fully examine the implications of remote execution.

Another area for further investigation is the variations in simulation outcomes based on the same travel demand. The selected simulators do follow different design principles and so some variations would be expected, but some of the analysed results introduce questions as to their validity. The developed interfaces demonstrate that these alternative traffic simulators can be integrated with a single knowledge-base. Every care has been taken in their development, but further investigation is needed to ensure the interfaces fulfil the input requirements or whether issues are present in the implementation of the specific traffic simulators.

# Chapter 8

# Conclusions and Future Work

This chapter provides an overview of the work contained in this thesis and identifies its contributions, research limitations and areas for future research.

## 8.1 Overview of the Work

The research conducted in this thesis has investigated the process of activity-based travel demand generation and traffic simulation. The closer modelling of human behaviour, the emergence of new technologies and the availability of more detailed datasets is leading to greater complexity for users in preparing and undertaking investigations, while the adoption of activity-based models has been limited despite their potential advantages. The robustness of conclusions in these investigations is supported by consideration and comparison of multiple techniques and models yet the variations in the platform, data requirements and dataset availability present further barriers to the breadth of modelling in investigations.

It has been proposed that the development of a core data schema to model the fundamental concepts of travel demand generation and traffic simulation will assist in the alignment of models in the different stages of the process. These fundamental concepts can be extended with additional concepts of the user and selected models to form a knowledge-base on which the modelling process is performed. It is further proposed that this modelling process can then be supported

by a framework to allow the configuration and selection of alternative models and datasets, which can be deployed in local or remote contexts. It is intended that these improvements will reduce the time and resources required to assemble travel scenarios and facilitate greater comparison and validation between models and implementations.

The technology of choice in this investigation has been the Semantic Web. This technology has been selected due to its extendible data modelling, platform independence and established technical standards. These technical standards have been implemented in a range of software libraries across multiple platforms and so assist the application of the investigated approaches. A fundamental design feature of the Semantic Web is the open access and transformation of data and there has been increasing availability in the provision of online datasets, including examples utilised in travel demand generation. This section will consider the findings of investigating the four research questions explored in this work.

## RQ1 How can a loosely coupled modular Semantic Web knowledge-base be applied to travel demand modelling and traffic simulation?

The initial stage of the investigation focused on considering how a loosely coupled modular Semantic Web knowledge-base can be applied to travel demand modelling and traffic simulation. This considered the existing three-stage process of Population Synthesis, Travel Demand Generation and Traffic Simulation found in the literature and discussed in Chapter 3. The need for a Knowledge-Base Construction stage was identified to align and reconcile the initial input datasets to the schema. This schema would be formed from the core schema and additional concepts introduced by the user or modules being orchestrated by the framework.

A set of major components for an activity-based model were also proposed with a sub-set forming the basis of the implemented prototype. These components were identified through their discrete functionality and potential for alternative implementations to produce modified outcomes to scenarios by reviewing the literature and existing implementations for activity-based models, datasets and traffic simulators.

The configuration of the framework was also considered and identified the potential for constructing and executing the framework on knowledge-bases and modules in local, remote or hybrid configurations. Remote execution of modules introduces the potential for users to only supply the scenario data without the hardware and setup considerations of a local execution. The remote retrieval of data for the knowledge-base presents two opportunities for users.

The current trend in publishing datasets online with Semantic Web technologies will support the objective of enabling the faster construction of local knowledge bases. These online datasets can be directly retrieved and transformed to the required schema removing the need to develop ad hoc processes of extraction, transformation and loading for users. However, these datasets will still require the Knowledge-Base Construction stage to align datasets for use in the later stages.

A further development is the curation of datasets containing the outputs of the Knowledge-Base Construction stage and aligned with the schema. The data contained in the remote knowledge-bases can be directly accessed by modules with orchestration provided by the user to select those of interest. These datasets can then provide benchmark scenarios upon which new techniques can be developed and compared. Policy-makers can then use these comparisons to inform their selections during the construction of their own customised scenarios.

## RQ2 What data concepts are required to construct a knowledge-base for travel demand modelling and traffic simulation?

The next stage of the investigation considered the data concepts of the core schema that are required to construct a knowledge-base for travel demand modelling and traffic simulation as described in Chapter 4. This was undertaken through a review of the literature, published vocabularies, related datasets and implementation parameters. The schema was further reviewed and refined during the iterative development of the prototype described in Chapter 6. The development of the schema sought to distinguish between the core concepts and those formed from model assumption or user context to seek the minimal ontological commitment.

The design principles of Semantic Web support an extendible approach so that new concepts can be introduced. These concepts can sit alongside each other in the knowledge-base without interference. This facilitates multiple alternative models to operate on the same knowledge-base to re-use common data and exploit the additional data they require.

The extendible approach also includes the utilisation of published vocabularies which have been considered to address fundamental concepts of modelling geospatial and temporal domains. These have been selected to provide general representations so that diverse datasets can be included without introducing a language or encoding bias. The schema has been divided into those concepts related to the real-world and those abstract concepts derived from travel demand modelling and traffic simulation.

The grounded real world concepts are focussed around describing the people, activities, locations, vehicles and network infrastructure of the scenario. There has been a discussion of the interactions between these concepts and examples of the user or module extensions that can be applied within the user's schema. Modules can operate on the base concepts or introduce their own specific concepts as their modelling requires. It is only necessary for the schema to describe the base concepts and not enumerate instances or sub-classes.

The set of abstract concepts identify the data necessary for interactions between modules. These include the approaches to activity-based modelling implemented in the prototype but ultimately only requires the production of activity and travel schedules used by the traffic simulators. Therefore, only a small subset of concepts are required to be output by alternative travel demand models while input concepts are fundamental concepts commonly expressed in datasets.

The identification of abstract concepts also include the management of the varying permanence of some parameters in the knowledge-base. A user may wish to modify these parameters in different investigations to explore the impact on a model or scenario. These parameters can be associated with a specific instance of a *Travel Scenario* concept for consistent, and if necessary repeated, retrieval without altering the permanent data of the knowledge-base.

The logical organisation of the knowledge-base can also be configured and permits transient or results data generated during execution to be easily removed

from the knowledge-base. The division of data into graph also enables the selection and switching between different datasets and sources of data within the same knowledge-base.

### RQ3 How can alternative techniques and data be selected using a Semantic Web knowledge-base?

The third stage of investigation considered how alternative techniques and data can be selected using a Semantic Web knowledge-base. It was discussed in Chapter 5 that the process of orchestrating modules and datasets can be managed through an RDF definition of services and graphs and the use of SPARQL Federated Queries. The RDF definition allows the user to redirect the target of queries, or sub-queries, so that alternative data is retrieved or modules triggered.

It is identified that use-cases exist for both local and remote configurations, with local configurations requiring less setup and communication overhead, and there has been a discussion into how this can be supported while maintaining compliance with the SPARQL standard. A drawback to the proposed approach is the access of two local knowledge-bases in the same query, but this can be resolved through several mitigating choices that do not compromise the data or concept of the approach.

The SPARQL query mechanism design was also considered and the consequences for data retrieval and re-use by modules. Each set of parameters passed to a module from a SPARQL query is a discrete case rather than a batch of cases for iteration through. This can mean that data repeatedly used and unchanged across all the cases must be retrieved from the knowledge base for each case and is then discarded. This wastes resources in performing repeated retrieval activities for the same result. Therefore, it recommended that modules identify and temporarily store invariant data from the knowledge-base so that it can be re-used in later cases of the query.

The development of a core schema can reduce issues of misaligned data. Datasets and modules which are aligned to the schema can be more readily consumed and utilised. However, the enforced requirement of the core schema would limit the set of modules and datasets that can inter-operate even if they concep-

tually align or have minor schema differences. Users can mediate and adapt the data between datasets and modules by supplying replacement SPARQL queries to the framework for use by modules. These queries are substituted for the default query of the module and can transform the data in query or retrieve alternative available data to satisfy the requirements of the module.

The specification of these queries by users introduces a risk of error through misunderstanding the schema of modules and datasets being targeted. This is particularly an issue when modules are hosted on computer resources not provided by the user and the resolution of incorrect or misaligned SPARQL queries can dramatically increase computation. It is proposed that these issues can be mitigated by performing data and query validation prior to executing the queries.

This validation can partially be undertaken utilising existing techniques found in the SPARQL standards and literature but there has been the need to develop solutions to specific issues relating to variables in SPARQL queries. Applying these techniques reduces the risk of incorrect results or wasted resources. A mechanism is also established to provide feedback to the user when problems do occur to assist in their resolution.

The proposed framework uses SPARQL to construct, transform, redirect and execute scenarios. The re-use of this platform-independent language means that users do not need to learn multiple programming languages and can apply their developed skills repeatedly. The framework does not introduce any variation to the SPARQL standard and is instead an extension of its language and principles. Therefore, the barrier to using the framework is lowered and requires a narrower skill set than a conventional solution.

The requirements of the framework have been established with no domain-specific requirements identified. Therefore, it is proposed that the framework provides a general solution for accessing and configuring modular solutions for other similar problems. Implementation of the framework can be achieved for a platform by extending an existing Semantic Web library to meet the requirements of the framework. This extension can then utilised by modules for the management of their operational queries while datasets only need to be published as SPARQL endpoints or local knowledge-bases using a standard Semantic Web library.

### RQ4 Can a Semantic Web framework be implemented for the generation of travel demand?

The final stage of the investigation considered whether a Semantic Web framework could be implemented to generate travel demand. This was undertaken by implementing the framework and modular design of the activity-based model with traffic interfaces for two third-party traffic simulators as described in Chapter 6. This resulted in three discrete modules for the activity-based model proposed in Chapter 3 that exchanged information following the core schema described in Chapter 4 and were orchestrated by the framework proposed in Chapter 5. The implementation of the prototype and framework were then evaluated using a constructed scenario in Chapter 7.

The developed prototype provides the user with control over the activity patterns, schema, module parameters, module selection and discrete choice calculation. The implemented modules are intended to be generic representations with minimal design assumptions that cannot be modified through query. The core schema can be further enriched by the user defining instances or extending properties and classes to reflect their own additional modelling of the knowledge-base. The approach allows the user to select alternative modules based on those concepts; access and modify both local and remote datasets; and apply the generated demand to multiple traffic simulators.

The evaluation of the prototype in Chapter 7 considered the utilisation of a constructed scenario and the performance of the framework in alternative configurations. The construction of a scenario was necessary due to issues identified in locating datasets with spatial information to support a real-world scenario and techniques for aligning the demographic, activity and spatial data. These issues are referred to in research for existing travel demand generation processes and have yet to be resolved to the best of our knowledge.

The constructed scenario demonstrated travel demand generation across a range of activities and travel modes for different travel groups that were all specified by the user schema, which extended the core schema, and were not mandated by the prototype. The different travel groups were each configured by activity patterns that influenced their travel behaviour. The travel was distributed

throughout the road network with a wide range of trip distances and durations scheduled.

The lack of public transport routing and distribution of personal vehicles in the scenario resulted in a high prevalence of walking. The inclusion of public transport is an area of future work that can be resolved by the development of an alternative routing module and inclusion of relevant data in the knowledge-base. Consideration of public transport has been made in the development of the core schema and implemented prototype, but there was not opportunity to take this further.

The overall travel demand reflected the typical weekday travel pattern and was reflected in the results of the two traffic simulators. These simulators produced varying outcomes as would be expected from their different modelling approaches. However, closer inspection of the simulator output, particularly for one simulator, identified cases of unexpectedly large positive and negative differences to the generated schedules. These differences were not apparent in the aggregate simulation results, but emerge under further scrutiny and comparison between each simulator and the schedules. A pattern has not been identified in both simulator outputs to suggest an issue with the travel demand generation process and so may reside in the interface or simulator themselves.

An evaluation was undertaken into the performance of alternative configurations using the framework. This demonstrated that the framework was able to modify the execution based on the RDF configuration data provided to orchestrate several arrangements of local and remote modules and knowledge-bases. In a loop-back only network environment, it was found that remote configurations over HTTP took at least 7.2 times longer than the local configuration without HTTP. Further investigation is required into benchmarking in a local or wide area network environment which will likely increase this ratio. However, these timings do not take into consideration the set-up and configuration savings provided to users from not developing ad hoc interfaces between components; cleaning and reconciling datasets; and potentially directly accessing established knowledge-bases and modules through only the configuration.

Further investigation is also needed into using the framework to distribute the generation process across multiple computers. The application of multi-threading

to the implemented prototype batch scheduling of travel groups improved execution speed by 3.3 times. The convenient access to greater computational resources may offset or mitigate the likely networking cost. The SPARQL query execution mechanism was also identified as an area for improving execution performance. The caching of re-used and invariant data was found to increase execution duration by 72.3 times, while SPARQL query optimisations, ordering and narrowness were found to cause substantial variations in retrieval speed.

The proposed approach of a Semantic Web framework for the generation of travel demand was successfully implemented. There are areas of development to produce and validate a complete travel demand modelling process. However, these developments are not constrained by this approach and their resolution has in-part been limited by available datasets and established reference techniques.

The travel demand was generated using a core schema that identifies the main required concepts and can be extended by the user to meet their modelling requirements. This includes satisfying the varying data requirements of modules without interference within the same knowledge-base.

The framework for orchestration of different configurations has also been successfully evaluated in a loop-back network environment and found that this incurred a noticeable increase in execution times, although travel demand generation is not a process with real-time response requirements. These developments provide the potential for reducing the burden on users in setting up travel demand scenarios and comparing between alternative configurations of models and techniques. This is, in turn, should lead to greater focus upon the investigation and the production of more robust outcomes.

## 8.2 Thesis Contributions to Knowledge

This section discusses the contributions of this thesis and project. It is divided into the principal contributions derived from this thesis and the additional contributions from related work undertaken during the project.

## Principal Contributions of the Thesis

The principal contributions of this work are:

### Applying a knowledge-based approach to the process of Travel Demand Generation

An RDFS schema has been developed using knowledge-modelling to describe the fundamental structure of a knowledge-base for the generation of travel demand. This structure can then be extended by additional concepts that the user wishes to model or to describe concepts required by a module chosen for use in the framework. This contributes to the body of knowledge by comprehensively examining the travel demand generation process using a knowledge-modelling approach and represents a broader effort to consider the whole process than found in the literature.

The schema identifies the grounded concepts related to demography, activities, land use and network infrastructure that are found in datasets and research to form the input to activity-based travel demand models and traffic simulators. This includes RDF vocabulary that has more general use in other areas of the transport domain but has not been found in the literature, e.g. the road network vocabulary developed to satisfy geospatial datasets and multiple traffic simulators. Existing published vocabularies and design patterns have been utilised to express concepts, such as geospatial representations and N-ary relations, in a consistent and re-usable manner.

There has also been the identification of travel demand modelling specific structures that can be utilised to express the data passed between key modules of the framework. Alternative travel demand model implementations must only fulfil the activity and travel schedule structure at the boundary to the traffic simulator interfaces and so enabling the incorporation of alternative module arrangements or existing implementations.

A further concept was the encapsulating of scenario and experimental parameters separately from permanent and semi-permanent data so that multiple experimental investigations can be explored with configuration and results for each retained in the knowledge-base. The RDF graph structure permits alterna-

tive representations and data to reside coherently in the knowledge-base so that multiple configurations can be assembled and orchestrated around the same core of data.

### Development of a Semantic-based Framework for Travel Demand Generation

A framework has been developed to assist in the retrieval of data and orchestration of task modules. The premise of the framework is to provide the user with control over the configuration of their investigations by allowing them to select modules and mediating any schema misalignments that may occur between modules. This introduces the prospect of users providing incorrect information through badly formed queries. Therefore, there is also a consideration of the selected mechanisms for ensuring the validity of data and queries which are passed between modules at the direction of the user.

This framework forms an extension to the platform-independent SPARQL standard and can be implemented using any standards-compliant Semantic Web library. The configuration of the framework is expressed in an RDF structure that describes the location of data and modules as services. This configuration is utilised by modules to query for data or invoke other modules, which can exist in a local or remote context. The user is able to also describe replacement queries in the configuration to allow the re-alignment of data concepts, retrieval of additional data or the redirection to alternative modules.

This configuration and control of the travel demand generation process utilises the same RDF data structure and SPARQL language that is used in the knowledge-base. The user is alleviated of the burden of preparing and transforming data between module formats and across platforms. It has been shown that this approach can be seamless applied to a single local knowledge-base or extended to online access of remote datasets hosted as SPARQL endpoints or modules complying with the framework.

This presents the potential for an online infrastructure of datasets and modules that a user can pick and choose services before reconciling the schema through modified queries leading to accelerated investigative outcomes across a broader

range of techniques. The adoption of this infrastructure would represent a fundamental change in how investigations are constructed and executed so that users can produce investigative outcomes quicker and developers can focus more closely upon specific task models.

The characteristics of SPARQL queries have been explored with potential issues identified in users submitting modified queries with grammatical correctness, but not semantic or syntactic correctness, resulting in absent, improper or computationally expensive results. These issues are mitigated through identifying methods to assure SPARQL query validity prior to execution, particularly in variable name binding, and a feedback reporting mechanism to the user. The general challenge of SPARQL query optimisation for the language's complete expressivity still remains.

It has also been identified that modules would benefit from temporary caching of invariant and re-used data during query execution, leading in one case to a 72.3 times improvement in execution duration. The requirements of the framework have been established without travel demand generation specific elements presenting the opportunity for this approach to be applied to similar problems in other domains.

### Design and Demonstration of a Semantic-based Travel Demand Generation Framework

An implementation of the travel demand generation process was successfully constructed as loosely coupled modules using Semantic Web technologies. These modules were designed around specific stages of the scheduling process and interfaces with two third-party traffic simulators. The process utilised the data structures of the RDFS core schema and was orchestrated by the developed framework.

This demonstrated that the travel demand process could be decomposed into discrete components and that platform-agnostic travel demand data could be produced and integrated with multiple traffic simulators. Further, it represents the practical realisation of Semantic Web methodologies and technologies throughout the process, which have not been identified in the literature. Consideration was given to all stages of the travel demand process to inform and structure the

investigation and development, while also providing direction for future work.

A constructed scenario formed the knowledge-base on which the travel demand generation and traffic simulation processes were performed. The process produced temporally and spatially distributed travel demand across a variety of modes, routes and number of travel stages. The implemented approach provided the user with control over the demography organisation, vehicle definitions, choice parameters and module selection rather than these being model design assumptions as found in some implementations.

The interface between travel demand generation and traffic simulation was achieved through SPARQL query to produce RDF/XML which was transformed by XSLT templates. This provides a flexible mechanism that is open for users to inspect and modify for variations in their knowledge-base or traffic simulator input requirements. However, further investigation is needed as sub-optimal or broad SPARQL query and the selected RDF/XML serialisation made this a relatively slow process. The platform-agnostic RDF schema and these flexible interfaces provide the opportunity to develop travel demand models and traffic simulators integrations on a many-to-many basis, rather than the one-to-one basis as found in the literature. This, in turn, can lead to greater comparison in travel demand and simulator outcomes.

An investigation was undertaken into the performance of the implementation across multiple configurations. This included the performance cost of network communication over HTTP to remote modules and datasets compared to a local configuration. It was found that even in a loop-back only environment the execution duration increased by 7.2 times. While the process of travel demand generation is not subject to real-time response requirements this does represent a substantial increase in duration. However, further optimisation and alternative design choices of the prototype could potentially reduce this overhead.

The implemented modular design was found to be suitable to multi-threaded execution resulting in 3.3 times improvement in execution duration and indicating a distributed computing approach may be applicable to access greater computational resources. This may assist in mitigating the network communication costs that an online network infrastructure for the travel demand process may incur. The combination of a distributed network infrastructure, the expanding range of

published online Semantic Web datasets and the orchestration provided by the developed framework provides a basis for travel demand generation models and traffic simulators to increase in complexity and diversity to meet current and future sociological and technological needs.

## Additional Contributions of the Thesis

The additional contributions of this work are:

### Implementation and Benchmarking of the OGC GeoSPARQL Standard

The geospatial nature of travel means that a supporting framework for handling geospatial data is an underpinning component of delivering a Semantic-based Travel Demand Generation framework. The positioning and spatial relationship between entities have an influence throughout the travel demand generation process.

The GeoSPARQL standard has been developed to provide an RDF vocabulary and functionality for a Semantic Web geospatial framework. The sophistication of the full implementation of the geospatial framework in permitting multiple coordinates reference system, spatial shapes, and data serialisations make it more suited to Semantic Web principles than alternative simpler approaches.

Investigation of available implementations found there were no full implementations of the standard. Instead, implementations were limited in functionality and in some cases did not adhere to wider Semantic Web standards. In addition, these systems were developed using relational databases that required lengthy configuration procedures and were not developed specifically for RDF data, unlike graphstores.

Apache Jena is a popular Java library for Semantic Web applications and is compliant with the W3C Semantic Web standards. Therefore, it was selected for the implementation of the prototype. However, it has limited spatial support through a custom module and no support for the GeoSPARQL standard.

An investigation was made into the development of a full implementation of the GeoSPARQL standard using the Apache Jena library, including supporting automatic switching between coordinate reference systems and units of measure.

An additionally developed novel feature was the caching of invariant geospatial literals and other data that was repeatedly re-used in spatial queries and produced a performance improvement up to 20%.

A benchmarking framework was developed to compare the implementation with Parliament and Strabon, two existing partial GeoSPARQL implementations, using the published Geographica macro and micro benchmarking queries. This framework permits the submission of any SPARQL query for benchmarking; provides Java interfaces to incorporate additional test systems; resolved certain implementation inconsistencies in the Geographica framework; and forms the basis for developing a conformance framework for GeoSPARQL and other SPARQL related standards.

Details and results of the implementation and benchmarking framework are provided in Appendix B, which forms a journal article that has been submitted for publication. The source code for this GeoSPARQL-Jena implementation and benchmarking has been released as a contribution to the Semantic Web community. The Apache Jena project has since invited that the GeoSPARQL-Jena project is incorporated as a module of their framework to replace the existing spatial module.

## 8.3 PhD Research Limitation and Plans for Further Work

This section will discuss the identified limitations to this study and areas of future work that has resulted from its undertaking.

### Implementation of Knowledge-Base Construction Modules

The scope of the project has been focussed upon the second and third stages of the three-part travel demand modelling process. These stages were selected due to their bearing upon the proposed approach as the core process of activity-based travel demand generation and the output process of traffic simulation. This focus enabled the project to explore and evaluate the development of the schema, prototype and framework.

The initial stage of gathering, processing and aligning datasets has been partly explored during the project and with the limited discussion in this thesis for brevity. Instead, the focus has been on identifying the fundamental and boundary data concepts that interface between the first and second stage. Implementations of Population Synthesis and Activity Generation modules successfully produced RDF output adhering to the schema using public data sources. However, there was not opportunity to explore the breadth of techniques for these modules and those techniques necessary for Knowledge-Base Construction to enable the generated data to be incorporated into the evaluation scenario. This would then enable the evaluation of modules in real-world scenarios.

There is already an identified need in the literature for research into techniques for aligning these datasets which is of critical importance to the Knowledge-Base Construction stage. The development of these techniques as modules of the framework would enable the end to end process using the framework and present users with another area of choice in their investigations for the interchange of modules. In parallel, there is a need for the publication of detailed spatial datasets on which these techniques can be applied.

### Expanded Features of Activity-Based Modules and Integration of Existing Techniques

The modelling of human behaviour through activity-based travel demand models provides a broad range of design and implementation details. The implemented prototype consists of three modules in the travel demand stage that fulfil a number of features. In the discussion, it has been identified that certain features, such as activity priority, vehicle usage and time management, could be defined as sub-modules so that alternative techniques can be explored.

There is also the potential to expand these modules to include additional features that are required from current travel demand models, such as multi-day and long-term planning; intra-household trip planning; public transport routing; location popularity; further handling of road network semantics; weather events; and learning from trip feedback for future routing. A further area of development are tools to assist in the composing of the framework configuration and the

construction of the knowledge-base, including the incorporation of datasets and application of dataset aligning techniques. The exploration of alternative RDF serialisations may assist in reducing the time required to prepare the generated schedules for simulation.

It has also been identified that there is potential for existing techniques to be integrated either through re-implementation as collections of sub-modules or converting and aligning their input and output to the RDFS schema of the knowledge-base. Therefore, an area of future work is expanding the prototype modules to fulfil more features of travel demand modelling and incorporating existing techniques to operate within the framework.

### Investigation of Network Performance and Distributed Computing in Remote Configurations

The evaluation explored several alternative configurations of the framework. These included examining the effect on the performance of remote HTTP server communication. This communication was carried out on a single machine through the network adapter loopback, rather than a local or wide area network. Therefore, further work is necessary to determine the full effectiveness of the remote execution configurations or whether the current performance of network communications means that remote datasets should only be used for retrieval of data that is acted upon by modules locally.

In addition, there is the potential for the execution of the scheduling process to be distributed over multiple computers. The evaluation considered parallel execution across multiple threads and found an improvement in performance. The orchestration and consolidation of these distributed machines could be applied by expanding the framework configuration to include new concepts and descriptions.

### Application of Additional Semantic Web Technologies

The investigation and developed prototype have utilised several of the Semantic Web technologies, specifically RDF, RDFS and SPARQL. An area of future work is applying other technologies, such as OWL reasoning, SHACL data validation

and SPIN rules, as part of knowledge inferencing and implementing alternative travel demand models. These technologies may have particular application in the Knowledge-Base Construction stage to infer new data and determine logical consistency in the schema. These technologies are expressed in platform-agnostic format and can be shared as part of the schema to aid their dissemination. The further development of the framework could examine the incorporation of Semantic Web Services standards, such as SAWSDL, for discovery and composition of modules and data sources.

The interface between travel demand generation and traffic simulators was developed to extract RDF/XML from the knowledge-base and transform it into XML for simulator input. The RDF/XML format chosen provided a structure which simplified the design of XSLT templates but was relatively slow to output. Further work is needed into identifying an alternative RDF serialisation which provides the required structure and is quick to produce.

# References


[1] Richard G. Wilkinson and M. G. Marmot. *Social determinants of health electronic resource] YBMB ]*. 2nd ed.. ID: dedupmrg830770; Includes bibliographical references and index. Oxford: Oxford : Oxford University Press, 2006.

[2] P. Wockatz and P. Schartau. *IM Traveller Needs and UK Capability Study*. 2015. URL: `https://ts.catapult.org.uk/traveller-needs-and-uk-capability-study`.

[3] Edward Beimborn and Rob Kennedy. "Inside the blackbox: Making transportation models work for livable communities". In: (1996).

[4] G. Tamminga, P. Knoppers, and JWC Van Lint. "Open traffic: A toolbox for traffic research". In: *Procedia Computer Science* 32 (2014), pp. 788–795.

[5] Andreas Tolk. "Interoperability, composability, and their implications for distributed simulation: Towards mathematical foundations of simulation interoperability". In: *Distributed Simulation and Real Time Applications (DS-RT), 2013 IEEE/ACM 17th International Symposium on*. IEEE. 2013, pp. 3–9.

[6] Bruce Edmonds and Ruth Meyer. *Simulating Social Complexity*. Springer, 2013.

[7] Juan de Dios Ortúzar and Luis G. Willumsen. *Modelling transport*. John Wiley & Sons, 2011.

[8] Soora Rasouli and Harry Timmermans. "Activity-based models of travel demand: promises, progress and prospects". In: *International Journal of Urban Sciences* 18.1 (2014), pp. 31–60.

[9] Joe Castiglione, Mark Bradley, and John Gliebe. *Activity-Based Travel Demand Models: A Primer*. Tech. rep. Transportation Research Board of National Academies, 2014.

[10] Kay W. Axhausen. *Definition of movement and activity for transport modelling*. 2nd Edition. Emerald Group Publishing Limited, 2007, pp. 329–343.

[11] Linda Ramstedt, Johanna Törnquist Krasemann, and Paul Davidsson. "Movement of people and goods". In: Simulating Social Complexity. Springer, 2013, pp. 651–665.

[12] D. A. Hensher and K. J. Button. *Handbook of Transport Modelling*. 01371524. Elsevier, 2008. ISBN: 9780080453767. URL: https://books.google.co.uk/books?id=oYMgAQAAMAAJ.

[13] *2011 Census Analysis - Method of Travel to Work in England and Wales Report*. Tech. rep. Office for National Statistics, Feb. 2013. URL: http://www.ons.gov.uk/ons/rel/census/2011-census-analysis/method-of-travel-to-work-in-england-and-wales/art-method-of-travel-to-work.html#tab-England-and-Wales-picture-of-methods-of-travel-to-work-in-2011.

[14] Darren Stillwell et al. *England National Travel Survey: 2016*. Tech. rep. Contains public sector information licensed under the Open Government Licence v3.0. Department for Transport, July 2017. URL: https://www.gov.uk/government/statistics/national-travel-survey-2016.

[15] M. Tranter, D. Robineau, and G. Goodman. *England National Travel Survey 2014*. Tech. rep. Contains public sector information licensed under the Open Government Licence v3.0. Department for Transport, 2015. URL: https://www.gov.uk/government/statistics/national-travel-survey-2014.

[16] Chenyi Chen et al. “The retrieval of intra-day trend and its influence on traffic prediction”. In: *Transportation research part C: emerging technologies* 22 (2012), pp. 103–118.

[17] Guus Tamminga et al. “Toward GIS-Compliant Data Structures for Traffic and Transportation Models”. In: *Transportation Research Board 92nd Annual Meeting*. 2013. Chap. 13-2455.

