Knowledge representation, storage and retrieval for BIM supported building evacuation design

Calin Boje

March 2018

A thesis submitted to Cardiff University for the degree of Doctor of Philosophy

Summary

Safe evacuation design is a complex process, which relies on crowd simulation models when assessing the performance of large or complicated building layouts. Current simulation methods and tools lack automation and are limited to geometry when relying on BIM interoperability. The use of semantic web linked data is seen as a step towards integrating and leveraging current digital resources to facilitate intelligent and automatic design capable of knowledge processing. An intelligent software system has been developed which is capable of integrating multiple information sources and which can facilitate fast automatic construction and analysis of crowd simulation models for design decision support. The system includes several developed OWL ontologies and SWRL rules which represent design knowledge from the fire evacuation field, thus being able to process and store data about a multi-disciplinary design field. The work conducted towards the development of the system involved investigation into crowd analysis tools, evacuation and digital building models. The ontology and knowledge operators are presented and discussed, providing insight into future exploration of such methods with the aim of outlining their benefits and limitations. The system and knowledge engineered have been tested using a case study, proving they are capable of fast processing and correct interpretation of model data.

Acknowledgements

I would like to thank my supervisors Dr. Haijiang Li and Prof. Yacine Rezgui for their guidance and support.

Special thanks to the people from industry which have been involved with my research project by providing invaluable insight into crowd simulation and their help towards validating the system: Peter Debney and Rhys Lewis from Oasys; Frank Schuyer, Victoria Manolova and Geert van Gorp from Xinaps.

Special thanks to the developers of and contributors to IfcOwl, on which I have relied to conduct this work. I am grateful to Michael Dibley and Alex Bradley for the brainstorming sessions and helping me learn about ontologies. Additionally, I would like to thank Adbul Rahman Darras for his help on the case study.

A big thank you to all researchers, procrastinators, coffee addicts and philosophers in the making that passed through the BRE Institute over the last years. I would not have made it without their encouragement over the daily lunch and coffee breaks.

To my family for their support and encouragement even from far away, they will always be my inspiration.

To my girlfriend Ioana, for her constant support, love and patience to see me through this PhD.

DECLARATION

This work has not been submitted in substance for any other degree or award at this or any other university or place of learning, nor is being submitted concurrently in candidature for any degree or other award.

Signed ……………………………………………… (candidate) Date ………….………

STATEMENT 1

This thesis is being submitted in partial fulfillment of the requirements for the degree of …………..(insert MCh, MD, MPhil, PhD etc, as appropriate)

Signed ……………………………………………… (candidate) Date ………….………

STATEMENT 2

This thesis is the result of my own independent work/investigation, except where otherwise stated, and the thesis has not been edited by a third party beyond what is permitted by Cardiff University's Policy on the Use of Third Party Editors by Research Degree Students. Other sources are acknowledged by explicit references. The views expressed are my own.

Signed ……………………………………………… (candidate) Date ………….………

STATEMENT 3

I hereby give consent for my thesis, if accepted, to be available online in the University's Open Access repository and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ……………………………………………… (candidate) Date ………….………

STATEMENT 4: PREVIOUSLY APPROVED BAR ON ACCESS

I hereby give consent for my thesis, if accepted, to be available online in the University's Open Access repository and for inter-library loans **after expiry of a bar on access previously approved by the Academic Standards & Quality Committee.**

Signed ……………………………………………… (candidate) Date ………….………

Table of Contents

List of Figures ... xiii

List of Tables ... xix

List of Abbreviations ... xxi

Chapter 1. Introduction ... 1

1.1. Problem statement ... 1

1.2. Research motivation ... 2

1.2.1. Designing for human safety ... 2

1.2.2. Achieving collaboration and integration through BIM ... 3

1.2.3. Designing buildings with knowledge in mind ... 4

1.3. Research hypothesis ... 5

1.4. Research questions ... 5

1.5. Research contribution ... 6

1.6. Structure of the thesis ... 6

Chapter 2. Literature Review ... 9

2.1. Designing safe building egress ... 9

2.1.1. Determining evacuation time ... 9

2.1.2. Crowd simulation models and tools for evacuation ... 14

2.2. Building Information Modelling for collaboration ... 17

2.2.1. BIM level interoperability ... 17

2.2.2. BIM-based evacuation design ... 19

2.3. The Semantic Web and Linked Data paradigm ... 22

2.3.1. Semantic web languages – RDF to OWL ... 22

2.3.2. Ontologies for building design ... 25

2.4. Summary of literature findings ... 28

Chapter 3. System design and methodology ... 31

Chapter 4. Requirement analysis ... 39

4.1. Design of crowd simulation tools ... 39

4.1.1. Features and capabilities ... 40

4.1.2. Taxonomy of model concepts 43
4.2. Scenario automation requirements 48
4.2.1. Creating valid models 48
4.2.2. Creating analysis feedback 52
4.3. Technologies and tools requirements 54
4.3.1. Crowd simulation model. MassMotion 54
4.3.2. Digital building model. IFC and IfcOwl 57
4.3.3. Knowledge modelling tool. Protégé 58
4.3.4. Knowledge management server. Stardog 59
4.4. Summary 60
Chapter 5. Knowledge base development 61
5.1. Ontology development 62
5.1.1. ONTOCS overall alignment configuration 63
5.1.2. Representing the crowd simulation model 65
5.1.3. Representing the analysis feedback process 72
5.1.4. Representing design codes 76
5.1.5. Representing a crowd simulation tool 81
5.2. Ontological alignment of concepts 83
5.2.1. Aligning digital building and crowd simulation models 84
5.2.2. Aligning scenario and feedback analysis models 85
5.2.3. Aligning design codes and classification criteria 88
5.3. Ontology validation 90
5.4. Summary 91
Chapter 6. Storing and retrieving knowledge 93
6.1. Knowledge operator types 94
6.1.1. Reasoning rules. SWRL 95
6.1.2. Queries. SPARQL 98
6.2. Storing information and knowledge 100
6.2.1. Storing building model data and information 103