[18] Joachim Wahle et al. “Decision dynamics in a traffic scenario”. In: *Physica A: Statistical Mechanics and its Applications* 287.3 (2000), pp. 669–681.

[19] Hong Zheng et al. *A Primer for Agent-Based Simulation and Modeling in Transportation Applications*. Tech. rep. United States. Federal Highway Administration, 2013.

[20] Theo Arentze and Harry Timmermans. *Albatross: a learning based transportation oriented simulation system*. Citeseer, 2000.

[21] Farhana Yasmin, Catherine Morency, and Matthew J. Roorda. “Assessment of spatial transferability of an activity-based model, TASHA”. In: *Transportation Research Part A: Policy and Practice* 78 (2015), pp. 200–213.

[22] Ryuichi Kitamura and Satoshi Fujii. “Two computational process models of activity-travel behavior”. In: *Theoretical foundations of travel choice modeling* (1998), pp. 251–279.

[23] Wilfred W. Recker, Michael G. McNally, and Gregory S. Root. “A model of complex travel behavior: Part II—An operational model”. In: *Transportation Research Part A: General* 20.4 (1986), pp. 319–330.

[24] Dick Ettema, Aloys Borgers, and Harry Timmermans. “SMASH (Simulation model of activity scheduling heuristics): Some simulations”. In: *Transportation Research Record: Journal of the Transportation Research Board* 1551 (1996), pp. 88–94.

[25] Reginald G. Golledge, Mei-Po Kwan, and Tommy Gärling. “Computational process modeling of household travel decisions using a geographical information system”. In: *Papers in regional science* 73.2 (1994), pp. 99–117.

[26] Mei-Po Kwan. "GISICAS: AN ACTIVITY-BASED TRAVEL DECISION SUPPORT SYSTEM USING A GIS-INTERFACED COMPUTATIONAL-PROCESS MODEL." In: *Activity-based approaches to travel analysis* (1997).

[27] Joshua Auld and Abolfazl Kouros Mohammadian. "Activity planning processes in the Agent-based Dynamic Activity Planning and Travel Scheduling (ADAPTS) model". In: *Transportation Research Part A: Policy and Practice* 46.8 (2012), pp. 1386–1403.

[28] Matthew J. Roorda, Eric J. Miller, and Khandker MN Habib. "Validation of TASHA: A 24-h activity scheduling microsimulation model". In: *Transportation Research Part A: Policy and Practice* 42.2 (2008), pp. 360–375.

[29] Franziska Klügl and Paul Davidsson. "AMASON: Abstract Meta-model for Agent-Based SimulatiON". In: Multiagent System Technologies. Springer, 2013, pp. 101–114.

[30] Massimo Cossentino et al. "A holonic metamodel for agent-oriented analysis and design". In: Holonic and Multi-Agent Systems for Manufacturing. Springer, 2007, pp. 237–246.

[31] G. Fabio et al. "JADE White Paper". In: *Telecom Italia Lab* 3 (2003), pp. 6–19.

[32] Andreas Horni, Kai Nagel, and Kay W. Axhausen. *The multi-agent transport simulation MATSim*. Ubiquity Press London, 2016.

[33] Guilherme Soares et al. "Agent-Based Traffic Simulation Using SUMO and JADE: An Integrated Platform for Artificial Transportation Systems". In: Simulation of Urban Mobility. Springer, 2014, pp. 44–61.

[34] *Semantic Web*. Tech. rep. World Wide Web Consortium (W3C), 2018. URL: `https://www.w3.org/standards/semanticweb/`.

[35] *W3C DATA ACTIVITY Building the Web of Data*. Tech. rep. World Wide Web Consortium (W3C), 2016. URL: `https://www.w3.org/2013/data/`.

[36] Dean Allemang and James Hendler. *Semantic web for the working ontologist: effective modeling in RDFS and OWL*. Elsevier, 2011.

[37] Matthew Perry and John Herring. *GeoSPARQL - A Geographic Query Language for RDF Data*. Tech. rep. Open Geospatial Consortium (OGC), Sept. 2012. URL: http://www.opengeospatial.org/standards/geosparql.

[38] S. Cox and C. Little. *Time Ontology in OWL*. June 2017. URL: https://www.w3.org/TR/owl-time/.

[39] David Corsar et al. "The Transport Disruption Ontology". In: *International Semantic Web Conference*. Springer, 2015, pp. 329–336.

[40] Martin Hepp and Mirek Sopek. *Automotive Ontology Community Group*. Sept. 2018. URL: https://www.w3.org/community/gao/.

[41] J. Echterhoff. *INSPIRE as Linked Data*. July 2017. URL: https://github.com/inspire-eu-rdf.

[42] C. Mtral, G. Falquet, and A. F. Cutting-Decelle. "Towards semantically enriched 3D city models: an ontology-based approach". In: *Academic track of geoweb* (2009).

[43] S. Harris and A. Seaborne. *SPARQL 1.1 Query Language*. Tech. rep. World Wide Web Consortium (W3C), Mar. 2013. URL: https://www.w3.org/TR/sparql11-query/.

[44] Yoram Shiftan and Moshe Ben-Akiva. "A practical policy-sensitive, activity-based, travel-demand model". In: *The Annals of Regional Science* 47.3 (2011), pp. 517–541.

[45] Paul Davidsson and Harko Verhagen. "Types of simulation". In: Simulating Social Complexity. Springer, 2013, pp. 23–36.

[46] Daniel Krajzewicz et al. "Recent development and applications of SUMO simulation of urban mobility". In: *International Journal On Advances in Systems and Measurements* 5.3 and 4 (2012), pp. 128–138.

[47] Dominik Ziemke, Kai Nagel, and Chandra Bhat. "Integrating CEMDAP and MATSim to increase the transferability of transport demand models". In: *Transportation Research Record: Journal of the Transportation Research Board* 2493 (2015), pp. 117–125.

[48] Jessica Guo and Chandra Bhat. "Population synthesis for microsimulating travel behavior". In: *Transportation Research Record: Journal of the Transportation Research Board* (2008).

[49] Wenli Gao, Michael Balmer, and Eric Miller. "Comparison of MATSim and EMME/2 on greater Toronto and Hamilton area network, Canada". In: *Transportation Research Record: Journal of the Transportation Research Board* 2197 (2010), pp. 118–128.

[50] Konstadinos G. Goulias et al. "Simulator of activities, greenhouse emissions, networks, and travel (SimAGENT) in Southern California". In: *91st annual meeting of the Transportation Research Board, Washington, DC.* 2012.

[51] Mahmoud Javanmardi, Joshua Auld, and Abolfazl Kouros Mohammadian. "Integration of TRANSIMS with the ADAPTS Activity-Based Model". In: *Fourth TRB Conference on Innovations in Travel Modeling (ITM), Tampa, FL.* 2011.

[52] Dung-Ying Lin et al. "Integration of activity-based modeling and dynamic traffic assignment". In: *Transportation Research Record: Journal of the Transportation Research Board* 2076 (2008), pp. 52–61.

[53] Nuno David. "Validating simulations". In: Simulating Social Complexity. Springer, 2013, pp. 135–171.

[54] Ana LC Bazzan and Franziska Klügl. "A review on agent-based technology for traffic and transportation". In: *The Knowledge Engineering Review* 29.03 (2014), pp. 375–403.

[55] Fenghua Zhu, Ding Wen, and Songhang Chen. "Computational traffic experiments based on artificial transportation systems: An application of ACP approach". In: *Intelligent Transportation Systems, IEEE Transactions on* 14.1 (2013), pp. 189–198.

[56] Daniel Kahneman and Amos Tversky. "Choices, values, and frames." In: *American psychologist* 39.4 (1984), p. 341.

[57] C. Black et al. *Where is 2+ car sharing headed? A review of the journey sharing sector and opportunities for future development.* Tech. rep. Carplus Ride Share Working Group by Carplus, 2013.

[58] Arvind Thiagarajan et al. "VTrack: accurate, energy-aware road traffic delay estimation using mobile phones". In: *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2009, pp. 85–98.

[59] Juan C. Herrera et al. "Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment". In: *Transportation Research Part C: Emerging Technologies* 18.4 (2010), pp. 568–583.

[60] Xuegang Jeff Ban and Marco Gruteser. "Towards fine-grained urban traffic knowledge extraction using mobile sensing". In: *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*. ACM, 2012, pp. 111–117.

[61] Claus Stadler et al. "Linkedgeodata: A core for a web of spatial open data". In: *Semantic Web* 3.4 (2012), pp. 333–354.

[62] Bernhard Lorenz, Hans Jrgen Ohlbach, and Laibing Yang. "Ontology of transportation networks". In: (2005).

[63] INSPIRE Directive. "Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE)". In: *Published in the official Journal on the 25th April* (2007).

[64] Linda van den Brink et al. "Linking spatial data: automated conversion of geo-information models and GML data to RDF". In: *International Journal of Spatial Data Infrastructures Research 9,(2014)* (2014).

[65] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web". In: *Scientific American* 284.5 (2001), pp. 28–37.

[66] Apache Any23. *Apache Any23 - Extract structured data in RDF format.* 2013. URL: `https://any23.apache.org/`.

[67] M. Kay. *XSL Transformations (XSLT) Version 3.0*. Tech. rep. Jul, 14. World Wide Web Consortium (W3C), June 2017. URL: `https://www.w3.org/TR/xslt-30/`.

[68] Franck Michel, Johan Montagnat, and Catherine Faron-Zucker. "A survey of RDB to RDF translation approaches and tools". In: (2013).

[69] Thomas R. Gruber. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing". In: *International Journal of Human-Computer Studies* 43.5-6 (1995), pp. 907–928.

[70] Markus Lanthaler, Michael Granitzer, and Christian Gütl. "Semantic Web services: state of the art". In: *Proceedings of the IADIS international conference-Internet technologies and society 2010*. IADIS Press. 2010, pp. 107–114.

[71] Mark D Wilkinson, Benjamin Vandervalk, and Luke McCarthy. "The Semantic Automated Discovery and Integration (SADI) web service design-pattern, API and reference implementation". In: *Journal of biomedical semantics* 2.1 (2011), p. 8.

[72] Ben Vandervalk. "A Semantic Web Based Approach for Evaluating Queries Across Distributed Bioinformatics Databases and Software". MA thesis. The University of British Columbia, 2011.

[73] Ahmad C. Bukhari and C. Baker. "The Canadian health census as Linked Open Data: towards policy making in public health". In: *9th International Conference on Data Integration in the Life Sciences; July 11-12, 2013; Montreal, PQ URL: http://www2. unb. ca/csas/data/ws/dils2013/papers/DILS-SYS-EC-paper3. pdf*. 2013.

[74] Raffaella Aracri et al. "Publishing the 15th Italian Population and Housing Census in Linked Open Data". In: (2011).

[75] Christian Bizer, Tom Heath, and Tim Berners-Lee. "Linked data-the story so far". In: *Semantic Services, Interoperability and Web Applications: Emerging Concepts* (2009), pp. 205–227.

[76] Freddy Lécué et al. “Star-city: semantic traffic analytics and reasoning for city”. In: *Proceedings of the 19th international conference on Intelligent User Interfaces*. ACM. 2014, pp. 179–188.

[77] Spyros Kotoulas et al. “Spud—semantic processing of urban data”. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 24 (2014), pp. 11–17.

[78] Emanuele Della Valle et al. “Semantic traffic-aware routing using the larkc platform”. In: *Internet Computing, IEEE* 15.6 (2011), pp. 15–23.

[79] Susel Fernandez et al. “Ontology-Based Architecture for Intelligent Transportation Systems Using a Traffic Sensor Network”. In: *Sensors* 16.8 (2016), p. 1287.

[80] Abolghasem Sadeghi Niaraki and Kyehyun Kim. “Ontology based personalized route planning system using a multi-criteria decision making approach”. In: *Expert Systems with Applications* 36.2 (2009), pp. 2250–2259.

[81] *data.gov.uk*. Tech. rep. Cabinet Office, 2016. URL: `https://data.gov.uk/`.

[82] J. G. Kim and M. Hausenblas. *5 Star OPEN DATA*. 2015. URL: `http://5stardata.info/en/`.

[83] Open Street Map. *Open Street Map*. July 2014. URL: `https://wiki.openstreetmap.org/wiki/Main_Page`.

[84] Müller Kirill and Kay W Axhausen. “Population Synthesis for Microsimulation: State of the Art”. In: *Transportation Research Board 90th Annual Meeting*. 2011.

[85] H. Knublauch et al. *A Semantic Web Primer for Object-Oriented Software Developers*. Tech. rep. World Wide Web Consortium (W3C), Mar. 2006. URL: `https://www.w3.org/TR/sw-oosd-primer/`.

[86] Laron Smith, Richard Beckman, and Keith Baggerly. *TRANSIMS: Transportation analysis and simulation system*. Tech. rep. Los Alamos National Lab., NM (United States), Apr. 1995.

[87] Andreas Horni, Kai Nagel, and K. Axhausen. “High-resolution destination choice in agent-based demand models”. In: *Annual Meeting Preprint*. 2012, pp. 12–1989.

[88] Oliver Horeni. “Measuring Mental Representations Underlying Activity-Travel Choices”. PhD thesis. Eindhoven University of Technology, 2012.

[89] Ittai Abraham et al. “A hub-based labeling algorithm for shortest paths in road networks”. In: *International Symposium on Experimental Algorithms*. Springer, 2011, pp. 230–241.

[90] Reinhard Bauer and Daniel Delling. “SHARC: Fast and robust unidirectional routing”. In: *2008 Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM. 2008, pp. 13–26.

[91] Liping Fu. “An adaptive routing algorithm for in-vehicle route guidance systems with real-time information”. In: *Transportation Research Part B: Methodological* 35.8 (2001), pp. 749–765.

[92] Axel Wegener et al. “TraCI: an interface for coupling road traffic and network simulators”. In: *Proceedings of the 11th communications and networking simulation symposium*. ACM, 2008, pp. 155–163.

[93] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. Tech. rep. World Wide Web Consortium (W3C), Dec. 2012. URL: https://www.w3.org/TR/owl2-overview/.

[94] Michael Schneider, Sebastian Rudolph, and Geoff Sutcliffe. “Modeling in OWL 2 without Restrictions”. In: *arXiv preprint arXiv:1212.2902* (2012).

[95] Holger Knublauch and Dimitris Kontokostas. *Shapes Constraint Language (SHACL)*. July 2017. URL: https://www.w3.org/TR/shacl/.

[96] H. Knublauch, J. A. Hendler, and K. Idehen. *SPIN - Overview and Motivation*. Feb. 2011. URL: https://www.w3.org/Submission/spin-overview/.

[97] G. Schreiber and Y. Raymond. *RDF 1.1 Primer*. June 2014. URL: https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/.

[98] N. Noy and A. Rector. *Defining N-ary Relations on the Semantic Web*. Tech. rep. World Wide Web Consortium (W3C), Apr. 2006. URL: https://www.w3.org/TR/swbp-n-aryRelations/.

[99] PV Vinu et al. "Pattern Representation Model For N-ary Relations In Ontology". In: *Journal of Theoretical & Applied Information Technology* 60.2 (2014).

[100] Fabien Gandon and A. Th Schreiber. *RDF 1.1 XML Syntax*. Tech. rep. World-Wide Web Consortium, Feb. 2014. URL: https://www.w3.org/TR/rdf-syntax-grammar/.

[101] D. Brickley and R. V. Guha. *RDF Schema 1.1*. Feb. 2014. URL: https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[102] S. A. Abdallah and B. Ferris. *The Ordered List Ontology*. July 2010. URL: http://purl.org/ontology/olo/core#.

[103] Sean Bechhofer et al. *OWL Web Ontology Language Reference*. Feb. 2004. URL: https://www.w3.org/TR/owl-ref/.

[104] Natasha Noy, Michael Uschold, and Chris Welty. *Representing Classes As Property Values on the Semantic Web*. Tech. rep. World Wide Web Consortium (W3C), Apr. 2005. URL: https://www.w3.org/TR/swbp-classes-as-values/.

[105] Alan Rector. *Representing Specified Values in OWL: "value partitions" and "value sets"*. Tech. rep. World Wide Web Consortium (W3C), May 2005. URL: https://www.w3.org/TR/swbp-specified-values/.

[106] Open Geospatial Consortium. *Simple Feature Access - Part 1: Common Architecture*. 2016. URL: http://www.opengeospatial.org/standards/sfa.

[107] *Basic Geo (WGS84 lat/long) Vocabulary*. Tech. rep. W3C, 2003. URL: https://www.w3.org/2003/01/geo/.

[108] Jwe Van Lint and H. Van Zuylen. "Monitoring and predicting freeway travel time reliability: using width and skew of day-to-day travel time distribution". In: *Transportation Research Record: Journal of the Transportation Research Board* 1917 (2005), pp. 54–62.

[109] JWC Van Lint, Henk J. Van Zuylen, and H. Tu. "Travel time unreliability on freeways: why measures based on variance tell only half the story". In: *Transportation Research Part A: Policy and Practice* 42.1 (2008), pp. 258–277.

[110] David Peterson et al. *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. Tech. rep. W3C, Apr. 2012. URL: https://www.w3.org/TR/xmlschema11-2/.

[111] Lijun Sun and Alexander Erath. "A Bayesian network approach for population synthesis". In: *Transportation Research Part C: Emerging Technologies* 61 (2015), pp. 49–62.

[112] Kirill Müller and Kay W. Axhausen. *Hierarchical IPF: Generating a synthetic population for Switzerland*. Eidgenssische Technische Hochschule Zurich, IVT, 2011.

[113] Robin Lovelace, Dimitris Ballas, and Matt Watson. "A spatial microsimulation approach for the analysis of commuter patterns: from individual to regional levels". In: *Journal of Transport Geography* 34 (2014), pp. 282–296.

[114] *Census Microdata*. Tech. rep. Office for National Statistics, 2016. URL: https://www.ons.gov.uk/census/2011census/2011censusdata/censusmicrodata.

[115] Dan Brickley and Libby Miller. *FOAF Vocabulary Specification 0.99, Namespace Document 14 January 2014-Paddington Edition*. Tech. rep. 2014. URL: http://xmlns.com/foaf/spec.

[116] Ian Davis. *RELATIONSHIP: A vocabulary for describing relationships between people*. Apr. 2010. URL: http://vocab.org/relationship/.

[117] *Families and households QMI*. Tech. rep. Office for National Statistics, Nov. 2018. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/families/qmis/familiesandhouseholdsqmi.

[118] Dave Reynolds. *The Organization Ontology*. Jan. 2014. URL: https://www.w3.org/TR/vocab-org/.

[119] EU ISA Programme Core Vocabularies Working Group (Business Task Force). *ISA Programme Core Business Vocabulary*. May 2012. URL: https://joinup.ec.europa.eu/release/core-business-vocabulary/100.

[120] *ISO 3833:1977 Road vehicles – Types – Terms and definitions*. Tech. rep. International Organization for Standardization, Dec. 1977. URL: https://www.iso.org/standard/9389.html.

[121] David S. Burggraf. "Geography Markup Language". In: *Data Science Journal* 5 (2006), pp. 178–204.

[122] EU ISA Programme Core Vocabularies Working Group (Location Task Force). *ISA Programme Location Core Vocabulary*. May 2012. URL: https://www.w3.org/ns/locn.

[123] Ordnance Survey. *OS MasterMap Topography Layer upgrade 2016*. 2016. URL: https://www.ordnancesurvey.co.uk/business-and-government/help-and-support/products/os-mastermap-topography-layer-upgrade-2015.html.

[124] Yi-Chang Chiu et al. "Dynamic traffic assignment: A primer". In: *Transportation Research E-Circular* E-C153 (2011).

[125] Eleni I. Vlahogianni, John C. Golias, and Matthew G. Karlaftis. "Short-term traffic forecasting: Overview of objectives and methods". In: *Transport reviews* 24.5 (2004), pp. 533–557.

[126] Jinyuan Li et al. "A software architecture for artificial transportation systems-principles and framework". In: *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*. IEEE, 2007, pp. 229–234.

[127] Todd Litman. *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute Victoria, Canada, 2017.

[128] Larry Masinter, Tim Berners-Lee, and Roy T. Fielding. *Uniform Resource Identifier (URI): Generic Syntax*. en. Tech. rep. Internet Engineering Task Force (IETF), Jan. 2005. URL: https://tools.ietf.org/html/rfc3986.

[129] Matthew Kerwin. *The "file" URI Scheme*. en. Tech. rep. Internet Engineering Task Force (IETF), Feb. 2017. URL: https://tools.ietf.org/html/rfc8089.

[130] *Apache Commons JCS - Java Caching System*. Tech. rep. Apache Foundation, Aug. 2018. URL: https://commons.apache.org/proper/commons-jcs/.

[131] *EH Cache*. Tech. rep. Software AG USA, Sept. 2018. URL: http://www.ehcache.org/.

[132] *SOA Source Book*. Tech. rep. The Open Group. URL: http://www.opengroup.org/soa/source-book/intro/index.htm.

[133] Jesús M. Almendros-Jiménez, Antonio Becerra-Terón, and Alfredo Cuzzocrea. "Detecting and Diagnosing Syntactic and Semantic Errors in SPARQL Queries". In: *7th ACM International Workshop on Linked Web Data Management*. CEUR-WS, 2017.

[134] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. "A comparison of string metrics for matching names and records". In: *Kdd workshop on data cleaning and object consolidation*. Vol. 3. 2003, pp. 73–78.

[135] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Authentication*. Tech. rep. June 2014. URL: https://tools.ietf.org/html/rfc7235.

[136] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. Tech. rep. Aug. 2018. URL: https://tools.ietf.org/html/rfc8446.

[137] William G. Halfond, Jeremy Viegas, and Alessandro Orso. "A classification of SQL-injection attacks and countermeasures". In: *Proceedings of the IEEE International Symposium on Secure Software Engineering*. Vol. 1. IEEE, 2006, pp. 13–15.

[138] Apache Software Foundation. *Apache Jena*. 2018. URL: http://jena.apache.org.

[139] Robert Battle and Dave Kolas. "Enabling the geospatial semantic web with Parliament and GeoSPARQL". In: *Semantic Web* 3.4 (2012), pp. 355–370.

[140] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. "Strabon: a semantic geospatial DBMS". In: *The Semantic Web–ISWC 2012* (2012), pp. 295–311.

[141] Various. *OWL API*. Tech. rep. owl.cs, 2018. URL: http://owlcs.github.io/owlapi/.

[142] Various. *RDF4J*. Tech. rep. Eclipse Foundation, 2018. URL: http://rdf4j.org/.

[143] Various. *Protégé*. Tech. rep. Stanford Center for Biomedical Informatics Research, 2018. URL: https://protege.stanford.edu/.

[144] Moshe Ben-Akiva and Michel Bierlaire. "Discrete choice models with applications to departure time and route choice". In: *Handbook of transportation science*. Springer, 2003, pp. 7–37.

[145] Wei Zeng and Richard L. Church. "Finding shortest paths on real road networks: the case for A*". In: *International Journal of Geographical Information Science* 23.4 (2009), pp. 531–543.

[146] Peter J. Cameron. *Combinatorics: topics, techniques, algorithms*. Cambridge University Press, 1994.

[147] Charalambos A Charalambides. *Enumerative combinatorics*. Chapman and Hall/CRC, 2002.

[148] Michael Zilske, Andreas Neumann, and Kai Nagel. "OpenStreetMap for traffic simulation". In: (2011).

[149] *Highways England Network Journey Time and Traffic Flow Data*. Tech. rep. Highways England, 2018. URL: http://tris.highwaysengland.co.uk/.

[150] Neil Park. *Estimates of the population for the UK, England and Wales, Scotland and Northern Ireland*. Tech. rep. June 2018. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationestimates.

[151] *Java Measurement Harness*. Tech. rep. OpenJDK, 2018. URL: http://openjdk.java.net/projects/code-tools/jmh/.

[152] Michael Schmidt, Michael Meier, and Georg Lausen. "Foundations of SPARQL query optimization". In: *Proceedings of the 13th International Conference on Database Theory*. ACM. 2010, pp. 4–33.

# Appendix A: Contributions to Open Source Projects and Standards

The following list outlines the contributions made to public projects during the course of this thesis. These range from identification of errors to the submission of patches.

## Apache Jena: Semantic Web Java Framework

- Provided patch to allow variable assignment in ParameterizedSparqlString for SPARQL keyword VALUES. (Issue ID: JENA-1578 FIXED)
- Provided patch to include list/list Property Function signature. (Issue ID: JENA-1339 FIXED)
- Reported issue with closing and reopening TDB2 Datasets. (Issue ID: JENA-1521 FIXED)
- Reported issue with xsd:duration subtraction in XML Datatypes.
- Revised documentation to more fully describe process for TDB resource releasing.
- Revised documentation to more fully describe process for wrapping Dataset Inference Models.

- Revised documentation to more fully describe PropertyFunctions graph operations.

## Java Standard Edition JDK

- Reported issue with subtraction of java.xml.datatype.Duration in the Java API library. (Issue ID: JDK-8190835 FIXED)

## SUMO: Traffic Simulator

- Reported issue with DUAROUTER not utilising Traffic Assignment Zones.
- Reported issue with missing attributes in output TripInfo.xml files.
- Reported issue with ArrivalPos attribute being ignored in input route files.
- Reported issue with looping pedestrian paths resulting in infinite sub-lane switching.
- Revised documentation to correct person pathing only requiring only single edge "x" is required rather than from-to edges "x x".
- Revised documentation to correct that pedestrian walk stage vType value is ignored and only Person vType is supported.
- Reported issue with pedestrians occasionally performing walk stages at faster speed than maxSpeed (i.e. negative time loss). (Issue ID: 4385 FIXED)
- Highlighted the lack of impaired mobility class in abstract vehicle classes for people using mobility scooters and wheelchairs. This prevents the modelling of different non-vehicle modes as all are treated as pedestrians. Accepted as future enhancement to the simulator. (Issue ID: 4411 OPEN)

## OGC GeoSPARQL Standard Schema

- Reported that the GeoSPARQL standard (11-052r4) defines "hadDefaultGeometry" but the published schema v1.0.1 uses "defaultGeometery". This discrepancy interferes with RDFS and OWL inferencing. (Issue ID: 548 OPEN).

## Ordnance Survey MasterMap Highways Network Schema

- Reported discrepancy between proposed schema and the INSPIRE project Road Transport Network schema to which it is aligned. Properties for local speed limits and number of lanes used in traffic simulations were absent.

# Appendix B: GeoSPARQL-Jena: Implementation and Benchmarking of a GeoSPARQL Graphstore - Submitted Journal Article

This appendix contains the journal article submitted for publication in the Semantic Web Journal of work undertaken in relation to this thesis in investigating the implementation of the OGC GeoSPARQL standard. The GeoSPARQL standard was published in 2012 [37], but a full implementation has not been delivered. Existing implementations have partially implemented the standard; relied upon using relational databases for persistent storage; and have demonstrated lengthy data preparation and query execution durations.

The geospatial nature of travel means that a supporting framework for handling geospatial data is an underpinning component of delivering a Semantic-based Travel Demand Generation framework. The Apache Jena Semantic Web framework is a popular API for Semantic Web applications that is compliant with the W3C Semantic Web standards, but has limited support for spatial operations and does not follow the GeoSPARQL standard. Therefore, investigation and benchmarking was undertaken into implementing GeoSPARQL within the Apache Jena framework.

Initial investigation was undertaken into this by Haozhe Chen as part of the

Masters thesis "Implementing GeoSPARQL in Apache Jena" supervised by Dr. Taha Osman and supported by Gregory Albiston, the originator of the idea. This Masters thesis scoped the requirements of the work and implemented an initial set of spatial relation and geometry functions. These functions were then benchmarked against an existing partial implementation of the GeoSPARQL standard, Parliament. The benchmarking process was performed manually using a customised set of queries and required using the Parliament user interface via HTTP, introducing issues of comparability.

The implementation and benchmarking were then expanded by Gregory Albiston as a complete implementation of the GeoSPARQL standard. This included supporting automatic switching between coordinate reference systems and units of measure, caching of invariant data, and resolving incomplete features of the earlier collaborative effort. The inclusion of caching of invariant geospatial literals and other data repeatedly re-used in spatial queries represents a novel feature, and produced a performance improvement up to 20%.

A benchmarking framework was also developed to compare the implementation with Parliament and Strabon, another partial GeoSPARQL implementation, using the published Geographica macro and micro benchmarking queries. This automated benchmarking framework performs queries at the API level and removed certain inconsistencies found in the existing Geographica benchmarking framework and the initial investigation. The details and outcome of this effort are described in the enclosed journal article.

The source code for this GeoSPARQL-Jena implementation and benchmarking has been released online. The Apache Jena project has since invited that the GeoSPARQL-Jena project is incorporated as a module of their framework to replace the existing spatial module.

# GeoSPARQL-Jena: Implementation and Benchmarking of a GeoSPARQL Graphstore

**Gregory L. Albiston, Taha Osman and Haozhe Chen**

*School of Computing and Technology,*
*Nottingham Trent University,*
*Nottingham, United Kingdom*
*e-mail:*
gregory.albiston@ntu.ac.uk; taha.osamn@ntu.ac.uk; haozhe.chen2014@my.ntu.ac.uk

**Abstract:** This work presents a RDF graphstore implementation for all six modules of the GeoSPARQL standard using the Apache Jena Semantic Web library. Previous implementations have provided only partial coverage of the GeoSPARQL standard. There is discussion of the design and development of on-demand indexes to improve query performance without incurring lengthy data preparation delays. A supporting benchmarking framework is also discussed for the evaluation of any SPARQL compliant queries with interfaces provided for integrating additional test systems. This benchmarking framework is utilised to examine the performance of the implementation against two existing GeoSPARQL systems using the Geographica benchmark. It is found that the implementation achieves comparable or faster query responses than the alternative systems while also providing much faster dataset loading and initialisation durations.