6.2.2. Storing design process information and knowledge 108
6.3. Retrieving information and knowledge 110
6.3.1. Stage I – Scenario generation 112
6.3.2. Stage II – Analysis feedback 119
6.3.3. Linking models for future extensibility 127
6.3.4. Scenario vs element view 128
6.4. Summary 130
Chapter 7. System implementation, testing and validation 131
7.1. Introduction to the ONTOCS system 131
7.1.1. ONTOCS process workflow 132
7.1.2. ONTOCS architecture 135
7.2. ONTOCS Case study on Queen’s Buildings 137
7.2.1. Building description 138
7.2.2. Use cases – objectives, assumptions and rationale 139
7.2.3. Stage I – Scenario generation use case results 144
7.2.4. Stage II – Analysis feedback use case results 152
7.2.5. Discussion 165
7.3. Summary 170
Chapter 8. Discussion and future work 171
8.1. Revisiting the hypothesis 171
8.2. Research limitations 178
8.3. Future work 179
Chapter 9. Conclusion and contribution 181
Bibliography 185
Appendix A – Simulation tools concepts 197
Appendix B – Developed ontologies 212
Crowd Simulation Scenario (CSS) ontology 213
Feedback Analysis (FBA) ontology 216
UK Spaces Occupant Capacity (UKSOC) ontology 219

MassMotion (MM) ontology 223

Alignment between UKSOC and Uniclass2015 ontologies 226

Appendix C – SPARQL queries 235

Appendix D – ONTOCS system interface 252

Appendix E – Case study data 262

List of Figures

Figure 2-1. Components in evaluating RSET and ASET (PD 7974 2004) 10

Figure 2-2. Example evacuation time distribution of people leaving the premises of a building (adapted from PD 7974 2004).. 11

Figure 2-3. The computer modelling process for evacuation design, adapted from Kuligowski (2016a) 16

Figure 2-4. Increasing levels of semantics for data on the Web.................................. 22

Figure 2-5. Example of an RDF triple following the SPO pattern 23

Figure 3-1. The envisaged processes for achieving knowledge mining using the proposed ONTOCS system... 32

Figure 3-2. Pursuit of knowledge in parallel to system design and testing.................. 33

Figure 4-1. CST concepts by domain feature categories (plot data in Appendix A)..... 44

Figure 4-2. Crowd simulation scenario information requirements (categories which contribute to context are in Figure 4-3)... 49

Figure 4-3. Information sources contributing to simulation scenario context............... 51

Figure 4-4. Example model with a plotted Fruin's LOS map 54

Figure 4-5. MassMotion interface 56

Figure 4-6. Summary view of the IfcOwl ontology 57

Figure 4-7. Entities view tab of the FOAF ontology in the Protégé tool........................ 58

Figure 4-8. Stardog web interface pages for schema and tree browsers..................... 59

Figure 5-1. Alignment of ontologies for the ONTCS system and potential extensions to nearby knowledge domains... 64

Figure 5-2. The Crowd Simulation Scenario (CSS) ontology 65

Figure 5-3. Main CSS classes in direct relationship with the 'Scenario' class............. 66

Figure 5-4. CSS ontology 'ModelObject' class hierarchy 68

Figure 5-5. CSS ontology 'ScenarioAssumption' class hierarchy................................ 69

Figure 5-6. CSS ontology 'SimulationResult' class hierarchy...................................... 70

Figure 5-7. CSS ontology 'Agent' class relationships... 71

Figure 5-8. The Feedback Analysis (FBA) ontology... 73

Figure 5-9. Main FBA classes in direct relationship with the 'Scenario' class 75

Figure 5-10. FBA ontology concepts capturing user objectives and requirements 76

Figure 5-11. The UK Spaces Occupant Capacities (UKSOC) ontology 77

Figure 5-12. UKSOC ontology ‘Space’ class and its relationships 78

Figure 5-13. UKSOC ontology SWRL rules 79

Figure 5-14. The MassMotion (MM) ontology 81

Figure 5-15. MM ontology “Object” class with important subclasses and properties 82

Figure 5-16. Alignment of classes between the CSS and IfcOwl ontologies 84

Figure 5-17. Alignment of main classes between the CSS with FBA ontology 86

Figure 5-18. Alignment of all concepts between the CSS and FBA ontologies 87

Figure 6-1. Pyramid with increasing levels of meaning 93

Figure 6-2. Conceptualisation of a property chain using SWRL 97

Figure 6-3. Example SPARQL query and results 99

Figure 6-4. Example of comparison between OOP instances and OWL individuals .. 101

Figure 6-5. ONTOCS ontologies using RDF graphs as resources 102

Figure 6-6. Example of a space instance which is represented by 3 equivalent ontology individuals across several knowledge domains 103

Figure 6-7. Tree hierarchy of an IfcSpace individual in IfcOwl. 105

Figure 6-8. MassMotion and CSS storage of objects and data in RDF 108

Figure 6-9. Storing scenario assumptions (1.) and user objectives (2.) 110

Figure 6-10. ONTOCS system process workflow through its two main stages 111

Figure 6-11. SPARQL operators retrieving information from the IfcOwl model 113

Figure 6-12. SPARQL query Q-IFC-2 operating on the IfcOwl model, retrieving data about instances with results shown below it 115

Figure 6-13. SPARQL query Q-IFC-1 reasoning IfcOwl individuals which are also MassMotion ontology individuals according to ontology schema alignment 115

Figure 6-14. Example of contextual information being interpreted by rules from an IFC domain to provide context to a crowd simulation model 116

Figure 6-15. SPARQL query (Q-RES-4) reasoning individuals which are classified as ‘InhabitedSpace’ within the scope of the CSS ontology 117

Figure 6-16. ONTOCS interface for objective input page 120

Figure 6-17. Plotted egress progression results for several example scenarios and objective set 3 from Table 6-3 121

Figure 6-18. Static and dynamic information model progression considering changes in design and context 127

Figure 6-19. Example of scenarios where a group of agents decide to change the evacuation route 129

Figure 6-20. Example of objects and properties (contextual and geometric) influencing the final analysis result 130

Figure 7-1. ONTOCS system components with numbered workflow steps 134

Figure 7-2. ONTOCS system package diagram across functional layers; packages in green represent developed code; packages in grey represent imported code 135

Figure 7-3. Case study building layout, ground floor 138

Figure 7-4. Average query times for geometry retrieval 144

Figure 7-5. Average query times for context retrieval 145

Figure 7-6. Comparison of manual versus automatic model creation time 146

Figure 7-7. Merged manual and ONTOCS automatic models for object positions comparison 147

Figure 7-8. Plotted agent numbers versus time for scenarios SC1 (blue) and SC2 (red) 148

Figure 7-9. Plotted agent population versus time for scenarios 1 and 2 (ONTOCS), 5 and 6 (manual) 149

Figure 7-10. Plotted agent population versus time for scenarios SC3 (blue) and SC4 (red) 150

Figure 7-11. Plotted maximum density experienced during scenario SC1 for ONTOCS and manually constructed models 151

Figure 7-12. Plotted agent number versus final egress times for scenarios 1-36 152

Figure 7-13. Plotted agent numbers versus egress progression in time for scenarios 1 to 18 which use real building data stored in the IFC model. UKSOC scenario 26 (blue) at 100% population capacity was added for comparison 154

Figure 7-14. Plotted agent numbers versus egress progression in time for scenarios 19 to 36, which use factors reasoned from UKSOC. IFC scenario 8 (red) at 100% population capacity was added for comparison 155

Figure 7-15. Plotted query reasoning times for single objective input 158

Figure 7-16. Plotted query reasoning times for multi-objective input 159

Figure 7-17. Scalability for objective (a) – Total egress time 160

Figure 7-18. Scalability for objective (b) – Capacity egress time 161

Figure 7-19. Scalability for finding valid scenarios 162

Figure 7-20. Scalability for finding invalid scenarios 163

Figure 7-21. Scalability for finding fully valid scenarios 164

Figure B-1. CSS ontology classes and object properties connecting them 213

Figure B-2. CSS ontology metrics..214

Figure B-3. CSS ontology object properties...214

Figure B-4. CSS ontology data properties 215

Figure B-5. FBA ontology classes and object properties connecting them.................216

Figure B-6. FBA ontology metrics 217

Figure B-7. FBA ontology object properties 217

Figure B-8. FBA ontology data properties..218

Figure B-9. UKSOC ontology with main classes and individuals...............................220

Figure B-10. UKSOC ontology classes and object properties connecting them.........221

Figure B-11. UKSOC ontology metrics 222

Figure B-12. UKSOC ontology SWRL rules matching individuals of specific classes to factors 222

Figure B-13. MM ontology with main upper classes and object properties connecting them 223

Figure B-14. MM ontology object properties 224

Figure B-15. MM ontology data properties...225

Figure B-16. MM ontology metrics 225

Figure D-1. ONTOCS welcome page 253

Figure D-2. ONTOCS user log in page 254

Figure D-3. ONTOCS database selection and IFC model upload page 255

Figure D-4. ONTOCS geometry conversion report from IFC to MassMotion and ontology upload page 256

Figure D-5. ONTOCS scenario creation page 257

Figure D-6. ONTOCS scenario generation, validation and selection for analysis page 258

Figure D-7. ONTOCS scenario level results and feedback page 259

Figure D-8. ONTOCS objects creation and reasoning results viewing page............. 260

Figure D-9. ONTOCS object level results page... 261

Figure E-1. Case study building layout with numbered spaces................................. 262

List of Tables

Table 2-1. Route escape factors according to Shields and Silcock (1987) 13

Table 3-1. Knowledge mining components (Kaufman and Michalski 2005)................. 31

Table 4-1. List of CSTs investigated.. 40

Table 4-2. List of main CST feature categories 41

Table 4-3. Taxonomy of common concepts for a crowd simulation analysis domain... 47

Table 4-4. Identified PIs for building performance assessment during evacuation scenarios 53

Table 5-1. Metrics for developed ontologies.. 63

Table 5-2. Spaces occupant capacities (adapted from The Building Regulations (2015) Annex C3 – Methods of measurement).. 80

Table 5-3. Example of aligned concepts between the UKSOC and Uniclass 88

Table 5-4. Example of UKSOC and Uniclass alignment rules (Appendix B)................ 89