**Keywords and phrases:** GeoSPARQL, SPARQL, RDF, Apache Jena, geospatial, geospatial data, geospatial query language, geospatial index.

## 1. Introduction

This work discusses the implementation of a geospatial graphstore complying with the OGC GeoSPARQL standard [11]. Previous works have partially implemented the GeoSPARQL standard and have generally extended relational databases rather than utilising a RDF graphstore. This work describes the design features of the implementation, investigation of parameter tuning and the development of a benchmarking framework for comparison with other GeoSPARQL implementations.

The increasing usage of location-aware devices, e.g. smartphones, Internet-of-Things devices and in-vehicle navigation, has led to an increasing range of applications and datasets reliant upon geospatial data. The interpretation of geospatial data and relations is an established field of study upon which the GeoSPARQL incorporates Semantic Web concepts and principles. The Semantic Web is intended to increase the access and sharing of data through the internet and presents the opportunity to enrich data through knowledge modelling, automated inferencing and interconnection of datasets to provide deeper insights and experiences.

A key technology of the Semantic Web is the Resource Description Framework (RDF) [13] which is a data structure standard based upon a directed labelled graph of subject-predicate-object triples. It is intended that any data can be represented using RDF's design principles and extended by vocabularies, e.g. GeoSPARQL, to allow consistent interpretation across datasets. The SPARQL query language [9] provides a query mechanism to explore, retrieve and modify RDF data. It also includes many operations commonly found in the Server Query Language (SQL) used in traditional relational databases. The GeoSPARQL standard enhances SPARQL's query mechanisms to allow users to conveniently access geospatial data.

### *1.1. Challenges*

There are several challenges relating to geospatial graphstores. There is no implementation of the GeoSPARQL standard stating full compliance across all six components. In addition, there is no benchmarking framework to investigate GeoSPARQL compliance and consider user quality of life aspects, such as mixed coordinate reference system datasets, inferring geometry metadata and converting datasets between geospatial datatypes and coordinate reference systems.

Each of the reviewed implementations uses modified relational databases for persistent storage rather than an RDF graphstore. These relational databases have been adapted for RDF usage rather than developed specifically for the purpose. Therefore, there is potential for graphstore implementations to have design optimisations which improve performance. The extension of relational databases can also present a set-up and environment configuration burden that is justifiable in production scenarios but limiting for prototype development and research.

There is also a design challenge between the SPARQL query execution process and the dimensionality of the problem space defined by the GeoSPARQL standard. In SPARQL, query execution is performed on discrete cases of parameters which are represented in string literal form. In the case of geospatial calculations, information is extracted from the string literals, processed and then discarded for the next iteration of query execution. However, the next iteration step may contain one or more parameters used in the previous iteration. Later query iterations may also use the same parameters as a previous iteration after following a different route through the data graph. Therefore, computation is wasted repeatedly extracting information across multiple iterations.

Potential solutions to this are storing the results of previous computations or pre-computation of all potential values. The GeoSPARQL standard defines many functions but a central component is the three sets of *relation families*. These relation families each describe eight spatial relationships between geometry pairs. The relationship between two geometries is invariant and so suitable to computation and storing during data loading. This would change the extraction and processing described previously to a retrieval process. However, the dimensionality of these calculations is excessive. Calculating all the relationships of a very modest 10,000 geometry dataset would require 2.4 billion triples.

This represents an exceedingly large supporting set of data for a small geospatial dataset and introduces issues of pre-computation cost, storage and search durations. Particularly when only a small subset of geometries and relations may be utilised by a user at any time and the new or modified geometries will require re-computation. Graphstores have been developed to handle trillions of triples [8, 15] and so geometry datasets could be much larger than the scale described. Therefore, a compromise is required between pre-computing or retaining related information versus the potential scale and dimensionality of datasets.

### *1.2. Contributions*

The principle contributions of this work can be summarised as:

1. Full GeoSPARQL implementation of all six components, including automatic inferring geometry properties and query re-writing, using an RDF graphstore.
2. Benchmarking framework for GeoSPARQL and SPARQL queries with results and comparison of three GeoSPARQL systems.
3. Discussion and implementation of novel on-demand indexes to improve GeoSPARQL query performance.
4. Identification of areas for review and revision in future versions of the GeoSPARQL standard.

### *1.3. Related Work*

This section some of the work already undertaken in the development and implementation of GeoSPARQL. The GeoSPARQL standard is influenced by the Simple Features standard. This outlines spatial functions for use in SQL relational databases with geometry shapes represented in Well Known Text (WKT) and Well Known Binary (WKB) formats. Simple Features defines a set of geometry relations which are included in the GeoSPARQL standard as one of three *relation families*. Other functions from Simple Features have also been incorporated but with broader usage in the SPARQL context. Therefore, there is a straightforward route for SQL dependent implementations to achieve some GeoSPARQL compliance by adapting existing Simple Features functionality, but additional aspects need to be taken into consideration.

There are numerous implementations of RDF frameworks with many featuring geospatial and sometimes more specifically GeoSPARQL support. These RDF frameworks provide persistent storage through RDF graphstores or relational databases adapted for RDF with a geospatial extension, e.g. PostgreSQL with PostGIS. While these frameworks refer to supporting GeoSPARQL it has not been possible to find any explicit statements of the implemented GeoSPARQL extensions and requirements compliance. As far as we are aware, there has not been any systematic research or compliance testing of the various RDF frameworks in their support for GeoSPARQL.

The following gives a brief overview to these frameworks to illustrate the fragmented nature of GeoSPARQL support and different implementation approaches. AllegroGraph is a graph database that supports geospatial operations but takes a more generalised approach than the Simple Features or GeoSPARQL standards, with a set of functions focused upon points within a radius or polygons. Apache Jena is a Java based Semantic Web and Linked Data framework that provides various tools for developing applications including persistent storage, SPARQL query engine, and inferencing. It has a spatial module that supports a small set of functions which are unrelated to the Simple Features or GeoSPARQL standards. Apache Marmotta [16] is a Linked Data platform which uses PostgreSQL with PostGIS backend and provides functions from the *Geometry Extension* and *Geometry Topology Extension*, but geospatial support has not been included in any public release due to "portability issues".

Stardog is an RDF data unification platform that supports a subset of non-topological GeoSPARQL functions along with their own geospatial functions. Virtuoso is SQL database with RDF and SPARQL extensions that includes some geospatial support related to the Simple Features standard but not explicitly supporting GeoSPARQL. Parliament integrates the Apache Jena framework with a Berkley DB persistent storage to provide temporal and GeoSPARQL compliant spatial support. However, the Parliament is dependent on obsolete components of Apache Jena and has not been updated to the latest major version. Strabon [10] is a spatiotemporal store that uses the RDF4J framework backed by PostgreSQL with PostGIS to support the *Core*, *Geometry Extension* and *Geometry Topology Extension* components of the GeoSPARQL and its own stSPARQL syntax.

Parliament, Strabon and uSeekM (now unavailable) were examined in the Geographica benchmark [7], which was developed by the same project team as Strabon. In this work, the three systems were benchmarked for data loading durations, 28 individual queries and three sets of linked queries. Strabon generally outperformed Parliament and uSeekM. The query syntax used predated the finalisation of the GeoSPARQL standard and in some cases was for stSPARQL functionality. The datasets and queries have been published with some updates made to comply with GeoSPARQL.

The remainder of this work is organised as follows. In Section 2 is an overview of Spatial Analysis and key topics in the area along with a motivational scenario for its usage. Section 3 discusses the GeoSPARQL standard and the six extension modules. Section 4 describes the implementation design of the GeoSPARQL extension to the Semantic Web library Apache Jena and features to improve query performance. It also details the design and development of the Benchmarking Framework used to evaluate the implementation. Section 5 discusses the challenges encountered during the implementation process. In Section 6 is an evaluation of results obtained when comparing the implementation described in Section 4 with two existing GeoSPARQL systems, Parliament and Strabon, using the developed benchmarking framework. The final section provides a summary of the findings of this work and identifies areas for future work.

## 2. Overview of Spatial Analysis

This section provides an overview of key components and concepts for spatial analysis. These components provide the theoretical and practical foundations upon which the Simple Features and GeoSPARQL standards have been developed. A supporting illustrative use case is also provided to highlight the practical need for geospatial queries and support.

### *2.1. Topological models for spatial analysis*

A key feature of Geographic Information Systems (GIS) is spatial analysis and querying using topological models of the relationships between spatial objects [6]. Several models have been developed to interpret these relationships between different geometry shapes, including Egenhofer, Region Connection Calculus (RCC) and the Simple Features which are utilised by the GeoSPARQL standard.

Research has been undertaken into different interpretations of spatial information. These have included relationships based on direction and distance using logical operators and comparing spatial data using algebraic point sets. The approach of the Egenhofer Topological Model [6] seeks to overcome limitations of these two approaches by establishing relationships based on the intersection of two spatial objects in their interior and boundary. In this topological model the relationships are invariant to topological transformations, e.g. translation, scaling and rotation, of the spatial objects, focused upon two-dimensional spatial objects and produces a Boolean result. The Egenhofer relation family consists of eight relationships.

The RCC model is a boundary-free theory that seeks to provide a qualitative representation for reasoning based upon axioms that every spatial object is a region, i.e. has an interior, a region connects with itself and if a region connects with another region then the other region also connected to the region. The RCC-8 relation family based upon the assumption of *jointly exhaustive and pairwise disjoint* (JEPD) relations derives eight relationships [4]. The Simple Features relation family is established by the Simple Features standard [5] to describe a set of spatial objects and their relations which satisfy most real-world geospatial data use cases and re-uses relations defined by the previous relation families to form eight relationships.

The Simple Features relation family is represented by the Dimensionally Extended Nine-Intersection Model (DE-9IM). The DE-9IM notation [3] extends the concepts introduced by Egenhofer to describe relations through the intersections of the interior, boundary and exterior of spatial objects. The nine pair-wise intersections produce a matrix where the maximum permitted spatial dimension for the intersection is indicated. The spatial dimension of geometric objects is point 0, line string 1 and polygon 2. The boundary of a geometric object is formed from a set of geometric objects of the next lower dimension. The interior is those points remaining when the boundary is removed, and the exterior is those points not in the interior or boundary. This notation can represent all

the relations of the *relation families* while also allowing for alternative relations to be described.

### 2.2. Spatial Reference Systems (SRS)

The consistent interpretation of spatial coordinate information is managed by Spatial Reference Systems (SRS), also known as Coordinate Reference Systems (CRS) with the terms being used interchangeably here. A wide range of SRS types have been developed for different applications and only key points will be summarised. SRSs can be categorized into several different types with *projected* and *geographic* being two widely encountered. Each SRS contains a coordinate system that describes the relationship between the coordinates and the earth's surface according to a unit of measurement. Coordinate systems can range in dimension or axis number but two or three are common. Two SRS may use the same coordinate system but order the axis differently, e.g. the widely used WGS84 and the GeoSPARQL defined CRS84. Mathematical transformations between SRS have been published as part of SRS definition catalogues, such as EPSG [17].

A *geographic* SRS, e.g. WGS84, places the coordinates on an ellipsoid approximating the earth's shape. This allows the entire earth's surface to be represented using latitude and longitude coordinates in degree or radian units. Accurate distances between points requires consideration of the ellipsoid curvature through *great-circle* or *orthodromic* distance. A *projected* SRS, e.g. Universal Transverse Mercator or UTM, places easting/westing/x and northing/southing/y coordinates on a planar surface. The distortion of representing the earth's curvature on a plane is controlled and compensated with the SRS only being applicable to a small surface area. This area is often enough to represent a whole nation, e.g. United Kingdom has a single *projected* SRS OSGB36/BNG, while the global UTM system divides the earth into sixty SRS zones. This approach allows quick calculation of accurate distances using *Euclidean distance* in linear units, such as metres.

The importance of SRS support can be highlighted by considering the use case of a public SPARQL endpoint. In this case an endpoint is providing access to a dataset of geospatial data encoded in the widely used global *geographic* WGS84. A user is querying the endpoint with a local geospatial dataset encoded in their national *projected* SRS, as is typically the case. A geospatial system that does not support SRS transformations would require the user, or the data publisher, to transform the dataset or all queries into WGS84. This places the technical burden upon on the user and creates the potential for incorrect transformation. It also assumes that the user is aware that such transformation is required or is not supported.

### 2.3. Spatial Geometry Serialisations

The representation of geospatial data can be achieved using a variety of spatial serialisations, e.g. WKT, GML, GeoJSON and KML. These identify the geom-

etry shape, coordinates and, in some cases, the SRS of the geospatial data. The standards and scope of serialisations vary through application context, defining additional geospatial concepts, such as Features and Geometry, or following the design conventions of a parent serialisation, such as XML and JSON. These serialisations therefore encode equivalent geospatial information, but some variations occur, such as in geometry shapes and their semantics. Typical geometry shapes are Point, LineString and Polygon along with collections of these shapes.

### *2.4. Generating travel demand data for traffic simulation: a motivating scenario*

An example use case for geospatial data can be found from the domain of Traffic and Transport demand modelling and simulation [1]. This domain investigates the interaction of people, vehicles, locations and transport infrastructure. These interactions can be tangible between physical entities, such as a person's location being upon a pavement area, or abstract, such as a property being in an administrative area.

The undertaking of demand modelling and simulation is multi-stage process with spatial analysis being of use in preparing scenario data, modelling the behaviour of individuals in the scenario's context and analysing the outcomes of simulation. The type of interactions between the three main geometry shapes can be found across each of the stages. The following list provides some example spatial analysis applications that may be encountered:

- Distance between a person (point) and a bus stop (point), road (line string), access/entrance (point) or building (polygon).
- Identifying all the houses (polygon) within a local authority, retail shopping or school enrolment area (polygon).
- Determining the common routes (line string) and stopping locations (point) of individuals during their journeys.
- Identifying access permissions into resident (point) only areas (polygon) during route planning.
- Aligning national demography and transport data (projected SRS) with road and building infrastructure (geocentric SRS).

By applying spatial analysis, the results of these interactions can be consistently and accurately determined allowing greater comparison between models.

## 3. Extensions and Requirements of the GeoSPARQL Standard

The GeoSPARQL standard was developed to enable querying of geospatial data in RDF format and is structured around six modules supported by thirty conformance requirement clauses. The standard is developed from the Simple Feature standard, which is widely used in SQL databases, but provides a wider definition of spatial relations and incorporates RDF concepts, such as Unique Resource Identifiers (URIs), namespaces and RDFS inferencing.

The Simple Features standard is based upon a simplified spatial model where all coordinates are on a planar surface regardless of SRS. This simplification reduces calculation complexity but introduces potential error when using *geographic* SRS which is tolerated by the standard. Comparison between geometries with different SRS requires accounting for both numerical and positional differences.

### *3.1. Core*

This extension defines the fundamental vocabulary of the standard, which is the *Spatial Object* and *Feature* classes. A *Spatial Object* is the superclass of *Feature* and *Geometry*. A *Feature*, e.g. a building or lake, is defined separately from the *Geometry* that defines its spatial shape. Therefore, a *Feature* may have multiple *Geometry* for different purposes, timestamps or serialisations.

### *3.2. Geometry Extension*

In this module is the definition of the *Geometry* class, the *Geometry Literal* datatype and the *Non-topological Query Functions*. The relationship between *Feature* and multiple *Geometry* is defined using two properties, the one-to-one *hasDefaultGeometry* and the one-to-many *hasGeometry*. The *hasDefaultGeometry* is used in later extensions to select a consistent *Geometry* when determining inferred results.

The spatial coordinates of a *Geometry* are defined between it and a *Geometry Literal* through the one-to-one *hasSerialization* property. The *hasSerialization* property can be sub-classed to a specific serialisation, e.g. *asWKT* or *asGML*, to allow finer control during querying. Additional *Geometry* properties describe information that can be gained by interpreting the serialisation of the *Geometry* but are made available for direct use in SPARQL querying.

The *Geometry Literal* datatype provides the string representation of the geospatial geometry shape and defines two specific serialisations, *WKT* and *GML*, with specific conformance requirements for each. The *Non-topological Query Functions* are applied to one or more *Geometry Literals* to return a result and include distance, intersection and boundary. Calculations are performed in the SRS of the first *Geometry Literal*.

### *3.3. Topology Vocabulary Extension*

This module defines the vocabulary for property relationships, such as *contains*, *equals* and *disjoint*, between two *Spatial Objects* for assertion in a dataset. These relations are defined in three relation family of Simple Features, Egenhofer and Region Connection Calculus 8 (RCC8) with eight *spatial relations* in each family. Each *spatial relation* is defined using a DE-9IM intersection matrix which shows the maximum number of dimensions of the intersection between interior, boundary and exterior of the two geometries.

### 3.4. Geometry Topology Extension

In this extension the *Topology Vocabulary* is re-formulated as filter functions which accept *Geometry Literals* and return a Boolean result. This allows the calculation of *spatial relations* which have not been explicitly asserted in the dataset. Functions are defined for the same three relation family contained in the *Topology Vocabulary*. Calculations are performed in the SRS of the first *Geometry Literal*.

### 3.5. RDFS Entailment

This extension outlines the application of RDFS and OWL reasoners and geometry hierarchies to produce inferred statements in a dataset. This means that classes and properties, such as *Spatial Object* or *hasSerialization*, can be obtained from a schema applied to the dataset or the stated relationships rather than needing the dataset to consistently state their existence. This reduces the burden upon dataset publishers and allows the automated identification of inconsistencies.

### 3.6. Query Rewrite Extension

This extension builds upon the vocabulary, functions, and inferencing of the previous extensions to provide the most wide-reaching support to the user. The available usage of a vocabulary for relations between *Spatial Objects* in SPARQL queries, provided by the *Topology Vocabulary*, still requires the dataset to be populated with triples using the vocabulary. This requires the dataset publisher or user to determine and assert the relations. Determining these relations is the functionality of the *Geometry Topology Extension* described previously but is based upon retrieving the *Geometry Literal* of *Spatial Objects*.

This complicates query writing and requires the user to have detailed knowledge of the vocabulary. The *Query Rewrite Extension* allows the user to use the higher level *Spatial Objects* in their queries through syntactic sugar to obtain results regardless of their assertion in the dataset. The geospatial system expands the query as required to find either asserted triples or calculates the relations when absent using the *hasDefaultGeometry* property between *Feature* and *Geometry* to find *Geometry Literals*. This leads to four potential cases of *Feature-Feature*, *Feature-Geometry*, *Geometry-Feature* and *Feature-Feature* and any asserted triples for each *Geometry Literal* pair.

## 4. Implementation of the GeoSPARQL Jena and Benchmarking Framework

This section describes the design and implementation considerations of the GeoSPARQL Jena system and the supporting benchmarking framework. In the

first part is the GeoSPARQL Jena system, which implements the GeoSPARQL standard and extends the Semantic Web library Apache Jena. In the second part is the benchmarking framework, which has been currently been developed for testing three query sets across three target systems and provides facility for interfacing additional target systems and user defined queries.

### *4.1. Implementing the GeoSPARQL Standard as an extension of Apache Jena*

The primary focus of implementing the GeoSPARQL standard is to extend the query execution mechanism of Apache Jena to support all the extension modules. Filter and Property functions are registered with the ARQ query engine for use when matching function or property names are encountered within a SPARQL query. The key implementation difference between Filter and Property functions is that Filter functions only operate upon the parameters passed into the function while Property functions can access the underlying graph to retrieve or create additional triples.

Two design principles were to minimise user configuration requirements and minimise pre-computation through on-demand processing of queried data. This on-demand processing means that initialisation periods are short, only calculations relevant to utilised functionality are performed and new datasets can be incorporated quickly and with consistent query durations.

The extension approach allows all the existing functionality of Apache Jena, which closely complies with Semantic Web standards, to be utilised, e.g. RDF file handling, RDFS inferencing, HTTP endpoint and persistent storage. The GeoSPARQL class and property hierarchy are achieved by loading the published GeoSPARQL RDF schema into an RDFS inferencing model and applying to any dataset. This allows user and version modifications to the schema to be made without requiring alteration to the implementation source code. The deployment only requires an application developer to write a single configuration line of Java code. Various configuration options for the use of the GeoSPARQL Jena extension have been made available to the developer, such as index sizing and in-memory or persistent storage options. Additional utility methods have also been implemented using the available functionality, such as converting RDF files between spatial reference systems and geometry serialisations.

The implementation workflow, shown in Figure 1, is activated when the ARQ query engine reaches a registered Filter or Property function name. The following describes the workflow with relevant diagram points indicated by parenthesis. The ARQ engine processes query by checking triples in the dataset and passing values to the corresponding function (1). A key concept of SPARQL querying is the reduction of the full dataset graph down to a subset which are true for the graph described by the query. The ARQ engine seeks to optimise execution for the quickest resolution. Therefore, a set of partial results for the wider query may have been obtained prior to passing to the function. Although the function may only be defined once in a query it is called multiple times,

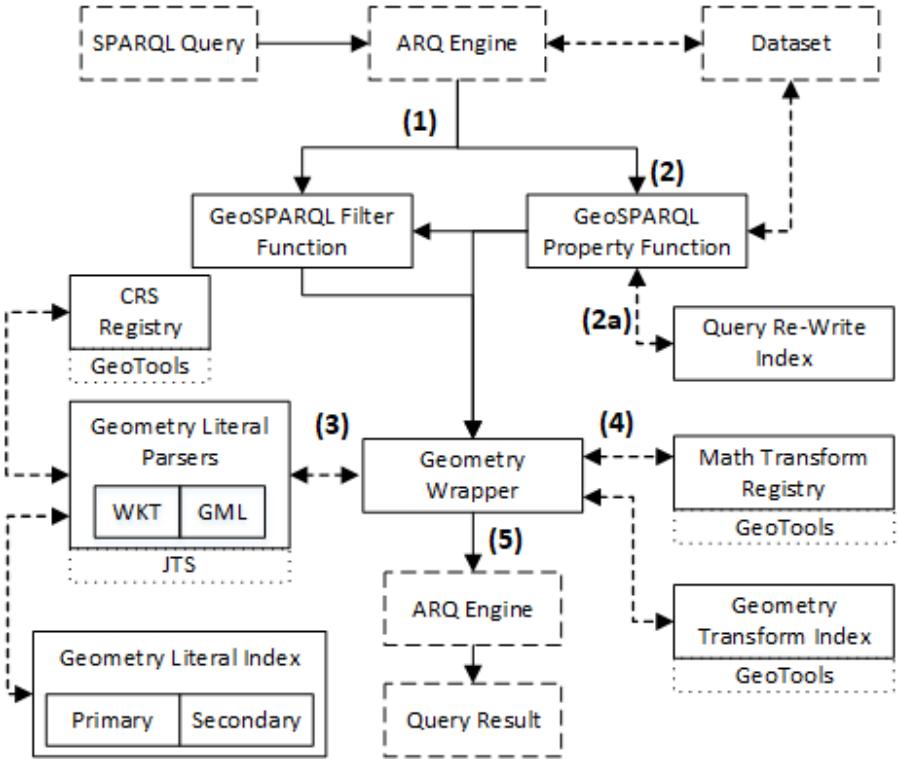


FIG 1. *The GeoSPARQL Jena query execution workflow from SPARQL query to result with key points from the description indicated by parenthesis. Workflow activated when a GeoSPARQL function or property is encountered. Implementation stages shown along with interfacing Apache Jena (dashed) and external library (dotted) stages.*

once for each potential result, passing in different arguments. There may also be additional parsing of literal values during query execution that is hidden during query writing.

GeoSPARQL Property Functions may retrieve additional values from the dataset (2). Many Property Functions in GeoSPARQL are syntactic sugar for a Filter Function which is used to perform the actual processing. The distinct Property Functions follow the same process as Filter Functions. The Query Re-Write Property Functions also checks for asserted Feature and Geometry relationships in the dataset to shortcut resolution, as required by the GeoSPARQL standard, followed by an additional check of the Query Re-Write Index (2a).

This index contains up to a maximum number of recently found relations between Geometry Literals. Each Geometry Literal pair could be involved in up-to four Feature and Geometry relationships for each spatial relation, without considering that a single Geometry Literal could be used by multiple Features or Geometries. Therefore, retaining recent results has been designed into

the workflow to address the re-working behaviour of SPARQL query execution highlighted previously but still respecting the large dimensionality that could quickly result from retaining every result.

Geometry Literals are unparsed into Geometry Wrappers via relevant parsers (3). The Geometry Literal Index is checked for recently unparsed Geometry Literals to allow re-use between iterations through on-demand caching. Geometry Literals are immutable and so later extraction yields the same Geometry Wrapper. The Geometry Literal Index contains two indexes that relate to positions of function arguments which is likely to be consistent between query iterations and allows prioritising the index search. If the Geometry Literal is not present in the initial index, then the alternative index is checked before a new Geometry Wrapper is extracted from the Geometry Literal and added to the initial index. The CRS Registry similarly retains recently used CRS data for any future Geometry Literals with the same CRS.

The Geometry Wrapper handles all processing to resolve the request, e.g. spatial relations and functions, with one instance for each Geometry Literal (4). The spatial relations and functions leverage the JTS library [18], which is an implementation of the Simple Features standard, and has been extended to incorporate the additional concepts of the GeoSPARQL standard. Mathematical conversions between CRSs are obtained as they are required and stored in a Math Transform Registry for re-use, as the same transformation converts any geometry between two CRSs. The geometry resulting from a CRS conversion is retained by the Geometry Transform Index for future re-use, up-to a configurable maximum number. Here again the design approach has been to retain information that may be useful in later query iterations. The functionality for CRS conversion is provided by the GeoTools library [14] which accesses coordinate transformations definitions provided by defining authorities. The outcome of the function is returned to the ARQ Engine (5) via the calling function to continue processing the query, potentially returning to step (1) for another function.

The retention of data in-memory can present issues when processing large datasets. To manage the sizes of the indexes and registries several strategies have been implemented. Firstly, the data retained within registries is common across multiple geometries and highly re-usable. Therefore, these have been not constrained in size or persistence and are intended to persist for the duration of the application. Secondly, to manage size the index contents expire after a fixed time-period (default: 5 seconds) which is refreshed whenever an item is retrieved. Checking for removals occurs in 1 second intervals. Therefore, memory usage will increase during periods of geospatial querying and then fall back during other activities.

The maximum number of items can also be limited to avoid excessive amounts of memory being used and exceeding available resources. When the index is full, query execution continues unblocked but there are no additions to index until items have expired. Investigation was undertaken into memory caching with overflow to disk storage [19, 20], but the Geometry Wrapper serialisation was too burdensome for on-demand caching to disk.

Obtaining CRSs, including units of measure, and math transformations can require internet connectivity for lookups from authoritative sources and has been used as justification for only permitting a single CRS in certain contexts [2]. In the context of the Semantic Web, internet connectivity is an underlying principle and typically expected to enable open systems; demonstrated by the SPARQL standard supporting federated querying through HTTP and a typical use case being the publication of data through an HTTP endpoint. Consideration has been made in the implementation for closed systems by allowing users to define the CRSs they require and adding them to the registry.

A practical issue for users of a geospatial system is the conversion of a heterogenous dataset or datasets, with varying CRS or serialisations, to become homogenous. Methods have been included to leverage the interoperability implemented for GeoSPARQL so that users can use GeoSPARQL Jena as an API to undertake these conversions and complement Apache Jena's existing support for serialising triples.

An additional use case is converting existing geospatial data to RDF formats as highlighted for future work in the GeoSPARQL standard. Many platforms and libraries are available for data conversion but are often intended for large scale or diverse formats. A common format for database export or manually prepared data is tabular separated value files, e.g. CSV and TSV etc. Therefore, a tabular file converter has been developed to facilitate the conversion of user geospatial data to RDF. This converter is provided as a pure Java application or API, does not require an external schema or configuration and does not create additional column and row index properties as found in some other conversion libraries. Configuration is through column headings with URIs able to be explicitly stated, formed from a base URI for the file, looked up from common URIs or defined by the user in a separate file.

Datatypes can also be referred in shorthand to standard XSD types [12] or be user defined in file or external prefix file. In RDF the number of properties can vary between individuals of the same class and therefore repeated properties, blank fields, and ragged rows are supported. Apache Jena inferencing and serialisation has been leveraged so that the user can output files for a dataset that incorporates, or separates, inferred triples in a wide range of standards-compliant formats.

### *4.2. Implementing the Benchmarking Framework*

The benchmarking framework was developed in Java and built as a Gradle multi-project. Java was selected due to all three systems providing a Java API and its prevalence in enterprise production systems. A common deployment use case for SPARQL enabled graphstores and databases is as an HTTP endpoint for public querying. All three target systems can be configured as an HTTP endpoint, but Java APIs have been used so that HTTP or network overhead are not factors in the results. An area of future work for the framework is the development of querying over HTTP allowing non-Java systems to be compared.