Table 6-1. List with developed queries for Stage I. 114

Table 6-2. List of SWRL rules for the CSS ontology classifying space types............. 118

Table 6-3. Example of objective sets inputted by uses for the analysis stage 119

Table 6-4. Developed queries retrieving knowledge from the FBA model 123

Table 6-5. SWRL rules operating with the FBA ontology for classifying scenarios 125

Table 7-1. List of constructed and analysed scenarios 140

Table 7-2. Simulation scenarios created using the ONTOCS system for analysis..... 141

Table 7-3. Set of analysis objectives inputted into the system for evaluating the 36 simulation scenarios 141

Table 7-4. SPARQL queries and their respective SWRL rules operating within the FBA ontology 143

Table 7-5. Input model size and ONTOCS conversion report 145

Table 7-6. Time for manual construction actions of the model using MassMotion..... 146

Table 7-7. Objective sets with ontology reasoning answers per query 153

Table 7-8. SPARQL query times measurements taken during the session with the 36 scenarios running on the ONTOCS system 157

Table A-1. MassMotion Concepts 197

Table A-2. Pedestrian Dynamics concepts 200

Table A-3. STEPS concepts 203

Table A-4. BuildingEXODUS concepts 207

Table A-5. Simulex concepts 210

Table A-6. Summary of common CST concepts from previous tables 211

Table B-1. Spaces occupant capacities (adapted from The Building Regulations 2015 Appendix C3 – Methods of measurement) 219

Table B-2. Aligned common spaces between UKSOC categories and Uniclass categories with comments 226

Table B-3. Alignment SWRL rules between the UKSOC and Uniclass2015 ontology. Implements Table B-2 229

Table C-1. SPARQL queries operating to IfcOwl 235

Table C-2. SPARQL queries operating on CSS and other resources 250

Table E-1. Building spaces data exported from Revit 263

Table E-2. STAGE I query times for the run of 36 scenarios in parallel 266

Table E-3. STAGE II query times for increasing numbers of scenarios (part1/3) 267

Table E-4. STAGE II query times for increasing numbers of scenarios (part2/3) 268

Table E-5. STAGE II query times for increasing numbers of scenarios (part3/3) 269

List of Abbreviations

AEC = Architectural Engineering and Construction

ASET = Available Safe Escape Time

BIM = Building Information Model

BREEAM = Building Research Establishment Environmental Assessment Method

CA = Cellular Automata

CS = Crowd Simulation

CSM = Crowd Simulation Model

CSS = Crowd Simulation Scenario (ontology)

CST = Crowd Simulation Tool

DBM = Digital Building Models

DLs = Description Logics

FBA = Feedback Analysis (ontology)

FOAF = Friend of a Friend (ontology)

IFC = Industry Foundation Classes

IRI = Internationalized Resource Identifier

LD = Linked Data

MM = MassMotion

ONTOCS = Ontology Crowd Simulation

OOP = Object Oriented Programming

OWL = Web Ontology Language

PF = Path-Finding

PI = Performance Indicator

RDF = Resource Description Framework

RSET = Required Safe Escape Time

SF = Social Forces (model)

SPARQL = SPARQL Protocol and RDF Query Language

SPO = Subject -> Predicate -> Object

SQL = Structured Query Language

SW = Semantic Web

SWRL = Semantic Web Rule Language

UKSOC = UK Spaces Occupancy Capacities (ontology)

UML = Unified Modelling Language

IRI = Internationalized Resource Identifier

URI = Uniform Resource Identifier

W3C = WWWC = World Wide Web Consortium

WWW = World Wide Web

Chapter 1. Introduction

1.1. Problem statement

The building design process is multi-disciplinary, complex and is faced with increasing expectations for better building performance across all fields, most importantly energy efficiency and human safety.

The rise in technologies over the past decades has made it possible for designers to easily share data across design-disciplines. The use of Digital Building Models (DBM) and Building Information Modelling (BIM) processes allows computer tools and designers to exchange data and collaborate seamlessly. More recent developments have focused on ways to integrate building data with other knowledge fields, making design systems more intelligent and more comprehensive.

While significant strides have been made in applying the BIM paradigm to cost and energy modelling where automation and interoperability is high, there is still a significant gap when looking at the field of fire evacuation analysis. This is mainly due to the complexity of fire safety design which has traditionally relied on various fields of knowledge ranging from building design to psychology.

Global population growth and urbanisation put ever increasing pressure on engineers to ensure high standards of safety. The use of Crowd Simulation Models (CSMs) to assess building performance in various scenarios, especially evacuation design, is becoming more prevalent when dealing with highly populated buildings such as airports. However, these are niche tools requiring significant amount of time to invest in scenario construction and analysis being reliant on many sources of information and often bringing little added benefit.

1.2. Research motivation

1.2.1. Designing for human safety

"Buildings are designed and constructed to accommodate people and processes, among other things. Consequently, the flow of people, materials, and products must be incorporated in such a way to achieve a pre-determined level of fire safety. The minimum level of fire safety to be achieved may be prescribed by building legislation or be determined by insurance requirements or other influences." (Shields and Silcock 1987)

The British published documents concerning fire regulations and guides in the UK, divide the entire process into 7 independent sub-systems (PD 7974 2004), each of which need to be assessed individually:

1) Fire growth – the way in which fire and heat spread depending on flammability of materials, presence of oxygen, and how it can be modelled;
2) Smoke spread – the propagation of gas and fire emissions through spaces, its potential effects on building materials and occupants and ways to minimise them;
3) Structural protection - the structure of the building, what designers should consider to effectively increase the building's resilience in case of fire;
4) Detection and suppression – considers ways in which fire can be detected early-on so its effects can be minimised;
5) Intervention – concerning the fire action plan of isolating the incident, extinguishing the source of fire and ensuring the safety of the structure and its occupants;
6) Human factors – tackles the complexities of the nature of the occupants and their behaviour during evacuation events, as well as ways of simulating them using different computer models;
7) Risk assessment – ways to identify fire related risks and their impact.

Considering the above, designers must deal with a complex system of assessing the performance of the building in different perspectives which are interdependent (Chitty and Fraser-Mitchell 2003).

The 2017 Grenfell Tower disaster in London (Government Digital Service 2018) is only one example of the sort of tragedies that can occur when the fire safety requirements are neglected by authorities. According to the England statistics on fire incidents (Home Office 2018), the magnitude of this incident alone gave rise to the percentage of casualties by 37% compared to the previous year, even though the trend of casualties has been on decline since 1981. In cases such as these, building designers and managers have to ensure a strict adherence to fire safety regulations. Baiche et al.

(2006) found that building regulations are not always complied with, due to lack of knowledge of those designing or checking. Alper and Karsh (2009) offer a comprehensive review over regulation violations, suggesting similar causes. In the scope of building design where designers cannot comply with regulations, they are required to prove a certain degree of safety via modelling techniques.

The scope of this research is focused on assessing building performance with regards to human factors. In practice this is evaluated using Crowd Simulation Tools (CSTs), which are able to simulate in different levels of detail how people behave during an evacuation event. This in turn allows designers to assess the performance of the building in such scenarios (Thalmann et al. 2007, Ronchi et al. 2014). The entire process relies on expert designers using CSTs to create, run and analyse scenarios using several iterations, which is time consuming (Khan et al. 2014). Additionally, each building layout is different and so is each scenario in terms of context (Nilsson and Fahy 2016). The challenge lies in being able to assess the building performance in an efficient manner and on a larger scale, thus being able to identify flaws in the building design in a speedy manner.

Considering the aspect of fire safety design assessment within the BIM paradigm, the level of integration and automation concerning Crowd Simulation (CS) analysis is relatively low compared to other aspects of design, such as energy modelling. This will become clearer in the literature review chapter. For now, it is worth mentioning that the modelling process is complex in itself and has to catch-up with current BIM technologies.

1.2.2. Achieving collaboration and integration through BIM

In the last 20 years, the construction industry has seen major advances in digital modelling, but this has also raised industry expectations in terms of fast project delivery. A prime example is the UK Government’s initiative to impose level 2 BIM on public projects (Cabinet Office 2011) , which came into effect in 2016. For BIM level 3 and beyond, the most prevalent issues under investigation were related to standardisation efforts for interoperability and collaboration of digital information. One of the main drivers of this standardisation was the expanding use of common and interoperable formats, mainly evolving around the Industry Foundation Classes (IFC). The use of the IFC schema and format ensure the exchange of structured information for all the domains included in the Architectural Engineering and Construction (AEC) industries. IFC is especially useful in design and construction stages, being used to exchange data, transfer model views, facilitate prescriptive rules checking and most importantly import/export of models across many tools. Although there are a myriad of BIM tools and platforms aimed at improving the multidisciplinary design flows for construction and design offering IFC support, they are limited in many ways and are not always the best way to collaborate across disciplines (Díaz et al. 2017). The use of IFC in its current state cannot solve all problems related to integration and collaboration, but it represents a foundation stone for structured data.

Attempting to predict the roadmap of future BIM developments, Succar (2009) introduced a BIM ontology of intersecting knowledge domains and describes a network of integrated models and services from BIM level 3 onwards, which can be used beyond just the semantic properties of the building models. Thus, it is expected that BIM models will be able to provide more than just data and information, but also knowledge (Bellinger et al. 2004) about the building environment.

1.2.3. Designing buildings with knowledge in mind

"What is the difference between knowledge and information and can computer applications really deal with knowledge?" (Stenmark 2002)

Modern computing methods can be used to integrate databases, knowledge and operators which can be used for decision making processes. The term 'knowledge base', refers to the concept of formalising human knowledge in computer understandable format (Kaufman and Michalski 2005). Knowledge in this context may refer to professional experience, data from simulations and analyses, predictions or best practices. It can be argued that knowledge should encompass all the mentioned factors to truly facilitate a good design solution.

The Semantic Web (SW) and Linked Data (LD) paradigms have become increasingly popular around the BIM domain (Abanda et al. 2013). SW and LD tools in construction design extend interoperability by including web resources, while also giving data a higher degree of meaning with the use of languages such as the Web Ontology Language (OWL) (Schevers and Drogemuller 2006), while being able to conceptualise knowledge models. With the development of IfcOwl (Beetz et al. 2009), the OWL representation of the IFC schema, various knowledge fields can be included from a wide spectrum of applications. Integration of information is the clear benefit on using BIM (Azhar 2011) on construction projects, but whether the SW will be able to fill this need remains to be seen. LD concepts work with large datasets across the web and rely heavily on open data sources, but the AEC industry culture remains reluctant to sharing data openly (Zanni et al. 2017). Apart from the linking of data, which is of great benefit to the AEC industry, OWL ontologies are widely used today to represent knowledge in computer understandable formats, and enable machine reasoning (Gibbins and Shadbolt 2009), which can facilitate its storage and retrieval by intelligent web-based systems.