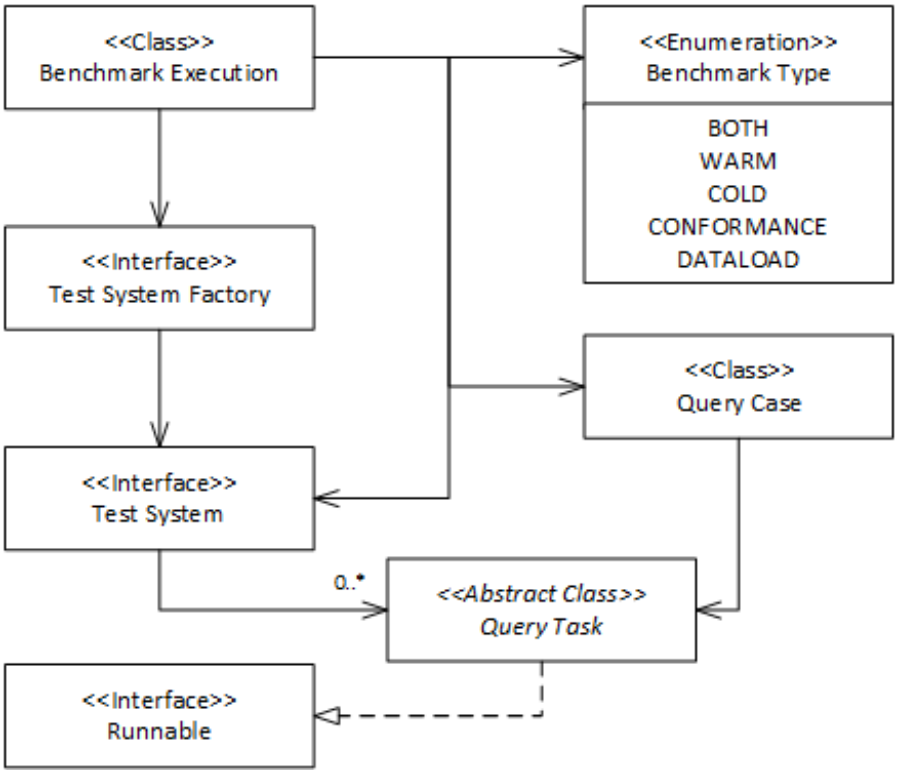


FIG 2. *Class diagram of the main Benchmarking Framework classes and interfaces. The Benchmark Execution requests instances of the Test System from the Test System Factory. The Test System executes Query Tasks for each Query Case that the Benchmark Execution has been set to iterate over and according to the Benchmark Type.*

The Gradle build tool was selected to assist in dependency management, distribution, and separation of each system into its own project to avoid dependency conflicts. The Parliament test system uses a legacy version of Apache Jena which conflicts with the more recent version used in the implementation. A core project contains the benchmarking framework for the loading and execution of queries and output of results.

In common with Geographica's approach, a Test System interface is defined which is implemented by the target system to integrate with the framework, see Figure 2. However, in our approach new instances of the Test System are produced using a Test System Factory interface to create separate instances. The initialisation period of these instances is recorded as a separate duration from dataset loading and query execution, again varying from Geographica, and is incurred when a system is initially accessed via the Java API, which may be once, e.g. during start-up for production usage, or multiple times, e.g. during application execution or development.

The execution of queries is performed through a separate Query Task class, varying from Geographica, that is extended by each Test System to standardise the three stages of query execution, i.e. preparation, processing and closure, and recording of durations. This execution standardisation means that all target systems perform the same work and are monitored consistently. The Query Task is executed on a separate thread and is monitored for its duration to limit excessively long query executions.

Investigation was made into incorporating the Java Measurement Harness (JMH) [21] for the benchmarking framework and index size experiments discussed later. JMH is designed to provide consistent and accurate micro-benchmarking by protecting against JVM optimisations and other benchmarking factors. However, the initialization process of Apache Jena conflicted with the benchmarking process of JMH. Therefore, an approach consistent with that adopted by Geographica was followed. This did allow finer reporting of sub-benchmark durations and output results to be developed but an area of future work is the incorporation of JMH.

The results of a query are unpacked in the processing stage as String maps and so decoupled from the SPARQL query engine implementation. The query performance in speed and number of results are summarised to file for comparison across queries and iterations. The executed query is also written to file for cross-comparison and the detailed values returned by the query can be optionally output. The query design in Geographica places filter functions in the SELECT part of the query rather than the WHERE body. This means that even if a function fails due to an exception, a null result is returned for every binding in the WHERE body and can be interpreted as false positives, i.e. a result with no actual data. The framework reports both the number of results returned and the number which contain data, so that comparison can be made and failing queries identified.

The execution of the framework can be performed in four modes: Cold, Warm, Both and Dataload. In Cold mode, each iteration is a new instance of the Test System. The Warm mode, an initialising query is performed on a single Test System which is then re-used for the target iterations. The Both mode allows the Cold and Warm modes to be run consecutively. In the Dataload mode, a specified dataset is loading into the Test System's persistent storage for the target iterations with the storage being cleared after each iteration. These modes follow the principles described in the Geographica benchmarking paper [7].

Two default query sets have been defined in the framework, i.e. Geographica Microbenchmark and Geographica Macrobenchmark which are each discussed in more detail later. These query sets are all loaded from external SPARQL query files and are not programmed into the framework, unlike the design of Geographica. This means that additional query files can be read into the framework for user benchmarking and allows visibility and customisation of the existing queries as standards evolve. The queries relating to a results set are also generated for user inspection.

The Macrobenchmark set utilises additional data which is substituted into queries to provide variation between iterations. This additional data is again

provided from external files to allow visibility and customisation. Development of support for using additional data in user-defined benchmarking and conformance testing is an area of future work in the framework. Each usage of the Macrobenchmark produces a set of consistent queries for each iteration across the set but which vary between generation and so between target Test Systems. Reproducibility is achieved by using the output of queries generated for one target system as user-defined queries for testing on subsequent target Test Systems.

The described framework can be extended to provide automated testing of GeoSPARQL standard conformance and other SPARQL standards. This can be achieved by developing a conformance query set and dataset for comparison with the results of the test system. Below is a comparison of the Geographica benchmark query sets with the standard to highlight the areas where conformance testing can be applied:

- Only WKT serialisation is used with no GML serialisation testing.
- Not all Geometry extension non-topological functions are utilised and there is inclusion of Strabon specific functions.
- The RCC8 and Egenhofer relation families are not included and only Simple Feature relations used.
- There is no conversion between different SRS, either geographic or projected, and the datasets all use the default WKT SRS URI.
- There is no testing of the Geometry properties, Topological Vocabulary, Query Rewrite or RDFS Entailment extensions.

Inclusion of these requirements and use cases in a conformance query set allows demonstration of compliance and functionality between test systems and are not covered by the Geographica benchmarking queries. The defining and automated testing of conformance queries is an area of future work for the framework.

## 5. Implementation Challenges

This section outlines the notable challenges that have been encountered when developing the GeoSPARQLJena library.

### *5.1. Well Known text (WKT)*

The WKT serialisation is a widely used serialisation for geospatial data and referenced in the GeoSPARQL standard. Parsers are available to extract the geometry data from the serialisation for processing. However, the GeoSPARQL standard allows the encoding of Coordinate Reference System URIs as part of the WKT string, which is non-standard and so not compliant with existing parsers. In addition, support of the Geometry Property Extension in the GeoSPARQL standard requires additional metadata for the geometry to be gathered that was not produced by the parsers.

A final issue is the support for more than two coordinate dimensions. The WKT definition permits between 2 and 4 coordinates (XYZM) to be defined but the available parsers only supported two dimensions (XY). The GeoSPARQL standard only considers the X and Y dimensions for spatial operations but it does not explicitly preclude the geometries containing 3 or 4 dimensions from being present in a dataset. Therefore, a WKT parser was implemented to overcome these issues and produce a geometry for the JTS library.

### *5.2. Geographic Markup Language (GML)*

The GML serialisation is an XML based geospatial serialisation that is widely used and referenced in the GeoSPARQL standard. There are several versions with varying cross compatibility and GeoSPARQL implementations are required to state their GML version support. The current implementation of GeoSPARQLJena supports the GML Simple Features Profile 2.0, which is a simplified profile of GML 3.2. This profile is intended for compliance with the Simple Features standard and therefore aligns with the GeoSPARQL standard.

GML parsers are available to extract the geometry data but present several issues. The examined parsers are designed to parse GML on a document basis using XML preamble and schema. The GeoSPARQL usage of GML is small isolated fragments of GML geometries that have been encoded within RDF. It was also not possible to locate a GML Simple Feature schema to inform the parser and a comprehensive set of example serialisations for the profile were not available. The extraction of geometry metadata and supporting a range of coordinate dimensions, as described in the previous section, was an additional issue affecting the parsing process. Therefore, a specific parser was implemented to overcome the identified issues. While compliance has been sought the development of this parser and support for other GML versions is an area of future work.

### *5.3. Units of Measure/Buffer and Distance Conversion*

It has been outlined previously that each SRS can use one of several unit systems. These can be dependent upon the type of SRS, e.g. *projected* uses linear metres and *geographic* uses non-linear degrees. While a standard international system of units has been established there is still a variety of distance units and sub-units in local usage, such as the standard mile in the United States and United Kingdom. URI definitions have been added for degrees and standard miles, along with their sub-units, for user convenience. Additional units can be registered using the Java Measure API. Conversion between this non-standard and standard units can then be performed consistently.

The GeoSPARQL standard supports *distance* and *buffer* functions which accept arguments in different unit systems, e.g. a geometry in degrees and an expected result in metres. Conversion directly between these units is problematic when considering that a degree in metres for coordinates near the north pole

is different to that of coordinates near the equator. The *buffer* function, which expands an enclosing area around a geometry, presents similar difficulties.

These issues are addressed by converting the geometry's SRS to an appropriate SRS for the units, if necessary, upon which the function is applied. For the *buffer* function the resulting geometry is converted back to the original SRS, while the *distance* function ensures the correct units are returned. When the functions are performed on a *projected* SRS but requiring non-linear units a conversion is made to the WGS84 SRS and then the *Euclidean distance* is still found, following the acceptance of error found in the Simple Features [5] and GeoSPARQL [11] standards.

### *5.4. Relation Families Intersection Patterns*

The GeoSPARQL standard describes three Relation Families for identifying spatial relationship between geometries. This includes definitions of the DE-9IM intersection patterns. The Simple Features and GeoSPARQL standards are consistent in stating that the DE-9IM intersection pattern for the *equals* relation, as used by the Simple Features, Egenhofer and RCC8 relation families, is "TFFFTFFFT". However, this does not yield a true result when comparing a pair of point geometries, which are equal. The Simple Features standard states that the boundary of a point is empty. Therefore, the boundary intersection of two points would also be empty.

An alternative intersection pattern of "T*F**FFF*" is used in several spatial libraries [18, 14] and has been applied in the GeoSPARQL-Jena library. This intersection pattern is the combination of the *within* and *contains* relations and yields the expected results for all geometry types.

### *5.5. getSRID Function*

A source of contradiction between the GeoSPARQL and Simple Feature standards is the use of a Spatial Reference System Identifier or SRID. In Simple Features, this provides an integer value for look up against a local catalogue of spatial reference systems and reflecting an internet disconnected design. In GeoSPARQL, the same term is used in the getSRID function, Requirement 5., for the URI string of the geometry literal, which uniquely identifies the spatial reference system and reflects an internet connected design. In seeking cross-compatibility these two different pieces of information, an integer and a URI, are being related together when they are conceptually different. Any future revision to the GeoSPARQL standard would benefit from the two concepts being separated into two functions or renaming the getSRID function.

### *5.6. GeoSPARQL Schema*

The OGC have published the GeoSPARQL v1.0 standard as an RDF/XML schema v1.0.1. This can be imported to provide RDFS and OWL inferencing

on a conforming dataset. However, the published schema does not conform with the standard. The property *hasDefaultGeometry* is missing from the schema and instead the *defaultGeometry* property is stated. This prevents RDFS inferencing being performed correctly and has been reported to the OGC Standards Tracker. A corrected version of the schema has been used in the remainder of this work.

### *5.7. Conformance Dataset & Queries*

The GeoSPARQL standard provides several specific query examples with a small dataset. However, these examples are not exhaustive for the general statements of the requirements and in some cases refer to other standard documents, e.g. Simple Features. These standards are highly technical and lengthy making identifying and extracting the required information for an implementation problematic.

Semantic Web technologies are designed to be platform independent. This means that there is opportunity for a compliance dataset and queries to be published to accompany the GeoSPARQL standard. These can then be used or re-purposed by any Semantic Web system on a wide variety of platforms. This would allow GeoSPARQL implementations, and the automated conformance and benchmarking framework discussed previously, to test functionality and evidence compliance more clearly and consistently.

## 6. Experimental Results and Discussion

This section outlines the evaluation, results, and analysis of the GeoSPARQL Jena implementation against two existing GeoSPARQL systems, Parliament and Strabon. The first set of experiments analyses the effect of varying the size of indexes implemented in GeoSPARQL Jena. This is followed by an overview of the methodology and configuration of the benchmarking framework. The final sections discuss the results obtained for the three systems using the framework.

### *6.1. GeoSPARQL Jena Index Size*

The implementation design described in Section 4 utilises several indexes and registries for the on-demand caching of data. The term *index* has been applied for data that is retained for short-term re-use, e.g. extraction of Geometry Literals for queries, while *registry* is applied to data that has potential long-term re-use, e.g. transformations between Spatial Reference Systems. The registries are expected to be small and persist for the duration of the application while indexes vary in size as data is added and removed through expiry.

The performance benefit and sizing of the indexes are investigated in this section as the size of the indexes presents a compromise between the data extraction process, in-memory storage requirements and retrieval time for existing entries. A test dataset of 100,000 Feature/Geometry and Geometry/LineString

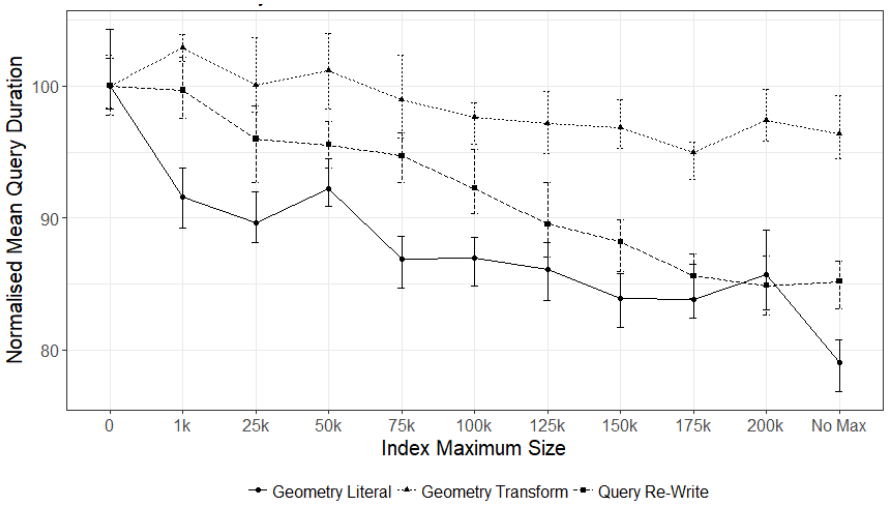


FIG 3. *Mean Query Duration by Index: Mean durations over 10 iterations. Graph shows means normalised by the mean duration of the no/zero size: Geometry Literal: 13.728s, Geometry Transform: 16.839s, Query Re-Write: 41.814s. Error bars show min and max durations.*

triples were randomly generated with GeoSPARQL schema and RDFS inferencing. This dataset was then tested against three queries using the *geof:intersection* and *geo:sfIntersects* functionality and four fixed test values. Each query was designed to test one of the three indexes: Geometry Literal, Geometry Transform, and Query Re-Write.

The queries were performed with a warm up execution followed by ten recorded iterations. Indexes were emptied between each iteration. A range of maximum index sizes were executed from zero, or no indexing, to no maximum size. The expiry time was set to 5 seconds with cleaning occurring at 1 second intervals and was selected based upon experimentation that indicated sufficient duration for data re-use but allowing cleaning to take place during the queries and therefore not retain all generated data.

The results of these experiments are shown in Figure 3. The overall trend in query duration can be seen to fall as the maximum index size is increased. The benefits to each index vary and range between 21.95% improvement for Geometry Literal, 15.13% for Query Re-Write and 5.03% for Geometry Transform. However, the improvement is not continuous as some maximum sizes decline the performance. These are likely caused by retention of values that are generated and not re-used later in the query but preventing more useful values being stored. Therefore, the overall benefit of using the indexes can be seen but the identification of specific maximum sizes is problematic.

It should be noted that identifying the maximum size for the indexes is complicated by background requests using the Geometry Literals during query processing. The test query applied an intersection function that produced 400,000

new Geometry Literal results, although many of these would be empty points, in addition to the 100,000 in the dataset and the 4 test values. During resolving a single query, the Geometry Literal Index received just over 1.7 million requests and peaked in size at 250,348. The peak sizes for the other indexes were Geometry Transform 100,004 and Query Re-Write 200,000, which can be predicted from the dataset and query structure. The queries for these two indexes still required large number of requests to the Geometry Literal Index, which was switched off, and so a balance across all three indexes is required depending upon the functionality utilised.

Any repetition of Geometry Literals in a dataset will also have an impact on the ratio of index max size to unique triples. The test dataset had no repeating Geometry Literals, but a dataset can contain non-unique literals yet still consist entirely of unique triples. There would also be variations in query structure and functionality that would affect the number and throughput of items processed and indexed before considering available memory and processing capability.

The benchmarking experiments in the following section were all performed using no max index size and an expiry time of 5 seconds. All benchmarks were executed with limitation to RAM and no memory issues were encountered with this approach. Further work is needed to investigate the effect on performance of dataset size and expiry time in conjunction with the max index size.

### *6.2. Experimentation configuration and methodology*

All benchmarking experiments were performed on the same desktop computer which was a x64 Windows 10 operating system with Intel Core i7-4820K with 16GB and Samsung 850 EVO 500GB. Previously outlined is that all three test systems provide a Java API. Identical Java JVM settings were used for each target system with a maximum heap size of 3GB and parallel garbage collection enabled.

In setting up the different systems there are several configuration points of note. Optimisation of database and graphstore performance can be achieve through tuning various parameters. A general principle has been followed of not optimising to the dataset available with each target system setup *out of the box* following only generally provided guidelines.

In the case of Parliament this meant that tuning parameters were not altered, spatial indexes were created during data loading and RDFS inferencing rules were enabled. GeoSPARQL Jena applied RDFS inferencing during data loading with an automated script applied upon completion. The script is provided by Apache Jena and counts the occurrences of properties and classes in the dataset to provide weightings during query execution.

The Strabon documentation provides several suggested PostgreSQL parameter values for use with different levels of RAM availability. These suggestions were applied but resulted in crashes during execution or exceeded the permitted range of PostgreSQL. Therefore, the default values have been used meaning a different approach to that used in the Geographica benchmark. Inferencing was activated during query execution.

TABLE 1
*Dataset loading and initialisation durations*

| | Loading | | Initialisation | |
|---|---|---|---|---|
| Test System | mean (s) | sd | mean (s) | sd |
| GeoSPARQL Jena | 90.694 | 7.401 | 0.047 | 0.003 |
| Parliament | 337.976 | 4.217 | 0.539 | 0.041 |
| Strabon | 374.439 | 6.194 | 80.317 | 1.882 |

The framework measures the duration of each query execution with three values: query preparation, query processing and their sum for the total duration. For brevity the total durations are reported here, unless specified. A general observation is that the Apache Jena based test systems, Parliament and GeoSPARQLJena, perform more work during iteration over the results in the query processing stage while Strabon spent relatively longer periods in the query preparation stage. This highlights the different strategies deployed by the target systems and the importance of considering the whole execution duration of queries.

In all cases a maximum of one hour was permitted for each iteration. Those cases which timed out are indicated in the text with any non-visible values in graphs being relatively small rather than missing. Each scenario was performed for five iterations.

### *6.3. Dataset Loading & Initialisation*

The six datasets published by Geographica have been used for the Microbenchmark and Macrobenchmark. These datasets use a pre-version 1.0 GeoSPARQL spatial reference system URI which was converted to the version 1.0 URI. This did not change the coordinate values but simply removed a legacy reference from the benchmarking.

The target system each use their own persistent storage method with all supporting multiple graphs. Each dataset was loaded into their own graph within the storage. In the implementation the Apache Jena Java API loading mechanism was used and not the Bulk Loader for large datasets, which may provide faster results. Table 1. shows the data loading and initialisation durations for the target systems.

The loading durations include both the import of triples and any associated one-time spatial indexing setup. The initialisation duration is the duration required to initially access the dataset through the Java API. Therefore, in a production environment this may occur once but in an iterative development environment will be incurred at every execution. In all cases the operations were performed five times with arithmetic mean and standard deviations shown.

The table demonstrates that the implementation, GeoSPARQL Jena, is noticeably quicker at both loading, 1.5 minutes compared to over 5 minutes, and

TABLE 2
*Microbenchmark test system rankings for Cold and Warm runs*

| | 1st | | 2nd | | 3rd | |
|---|---|---|---|---|---|---|
| Test System | Cold | Warm | Cold | Warm | Cold | Warm |
| GeoSPARQL Jena | 8 | 16 | 18 | 10 | 0 | 0 |
| Parliament | 0 | 0 | 0 | 0 | 26 | 26 |
| Strabon | 18 | 10 | 8 | 16 | 0 | 0 |

initialising the datasets. This makes it very suited to a development environment when coupled with its minimal setup requirements. Parliament has a more prolonged setting up period when it is building its spatial indexes but is then quick to access prior to querying. Strabon is the slowest to load the datasets and requires a lengthy initialisation period of 80 seconds. This is incurred prior to any query execution and regardless of its data requirements. Therefore, Strabon would be unsuited to a developmental or exploratory environment where an application is being frequently restarted. The following sections discuss the query performance excluding the identified initialization periods.

### *6.4. Geographica Microbenchmark*

The queries executed in this benchmark are those published by the Geographica project and follow the same numbering system. These utilise the non-topological query functions of the Geometry Extension and the filter functions of the Geometry Topology Extension for the Simple Feature relation family and therefore are a subset of potential queries for the GeoSPARQL standard. It was necessary to fix several namespaces in the published queries, but the body of the queries were unchanged. The queries Q6, Q28 and Q29 have been excluded as they contained spatial functions which are not defined in the GeoSPARQL standard and therefore only Strabon could execute.

The results of the microbenchmark are shown for Cold runs (Figure 4), Warm runs (Figure 5) and combined rankings (Table 2). Parliament did not complete any of the queries in the Spatial Joins section within the one-hour time limit. Results are shown in all cases for GeoSPARQL Jena and Strabon but in several cases results were achieved in less than a second. In all cases, except Q15, the number of results returned by all three systems were identical. Q15 uses the distance function to identify geometries within a radius of a fixed point. The difference in results may be attributable to precision of calculations and different approaches to handling the conversion of distance units in query. GeoSPARQL Jena is using the JTS library for calculations which does not reduce precision when other geospatial libraries may do so.

The query uses linear *metre* units for distance between two geometries. All geometries in the datasets are in CRS84, which is a geocentric SRS using non-linear *degree* units. In GeoSPARQL Jena, the handling of linear units with non-linear SRS geometries is achieved by temporarily transforming the geometries to a linear SRS. This is necessary as the ratio of *metres* to *degrees* is not fixed

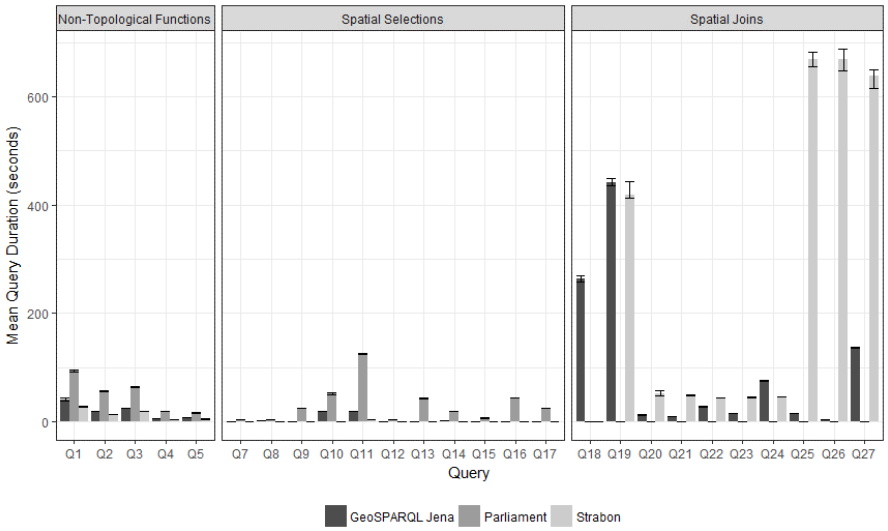


FIG 4. *Mean Query Duration (seconds) by test system microbenchmark in Cold run. Error bars show min/max. Parliament timed out in Q18-Q27.*

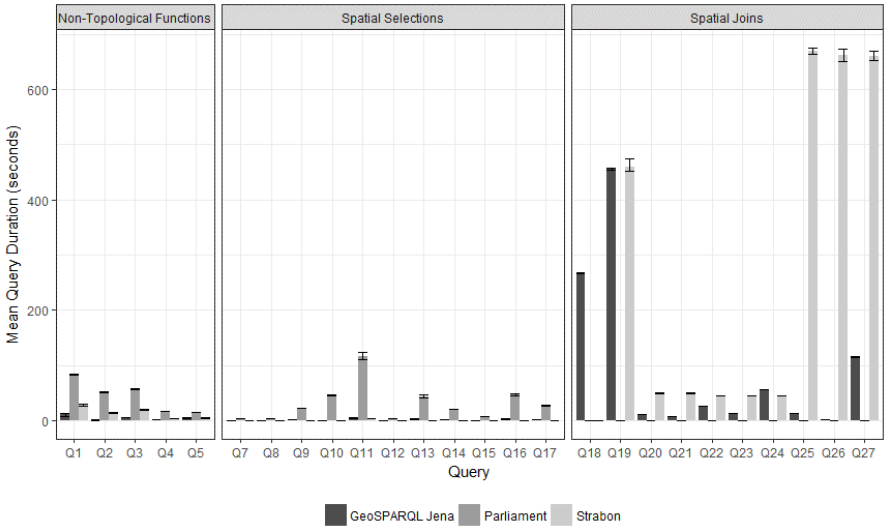


FIG 5. *Mean Query Duration (seconds) by test system microbenchmark in Warm run. Error bars show min/max. Parliament timed out in Q18-Q27.*

and varies according to the latitude of the coordinates. Alternative approaches may use a fixed ratio and tolerate the error as latitude varies.

Another query of note is Q18 which tests for the Simple Features *equals*

relation. Strabon is noticeably quicker and completes this in less than 1 second when GeoSPARQL Jena requires just under 4.5 minutes and Parliament times out. This query checks the spatial equality between two graphs numbering 7,551 and 21,993 geometry literals to produce 166 million combinations. Therefore, the speed of processing this scale of combinations by Strabon is noticeable. Further investigation suggests that Strabon is not checking for spatial equality but only lexical equality by matching strings. Spatial equality, but not lexical equality, can exist when two geometries have the same coordinates but use different SRS that reverse the axis order or when one geometry shape has additional coordinates that do not alter the shape, e.g. two straight lines that start and end at the same coordinates, but one has intermediate coordinates. Therefore, the results of Strabon in this query may be accurate but may not fully comply.

A final query of note is Q26 where GeoSPARQL Jena completes in less than 4 seconds while Strabon requires approximately 11 minutes and Parliament times out. This query tests for the Simple Feature *touches* relation, where the boundaries of two geometries align but do not cross into each's interior. The two graphs used are the same graph which contains 325 complex multi-polygon geometries for 105 thousand combinations. GeoSPARQL Jena on-demand indexing will mean that these geometries are readily available while optimisations for this relation, e.g. bounding boxes around the complex shapes, mean that cases can be rejected quickly. Similar optimisations can be applied in Q25 and Q27 and could explain why GeoSPARQL Jena is noticeably quicker than Strabon.

Table 2 shows that Strabon achieves the most frequent fastest completion times in the Cold runs but GeoSPARQL Jena is more often fastest in the Warm runs. This reflects the on-demand indexing implemented in GeoSPARQL Jena. Parliament is consistently the slowest performer and did not complete one set of queries in the time limit. Strabon is generally quicker in the Non-Topological Functions and Spatial Selections queries. These queries utilise a single graph dataset while the Spatial Joins require geometries from two separate graph datasets. This indicates that Strabon is performing optimisations for intra-graph operations, e.g. during the lengthy initialization period, that may not be possible or tractable inter-graph, i.e. the *Curse of Dimensionality*.

Although it would be expected that the Warm runs complete quicker than the Cold runs a comparison between the mean completion times finds this is not always the case with GeoSPARQL Jena 6 cases, Parliament 5 cases and Strabon 10 cases. In many cases the differences are fractional or less than a couple of seconds but the largest are exhibited by Strabon in Q19 and Q27 and GeoSPARQL Jena in Q19. It should also be noted that GeoSPARQL Jena is only marginally slower than Strabon in many queries, expect Q10 and Q11 and Q18 discussed earlier, while there is a noticeably lengthy difference in many queries in the Spatial Joins section for Strabon.