The use of LD could greatly benefit the better integration of information models between BIM and CS domains not just on a structured data level, but also on a knowledge process level, facilitating speedy design analysis and automation.

1.3. Research hypothesis

In light of the problems identified above, this research aims to tackle current limitations concerning evacuation design by adopting the following overarching hypothesis:

A knowledge processing-based approach can allow a fast retrieval of information and automatic construction of evacuation models by leveraging existing BIM data and design knowledge to enhance the decision-making processes about building performance by considering different simulation scenarios on a large scale.

1.4. Research questions

Aiming to contribute to knowledge in the process, the hypothesis is decomposed into the following questions:

Q1: How are evacuation models and tools used for assessing design performance while considering their scope and limitations?

Q2: What is the current level of interoperability between CS for evacuation and BIM?

Q3: What are the benefits of using ontologies for evacuation design, considering the BIM paradigm?

Q4: What are the requirements for an intelligent system capable of integrating resources relevant to the CS field within the context of automation and analysis feedback, whilst considering practical deployment and future extensibility?

Q5: What are the challenges concerning information models and workflow processes being represented in a knowledge base, considering the requirements for integration and knowledge retrieval?

Q6: What needs to be considered for design knowledge storage and retrieval concerning building egress performance using evacuation models?

Q7: How reliable is a knowledge-based system in understanding the building model and other linked data resources in facilitating correct and efficient design support?

1.5. Research contribution

The work carried out during this research project has contributed with several practical developments and with knowledge about the methodology adopted in delivering their implementation and in testing.

The core contribution of this work is the implementation and development of an intelligent knowledge-based system which can reason over linked data and knowledge resources represented in web languages. This in turn allows semi-automatic creation of simulation scenarios on a large scale which can be used in real design cases. Once these scenarios are executed using conventional simulation tools, their results can be aggregated by the system and feedback is provided to the users about the performance of building. The methodology adopted in the development of the system and the definition of the processes involved is also another core contribution, which can be replicated to other fields, extended or optimised for future developments.

The system makes use of several ontologies and knowledge rules which represent another core contribution of this research. These were developed based on reviewing of current academic research and industry guidance from manuals, and discussions with field experts. The representation of such knowledge in a machine-interpretable format will be useful to extend the current use of BIM and semantic web technologies to the fields of crowd simulation and evacuation design.

In parallel with the developments above, this work contributes to the overall pool of knowledge concerning the interoperability levels between BIM and CS, from the review conducted on academic research and practical assessment of several simulation software tools used in practice. Additionally, the methodology adopted for this research which was directed and applied to solving practical problems, brings insight into crowd simulation concepts, their integration with BIM and IFC, the reliance on design guidance resources and the benefits and limitations of using a knowledge-based system in this context. All these contributions to knowledge can be used to replicate this work into nearby design fields or to optimise current developments to become faster and more reliable in the future.

1.6. Structure of the thesis

The thesis is divided into several chapters, each pursuing answers for the main research questions.

Chapter 2 brings arguments against the gaps indicated and aims to answer the first three research questions by investigating the broader field of knowledge in the domains which are relevant to this research: evacuation design, crowd simulation tools, BIM and the use ontology-based systems. Background reading was conducted concerning the essential relevant concepts and where these fields overlap. The chapter is split into three sections, each focusing on one question. The first section outlines the role of crowd simulation models and tools in the field of fire safety. The second investigates the existing level of interoperability between CS developments and BIM. Finally, the third section explores the use of ontologies, indicating towards the benefits for the evacuation design field. The main findings from the literature are used to propose a novel solution.

Chapter 3 presents the conceptual framework of ONTOCS, an intelligent ontology-based system which is aimed at representing and retrieving new knowledge about design. The methodology employed to prove the research hypothesis using the prototype system is outlined.

Chapter 4 identifies the requirements for developing the ONTOCS platform and thus aims to answer the fourth question. The requirements include establishing a common CST taxonomy, identifying suitable tools for achieving functionality in practice, and identifying available sources of information and knowledge which enable automation and can provide relevant feedback for the design decision making process.

Chapter 5 aims to answer the fifth research question by presenting several developed OWL ontologies. The framework proposed relies on defining the knowledge domains using these ontologies for semantic web integration. The main concepts identified were defined, categorised and connected semantically. Some of the created ontologies were aligned with external ones to extend and test the benefits of shared information.

Chapter 6 addresses research question six and outlines the methods used to store and retrieve knowledge about the various information models. It begins with an introduction of knowledge operators, then presents ways and the challenges of working with knowledge bases in the context of CSM performance assessment whilst considering user input.

Chapter 7 is focused on the testing of the ONTOCS system and commenting on the overall methodology of this research. The chapter firstly introduces the system design and the workflow of the process it facilitates. Secondly, a case study on a real building is presented, with specific use-case scenarios, aimed to test the system and thereby validate the ontologies and knowledge operators developed in the previous chapters. A discussion is provided based on the decomposition of the final research question.

Chapter 8 concludes the work presented in previous chapters by outlining the main findings within the context of the research hypothesis. Research limitations and future work are then presented.

Chapter 2. Literature Review

This chapter reviews the status-quo of CSM and BIM technologies used to facilitate design support. The contents are divided into three sections, each aimed at a research question: The first section (3.1) investigates CSM in research and practice, the second (3.2) identifies the level of interoperability of CS with BIM, and the final section (3.3) outlines developments around knowledge processing tools and methods used for building design. Each sub-section begins with introducing the three relevant fields for this research: CS, BIM and OWL ontologies.