### *6.5. Geographica Macrobenchmark*

The queries executed in this benchmark are those published by the Geographica project and follow the same numbering system. In keeping with the Geograph-

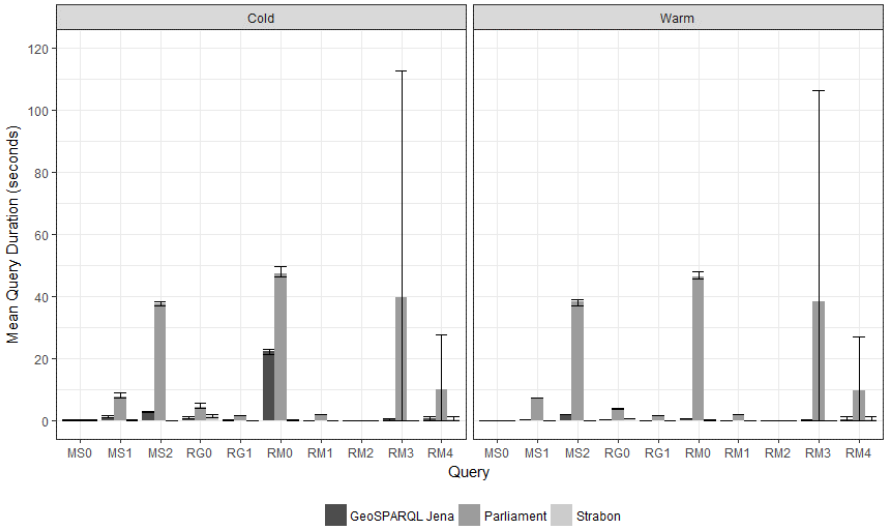


FIG 6. *Mean Query Duration (seconds) by test system for both macrobenchmark runs. Error bars show min/max. Incomplete iterations for Parliament in RM5 and Strabon in MS2, RG1 and RM3. RM5 not shown due to mean completion by GeoSPARQL Jena in 30 mins compared to Strabon in 3.4 seconds (see main text).*

ica approach there is variation between query iterations. Each block of queries within a set uses related data but the data selected varies between iterations, e.g. MS0, MS1 and MS2 have consistent data in the first iteration but a different set of data is used in the second iteration. The same resulting queries has been used across each of the test systems to allow direct comparison. This variation between iterations can be seen in Figure 6 where there is more noticeable variation in maximum to mean completion durations. This can partly be attributed to many iterations returning zero results and resolving very quickly.

There is consistency in the number of results returned in all but two queries. Strabon returns zero results in four iterations of MS0 when Parliament and GeoSPARQL Jena both return a single result. The second case is MS2 where there is a difference in result count for the only non-zero iteration between Parliament and GeoSPARQL Jena. Strabon did not complete, repeatedly, this iteration and so no completion duration is reported. Strabon also did not complete all the iterations for RM3 and RG1. In all cases where Strabon did not produce results the other test systems had non-zero result counts and Strabon produced an error rather than timing out, suggesting an issue with the processing of results within Strabon.

Parliament timed out after one hour in the RM5 query. This was also the most intensive query for GeoSPARQL Jena and required approximately 30 minutes in each iteration. The RM5 query uses two *intersection* filter functions and a geometry *difference* function with data from two graph datasets. In all iter-

Table 3
*Macrobenchmark test system rankings for Cold and Warm runs*

| | 1st | | 2nd | | 3rd | |
|---|---|---|---|---|---|---|
| Test System | Cold | Warm | Cold | Warm | Cold | Warm |
| GeoSPARQL Jena | 6 | 7 | 5 | 4 | 0 | 0 |
| Parliament | 1 | 1 | 3 | 3 | 7 | 7 |
| Strabon | 4 | 3 | 3 | 4 | 4 | 4 |

ations GeoSPARQL Jena required a similar duration to complete, even when no results were returned. Strabon was able to resolve the zero results in less than 1 second and only took 15 seconds for the non-zero result iteration. The *intersection* filter function is also used in microbenchmark Q19 and both test systems achieved similar results. This suggests that Strabon's query optimisation selected a resolution strategy which reduced the problem space much quicker than GeoSPARQL Jena.

Further examination of GeoSPARQL Jena finds that the first *intersection*, requiring a cross product of 231.8 million cases, was resolved before the second *intersection* that only required 6,285 cases. If order of resolution is reversed then the former is only 592 thousand cases. SPARQL query optimisation for resolution is controlled by the underlying Apache Jena query engine and persistent storage. Simple manipulation of the query, without changing content, dramatically reduces execution time to approximately 8 seconds, which is quicker than Strabon's result with the original query.

The manipulations required are: 1) moving a *intersection* filter function into the relevant graph clause and 2) reversing the order of the graph clauses. This allows the Apache Jena query optimisation to apply the more efficient resolution strategy. In Strabon, applying the manipulated query returns no results despite it being valid SPARQL 1.1 query. This highlights the different optimisations but also potential non-compliance of the examined query engines.

The rankings across the three test systems in Table 3 show that GeoSPARQL Jena achieves the fastest results in the majority of queries. These rankings do not consider the discussed query manipulation for RM5 so Strabon is still ranked as faster than GeoSPARQL Jena in that case. Strabon achieves very quick results in certain cases but did not complete all iterations for three queries. Parliament did not resolve one query and is generally the slowest to complete.

## 7. Conclusion

This work sets out the design of a fully compliant implementation of the GeoSPARQL standard utilising an RDF graphstore. Previous implementations have achieved partial compliance of a sub-set of extensions. The examined implementations have generally provided geospatial and RDF functionality by extending relational databases requiring additional configuration and setup. Additional design points that have been achieved were Semantic Web standards compliance,

minimal configuration and a short initialisation period. This means that the implementation would be suited to both development and production environments.

An on-demand indexing approach was designed and implemented to retain geospatial and supporting data for improved performance. Experimentation was performed to consider the impact of controlling index size and retention periods on query duration. This innovative approach of short and long term caching of key data has been demonstrated to reduce query completion times by up to 20% without incurring initialisation delays. This on-demand indexing also has general utility for other SPARQL applications that require repeated processing of datatypes that are computationally expensive to de-serialise.

To understand the implementation's performance capabilities, a benchmarking framework has also been designed and implemented to perform a comparison with two existing GeoSPARQL systems: Strabon and Parliament. This framework can be expanded for additional test systems through Java interfaces to ensure consistent querying and minimal integration effort.

The design of this framework is based upon importing, processing and reporting on SPARQL queries, rather than a hardcoded queries as developed in the Geographica benchmarking framework for GeoSPARQL. Therefore, it has utility for benchmarking SPARQL queries broader than GeoSPARQL. This approach also provides transparency in the query content and data being benchmarked while variant queries can be easily written and processed. Queries can also be benchmarked on alternative datasets.

An area of future work is extending the benchmarking framework with queries to test GeoSPARQL module conformance and report results. Several areas for conformance testing have been identified and feedback is welcome on further additions. Additional areas include incorporating additional GeoSPARQL systems for testing and allowing user data to be incorporated into SPARQL query templates. Finally, the incorporation of the Java Measurement Harness would provide more robust benchmarking timings. Its usage was investigated but issues with the detail of benchmarking results and conflicts with Apache Jena's initialisation process were not able to be resolved.

The reported benchmarking results show several advantages of the GeoSPARQL Jena implementation over the two benchmarked systems. The dataset loading and initialisation of the implementation are noticeably quicker. In the alternative systems, pre-query spatial index preparation is undertaken that is not incurred by the GeoSPARQL Jena implementation. Despite this, the benchmarking process has demonstrated that the GeoSPARQL Jena implementation has query times comparable or better than the alternative systems in all but one query case (RM5). The GeoSPARQL Jena implementation also completed all the GeoSPARQL benchmarking queries of the Geographica query set.

It has also been identified that minor manipulations to the benchmarking queries can trigger dramatic improvements in query resolution. This was found in the single query case (RM5) where GeoSPARQL Jena was dramatically slower than the Strabon test system. The application of two structural changes to the query, without altering content, reduced query time from 30 minutes to 8 seconds

and overtook Strabon's performance.

This demonstrates that SPARQL query writing and optimisation can be very specific to the query engine being utilised. This highlights the challenge in preparing benchmarking queries which do not over accentuate the performance of a specific test system. It further emphasises the need for benchmarking frameworks to be adaptable in processing alternative versions of queries supplied by the user; as designed and implemented in the benchmarking framework.

The GeoSPARQL-Jena [1] implementation, which has been invited for integration as a module by the Apache Jena project, and GeoSPARQL Benchmarking framework [2] have both been published as open source projects.

## References

[1] Albiston, G. L. and Osman, T. (2018). Semantic Model Assembly Framework for the Generation of Travel Demand. *Manuscript in preparation.*

[2] Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S. and Schaub, T. (2016). The GeoJSON Format Technical Report, Internet Engineering Task Force.

[3] Clementini, E., Felice, P. D. and Oosterom, P. V. (1993). A small set of formal topological relationships suitable for end-user interaction. In *International Symposium on Spatial Databases* 277-295. Springer.

[4] Cohn, A. G. and Renz, J. (2008). Qualitative spatial representation and reasoning. *Foundations of Artificial Intelligence* **3** 551-596.

[5] Consortium, O. G. (2016). Simple Feature Access - Part 1: Common Architecture.

[6] Egenhofer, M. J. and Franzosa, R. D. (1991). Point-set topological spatial relations. *International Journal of Geographical Information System* **5** 161-174.

[7] Garbis, G., Kyzirakos, K. and Koubarakis, M. (2013). Geographica: A benchmark for geospatial rdf stores (long version). In *International Semantic Web Conference* 343-359. Springer.

[8] Guo, Y., Pan, Z. and Heflin, J. (2005). LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web* **3** 158-182.

[9] Harris, S. and Seaborne, A. (2013). SPARQL 1.1 Query Language Technical Report, World Wide Web Consortium (W3C).

[10] Kyzirakos, K., Karpathiotakis, M. and Koubarakis, M. (2012). Strabon: a semantic geospatial DBMS. *The Semantic Web–ISWC 2012* 295-311.

[11] Perry, M. and Herring, J. (2012). GeoSPARQL - A Geographic Query Language for RDF Data Technical Report, Open Geospatial Consortium (OGC).

[1] https://github.com/galbiston/geosparql-jena

[2] https://github.com/galbiston/geosparql-benchmarking

[12] Peterson, D., Gao, S., Malhotra, A., Sperberg-McQueen, C. M. and Thompson, H. S. (2012). W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes Technical Report, W3C.
[13] Schreiber, G. and Raymond, Y. (2014). RDF 1.1 Primer.
[14] GeoTools - The Open Source Java GIS Toolkit.
[15] (2016). Oracle Spatial and Graph: Benchmarking a Trillion Edges RDF Graph Technical Report, Oracle Corporation.
[16] (2018). Apache Marmotta Technical Report, Apache Foundation.
[17] (2018). EPSG Geodetic Parameter Dataset Technical Report, International Association of Oil and Gas Producers (IOGP).
[18] (2018). Locationtech Java Topology Suite Technical Report, Eclipse Foundation.
[19] (2018). Apache Commons JCS - Java Caching System Technical Report, Apache Foundation.
[20] (2018). EH Cache Technical Report, Software AG USA.
[21] (2018). Java Measurement Harness Technical Report, OpenJDK.

# Appendix C: Semantic-based Assembly Framework for the Generation of Travel Demand - Prepared Journal Article

This appendix contains the journal article prepared for publication of the work undertaken during this thesis. It focuses on the application of Semantic Web to the travel demand generation process from Chapter 3; the concepts and structures of the travel demand process in Chapter 4; and the evaluation of generated travel demand in Chapter 7.

# Semantic-based Assembly Framework for the Generation of Travel Demand

**Gregory L. Albiston, Taha Osman and Evtim Peytchev**

*School of Computing and Technology,*
*Nottingham Trent University,*
*Nottingham, United Kingdom*
*e-mail:* gregory.albiston@ntu.ac.uk; taha.osamn@ntu.ac.uk; evtim.peytchev@ntu.ac.uk

**Abstract:** This work applies a knowledge modelling approach in the design of a framework for the generation of travel demand for traffic simulation. The proposed framework is based on interchangeable modules to integrate the main stages of travel demand modelling supported by an engineered knowledge base. This approach is intended to promote greater behavioural diversity, incorporation of more diverse contextual data, facilitate access to online datasets, and support users to undertake and validate investigations across a range of models and implementations. The framework provides the user with direct control to modify the schema, control the selection of data, select alternative modules to execute, and the potential to remotely retrieve data and execute modules. There is a discussion of the travel demand modelling process, identification of the primary stages and the proposal of a core schema to describe fundamental concepts for the construction of the knowledge base and integrate between modules. The framework is investigated through a prototype, which generated travel demand across a full day and performed simulation utilising two third-party traffic simulators based upon the configuration by the user schema. The problem of travel demand generation was separated into discrete task modules with identification of features for alternative design or further modularisation. The prototype evaluation identified no significant barriers to the proposed approach using Semantic Web technologies with only minor technical details providing challenges and illustrated the diverse results of traffic simulation. It is proposed that the framework provides a basis to develop new and existing approaches to travel demand generation to improve modelling outcomes and adoption.

**Keywords and phrases:** Activity-Based Models, Travel Demand, Traffic Simulation, Knowledge-Based Systems, Semantic Web.

## 1. Introduction

Traffic and transport planning is a wide ranging field that impacts on the daily lives of most members of society through transport congestion, infrastructure investment, safety, pollution etc. The generation of travel demand is one stage in the process of constructing traffic simulations to investigate transport planning problems by modelling the movement of residents, non-residents and freight [28]. Travel demand models have evolved from initial aggregate four-step demand models (trip generation, trip distribution, mode choice and route assignment) to disaggregate activity-based travel demand models [8].

The transition to activity-based models has been limited, partly due to perceptions of additional data requirements and implementation burden [8]. The advantages of activity-based models include developing behavioural realism, integrity between components, greater spatial and temporal resolutions, and supporting disaggregate micro traffic simulators. However, these objectives have been met to varying degrees through a wide variety of implementations [28],[8].

In this work, we propose a Semantic Web based framework that identifies data structures for fundamental components of activity-based travel demand models. It is proposed that the semantic modelling of these concepts into a structured knowledge base will allow the development of modular components that can be more easily interchanged, while still allowing users control over the organisation of their experimental scenarios. Further, it is proposed that a Semantic Web basis will enable the assembly of models and simulations from both local and online sources enabling quicker modelling and improved comparability of results.

### *1.1. Current Challenges in Travel Demand Modelling*

Several challenges are presented to users when selecting and utilising travel demand models and traffic simulators. These frameworks are generally developed as collections of tools and models that fulfil the distinct functions of the modelling process [8]. Due to development focus, one tool in the collection may provide advanced features or design while another is more limited, such as the supported transport modes, activity model, routing algorithms or human behaviour model. Therefore, a user must evaluate between frameworks and compromise on certain features to utilise others.

The file formats supported by a framework may also force its selection. Geospatial and road network data are provided in a wide range of standard file formats with each framework supporting a subset and potentially its own bespoke format. A user may not have the technical skills or resources to convert their own data file format into one of the supported formats. Other input data will rely on the framework's generation tools or need conversion to each framework's schema as no common standard exists.

Given that all models and simulations are incomplete representations of the physical world, it is good practice for results to be compared across multiple implementations [11]. This is particularly the case in activity-based models that are often reliant upon population sampling to reduce data gathering and computational complexity but introduces greater uncertainty [28]. A range of model designs are also employed for determining variables such as mode, destination and scheduling time period [8]. Therefore, users should expect to be able to perform investigations into multiple parameter values and across multiple frameworks with minimum investment of time and resources. Setting up multiple frameworks can require repeated dataset conversion with a thorough understanding of each implementation. The analysis of framework output also requires conversion of each result set to the desired analytical format.

There is potential to convert the output of one framework's tool for reuse in another framework. However, the user would again need to have a thorough

understanding of both tools' configurations to avoid error and integration gaps. Frameworks are also developed in a variety of programming languages and platforms, therefore transferring data between tools or utilising domain libraries may require the user performing a manual process unsuitable for large numbers of investigative cases and risking user error.

The process of selecting, obtaining and preparing input data for these frameworks also places a burden upon the user. Neither MatSIM [18] or SUMO traffic simulators [23] provide tools for the generation of activity-based travel demand. Geographic, road network, activity pattern and population data are inputs to the framework and their tools. Each dataset is typically published by a different agency or organisation. A user will need to identify the required data for their target area, source from the multiple providers, clean, combine and reformulate for use in their model and the selected framework. The data requirements between users are likely to be very similar with only relatively specific enhancements for their interest area. Yet no unified datasets or combining mechanisms have been identified.

Each of these identified challenges requires investing additional resources and potentially developing ad hoc solutions which could compromise the investigation by not accurately integrating the appropriate travel demand models and traffic simulators. Users also face the barrier that the outcome of these activities are not necessarily portable to another framework. They will need to be repeated to include the additional framework in an investigation.

The development of travel demand models also requires several further features. There has been a primary focus upon week day commuter car and freight transportation to evaluate the impact of traffic control [28]; however commuting represents 15% of trips and 20% of distance travelled [33] with flexible working patterns and business hours becoming more common [16]. Technological developments also present alternatives for co-ordination and planning, e.g. carpooling, car-sharing, automated vehicles and vehicle communication [34], [6]. There is an identified need to model multiple days, improve co-operation between household members, incorporate social network relationships and develop non-utilitarian human behaviour and decision making [8]. Human behaviour modelling tends to apply a single or a few approaches to all individuals and not consider all the contextual information that could be utilised [28]. These design goals further increase the breadth and depth of data requiring management.

The recent trends of Big Data, Open Linked Data and volunteer initiatives, such as Open Street Map, has seen the increased gathering, processing and availability of large detailed datasets. This presents an opportunity for traffic demand models to incorporate a greater range of information and modelling processes. Travel demand models have also been identified as having contributions to domains outside of transportation, e.g. environment and health [8]. This introduces greater complexity in the breadth and depth of data, which knowledge modelling can assist.

A knowledge base can be constructed that uses a common data schema, which describes the relationships between fundamental concepts, applied to the investigative data. The framework functionality can then be defined as loosely

coupled inter-operable modules, which are configured through external module parameters in the knowledge base. These modules can implement innovative approaches or wrap existing tools to incorporate state of the art research. Semantic Web technologies are well placed to support inter-operability and a knowledge modelling approach with their design principles of structured data, machine to machine processing and open data exchange between applications [5]. Technologies have also been developed to convert existing data sources in flat files, structured files [19] and relational databases [25] into Semantic Web formats, while Semantic Web standards support data retrieval from remote online sources.

The development of a knowledge model for the travel demand requires the identification and expression of the problem domain's common concepts and their relationships. Research has been undertaken to develop standardised and consistent definitions for transport modelling of movement and activity [4]. The integration of separate travel demand models and simulators or the transference to new geographic areas has been investigated, but on a case-by-case basis [38], [13] with constructing interfaces highlighted as a time-consuming process.

Research has been taking place into transferring existing data sources to the Semantic Web, including demographic census and similar data that form inputs to traffic demand modelling [7]. Relevant to these efforts, semantic ontology based traffic forecasting management solutions have also been successfully developed [24], [26] and transportation and land use related ontologies have also been published [32], [10].

The EC INSPIRE project [12] seeks to provide a standardised spatial infrastructure data format across the EU, including transport networks. Other research efforts have investigated the additional data requirements to incorporate transport models into the CityGML format [35] and the conversion of the GML format to RDF [36] but to the best of our knowledge, there are no published works focusing on traffic demand modelling.

### *1.2. Contributions*

This paper presents a novel semantically-modelled assembly framework for travel demand generation. We claim that the paper has made the following contributions to the field:

1. The developed framework integrates the multiple stages of travel demand modelling by providing for the development of interchangeable modules; resource online datasets; and assisting in the modelling process. The user has direct control of module and data selection enabling execution control between modules and extension of the schema to their use case.
2. The work identifies the key stages of travel demand modelling and proposes RDF schemas for the development of travel demand knowledge bases to integrate between stages and describe road networks.
3. The mechanisms are described to locally and remotely assembly and execute travel demand models across multiple data and service providers.

4. The framework has been implemented as an activity-based travel demand model, integrated with third-party traffic simulators, using Semantic Web technologies and we discuss its design, results and challenges.

The remainder of this work is divided into the following sections. The first section provides an outline of the key technologies of the Semantic Web and their relevance in this work. The second section describes the key components of the framework for traffic demand models. The third section outlines the core schema developed for the framework. The fourth and fifth sections discuss the implemented prototype and features for the configuration and remote execution of the framework. The sixth and seventh sections describe the experimental scenario results and implementation challenges. The closing section provides our conclusions and future work in this area.

## 2. Semantic Approach to Travel Demand Modelling

This section provides an overview of both Semantic Web technologies and traffic demand modelling. Current travel demand models cover a broad range of design decisions and concepts that it is impractical to exhaustively explore and discuss. Therefore, general components are identified with specific reference to aspects of activity-based models, but potential exists to apply the framework more generally. Future developments to these models will likely increase data complexity that an extendable knowledge base can assist in managing.

The many tools in the travel demand and traffic simulator frameworks would each form a module of the framework. These modules would then interact through a core extendable schema. The purpose of the framework is to enable users to retrieve, transform and re-use their data across multiple travel demand, traffic simulator and supporting modules for their investigations, while also incorporating their own modules and data schema. The Semantic Web is the technology of choice for building the framework.

### *2.1. Overview of Semantic Web Technologies*

The Semantic Web is not a single technology but a hierarchical collection of formal standards and recommendations with supporting tool implementations. Its objective is to enable the structuring of data for automated interpretation and facilitate exchange between applications.

The basis of Semantic Web technologies is the modelling of data to develop a knowledge model, i.e. a domain ontology, of common concepts and their relationships. Contextual facts relevant to the ontology can be asserted to construct a knowledge base. Semantic modelling upon a knowledge base using the relations, and their defined meanings, enables the inferencing of additional implied facts or the identification of inconsistencies.

knowledge bases can then be shared and utilised as the basis for applications and task solving models. Sharing and re-use of commonly defined relations

through ontologies allows the incorporation of facts from multiple knowledge bases to extend a dataset. The Semantic Web uses these modelling benefits to retrieve, join and transform data.

A mechanism for interoperability between knowledge bases is the sharing and extension of vocabularies and ontologies. These concepts and relations can be applied to provide consistent understanding and structure. Numerous vocabularies have been developed including spatial (GeoSPARQL), temporal (OWL Time) and domain specific (transport domain topics of traffic disruption, automotive, infrastructure).

The main components of the Semantic Web are:

1. Resource Description Framework (RDF): the fundamental data structure using a directed labelled graph approach based upon *subject-predicate-object* triples.
2. SPARQL Protocol and RDF Query Language (SPARQL): query language to explore, retrieve and modify RDF data from local and remote graph-stores.
3. Schema Languages: standardised vocabularies to describe relationships and inferences that can be derived when applied to a compliant dataset using an inference reasoner, e.g. RDFS and OWL.
4. Rule Languages: express additional relationships within the schema using *if-then* structured statement and can include functions to produce new data, e.g. SWRL and SPIN.

The primary objectives of this work is to provide a data focused, modular approach that can be explored and adapted by the user. Therefore, the focus is upon investigating SPARQL based querying with RDFS inferencing as the combination provides flexibility and control without certain complexities introduced by OWL schemas or rules-based extensions, e.g. modelling restrictions [29], Open World Assumption, computational complexity. However, overlap exists such that certain outcomes could be achieved using several approaches, each with their own advantages.

### *2.2. Architecture of the Travel Demand Generation Framework*

The core stages for activity-based traffic simulation in sequence are population synthesis, travel demand generation and traffic simulation [8]. Data is passed from each discrete stage with iterative feedback sometimes occurring from traffic simulation into travel demand generation. Each stage is also reliant upon a variety of input datasets, such as demographics, network supply, travel diaries and land use, see Fig. 1. The knowledge base provides a common repository for data that modules, e.g. Scheduling, Trip Planning or Network Routing, can access through SPARQL queries.

An immediate benefit of a Semantic Web approach is the storage of all this data in the extensible graph structure of a graph database to form, with an engineered schema, the knowledge base. This knowledge base provides the structure for interactions between different modules and data concepts, e.g. the Trip

Planning module requesting an estimated travel time from the Network Routing module.

The input and output data from each stage, scenario or execution can be partitioned into different graphs within the graph database for reuse or extraction through SPARQL queries. Users can convert existing datasets or the output from tools into RDF for import into the graph database. Additional data, e.g. vehicle characteristics or social network relationships, can be linked and stored without interference with the core data in the same graph database.

This means that implementations can extend the core knowledge base but still operate on the same graph database. Therefore, modules with alternative design principles or providing new functionality, e.g. environmental or communication models for vehicles, can be applied to the same set of underlying data. The structure of input data can be transformed or linked to existing concepts by extending the core schema using SPARQL queries.

Execution of modules is achieved in SPARQL queries using *property functions* for each module. Therefore, the SPARQL query can control the data selected and its processing. This allows the user to change between modules by simply modifying the query. Modules can also be selected according to the data's characteristics through class membership, data properties or inter-relationships.

This is illustrated by the example in Fig. 2 where two sets of *persons* are selected from the population, one based on their membership of an income quartile class and the other based upon being an employee with an income greater than the stated threshold. The modules are *property functions* and can be identified here by the *rou* namespace prefix. Each set of *persons* has a different routing approach selected to generate travel demand based on their contextual data, which has been a criticism of many travel demand models. Therefore, different behaviours can be described for subsets within a population but as part of the same query. Modules can interact based on the data provided rather than how that data is generated.

Another advantage of SPARQL are federated queries that can retrieve data

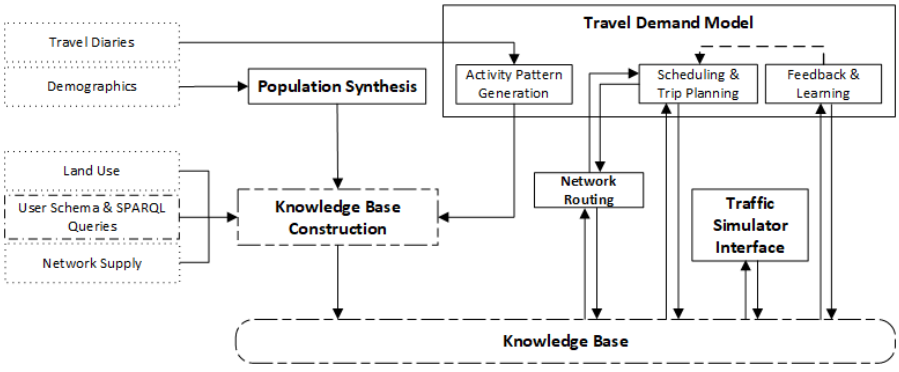


Fig 1. *Main stages for travel demand modelling.*

from local and online graphstores. These graphstores can be queried to retrieve relevant RDF triples and accelerate knowledge base construction, e.g. LinkedGeoData for land use and network infrastructure data [32]. A step further is remote services providing *property function* modules to perform functionality of travel demand modelling based on input parameters. A user can focus upon obtaining the simulation stage output using queries, containing configuration parameters and data selection patterns, to target remote services for data and functionality.

These queries can be separated into multiple stages to allow retrieval and manipulation of partial data in the local knowledge base or to redirect to alternative module implementations. Parameters are supplied as URI references accompanied by the URL reference of the remote service. Modules can query against the service URL to retrieve the data associated with the parameter URI and then perform their task. Named graph URI can also be used to partition and manage data within each graph database, which can assist with data disposal and dataset alternatives, e.g. time periods, geographies, schema compliance and user requests.

National, or international, knowledge bases would allow re-use of quality and consistent datasets while remote modules can provide best-practice implementations. Currently users construct a local dataset and model for their specific problem or geographic area by establishing their own infrastructure and sourcing, processing and transforming data files for the selected tools. The proposed approach would reduce these requirements as the data and interfaces are already designed around the same modelling paradigm.

To conclude, the knowledge-based approach to integrating the population synthesis, travel demand and traffic simulation processes results in a novel frame-

```
PREFIX rou: <http://example.org/routing#>
PREFIX ex: <http://example.org/local#>

SELECT ?person ?schedule
WHERE{
    {
        ?person a ex:Quartile4IncomeEmployee .
        ?schedule rou:routingMethodA ?person.
    }UNION{
        ?person a ex:Employee .
        ?person ex:income ?income .
        FILTER(?income > 50000)
        ?schedule rou:routingMethodB ?person .
    }
}
```

Fig 2. *Example SPARQL query to select different routing modules.*

work that improves the integration between independently developed implementations, assist users to manage and structure local data, allows users flexibility to compare alternative implementations and provide the basis for accessing standardised datasets, scenarios and tools as internet services.

## 3. Constructing the Travel Demand knowledge base

The purpose of this stage is to bring together the synthetic population, activity patterns, land use, network infrastructure and local modelling design as illustrated in Fig. 1. The user can develop a formal schema using inference languages and apply a reasoner to the data to perform inferencing of virtual triples, i.e. triples not present in the knowledge base without a reasoner.

The inferencing process can automatically allocate instances to classes, create data properties, infer relationships and identify contradictory data in the knowledge base, e.g. a child who possesses a driving licence. Class membership can be assigned based upon relationships and data e.g. vehicle ownership, age or income. Alternatively, SPARQL queries, or a rules language engine, can perform these tasks with varying flexibility and restrictions.

SPARQL queries can also permanently transform, remove or add data to the knowledge base. These queries are stored as text files and so can be distributed to share best practice with users able to make local modifications before applying to the knowledge base.

Traffic and transport, particularly in microsimulation, heavily utilise spatial relationships between objects. Graph databases can be extended to natively support these spatial relationships, as in some relational databases. Extensions complying with the GeoSPARQL standard [27] enable SPARQL queries to perform spatial searches without the need for external processing. Four general stages of knowledge base construction have been identified:

### *3.1. Population Synthesis*

This stage converts aggregate demographic data into a dis-aggregated set of persons grouped into households. Each person and household are described by a set of characteristic variables. In some existing travel demand models, an integrated synthesiser is available, but the user is then limited to the chosen algorithm and implementation choices in this active field of research. In the proposed approach, any population synthesiser can be utilised once its output is serialised into RDF and published locally into the knowledge base. The population's characteristic variables can be reformulated to a user's chosen schema as part of the knowledge base construction.

### *3.2. Spatial Allocation*

The synthesised population must be aligned with the land use data for the zones and regions of interest. The generated households are produced for an entire

zone but must be allocated to the individual sub-zone locations of homes. The linking process between households and homes can be achieved through user defined SPARQL queries, a library of best practice approaches or implemented modules for more sophisticated techniques. This step is described in literature as part of the population synthesis step but there is limited reporting of techniques and theoretical results [21].

By considering it as separate to population synthesis, the user can adapt the allocation process to their chosen schema, available data and selected approach, rather than the implementation of the selected tool. For example, an allocation based upon house prices and number of bedrooms would produce a different knowledge base instance to an allocation based upon aligning household characteristics of a property with a generated household. Numerous alternatives can be explored with consideration of the varying impact on the traffic simulation outcome.

### *3.3. Individual Classification & Linking*

In existing activity-based models, the functionality is typically based upon hard-coded characteristic values, such as specific household compositions or types of locations, or a fixed set of configurable parameters, e.g. school and retirement age. This functionality includes decision making processes with a criticism that existing models apply a single or a few approaches to all individuals [28], i.e. all employees make travels decisions in an identical manner or commuter and tourist decision making is identical.

It is proposed that these shortcomings can be overcome through local user control and generic module design. The user's control, over the local schema and in manipulating the knowledge base data, provides choice in the characteristic values and their inter-relationships that are present in the user's scenarios. For instance, a core schema would define *location* and *activity* concepts, but it is the user's local schema that extends these to define the specific types of locations and activities. Therefore, it is the user that determines the design of the local schema and data rather than fitting the data to a module's design assumptions.

Modules should be designed against generic concepts, rather than specific instances, so that the user has as much flexibility as possible. For example, a routing module would not specify the modes of transport supported but instead the data parameters required to perform routing for any mode. Certain cases may require a module to extend the breadth of the core schema, but should be minimised, e.g. vehicle routing module that considers road conditions, e.g. low bridges, requires additional vehicle characteristics not required by a generic router.

In the Semantic Web, classes are sets of individuals and can be sub-classed to any hierarchical depth [14]. A person, household or activity are all examples of individuals in this context. An individual can belong to multiple classes that can be asserted or inferred. Classification can be based on context using the individual's existing class membership and the values, cardinality and inter-connection of properties. Illustrative examples are:

1. A person with access to a car and possessing a driving licence belongs to the Car Driver class.
2. A person aged between 5 and 11 belongs to the Primary School Student class.
3. Leisure activities sub-classed into Exercise, Sport or Culture classes but also Indoor or Outdoor.
4. People working at the same location are inferred to be colleagues of each other.

The hierarchy depth of classification and properties becomes a user choice to an arbitrary level of detail based on data sources, design assumptions and implementation context. e.g. the core schema triple "Person hasActivityAt Location" can be sub-typed as "Employee hasEmplyomentAt Workplace" and "Student hasEducationAt School". The user asserts the data according to the sub-types, but generic modules can still operate upon the core concepts through the inferred memberships.

Filtering according to temporal, spatial or any other characteristic, e.g. opening hours or activity location, allows different contexts to exist within the same knowledge base. Locations modelled with an area of effect, e.g. school catchment or retail operational area, enable partitioning and selection rather than assuming a pervasive effect as in many existing models.

The allocation of an individual to a class, or their existing relationships to other individuals, can be used to apply default values or create new relationships, e.g. a person belonging to a household is inferred to be resident at the household's location. The creation of new relationships can also be constrained by applying filtering. Persons could be associated or limited to activities in a certain geographic area or specific types. Different derived schema and data will produce alternative knowledge bases that function with generic modules. Modules that extend the core schema would operate without interference to core schema modules when applied to the same knowledge base.

### *3.4. Network Conversion & Land Use Relations*

This stage consists of two parts: the conversion of road network, and other transport infrastructure, data into RDF format and the linking of land use data to the road network. Formats for road network information typically follow a node (junction) and edge (road) graph structure but there is a need for a standardised RDF vocabulary for transport networks and supporting infrastructure.

The INSPIRE project [12] includes a transport infrastructure theme. Work is in progress to develop RDF vocabularies but no vocabulary yet encompasses the whole transport domain for simulation purposes. Other research has investigated additional data requirements of transport models for CityGML [35] and the conversion of GML to RDF [36]. A standardised schema and tools would allow routing operations and interpreting of road semantics to be performed at the data level without the current dependency on traffic simulator or GIS systems.

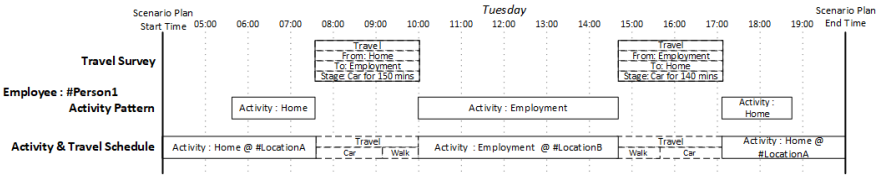


FIG 3. *Time-line representation of Travel Survey as basis for Activity Pattern template and contextual Activity & Travel Schedule.*

Once the network infrastructure has been stored in the knowledge base then geospatial relationships are identified between infrastructure and land use locations. This primarily consists of identifying the proximity of roads, buildings and public transport access points to each other.

## 4. Travel Demand Generation

A variety of travel demand models have been developed based upon different techniques, e.g the Four Step Model, Activity-Based and Agent-Based. The focus in this work is upon constructing activity and travel schedules based on template activity patterns as part of a modular design as illustrated in Fig. 1. The generation of activity pattern templates takes place prior to the knowledge base construction as they serve as an input to that stage. However, since travel demand models exist that do not require activity patterns it is discussed at this stage.

### *4.1. Activity Pattern Generation*

The activity pattern is a typical data structure of activity-based and agent-based models and provide a template, or *skeleton* [37], [3], from which a person's activity and travel schedule are assembled according to the context, see Fig. 3. The generic templates are populated with instance data to form an individual's schedule. The initial and final activities are extended to fill the entire scenario time-period. The locations and travel choices will create varying travel durations between activities. Waiting times are included in the activity or travel stage [4] to remove empty periods, but time gap filling approaches vary, e.g. extend activity duration; travel time contingency or plan extra activities.

The patterns available may be fixed [28] or derived from travel diaries of a sample population using classification algorithms [3], [30]. The travel diaries may be used to derive the activity choices, durations, indicative start and end times, journey mode and household co-operation. Sets of activity patterns can be associated with households and individuals in the synthetic population based upon the corresponding characteristics. Any activity generator could be utilised once its output is serialised to RDF.

### 4.2. Scheduling

The activity pattern templates are applied to a context, i.e. population, geography and scenario, to form a schedule. Adapting to an alternative context requires consideration of preserving minimum activity duration; tolerance for timing slippage; and extending or including additional activities to fill time gaps. The scheduling process typically covers a single day, but with development needed for multi-day schedules; improved co-operation between household members; and incorporation of social network data [28]. Schedulers also need to ensure that consistent travel occurs with tours returning to the start location and journeys returning to an individual's reference location [4].

Activity prioritisation is used in some scheduling approaches to allow co-operation between household members. Mandatory activities, e.g. education and employment, are determined first with invariant start and finish timings. Maintenance, e.g. food shopping, and discretionary, e.g. leisure, activities are then assigned with strategies for duration and inclusion [8]. Schedule coordination, e.g. adults escorting school children and shared vehicle usage, is modelled by mandatory activities being scheduled on an individual basis and then reviewed for co-operative travel across the household before allocating lower priority activities.

### 4.3. Trip Planning

This stage is a key distinction between travel demand models with travel decisions typically consisting of activity location; trip mode and activity time frame [17]. Design decisions are influenced by choice type and resolution order due to their interdependence and impact on later decisions [8], [3], [17]. Route choice is a further development in activity-based models [28], [8], but already a feature in some agent-based models [37]. These processes represent a range of design approaches which some implementations combine. Below we briefly outline four predominant approaches for activity-based travel demand models [28], [37] but other techniques can be built upon the proposed knowledge base:

- Constraints Based: All combinations of route, mode and locations between the activities are found as a sequence of travel. A schedule is checked for its feasibility according to travel and time constraints.
- Discrete Choice/Econometric: A person is assumed to be an entirely rational entity that chooses from a finite set of weighted probability choices. Typically, the probabilities are calculated to maximise *utility* using attributes related to the choice and person.
- Computational Process Model: Designed as heuristic responses so that behaviour is considered more habitual than self-optimising. Behaviour is modelled through *if... then...* rules activated by contextual variables.
- Agent Based: A system that defines discrete, self-contained agents that possess a set of characteristics and operate within an environment. These agents react to external events, exhibit control over their actions, learn

from past experience and communicate with other agents to pursue their goals.

### *4.4. Feedback & Learning*

The process of feedback and learning is based upon the relative success of the proposed travel plan. The outcome of the simulation process is fed back into the traffic demand model to inform the decision-making process. Generally learning is performed following a batch simulation of a schedule. However, iterative schedule adjustment due to travel delays during simulation are being developed [28].

In the proposed framework the whole knowledge base can be made available for interface with simulator APIs or as part of an artificial transport framework [31]. Therefore, the simulation stage can be performed as an iterative stepwise process with travel demand being adapted as simulation conditions change. This would facilitate both the current inter-simulation scheduling and rescheduling intra-simulation and make available all contextual information for learning.

### *4.5. Network Routing*

The topology of the transport network has an influence on the travel decisions taken by persons and the connectivity between locations, infrastructure and services. Many transport simulators provide tools to perform routing using a variety of algorithms, e.g. A* and Dijkstra. This module interprets the network supply information in the knowledge base to inform the travel demand models and removes dependency on transport simulator tools. This module can be executed either prior or during demand modelling, with generated routes stored in the knowledge base for re-use or reference. However, the prior option produces an exhaustive set of all route combinations which quickly becomes very large as the number of locations increases.

An area for future work is consideration of semantics present in road network datasets, e.g. temporal context, trip purpose and physical characteristics. These features are lacking in the examined routing tools such that routes ignore private or resident only access, road closures at specific time periods and tall vehicles under low height bridges etc. These are characteristics that affect routing and can be accommodated as additional contextual data in the proposed knowledge base framework. Alternative services can also be modelled as distinct modules, e.g. taxi services, car sharing, lift sharing and autonomous vehicles.

### *4.6. Travel Simulator Interface*

The outcome of a travel demand model is a person's activity and travel schedule. Simulator interface, or aggregation into Origin/Destination matrices, can be achieved through knowledge base query and data conversion into the required

format. The information required by a specific traffic simulator may not require the complete schedule with requirements varying. For example, MATSim [18] as a minimum requires people to be identified with a plan of activity type, end time, travel mode and location in XML format. SUMO [23] requires both person schedules and vehicle routing with start and stop locations and departure times in XML format. TRANSIMS [30] requires person and vehicle information including the household, person, purpose, mode, vehicle, start and end locations with departure and arrival times in CSV format.

Network infrastructure information is an additional input that is already needed in the knowledge base for travel demand generation. Simulators typically support their own bespoke file format for configuration parameters and network supply, but some standard network topology formats are supported. Therefore, interfacing to a simulator will require specific interfacing modules but some topology serialisations would be re-usable.

## 5. Assembling the Travel Demand Generation Framework

The previous section discussed the components for travel demand generation with focus on activity-based models and the advantages of applying a Semantic Web approach. This section discusses the knowledge modelling of the framework (Fig. 1) as schemas of the main concepts and their interaction. These have been implemented in a modular prototype discussed in the next section.

The core schema design diagram notation in this section has classes linked by key *object property* or *sub-class* relations, including cardinalities. Additional *object* and *data properties* are shown beneath class names. Example non-core extension properties and classes are shown by dashed lines.

### *5.1. Person, Travel Group, Activity and Location*

The core concept of traffic demand models is the Person and their relationships to Locations and Activities, illustrated in Fig. 4. A Person represents any individual who travels in the scenario, so could be sub-classified by the user. Each Person can be a member of a grouping for organisational and travel purposes, e.g. households. The use of *sub-class* and *sub-property* relationships enrich the knowledgebase's data but still retain the validity of the schema. A Location represents a spatial point of interest, including building and open spaces. Accessibility to these locations can vary for different modes, e.g. pedestrian access, motor vehicle parking, public transit links and freight delivery.

A multi-dimensional relationship can be formed between Person, Activity and Location. In a single model a Person could be linked to a single Location for certain Activities, e.g. employment, education and residence, and multiple Locations for other Activities, e.g. retail and leisure. However, it is a modelling assumption, i.e. determined by the user's schema and selected modules, if all Persons have a single Location for certain activities and not a requirement of the core schema.

Each Location can also provide zero or more Activities. For example, a school can provide both primary and secondary education which have different effective time periods and eligible school ages. A school is also a place of employment for teaching and administrative staff. Similarly, homes are the residence of individuals but also a place to visit for social interactions between friends and relatives.

The Activity itself may be modelled as unique to a Location or shared between multiple Locations. Each Activity has an effective time and days to reflect availability, such as morning and afternoon opening times. Therefore, a Location can have multiple Activities with different characteristics but the same Activity Type. The Activities can be sub-classed according to their characteristics while retaining a grouping through the enumerating of Activity Types.

The enumeration of the Activity Types forms a *Value Set* and ensures OWL 2 compliance as Object Property relationships must be between individuals and not classes [14]. This approach means that a single Activity Type can form a relationship that links many Activities to Activity Patterns. Each component (Activity and Activity Pattern) identifies a single Activity Type, which together forms a multiplicitous relationship, e.g. employment would be an Activity Type while employment at a specific office would be an Activity. Otherwise a user would have to identify and link every relevant individual Activity to an Activity Pattern and so impose a modelling burden.

A similar approach is taken to express a Person's travel modes as defined by the Mode class. These are either the personal or public transport modes a Person uses or those of their Vehicles. Locations can also identify Modes which have access and in turn those that do not. For example, city centre locations with no parking facilities would not be the direct destination for people using a car. Similarly, locations without wheelchair access would not be selected as

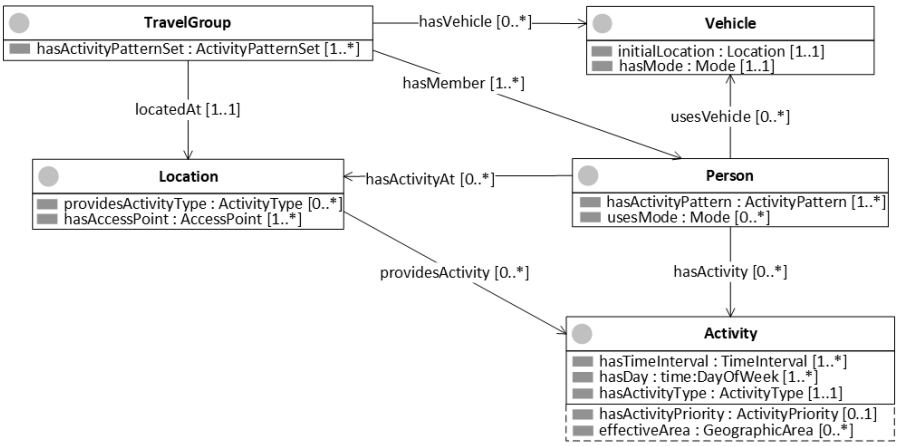


FIG 4. *Schema for Travel Group, Persons, Locations and Activities.*

viable for those people using that mode.

The relationships can be formed using the class, characteristics (data properties), geographic relation or arbitrarily asserted. An example SPARQL query is shown in Fig. 5 to both classify Persons as school age and link them to their local school according to its geographic "catchment" or effective area. The *OPTIONAL* clause ensures that all are classified, even when not in a school catchment area. Given a Location may provide multiple Activities then effective areas are applied according to Activity rather than Location. Extending properties for the Locations would be to apply comparative weighting for popularity based on the time and day of the week.

Further detail in terms of sub classes, relationships and characteristics can be included in the knowledge base by the user as required. For example, a highly detailed set of land use data could distinguish between several types of buildings, their occupants and the types of activities they provide in a hierarchy and structure specified by the user.

### *5.2. Activity Pattern and Activity Pattern Sets*

In an activity pattern approach, travel diary data is used to derive contextless templates that describe a series of general activity types with non-continuous time durations, see Fig. 3. This is modelled by an Activity Pattern that consists of a day of the week identifier and a time ordered list of items. Each item identifies an Activity Type, applicable time-period and travel distance range. The travel distance range specifies the minimum and maximum distance that should be travelled to reach the activity, whether asserted during knowledge base Construction or used to search for locations by the Scheduler module.

These Activity Patterns are grouped into Activity Pattern Sets, so that con-

```
PREFIX ex: <http://example.org/myKnowledgeBase#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>

INSERT{
    ?person a ex:PrimarySchoolChild .
    ?person ex:hasEducationAt ?school .

}WHERE{
    ?person a ex:Person ; ex:age ?age.
    FILTER( 5 <= ?age && ?age <= 11)

    OPTIONAL{
        ?person ex:hasResidenceAt ?house .
        ?school a ex:PrimarySchool ; ex:providesActivity .
        ?education a ex:PrimaryEducation ; ex:effectiveArea ?catchmentArea .
        ?house geo:sfWithin ?catchmentArea .
    }
}
```

Fig 5. *Example SPARQL query to classify Persons aged 5 to 11 as Primary School Children and link them to a local Primary School location.*

sistent planning and cooperation for joint activities, e.g. escort travel, can take place across Activity Patterns. Travel Groups can be linked with one or more Activity Pattern Sets according to their matching characteristics or directly asserted.

### *5.3. Travel Scenario and Activity & Travel Schedule*

The key output of a travel demand model is each person's schedule of activities and travel, see Fig. 3. This schedule is derived from a Person's selected Activity Pattern based upon their own and the scenario's contextual information. The Travel Scenario structure specifies the scenario information, e.g. start time, end time and day, along with execution parameters used by the modules, e.g. mode and scheduling characteristics, as illustrated in Fig. 6. Additional scenario parameters could include activity priorities or weather events. The Travel Scenario's URI reference can also be used as a graph URI so that all data generated for a scenario can be stored, exported and removed.

This allows multiple scenarios, with shared or varying parameters, to be executed upon the same knowledge base without interference. The absence of parameters can also control the scenario configuration. For example, a Person may have a specified Mode in the knowledge base but if the current Travel Scenario does not have those parameters then the Mode would be excluded.

There is no constraint placed on the user as to the number and definition of Modes. These can distinguish between distinct vehicle or travel types but

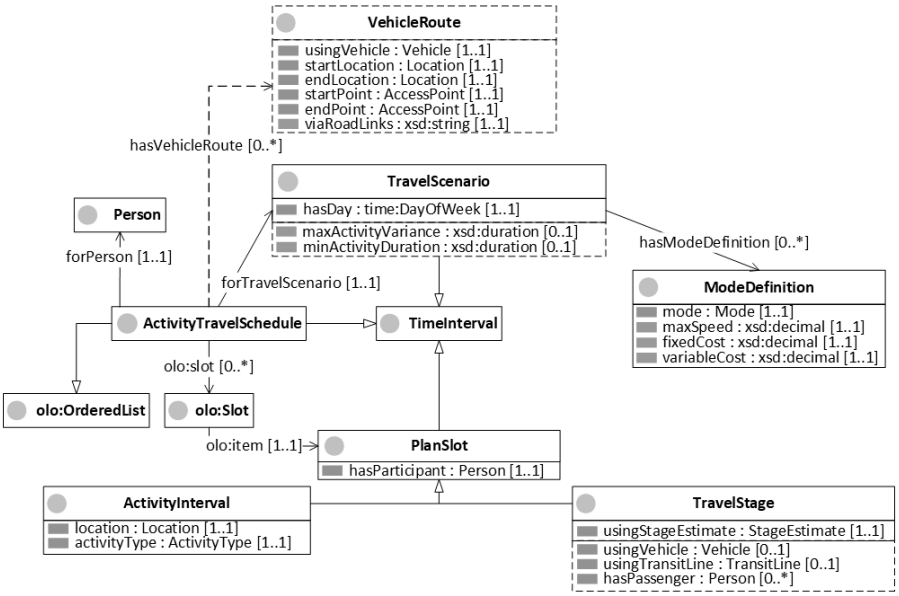


FIG 6. *Schema for Travel Scenario and Activity & Travel Schedule.*

also variation within the types. For example, a user can define multiple *personal* modes, e.g. walking and wheelchair, with different characteristics to reflect varying speeds between age groups.

The Activity & Travel Schedule is itself a time ordered list of continuous Travel Stages and Activity Intervals. The multiple Travel Stages, which form a trip between activities [4], provides the routing detail for moving between Locations using a single Mode. Multi-mode trips consist of consecutive Travel Stages. The routing detail are described as geospatial coordinates (Points), road segments (Road Links) and road junctions (Road Nodes) to support varying application and simulator requirements. The specific optional Vehicle or public Transit Line selected for travel is also identified in each Travel Stage so that their locations can be tracked or to influence later decisions, e.g. a return tour using public transport or a vehicle [4].

### *5.4. Trip Context, Stage Request and Trip Plan*

The travel stages of the schedule are determined by several factors. These include the person's characteristics, potential destinations, possible modes and the travel time-frame. These are described as a Trip Context which is passed from the Scheduling stage to the Trip Planning stage. The Trip Planning stage then determines the set of Travel Stages that will be selected to satisfy the Trip Context, see Fig. 7.

The Trip Context is split into multiple Stage Requests depending upon the number of modes and destination locations. Intermediate Stage Requests may be produced if multi-modal routing is taking place, e.g. walking to a bus stop or parking before walking. Information on available vehicles and their current location is specified and whether any resulting Trip Plan must position all vehicles at their *requiredLocation*, e.g. returning a vehicle home.

The Stage Request identifies a single origin, destination and mode that is sent to the Network Routing module. Other parameters may be supported by the Network Routing module, e.g. time-frame for multi-modal travel or interpreting road semantics. Travel by personal means, e.g. car, cycling or walking, is not temporally dependent, unless considering road semantics or traffic forecasting. Public transport is time dependent due to service availability.

The response to each Stage Request is a Stage Estimate. The Stage Estimate includes the routing information between origin and destination along with the stage's calculated values of duration, distance and cost based upon the scenario's mode definition. These values can be used for decision-making, e.g. utility in a Discrete Choice model, and considering the viability of a stage, or series of stages, in the trip's context, e.g. rejecting long duration stages. Further contextual information, such as heavily penalising or excluding walking during night time or cycling in the rain, can also be applied to the Stage Estimates. These discrete stages are chained by their origins and destinations to form multi-stage trips.

The generated ordered lists of Stage Estimates are narrowed to a single list by the Trip Planning modules decision-making process. The contextual information

is used to convert the selected list into Travel Stages within a Trip Plan, as the outcome of the Trip Planning stage.

### 5.5. Road Network

Road network topology file formats utilised in traffic simulators typically follow a node (junction) and edge (road/link) graph structure. This structure forms the basis for the RDF representation in Fig. 8 based upon INSPIRE concepts [12], SUMO [23] and MATSim [18] simulator formats and the GeoSPARQL standard [27].

The Road Network provides metadata and contains a collection of unidirectional Road Links that begin and end with Road Nodes. The Road Nodes provide the geospatial coordinates while the Road Links include local contextual data with global context provided by Road Link Types. Lane turning restrictions between Road Links at junctions is provided by Road Connections.

### 5.6. Activity & Travel Result

The Activity & Travel Schedule produced by the Travel Demand Model provides the intended activity and travel timings for a person during the scenario. The outcome of the Traffic Simulation stage is the activity intervals and travel stages of the schedule after experiencing the simulated environment and the travel plans of other participants. This information can then be used for analysis or incorporated into future scheduling actions, e.g. weighting Stage Estimate values, as feedback for learning.

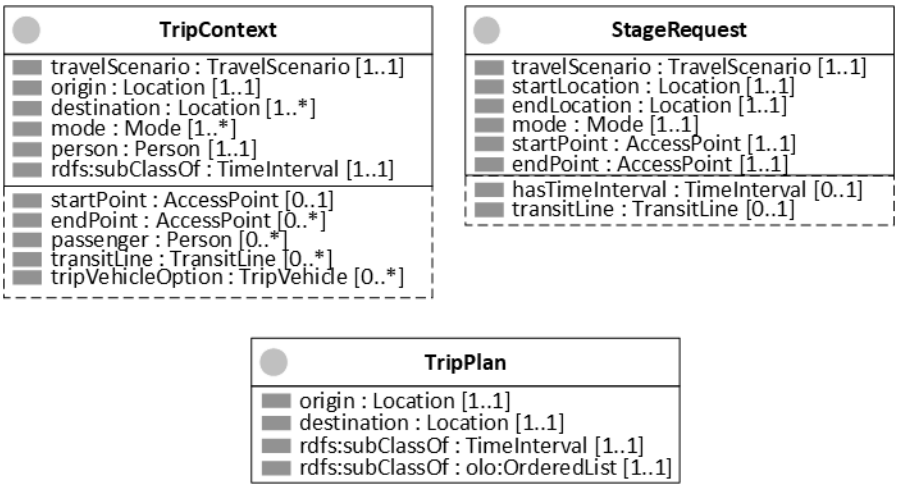


Fig 7. *Schema for Trip Context, Stage Request and Trip Plan.*

These activity and travel time intervals are captured in the Activity & Travel Result, which follows a similar structure to the Activity & Travel Schedule, see Fig. 6. It identifies the stages and links to the contextual information of the relevant Person, Traffic Simulator, Travel Scenario and Activity & Travel Schedule.

## 6. Implementation of Framework Prototype

The prototype was implemented using the Apache Jena Semantic Web API in a Java environment. Extension *property functions* are used to implement modules based upon SPARQL queries for data retrieval and execution. The use of queries allows user modification so that connectivity between modules can be re-routed to alternative choices. For example, additional modes or routing algorithms can be selected by modifying the SPARQL query to utilise alternative Network Routing modules according to the user's own criteria, e.g. person characteristic or class.

The knowledge base has been divided into multiple graphs based upon domain, e.g. road network, spatial locations and travel groups. Access configuration is controlled through an RDF graph which specifies the domain, graph URI and service URI. The service URIs can be a single local file URI or multiple HTTP URLs. This means that the knowledge base can be physically distributed over multiple datasets accessed using SPARQL's Federated Query standard. Execution can also be separated into multiple batches for multi-thread and multi-computer execution.

An extension framework complying with the GeoSPARQL standard [27] was also developed for geospatial querying [1]. This was necessary as Apache Jena currently has limited support for geospatial querying and alternatives required persistent graph database with varying GeoSPARQL

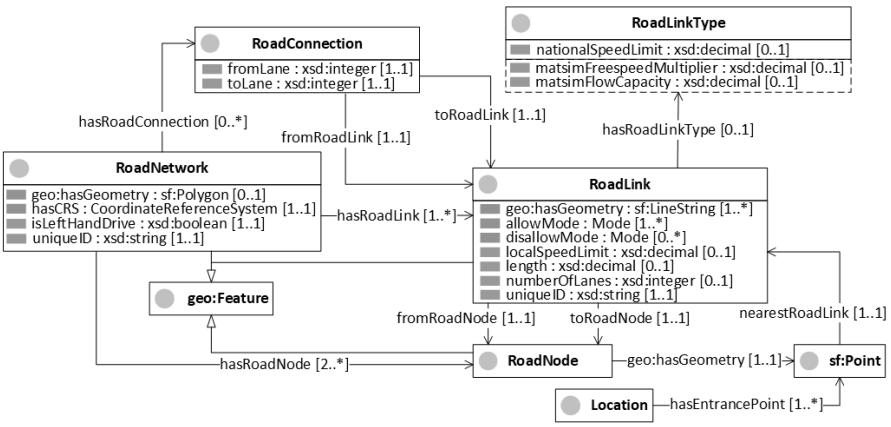


Fig 8. *Schema for road network described as edge and node graph.*

compliance, while the implemented extension provides full compliance with GeoSPARQL and allows flexible deployment of in-memory or persistent graph database.

Data generated by all modules is stored in a named graph, so that it can be easily exported or removed. The implemented Scheduler module takes an Activity Pattern Set for each Travel Group and then builds the schedule for each Person forwards in a single pass. Each schedule always contains at least one activity and must start and end with an activity.