2.1. Designing safe building egress

2.1.1. Determining evacuation time

The entire fire safety design process is a complex multi-disciplinary process which spans across different knowledge fields from structural fire resistance to human psychology. Fire design employs many regulations which were improved over the years to enforce a certain standard of safety. Regulations are usually set as a minimum requirement on the building design and they are usually a compromise between optimal safety and economic feasibility, with the purpose:

“1. To impose a level of fire safety such that it is unlikely that people occupying a building would suffer hurt in the event of an unwanted fire, and

2. To protect the community at large from the consequences of fire in an individual building.“ (Shields and Silcock 1987)

Fire safety engineering is governed by many sets of prescriptive rules concerning different design aspects, which were developed and improved through empirical methods. When certain design rules are not met, proving adequate building performance is necessary. Performance design needs to specify the objectives which should be aimed for. Meeting these requirements deems the safety is adequate.

To assess building performance objectively designers rely on balancing the Available Safe Escape Time (ASET) with the Required Safe Escape Time (RSET). The basic principle here is to always make sure ASET is greater than RSET, as shown in Figure 2-1. *"An appropriate margin of safety takes account of the risks associated with different potential fire scenarios and the uncertainties in the prediction of ASET and RSET for particular design scenarios."* (PD 7974 2004)

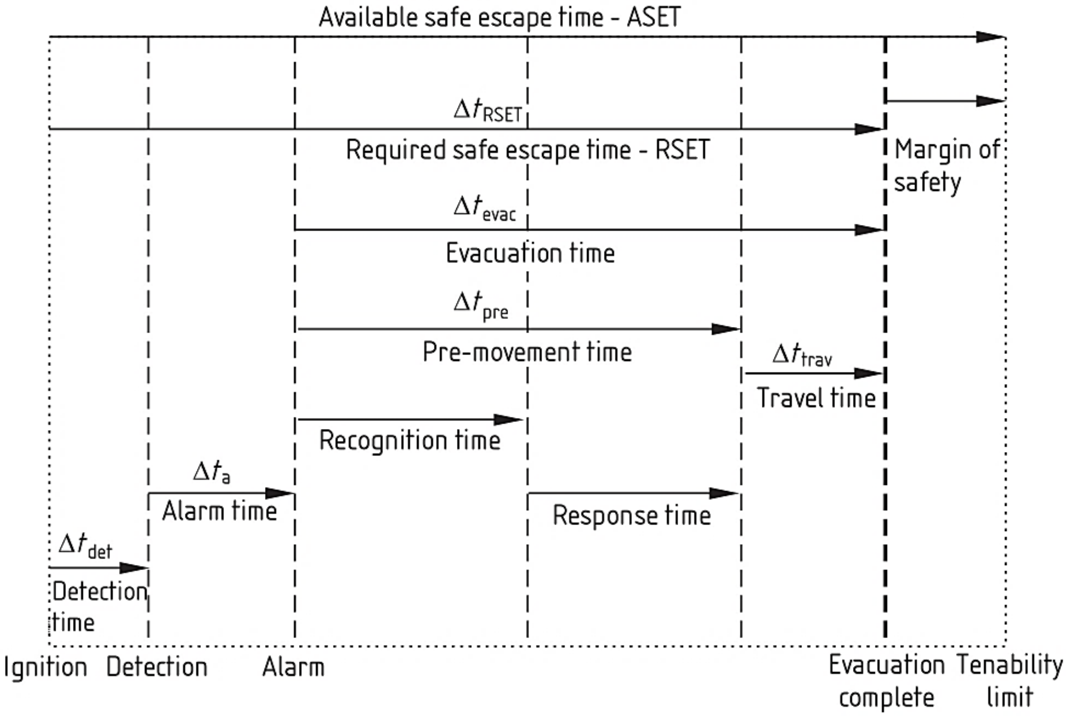


Figure 2-1. Components in evaluating RSET and ASET (PD 7974 2004)

Each of the components in the calculation of ASET and RSET are assumed individually:

a) Δt_{det} = time from fire ignition to its detection; dependent on the method of detection, usually an alarm system;

b) Δt_{a} = alarm time from detection to action being taken to evacuate;

c) Δt_{evac} = effective evacuation time, which consists of Δt_{pre} + Δt_{trav} ;

d) Δt_{pre} = the pre-movement time, is influenced by the behaviour of occupants which has two components: ***recognition*** – time taken for each individual to respond to cues and begin taking action, and ***response*** – actions taken immediately after recognising that a fire event is real and evacuation is necessary (dependent of occupant roles);

e) Δt_{trav} = the time for occupants to reach a safe refuge point;

Different combinations of assumptions can be made about the factors above, each influencing the performance of the next, and consequently RSET as a whole. Detection time (Δt_{det}) and alarm time (Δt_{a}) depend highly on the fire strategy in place and technological system incorporated within the final building design and are considered out of scope for this research. The evacuation time (Δt_{evac}) is the factor where human behaviour plays an important role, and it can be estimated using a mixture of historical data, live drills and CSMs. From its two main components, the first (Δt_{pre}) is more reliant on observation data from previous real or mock evacuation scenarios, with Figure 2-2 showing an example of a typical evacuation curve. The pre-evacuation time is characterised by occupants initially seeking confirmation of a fire, with some occupants reacting sooner than others, expressed as the pre-movement of the first occupants $\Delta t_{pre\ (first\ occupants)}$. Once the first occupants begin to evacuate, more people become aware of the event and the number of evacuees increases rapidly in a short period of time, following the peak in Figure 2-2, defining a distribution of occupant times before they begin travelling towards the exits $\Delta t_{pre\ (occupant\ distribution)}$.