The Activity Pattern for a Person consists of an ordered list. The items are processed with a Trip Context being sent to the Trip Planning module to request a Trip Plan of one or more Travel Stages. These Travel Stages are added to the schedule and used to inform the start time of the next activity. Each item in the Activity Pattern has an Activity Type. This Activity Type is intersected with a Person's related Activities, provided by one or more Locations, to produce a short-list of potential destination Locations. Therefore, Activities with a single Location, e.g. education, employment and residence, will always be respected and the Person will return to them during a tour.

If no Locations with the current Activity Type are asserted for a Person, e.g. leisure and retail activities, then the Scheduler searches for potential Locations. These Locations are selected based on the Activity Type and distance from the current Location within the minimum and maximum travel range radius of the Activity Pattern item. The travel range is iteratively expanded, according to a Travel Scenario parameter, until at least one potential Location is found. These Locations may later be rejected due to insufficient travel time but a travel attempt is made. Therefore, a Location can be selected based on the current context or the assertions in the knowledge base Construction stage.

All time in the scenario period prior to the first activity and beyond the last activity are expanded so that a full schedule is produced for the scenario. Trips between activities are scheduled for arrival by the next activity start time, with the earlier activity being extended to fill any gap prior to travel. The start and end times of activities are randomly varied using Travel Scenario parameters for maximum variation and minimum duration.

Should no trip be found in the available time period then a second attempt is made with travel time maximised with the activity as late and brief as permitted by Travel Scenario parameters. Activities may be merged when two identical activity types occur at the current Location; followed on without travel for activities at the same Location; or skipped due to insufficient travel time for a destination. Consistent vehicle usage between travel stages is ensured so that used vehicles are returned to the start location. Therefore, commuters do not abandon their vehicles after travelling to other locations but also do not insist on a single mode for an entire schedule. An area of future work is utilising activity priority and travel group co-operation in scheduling.

The Trip Planning module constructs a choice set of feasible Trip Plans and selects a single plan using a Random Utility Model (RUM) [8]. Taking the case of a single choice as in (1), an individual ($J$) has multiple influencing attributes ($x$). The fitted model's observed coefficients ($\beta^{\tau}$) are multiplied by the attributes

and summed to derive the observed value ($V$). Negative coefficients are applied when a smaller attribute value is preferred e.g. cost, time or distance.

$$U_J = \beta_J^\tau x_J + \epsilon_J = V_J + \epsilon_J \tag{1}$$

The observed value ($V$) and the unknown error term of unobserved variables ($\epsilon$) form the choice utility ($U$). The probability of selecting a choice is found from the choice set for an individual as in (2),by taking the exponential of the observed value and normalising it against the exponential observed values for all choices in the set.

$$P_i = \frac{e^{V_i}}{\sum_j e^{V_j}} \tag{2}$$

The Trip Plan utilities are calculated by summing the utility of each Stage Estimate using coefficient weightings for trip cost, duration and distance as in (3) of the individual Person (Fig. 9). This calculation was derived to provide differentiation between modes based upon varying trip distances. The utilities are calculated through query of the knowledge base and therefore can be directly modified by a user to retrieve alternative individual or global weights and utility equations without modifying the Trip Planning module.

$$U_J^i = \beta_{cost}^i x_J^i + \beta_{distance}^i x_J^i + \beta_{duration}^i x_J^i \tag{3}$$

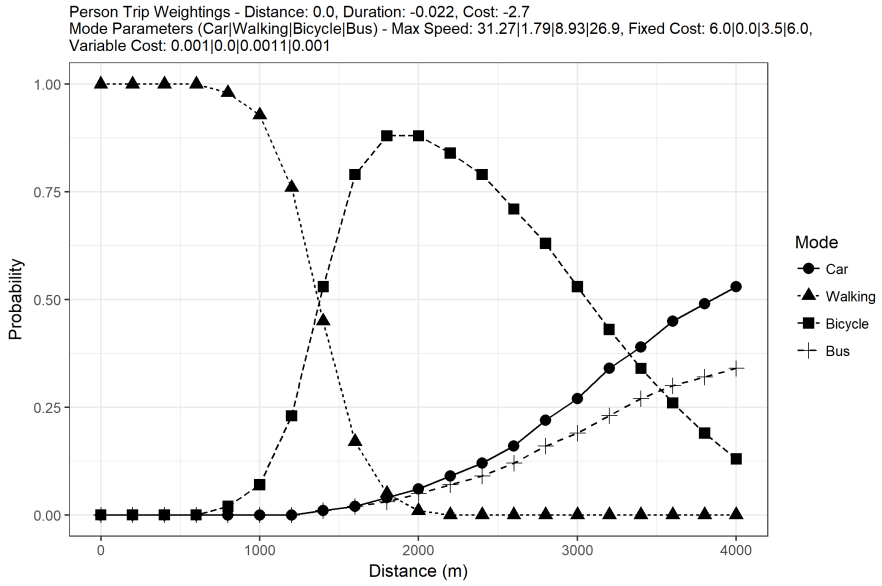


Fig 9. *Probability distribution over distance by mode.*

The utility of the whole choice set can be derived by firing semantic queries (SPARQL) against the knowledge base to retrieve the required contextual data and calculating the results. However, the final step of forming the probability set as in (2) and selecting a choice are not suited to standard SPARQL syntax due to repeated recalculation and increased query complexity. Instead, a generic *property function* module can achieve this more efficiently using imperative programming.

The choice set assembly algorithm utilises the provided modes, vehicles and transit lines to recursively build multi-stage, multi-mode trip permutations for each destination. Transfer Locations are identified to change between the current and next mode, meaning a new stage. The transfer locations are sorted by proximity to the overall destination to achieve the greatest progress in the current mode. As proximity does not always equal the shortest path between locations, e.g. routing via a river bridge, several locations are selected for each transfer, using a scenario parameter.

Those modes which are utilised by vehicles or public transit lines are required to satisfy additional conditions. These include the vehicle being already located at the transfer location for stage start and the stage ending at any required location. Similarly, public transit modes must use locations utilised by a specific Transit Line at both the stage start and end.

These potential stages of a trip are given routing detail by the Network Routing module. The routes between origin and destination with the selected mode are found using A* shortest path algorithm. Bidirectional routing along edges is applied when the Person's *personal mode* is being used.

The Network Routing module does not assume any specific distance units so can be applied to any road network with consistent units. The maximum mode speed, defined in the Travel Scenario's Mode Definition in Fig. 6, is compared to the current link's maximum speed with the lower value selected. It is assumed that the maximum speed is achieved instantaneously as physical acceleration and human behaviour exceeding speed limits are traffic simulator concepts. The parameters are also used to calculate a non-denominational cost for the stage based upon an upfront *fixed cost* and distance based *variable cost*.

Traffic Simulator interfaces were implemented for MATSim and SUMO simulators. The Activity & Travel Schedules, Road Network data and other information were extracted and converted to the simulators's XML input formats. Simulator outputs were then converted back into RDF for the knowledge base. The RDF to XML conversion was undertaken using SPARQL queries to extract triples into RDF/XML followed by conversion using XLST templates [19] into the traffic simulators' XML schema, see Fig. 10. This approach exposes the entire data extraction and transformation process to the user using standardised technologies. The user can modify the interface to adapt to local variations in the schema and knowledge base; focus upon a specific scenario; or to adapt for changes in simulator functionality.

In summary the prototype provides the user with control over the activity patterns, schema, module parameters, module selection and discrete choice calculation. These can be applied based on the class and properties present in the

data with minimal design assumptions, e.g. modes are defined in the data and not a fixed hierarchy. The implemented Scheduler produces full day schedules and is discrete from the Trip Planning and Network Routing stages. The Trip Planning module produces trips with the number of stages dependent on the modes and transport resources, rather than a fixed number, and considers spatial access constraints. The Transport Interfaces allow data to be extracted in the user's chosen schema and adapted for used with alternative simulators.

## 7. Implementation of Access Configuration Framework

This section discusses the access configuration framework to retrieve and process data with alternative datasets and modules. These datasets describe the parameters and scenario data under investigation, e.g. household, transport infrastructure, and land usage, which modules consume to produce elements of the travel modelling process, e.g. personal travel schedules, travel choices, and route cost estimates. It is intended to ease the burden on users in assembling, controlling and comparing their investigative scenarios across multiple implementations of travel demand generation models and traffic simulators.

The Semantic Web is based upon an open network of linked datasets accessible through the HTTP protocol. Therefore, the knowledge-base and modules of the framework do not have to be on a single computer but can be remotely located and split across multiple HTTP services. Experimental investigations into travel demand, traffic congestion, policy making etc. can be constructed by directing to HTTP services publishing datasets or hosting modules. The user would specify the URL address to access their chosen datasets and modules. This removes the necessity to retrieve, clean and format convert datasets or install and configure modules. The results of this process would be gathered

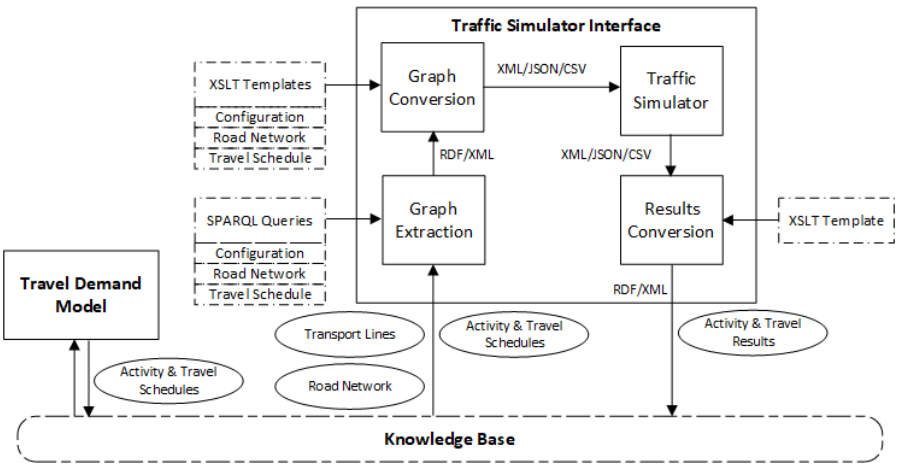


Fig 10. *Traffic Simulator Interface process utilising SPARQL and XSLT.*

by the user hosting their own online service, which provides the configuration information of the investigation and any of the user's own datasets or modules.

The core schema discussed in Section 5 provides a basis for modules and the knowledge-base to align, as a commitment to the vocabulary means it will be used coherently and consistently [15]. This removes data transformations, either prior or during execution, enabling datasets and models to interoperate without user intervention. However, the breadth of the travel domain means that establishing and maintaining a single all-encompassing schema for all modules to adhere to is problematic. Therefore, users need the flexibility to specify modifications and transformations as variations and developments emerge.

A specific module may vary from the core schema, or establish their own, and the user would mediate between that module and the other modules being targeted. This mediation is necessary in existing approaches through data format transformations between each component. This is inconvenient when setting up one configuration and becomes highly burdensome or inhibitive when applied to multiple configurations.

The proposed approach would allow the user to undertake this knowledge base construction, travel demand generation and traffic simulation process through the SPARQL query mechanism. The same skills and environment are used to select datasets, target modules and mediate between modules for all the various module configurations they choose to investigate. Compliant modules would follow the processing steps described using an API to achieve the extended functionality. The following sections describe the general features for configuring the framework and the specific data structures and functionality to achieve the outlined behaviour.

### *7.1. Features for Configuring the Framework*

Accessing data from remote uncontrolled sources introduces several issues that can waste resources or unnecessarily increase execution times. These issues can also be encountered in local controlled environments, but the transmission of data across networks should be minimised if possible. Errors originating in user input should be discovered as early as possible to prevent actions being processed that ultimately fail, after wasting resources undertaking the processing, or can return unexpected and incorrect results [39]. The following features have been identified to assist the process:

- Invariant Data Caching: Certain data items do not vary in the time-frame of a scenario, e.g. building coordinates, vehicle characteristics, road network topology, and therefore are consistent for the scenario's duration. SPARQL queries iterate through matching graph patterns which typically produce multiple parameter sets that consecutively vary by a single value and so a value may be required multiple times before no longer being required. Modules should retain data objects locally to avoid unnecessary retrieval and processing of repeated data, with possible time-based retention policies applied to manage resource usage.

- Data Schema Conformance: The effectiveness and accuracy of any model or system is heavily influenced by the quality of the input data. Publishers produce data that is acted upon by consuming modules, which in turn produce output data. Both parties must ensure that input and output data conforms to the schema that they publish or utilise. The Shapes Constraint Language (SHACL) [22] uses RDF triples to describe the shape and constraints of a dataset. These triples can be encoded as part of a published schema for automated validation and reporting to assist in designing transformation queries or identifying errors.
- SPARQL Query Validation: An open system can accept queries from users who lack understanding and resources to thoroughly test queries. The graph pattern matching of SPARQL queries can result in zero results due to misalignment between data schema and query structure, despite the queries being syntactically correct. This misalignment persists through all instance data so that initial failure of the first instance is repeated through all instances.
- Local and Remote Configuration: The framework must support local only, remote and hybrid configurations. In a local only configuration a knowledge-base is constructed at the start of the process, modules are available in the user application and network communication costs are not incurred. In a remote and hybrid configuration data and modules are accessed from services through the HTTP protocol. The configuration data is provided as a data service with URI references to the framework and service being passed to modules so that the details can be retrieved as required.
- Error and Warning Reporting: The orchestration of multiple software components can produce a range of undesirable issues including implementation errors, no responses from remote services, and validation and conformance outcomes. Users need visibility of these issues to investigate and make adjustments.

### *7.2. Framework Configuration*

The Framework Configuration describes the data structure and processes to control the configuration of the framework. This control relates to two primary tasks: 1) the direction to datasets and modules; and 2) mediating any schema misalignments between datasets and modules. The schema for this data structure is shown in Fig. 11. Each Framework Configuration instance is described by properties of Service Definition, Query Definition and Module Definition with each investigation having a single Framework Configuration.

The configuration information can be stored in the knowledge-base in a specific graph or together with other parameter data for the Travel Scenario and execution results. These three sets of information have use in post-execution analysis or the reconstruction of the investigation for reproduction studies.

The Framework Configuration provides a central reference for associating any global values required by module, e.g. traffic simulator configuration parameters. The optional Framework Service property states the URL HTTP service

from which the Framework Configuration can be retrieved. This permits the configuration to be passed between, and retrieved by, remote modules using only references to the Framework Configuration URI and its service.

### *7.3. Service Definition*

The Service Definition describes a service, graph name and the type of service/s. The Service Type states which particular parts of the core schema the data in the graph satisfies. Additional Service Types may be defined by modules, and published with accompanying schema, if their data requirements are broader or vary from the core schema. The user would fulfil these requirements in the knowledge-base and then signpost to them using the Service Definition of the Framework Configuration.

The service URI indicates the address of the SPARQL endpoint where the data is located. The graph URI indicates which graph within the endpoint holds the required data. This allows two Framework Configuration to point to the same service and retrieve different versions of data, e.g. Year 1 and Year 2. Alternatively, the two Framework Configurations could point to different services and retrieve their alternative versions of the same data, e.g. Year 1 from Service A and Service B.

The knowledge-base can be organised by the user to follow their own graph structure or separate the graphs onto multiple endpoint computers. The Service Type identifies which data schema can be satisfied by each service URI and graph URI pair. The module will seek the Service Type it requires without concern for the underlying organisation.

In the most simple configuration a single knowledge-base could have a single graph. However, a minimum separation of data into three graphs, i.e. configuration, scenario and results, is recommended to assist management and clarity.

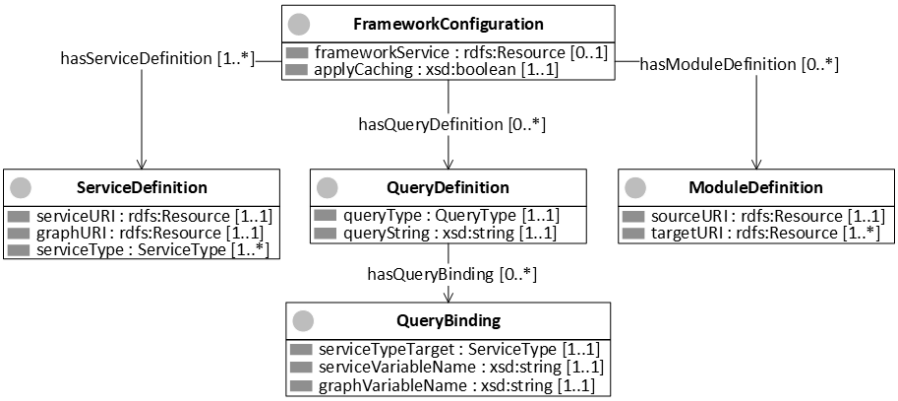


Fig 11. *Schema for Framework Configuration.*

Similarly, while a module may distinguish two areas of the data as being separate, e.g. person and vehicle data, a user can place them in the same graph without interference. Therefore, the Service Definition may refer to multiple service types, which are using the same service URI and graph URI.

The SPARQL queries of a module are defined as templates containing SERVICE and GRAPH clauses. These clauses use place-holder keywords instead of an explicit URI. During execution these place-holders are substituted for the service URI and graph URI of the Service Definition with the appropriate Service Type, or module defaults when none is defined. When these replacements match the base service or graph of the query then the clause is removed.

URIs can define a variety of schemes, including HTTP and File [20], but SPARQL only permits HTTP for service and graph URIs within queries due to its focus on online access. The use of a File scheme is dependent upon the implementation provided by the user's Semantic Web library and its interpretation of the content, so is not universally accessible. However, direct file access provides quicker access when operating in a local only context by not incurring the cost of HTTP encoding and communication. Therefore, benefits to set-up requirements and performance can be achieved by supporting File scheme URIs.

Access of local datasets can be achieved by defining File service URIs to point to the data as part of the configuration, but not forming part of the final processed SPARQL query. This requires that only a single File URI be used in a query and it must be the base service of the query. This will strip out service clauses for graph data or modules accessible locally, while still permitting a hybrid approach to other remote datasets and modules. Configurations requiring multiple local, to the user, services would require all but one to be hosted as a HTTP service to in effect become a remote service.

### *7.4. Query Definition & Query Binding*

The modules of the framework utilise SPARQL queries to retrieve data from services and invoke other sub-modules to generate data. These queries can be interpreted at runtime as text strings meaning that modifications can be applied or transmitted without requiring alteration or recompilation of modules. The Query Definition permits a user to define a replacement query for substitution of the module's default query. The modules state the default query string and its associate Query Type.

Users can re-write the queries to retrieve alternative pieces of data for a module. Similarly, sub-modules can be called to perform additional or alternative processing of the data within the module as *property functions* using standard SPARQL syntax. The usage of SPARQL syntax enables existing protections against exploitation or injection attacks of an open system to be maintained when processing queries. The only requirement is that the SELECT and CONSTRUCT variables are unaltered and bound so that the modules can retrieve the expected data from them. The queries used by modules can be considered to be data retrieval and sub-module execution.

In the former type, the module is adapted to the available data structure rather than transforming the data to the module's requirements; easing the burden of maintaining multiple knowledge-bases for different configurations. Another use case is the modification of calculations and equations, e.g. utility in discrete choice models. The query strings and types of these queries are recommended for publishing, but not mandatory as users can still utilise a module by aligning the knowledge-base with the modules' schema and there may a large quantity of trivial queries used by a module. The latter type of queries provide control over which sub-modules are selected and executed. These should be mandatory for publishing so that users have control to select alternative functionality and configure their investigations.

In both cases the user may wish to redirect part of the query to one or more services and/or graphs. This can be achieved by replacing, or inserting, explicit service and graph clauses in the query. However, these would require modifying each query string in each Framework Configuration that changes from the default. Instead the Query Binding structure is provided so that modules can check for variations and cross-reference to existing Service Definitions in the Framework Configuration. The place-holder variable names within a query are substituted for the defined service and graph URIs. Redirection across multiple queries within the configuration, as many queries may refer to the same service, can be achieved by changing the single Service Definition.

The SPARQL query language is expressive and flexible for users to structure queries and retrieve data from the graph data of the knowledge-base. However, the validation of the SPARQL queries have focussed upon grammatical checking of queries [2]. These grammatical checks identify when keywords are mistyped or functional requirements cannot be fulfilled, e.g. variables named as a result, but are not included in sorting or grouping statements.

They do not pro-actively ensure that queries are valid for logical or schema constraints and instead reactively error and fail during query processing. These logical and schema constraints have been categorised into syntactic and semantic validation [2]. The identified syntactic rules cover several cases including positioning errors, e.g. a literal being used as a subject or property, and filter conditions using literals of different data-types. The semantic rules are formed into an OWL ontology to use inferencing to check for logical consistency in the query. The application of these rules in addition to the grammatical checks will reduce the potential for invalid queries being processed.

It is proposed that the potential for meaningful results can be further enhanced by validating the explicit URI present in the query and through the checking of variable name usage in the query. The validation of explicit URI in a query can initially be performed by checking against the schema of the module. However, sub-modules may not exactly align with a module's own schema, which the replacement query is itself seeking to mediate. Therefore, explicit *class* or *property* URIs can be extracted from the query and then used in ASK and DESCRIBE queries to the sub-module's service. When these explicit URIs are valid to the sub-module then the query is valid for the module.

The checking of variable name usage is intended to highlight variables that

occur once in a query, and so serve no purpose, or have strong similarity to other variables, and so may be misaligned. The mismatching of variable names can have implications for query optimisation, results range and the binding of explicit values. The identification of single use variables is performed on the clauses in the results, body, and aggregation sections of the query.

The similarity of variable names is found using the Levenshtein distance [9]. This measures the edit distance, i.e. number of insertions, deletions or substitutions, required for two strings to match with a zero being an exact match. Variable names longer than three characters are checked for one or two edit distance. In one edit distance, equal length strings are ignored as enumerations if the initial or final characters are not the same, unless the variation is in case, e.g. "vara" and "varA". There is currently no checking for consistent enumeration or handling of more than a single enumerating character, e.g. "var1" and "var10". In the two edit distance, strings of equal length are checked for the transposing of adjacent characters. The cases that result in two insertions, two deletions, non-adjacent substitutions and non-transposing substitutions, i.e. more than two character values, are ignored as being too dissimilar.

These represent warnings, as similar variable names may be the user's intention, and so should not prevent execution, but can indicate the cause, or potential existence, of error in a query. However, the single use of variable names can represent errors as a value must be achievable for the query to execute. Once a query has been validated the outcome can be short term cached as its structure should not change for any following queries, while the data instances are iterated through, and invalid queries can be quickly rejected. The further development of these validation steps for the framework is an area of future work.

### *7.5. Module Definition*

The final component of the configuration is the Module Definition. This identifies a module and the additional sub-modules that a user has defined in the replacement query strings. The query validation process checks the explicit URIs present in a query against those of the schema. The URI of a replacement *property function* for a sub-module does not feature in the schema, and will not be present in the knowledge-base of a service, and so the query would be regarded as invalid.

A module can know its own URI and those of the default sub-modules so that they can be excluded during validation. However, a module cannot distinguish between user defined sub-module properties and incorrectly entered property URIs. Therefore, the Module Definition permits the user to state the sub-modules URIs that are permitted in replacement queries.

### *7.6. Validation Result*

The framework is developed based on the Semantic Web design principles of an open network. Information is transferred between modules and knowledge-bases

of the framework with customisation by the user. The previous sections have outlined the mechanisms available and proposed for ensuring that the data being produced and consumed is valid and to ensure that queries have the potential to produce meaningful results.

Once these validation steps have been performed it is necessary to report back to the user the outcome so that remedial action can be taken. The modules can also check for errors reported by their sub-modules and abort their execution. The inclusion of this validation reporting has more general usage as a means for modules to also report other information that may assist the user, e.g. policy, execution errors or additional meta-data, without it being included with the results of executing the framework.

The data structure for capturing the data and query validation reports is shown in Fig. 12. The structure has properties for a text summary of the validation results, e.g. the variable names or URI found to be invalid in a query, and whether the result constitutes an advisory warning or a critical error. Additional properties are defined for the identified subclasses of query and data validation errors. Data Validation Results provide specific references to the data source through service and graph URIs. Query Validation Results identify the URI of the invalid query used in the Framework Configuration. Each Query Validation Result also indicates the result type so that further background information into the cause can be found.

In summary, the Framework Configuration provides a mechanism for directing the execution to the services and graphs containing the data necessary to satisfy the travel demand generation modules. Published datasets or SPARQL endpoints adhering to the core schema for the travel demand model can reduce issues of misaligned data. However, the framework also supports adaptation and transformation of data for new purposes or incorporating alternative datasets and modules by the user providing modified module queries.

These user defined queries introduce greater potential for data and schema misalignment or invalid queries due to misunderstanding or inadequate testing. It is proposed that performing schema and query validation, by applying ex-

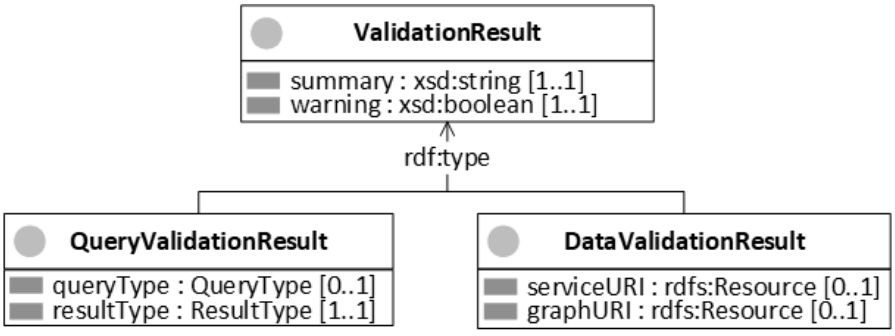


Fig 12. *Schema for Validation Result.*

isting techniques and proposed solutions to address specific issues, can prevent redundant usage of resources. A feedback mechanism is included in the Framework Configuration to assist the user in understanding and rectifying the source of errors and potential issues identified by these techniques.

The proposed Framework Configuration can support local, remote and mixed configurations. The accessing of these configurations is achieved using the HTTP support provided by SPARQL federated queries, but also allows direct access to file system knowledge bases to provide efficiency and simplify set-up.

The use of SPARQL is applied throughout the framework to construct, transform, redirect and execute scenarios. This provides a single language that is platform independent so that users do not need to learn multiple programming languages and can apply their developed skills repeatedly. The framework does not introduce any variation to the SPARQL standard and is instead an application of its language and principles. Therefore, the barrier to using the framework is lowered and potentially requires a narrower skill set than a conventional solution for the travel demand generation process.

The requirements of the Framework Configuration have been established with no domain specific requirements identified. It is put forward that the framework provides a general solution for accessing and configuring modular solutions for other problems. The further development of the framework would seek to develop, or expand upon existing, mechanisms for the discovery and negotiation of remote services of modules and datasets to assist the user in the configuration process.

## 8. Evaluation of Travel Demand Generation

This section considers the implemented prototype that has been developed using the schema and the access configuration framework discussed in the previous sections. The described experimental scenario highlights areas controlled through the knowledge base rather than implementation design decisions. The travel demand generated by the prototype using the experimental scenario is considered along with the variations in travel dynamics when applied to two third-party traffic simulators. Finally, there is evaluation of several local and remote configurations of the knowledge base and modules to demonstrate the impact these configurations on the performance of the demand generation process.

The experimental scenario knowledge-base of the prototype was constructed using a randomly generated road network of 14km by 8km using SUMO simulator's NETGENERATE application [23] and converted into RDF, see Fig. 8, using an XSLT template. RDFS inferencing was applied to the knowledge base using the RDF schema described previously, and published public schemas, to provide automatic inferencing and data validation, e.g. datatype checking, cardinalities and inferred sub-class membership and sub-property relationships. Applying OWL2 inferencing and additional property relationships would enable more diverse inferencing, e.g. relationships between locations and persons based on common activity types and person mode usage based upon vehicle usage.

A dataset was produced based on a road network containing one thousand *residence* locations, five *education* locations, one hundred *employment* locations, five *freight depot* locations and thirty locations each for *retail*, *leisure*, *personal business*, and *freight delivery*. Each location was assigned geospatial coordinates randomly selected from a set of evenly spaced points running alongside road links. Locations were also selected at the road links closest to the cardinal points and a central train station as starting points for external non-resident travellers using *transport link* activities.

Each *residence* location contains a single *household* Travel Group consisting of four persons to simulate four thousand *resident* individuals. *Households* were assigned one of the ten Activity Pattern Sets with each person in the group being allocated a single Activity Pattern. Ten Activity Pattern Sets were manually created with each consisting of four Activity Patterns. The Activity Patterns started and ended with *home* activities and consist of one or more activity blocks ranging from half hour to nine and a half hours. The start and end activities could be any Location or Activity Type but all were assigned to *residence* Locations.

The activity pattern's start and end times were chosen from the four quarters of the hour, with later random variation of plus or minus fifteen minutes. Lunch time and evening activities were included around core day time *education* and *employment* activities, but interrupted by lunch time, with a *home* activity prior to other evening activities.

Each *resident* Person was randomly allocated activities at locations according to activity types with one *employment* and *education* location and ten each for *retail*, *leisure* and *personal business* locations. Locations were assigned multiple activity types. House residences provide *home* and *leisure* activity types due to *leisure* activities also including socialising with friends and family. Other locations provide *employment* and other related Activity Types. This demonstrates the potential for multiple activities and alternative activity types to take place at a single geographic location.

*Non-resident* persons were similarly assigned locations for activities but were not assigned *residences*. Instead these were allocated to *transport link* activities at edge of network Gateway Links and Train Station locations. Two hundred Travel Groups were split evenly between the five *transport link* locations with four Persons per group. Activity Pattern Sets following those of the *resident* persons were produced but with *residence* activities replaced by *transport link*.

*Freight driver* persons were allocated an activity at a *freight depot* location to start and end the schedule. Each *freight depot* was allocated a Travel Group consisting of ten *freight drivers*. All freight drivers were assigned the same Activity Pattern of deliveries every thirty minutes throughout the day but varying travel range.

No *freight delivery* locations were asserted for the *freight drivers*. Instead, potential locations were searched dynamically according to proximity of the current location and travel range of the Activity Pattern item demonstrating contextual selection. Destination selection was equalised through the *freight driver* utility coefficients and freight vehicle mode parameters. All the described Locations

and Activity Types could be intermixed so that *residents* and freight drivers may travel out to *gateway links* and *freight delivery* activities can take place at *residences*.

The described person types, activities and locations were applied in the user schema, rather than prototype design, and can therefore be modified. These have been selected to illustrate typical domain concepts a user may wish to model. The implemented prototype is able to operate upon these in a generic manner while the user can still apply selection to use alternative modules, e.g. trip planning for freight. This is in contrast to some travel demand models, e.g. CEMDAP [38], which divides the population into workers and non-workers with fixed activity travel patterns.

Each *resident* was potentially allocated a single private Vehicle from a distribution covering *car*, *motorbike* and *bicycle* Modes with children restricted to *bicycles*. All *non-residents* arriving via *gateway link* were assigned either *car* or *motorbike* Vehicles. Those *non-residents* arriving at the train station were not assigned vehicles. All *freight drivers* were assigned *Heavy Goods Vehicles*. All *residents* and *non-residents* were assigned personal *walking* Modes, while *freight drivers* were not assigned a *personal* Mode to enforce continuous usage of their vehicles.

Personal utility coefficient weightings for the Random Utility Model (RUM) were specified according to the three person types as a model simplification rather than technical requirement. Each Mode was assigned max speed, fixed cost and variable cost definition for the Travel Scenario. The RUM was tuned, except *freight drivers*, to provide a walking preference for stages shorter than 1.4km and using vehicles for longer trips as can be noticed in Fig. 9. This threshold is intended to provide a mix of mode usage and was based upon indicative distance of traveller walking [33].

The results of the traffic simulators from scheduling for all 5,000 individuals can be seen in Fig. 13, 14 and 15. The simulation has been executed over the period of one day. In Fig. 13, the trips in progress are shown between ideal schedule and simulator. The general M-shaped curve of weekday commuting can be seen along with lunch time and evening travel with the general pattern directly influenced by the activity pattern templates. Both simulators can be seen to have higher numbers of travel stages in-progress.

Fig. 14 shows the activity intervals in progress and is the inverse of the travel stages, as individuals who are not travelling are performing activities, as can be seen by correlation with the peaks seen in Fig. 13. 25,806 out of 26,280 (98.2%) activity intervals were scheduled requiring 26,765 travel stages. The figure shows the switch from *home*, *delivery* and *transport link* activities at night to day time activities. The structure and hierarchy of these activities has been chosen for illustration and the user is able to expand and modify as required.

The final graph illustrated in Fig. 15 shows the mean delay between the ideal schedule and the simulated travel of the traffic simulators. Interactions between individuals and also the traffic infrastructure means that longer travel durations would be expected. We can identify noticeable differences between simulations derived from the same input data. In SUMO, extreme values can be seen at both

ends of the day when travel volumes are lowest, indicating severe disruption for a few travellers, but a general alignment with traveller volume shown in Fig. 13.

In MATSim, the mean delay peak occurs during the lower volume midday period, excluding an end scenario spike, suggesting more generalised delays are being experienced. The maximum delay was consistently found in MATSim, peaking at 83 minutes. The generated travel demand is platform agnostic and can be transformed and configured according to the target application, e.g. traffic simulators, or analysis, e.g. aggregation by type or geographic area.

The prototype was applied in several configurations of the framework to explore the impact on execution durations of network communications. These configuration scenarios have been executed as in-memory storage and using local services on the same computer, i.e. *http://localhost* addresses through the loop-back of the network adapter. The configurations were as follows:

- Local: local configuration with knowledge-base, modules and results all local to the user application.
- Data: remote data configuration with the entire knowledge-base on a remote server and modules and results are local to the user application.
- Joined: remote data and module configuration with both on the same remote server and results are returned to the user application.
- Split: remote data and module configuration with each on a separate remote server and results returned to the user application.

The results of these different scenarios can be seen in Figure 16 and Table 1. The Local configuration is quickest to complete as would be expected. This

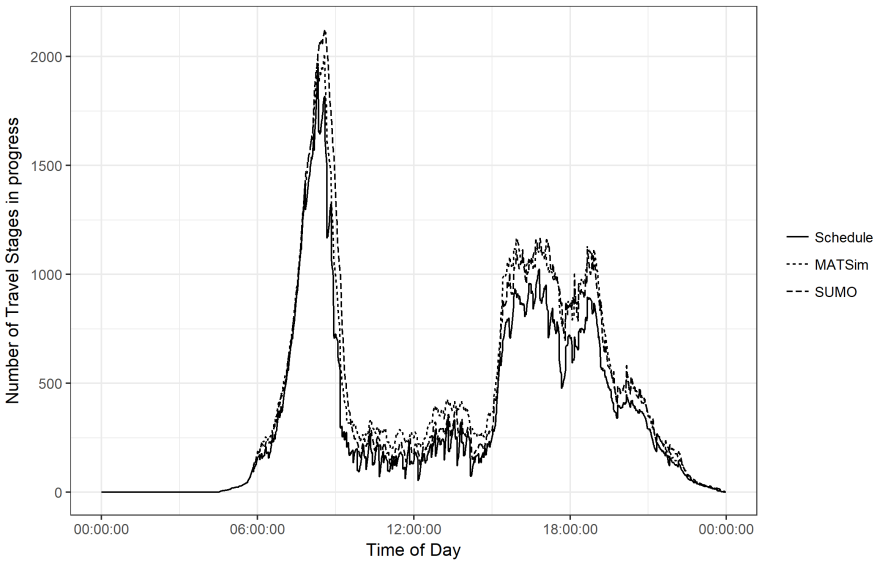


Fig 13. *Number of travel stages by simulator per one-minute interval.*

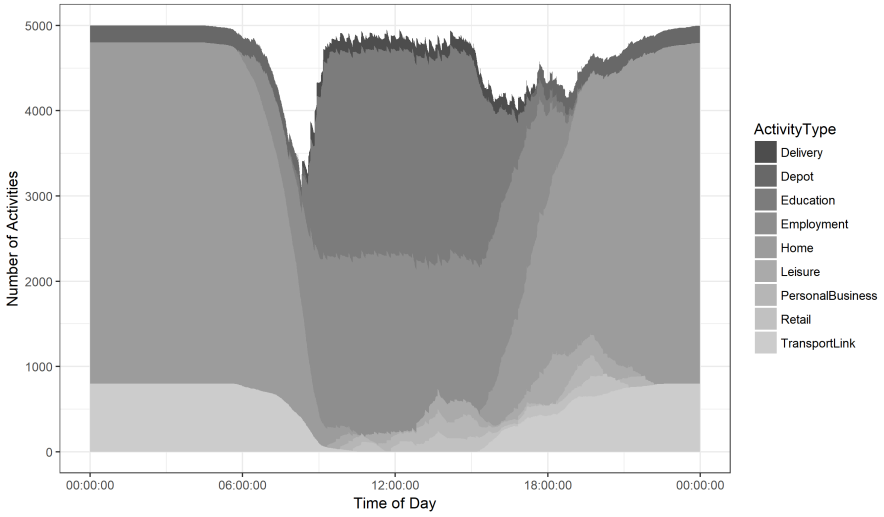


FIG 14. *Number of activities by activity type per one-minute interval.*

configuration does not incur the HTTP overhead as it can directly access the knowledge-base and results for all operations. The Joined configuration is next quickest as network communication only incurs when commencing the genera-

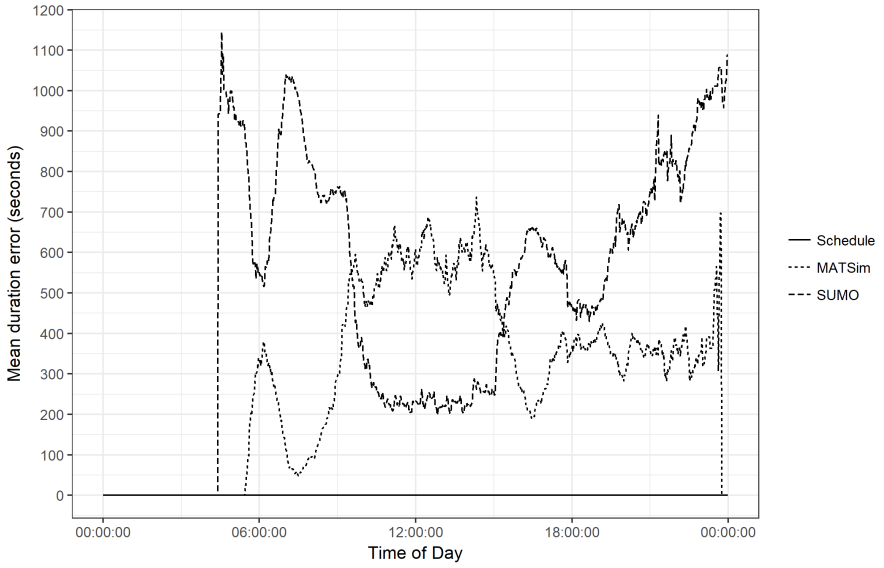


FIG 15. *Mean error of travel stages by simulator per one-minute interval.*

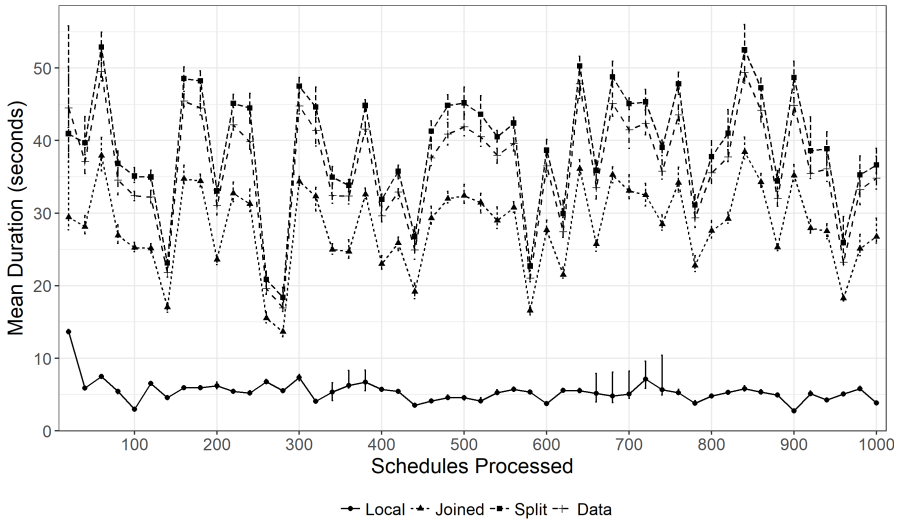


FIG 16. *Mean duration for completion of alternative in-memory configurations (10 iterations).*

tion of each Travel Group and when returning the generated results back to the user application. The Data configuration is then closely followed by the Split configuration. These incur the most communication as the modules have to retrieve data about the Travel Group being processed. The advantage of the Data configuration is that it has local access for the storage of results once generated.

The Split configuration represents the idealised scenario where the user only needs to set-up the configuration before all the required data of the knowledge-base is retrieved and processed by remote module. It is found to take 7.2 times longer to execute on the same scenario data as the local configuration even in this best-case scenario of local HTTP connection. However, there would be the potential for these remote services to employ greater computational resources or distributed computing as mentioned earlier to offset the communication costs.

It can also be seen that each of three alternative configurations follows the same general pattern with peaking and troughs in the same schedule batches. However, these do not follow the fluctuations shown in the Local configuration. Suggesting that the difference between the remote and local configurations can

| Configuration | Mean | Std. Dev. | Min | Max |
|---|---|---|---|---|
| Data | 1,820.389 | 12.741 | 1,802.013 | 1,838.979 |
| Joined | 1,407.687 | 10.102 | 1,392.702 | 1,421.278 |
| Local | 270.469 | 4.587 | 264.932 | 278.806 |
| Split | 1,952.167 | 16.220 | 1,931.723 | 1,981.499 |

TABLE 1

*Table of completion duration (seconds) of alternative in-memory configurations (10 test iterations per configuration).*

be solely attributable to the cost of HTTP communication.

Another area of investigation is the use of caching to store invariant data rather than continuously retrieve it from the knowledge-base. All the previous results reported using caching. Testing of a single iteration of a Resident only scenario for 1,000 individuals with in-memory storage saw performance drop from 4 minutes 30 seconds with caching to 5 hours 25 minutes 27 seconds without caching, an increase of 72.3 times. Therefore, the application of caching is not really an optional implementation decision. There cannot be a reliance upon retrieving data from the knowledge-base on-demand and instead opportunities to retain data for future use must be identified. These performance figures can be expected to decline for persistent storage and remote configurations.

In summary, the prototype has produced travel demand that varies by mode, destination and time according to local and global concepts and parameters. Expansion of concepts, e.g. activities, modes and travel users, and modification of parameters are controlled through the knowledge base rather than being explicitly designed into the modules.

The impact of utilising remote configurations has been shown to be noticeable, but not prohibitive. The travel demand generation process does not require real-time responses and so once in progress can be undertaken over a prolonged period, while preparing data and creating integrating interfaces requires a direct investment of resources and time by the user. The processes of caching and multi-threading have also been found to improve execution and an online approach creates the potential for accessing greater computational resources. In principle, continued progress in network communication technology and speeds should also reduce the difference between local and remote configurations, but would be offset by computational performance improvements and a difference will always exist in some form.

The retrieval of all necessary data to construct a local knowledge-base as the initial process would still be considered the most efficient approach. This also presents the opportunity for customisation and adaptation by users. These are still activities that the Semantic Web can support in achieving by simplifying and standardising the process of constructing and simulation transport travel demand, whether for local, remote or hybrid execution. The proposed Framework Configuration represents a further development to minimise the burden for users in preparing, accessing, and integrating datasets and module implementations and improve the quality and range of traffic and transport investigations.

## 9. Challenges in Utilising Semantic Web Technologies for Implementing Travel Demand Generation

The previous section discussed the prototype design for execution on a sample scenario. This section discusses the challenges identified during developing and implementing the prototype design of the Semantic-based travel demand generation framework.

### 9.1. Traffic Simulator Interface Design

The Traffic Simulator Interface stage involves the conversion of the Activity & Travel Schedules to the simulator input format, XML for both SUMO and MATSim simulators, and return of the results to the knowledge base. A conventional design approach is programming dedicated parsing interfaces for each simulator. However, these require maintenance for changes in both the simulator and knowledge base schema. This approach is also prescriptive in the supported platform and not easily adapted to user alterations of the knowledge base schema, e.g. property labelling or additional data.

This does not fit with the objective of providing flexibility and choice in assembling traffic and travel demand investigations. The XSLT technology [19] enables XML schema conversion using using template files read into platform independent engines. These templates can be modified in any text editor according to the data available and required. Several RDF serialisation formats are available including RDF/XML.

There are several versions of RDF/XML, but it is necessary to use the *plain* serialisation with XSLT due to template complexity and efficiency. The *plain* serialisation forces the use of a consistent *rdf:Description* label for all resources, instead of selecting an arbitrary class name, in data with individuals having multiple classes, and flattens the structure to remove nested child elements. Other RDF serialisations can introduce a more complex class labelling approach, due to RDF permitting multiple classes for each resource, which increases the complexity and maintenance of templates. A drawback of this RDF format is a relatively slow serialisation process.

### 9.2. SPARQL Language Expressivity

The SPARQL query language is a powerful tool for searching, extracting and transforming the data contained in the knowledge base. The language supports numerous built-in functions and the potential for performing complex operations solely within SPARQL queries. However, its execution approach does not support iteration over a set of data with actions dependent upon earlier iterations, e.g. scheduling of later activities and travel cannot be based on earlier decisions in a single query. Instead multiple queries would need to be executed, requiring management of this process, or implementing a *property* function module, as in the prototype.

Another area of inconvenience is the inability to express SPARQL queries as re-usable functions. The retrieval and transformation of data may require several steps which are used in multiple queries. SPARQL supports sub-queries, written out in full so increasing query complexity, or extending the query engine with *filter* and *property* functions requiring additional configuration. The SPIN and SHACL technologies use SPARQL syntax to describe queries encoded into the schema and executed by their engines. However, these are extensions and not standard SPARQL. Expressing re-usable SPARQL queries as functions within a core schema would allow easy distribution and reduce repetition.

Finally, sophisticated queries can be written using SPARQL to perform calculations and generate new values and triples. However, the complexity of expressing these operations and the query optimisation process can make performing these queries computationally expensive. Operations may be performed on data that is later discarded or the same values calculated repeatedly for alternative parts of the dataset.

In addition, SPARQL queries are checked for syntactic correctness but not consistency between variable names or the existence of predicates in the dataset. This can result in minor typographic errors or schema changes causing unexpected and unnoticed outputs. Therefore, while sophisticated functionality, e.g. calculating Discrete Choice probabilities and constructing Travel Stages from Stage Estimates, can be achieved within SPARQL queries the use of *property* functions to access an imperative programming language can improve maintenance and performance.

### *9.3. SPARQL Extension Property Function Arguments*

The processing of arguments passed into *filter* and *property* functions in SPARQL queries follows the Functional Programming paradigm. A function is called for each combination of parameter values without visibility of previous or following calls. Therefore, processing of encoded, invariant or externally retrieved data would happen repeatedly leading to an avoidable increase in execution time. This can be overcome by using a caching strategy within the property function to retain re-usable data. A caching strategy was applied in the prototype for invariant data, e.g. location coordinates, which reduced execution duration by a magnitude of approximately ten.

The Functional Programming paradigm also affects related items that might expected to be treated as a collected list and are instead separated into multiple discrete function calls. This can be mitigated by using the SPARQL *GROUP_CONCAT* aggregation term to produce a delimited string. Alternatively, a grouping structure, termed an N-ary relationship, can be defined. The root subject is passed into the function, which then retrieves the collection of related items.

## 10. Conclusion

The research described in this paper describes a novel knowledge-based approach that utilises Semantic Web technologies for the modelling and simulation of traffic and travel demand. It is proposed that this approach will enable the construction of a knowledge base, from local and remote data sources, upon which a set of discrete and interchangeable modules to generate components of travel demand can be developed and distributed.

Our knowledge-based approach developed a core schema that models the interactions between the major stages of population synthesis, activity-based

travel demand and traffic simulation. This schema has been applied to a prototype in order to explore the practical implications of this approach. An expanded user defined schema was combined with the prototype to model three types of travellers of resident, non-resident, and freight. These travellers use a range of modes and engage in various activities to generate travel demand that was simulated with two established third-party traffic simulators.

The implemented prototype and experimental scenario has demonstrated that the user can include their own schema of concepts; select alternative modules based on those concepts; access and modify both local and remote datasets and apply travel demand to multiple traffic simulators. The implemented modules are generic designs driven by the schema and data of the knowledge base. However, the overall modular architecture is designed to enable substitution of modules by users.

These features can help address the current shortcomings of singular behavioural models and the burden of comparing travel demand models. In the prototype, the user has control over the activity patterns, schema, module parameters, module selection and discrete choice calculation. These can be varied by class and properties present in the data. The applied query mechanism also allows results to be extracted in the user's chosen formulation.

Investigation has been undertaken into the performance of alternative configurations of datasets and modules to demonstrate the practical achievement and consider their effectiveness. These configurations were controlled using the proposed access configuration framework approach in a local network environment. This has shown that the modules and datasets can be separated into discrete components.

It was found that a local only configuration was 7.2 times quicker than alternative HTTP configurations over a loop-back network. Therefore, the availability of a local configuration for simpler investigations is highly useful. The utilisation of a caching strategy by modules was found to reduce execution durations by 72.3 times suggesting that caching strategies would always be recommended for modules. Further investigation is needed into the implications of the modules and datasets being hosted in an online environment, where execution durations are likely to be extended, but greater resources could be made available for processing.

The access configuration framework also supports the provision of user defined queries to reconcile between modules and datasets adhering to modified or alternative schemas. The need has been identified for the extended validation of these queries to prevent redundant resource usage and mechanisms are provided to inform users. Initial solutions to achieve this validation has been discussed and identified as an area future work. A mechanism is provided for feedback on the outcome of these validation steps to assist the user in understanding and rectifying errors.

The framework enables the user to select and apply modules of their own interest; incorporate alternative implementations for their own study; apply parameter and technique variations to explore their impact; and operate this functionality using existing SPARQL query language syntax. The access framework

configuration data can be combined with scenario parameters utilised in the experimental investigation to form a complete package of information to assist in reproduction of the investigation or its replication by other investigators.

Further work is needed to address and expand upon several identified features. There is a need to formalise module descriptions of data requirements and functionality to facilitate interchange. Tools are also needed to assist in knowledge base construction. The query validation process can be further developed to identify incorrect *class* and *property* URI and variable names.

The modules in the implemented prototype can be further developed to include: alternative forms of decision-making; intra-household cooperation; location popularity weightings and additional route planning, e.g. public transport. An area of investigation is the application of other Semantic Web technologies, e.g. OWL reasoning; SPIN rules; and SHACL validation, for knowledge inferencing and implementing travel demand functionality.

Finally, further investigation is required in applying the proposed remote access of data sources and modules as an online framework. The investigation has focussed upon loop-back network communication and there is no consideration of network latency, configuration, throughput or capacity. These results are intended to be indicative of the impact of applying these alternative configurations and represent best-case scenarios. Establishing a complete benchmarking process to evaluate a real-world networking environment would be an area of future work.

In conclusion, the proposed approach enables the integration of the multiple stages of travel demand modelling in a single framework. There is opportunity to utilise online data sources in a manner that can be easily extended and modified. The problem of travel demand modelling can be separated into inter-changeable modules based on common data requirements, which can be extended to apply new or alternative solutions. The SPARQL query language provides immediate benefits as a mechanism for retrieving, inferring, transforming and storing data in both local and remote knowledge bases. This allows user control of concepts for their own purposes or to incorporate concepts from outside of the transport engineering domain.

## References

[1] Albiston, G. L., Osman, T. and Chen, H. (2018). GeoSPARQL-Jena: Implementation and Benchmarking of a GeoSPARQL Graphstore. *Forthcoming*.

[2] Almendros-Jiménez, J. M., Becerra-Terón, A. and Cuzzocrea, A. (2017). Detecting and Diagnosing Syntactic and Semantic Errors in SPARQL Queries. In *7th ACM International Workshop on Linked Web Data Management*. CEUR-WS.

[3] Arentze, T. and Timmermans, H. (2000). *Albatross: a learning based transportation oriented simulation system*. Citeseer.

[4] Axhausen, K. W. (2007). *Definition of movement and activity for transport modelling*, 2nd edition ed. Emerald Group Publishing Limited.

[5] Berners-Lee, T., Hendler, J. and Lassila, O. (2001). The Semantic Web. *Scientific American* **284** 28-37.

[6] Black, C., Barrack, C., Ball, C., Keen, A., Clark, M., Gilbert, D. and McInroy, H. (2013). Where is 2+ car sharing headed? Technical Report, Carplus Ride Share Working Group by Carplus.

[7] Bukhari, A. C. and Baker, C. (2013). The Canadian health census as Linked Open Data: towards policy making in public health. In *9th International Conference on Data Integration in the Life Sciences; July 11-12, 2013; Montreal, PQ*.

[8] Castiglione, J., Bradley, M. and Gliebe, J. (2014). Activity-based travel demand models: a primer Technical Report, Transportation Research Board of National Academies.

[9] Cohen, W., Ravikumar, P. and Fienberg, S. (2003). A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation* **3** 73-78.

[10] Corsar, D., Markovic, M., Edwards, P. and Nelson, J. D. (2015). The Transport Disruption Ontology. In *International Semantic Web Conference* 329-336. Springer.

[11] David, N. (2013). *Validating simulations. Simulating Social Complexity* 135-171. Springer.

[12] Directive, I. (2007). Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE). *Published in the official Journal on the 25th April*.

[13] Goulias, K. G., Bhat, C. R., Pendyala, R. M., Chen, Y., Paleti, R., Konduri, K. C., Lei, T., Tang, D., Youn, S. Y. and Huang, G. (2012). Simulator of activities, greenhouse emissions, networks, and travel (SimAGENT) in Southern California. In *91st annual meeting of the Transportation Research Board, Washington, DC*.

[14] W3C OWL Working Group (2012). OWL 2 Web Ontology Language Document Overview (Second Edition) Technical Report, World Wide Web Consortium (W3C).

[15] Gruber, T. R. (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human-Computer Studies* **43** 907-928.

[16] Hensher, D. A. and Button, K. J. (2008). *Handbook of Transport Modelling*. Elsevier 01371524.

[17] Horeni, O. (2012). Measuring Mental Representations Underlying Activity-Travel Choices, PhD thesis, Eindhoven University of Technology.

[18] Horni, A., Nagel, K. and Axhausen, K. W. (2016). *The multi-agent transport simulation MATSim*. Ubiquity Press London.

[19] Kay, M. (2017). XSL Transformations (XSLT) Version 3.0 Technical Report No. Jul, 14, World Wide Web Consortium (W3C).

[20] Kerwin, M. (2017). The "file" URI Scheme Technical Report, Internet Engineering Task Force (IETF).

[21] Kirill, M. and Axhausen, K. W. (2011). Population Synthesis for Mi-

crosimulation: State of the Art. In *Transportation Research Board 90th Annual Meeting.*

[22] Knublauch, H. and Kontokostas, D. (2017). Shapes Constraint Language (SHACL).

[23] Krajzewicz, D., Erdmann, J., Behrisch, M. and Bieker, L. (2012). Recent development and applications of SUMO simulation of urban mobility. *International Journal On Advances in Systems and Measurements* **5** 128-138.

[24] Lécué, F., Tallevi-Diotallevi, S., Hayes, J., Tucker, R., Bicer, V., Sbodio, M. L. and Tommasi, P. (2014). Star-City: Semantic traffic analytics and reasoning for city. In *Proceedings of the 19th international conference on Intelligent User Interfaces* 179–188. ACM.

[25] Michel, F., Montagnat, J. and Faron-Zucker, C. (2013). A survey of RDB to RDF translation approaches and tools.

[26] Niaraki, A. S. and Kim, K. (2009). Ontology based personalized route planning system using a multi-criteria decision making approach. *Expert Systems with Applications* **36** 2250-2259.

[27] Perry, M. and Herring, J. (2012). GeoSPARQL - A Geographic Query Language for RDF Data Technical Report, Open Geospatial Consortium (OGC).

[28] Rasouli, S. and Timmermans, H. (2014). Activity-based models of travel demand: promises, progress and prospects. *International Journal of Urban Sciences* **18** 31-60.

[29] Schneider, M., Rudolph, S. and Sutcliffe, G. (2012). Modeling in OWL 2 without Restrictions. *arXiv preprint arXiv:1212.2902.*

[30] Smith, L., Beckman, R. and Baggerly, K. (1995). TRANSIMS: Transportation analysis and simulation system Technical Report, Los Alamos National Lab., NM (United States).

[31] Soares, G., Kokkinogenis, Z., Macedo, J. L. and Rossetti, R. J. (2014). *Agent-Based Traffic Simulation Using SUMO and JADE: An Integrated Platform for Artificial Transportation Systems. Simulation of Urban Mobility* 44-61. Springer.

[32] Stadler, C., Lehmann, J., Hffner, K. and Auer, S. (2012). LinkedGeodata: A core for a web of spatial open data. *Semantic Web* **3** 333-354.

[33] Stillwell, D., Pini, C., Cummings, J. and Fazil, A. (2017). England National Travel Survey: 2016 Technical Report, Department for Transport.

[34] Tamminga, G., Knoppers, P. and Lint, J. V. (2014). Open traffic: A toolbox for traffic research. *Procedia Computer Science* **32** 788-795.

[35] Tamminga, G., van den Brink, L., Lint, H. V., Stoter, J. and Hoogendoorn, S. (2013). Toward GIS-Compliant Data Structures for Traffic and Transportation Models. In *Transportation Research Board 92nd Annual Meeting.*

[36] van den Brink, L., Janssen, P., Quak, W. and Stoter, J. E. (2014). Linking spatial data: automated conversion of geo-information models and GML data to RDF. *International Journal of Spatial Data Infrastructures Research 9,(2014).*

[37] Zheng, H., Son, Y.-J., Chiu, Y.-C., Head, L., Feng, Y., Xi, H., Kim, S. and Hickman, M. (2013). A Primer for Agent-Based Simulation and Modeling in Transportation Applications Technical Report, United States. Federal Highway Administration.

[38] Ziemke, D., Nagel, K. and Bhat, C. (2015). Integrating CEMDAP and MATSim to increase the transferability of transport demand models. *Transportation Research Record: Journal of the Transportation Research Board* **2493** 117-125.

[39] SOA Source Book Technical Report, The Open Group.