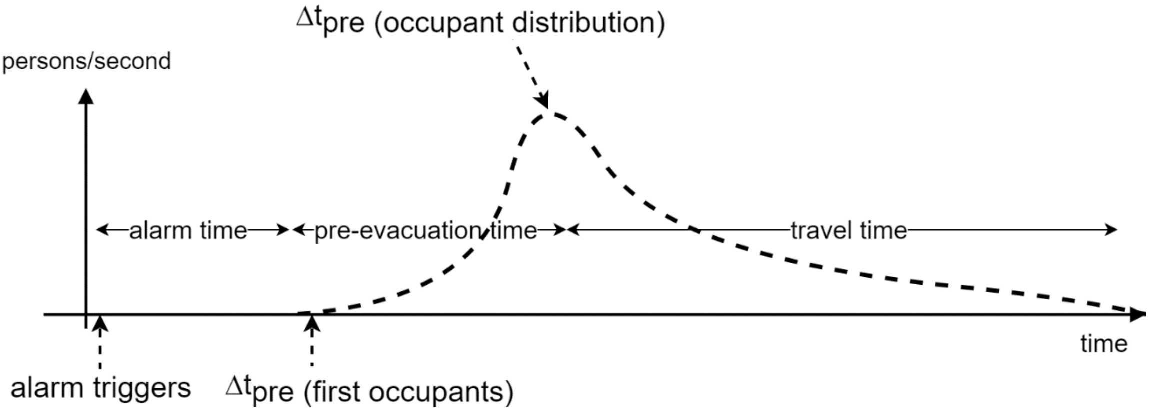


Figure 2-2. Example evacuation time distribution of people leaving the premises of a building (adapted from PD 7974 2004)

Travel time (Δt_{trav}) is characterised by two components:

1) $\Delta t_{trav\ (walking)}$ = walking time for people to reach safety; dependent on the walking speed of each occupant which can vary greatly; this factor can be considered as an average or per each individual;
2) $\Delta t_{trav\ (flow)}$ = flow time of occupants through the building; determined by the capacities of the doors and exits relative to the population number;

The evacuation time (Δt_{evac}) is estimated by evaluating each of its sub-components individually. The simulation and evaluation of Δt_{pre} using CSMs is not very common as it involves many behavioural factors. This is usually a simplified assumption based on empirical factors according to building type and the level of safety management to be achieved. For example, a combination of good detection and alarm system can assume that people will start evacuating sooner, therefore the pre-evacuation time will be significantly shorter than in a case with poor safety management. *"For evacuation times to become a viable component of fire safety engineering, it is, therefore, vital that a* *database of pre-movement times and pre-movement time distributions is obtained for a variety of occupancy types and a variety of building design and fire safety management strategies"* (Purser and Bensilum 2001). While this can be very useful, in many situations, due to the difference in building design, it can actually be of very little value for comparisons (Kuligowski 2016b). A simple change of an exit can produce very different results.

Travel time (Δt_{trav}) on the other hand, is a widely used factor which is measured during live drills and used extensively by CS tools and models, as the prime indicator of building performance. According to PD 7974 (2004), the two components which influence travel time can be simulated using CSMs in two very specific cases:

1) Scenarios at less than 33% population design capacity – the evacuation time is more dependent on the travel time of agents, $\Delta t_{trav\ (walking)}$. The low concentration of agents will not allow for bottlenecks to form, and is thus not influenced by the flow time through doorways;
2) Scenario at 100% population design capacity – the evacuation time is more dependent on the flow capacities of the exits, $\Delta t_{trav\ (flow).}$ This is because queuing is expected to form relatively quickly, reaching the maximum capacities of the exits.

When considering the estimation of evacuation time (Δt_{evac}), designers need to also be aware of the factors which influence it, as summarised in Table 2-1, from which two main categories emerge:

- building factors – refer to factors about the building environment, such as layout and position of objects; these are static in nature;
- human behaviour factors – refers occupant positions and distribution within the building environment, their movement and reaction speeds.

These two categories are interconnected: the building design is influenced by occupant needs and safety, while the occupant behaviour is influenced by the building layout (Nilsson and Fahy 2016), shape and components which occupants interact with (doors, stairs, lifts, furniture, alarm system, etc). While design regulations and standards emerged to cover both of the categories mentioned, the human factors have always proved difficult to account for. The fact that every building has in essence a unique design and layout, makes this even harder to assess.

Table 2-1. Route escape factors according to Shields and Silcock (1987)

Building factors	Human behaviour factors
Building type	Population density
Building contents	Population distribution
Evacuation time	Population mobility
Travel distance	Population reaction
Exits	Population discipline
Escape route width	
Enclosure of stairways	
Lobby approach stairways	
Doors in escape routes	
Lighting of escape routes	
Emergency lighting	
Construction of and egress from windows	
Fire detection system	
Alarm system	
Fire control system	
Smoke-control system	

The actual real fire safety performance of a building would be assessed during live drills. This however can only be done at the operation stage of the building lifecycle, where building layouts become too costly to change and so designers will try to justify their design decisions by strictly following rigid regulations (Gwynne et al. 1999). Purser and Bensilum (2001) conclude that while simple approximations of escape time for small buildings are acceptable, *"for larger more complex buildings, more sophisticated computer evacuation models may be required."*

Thus, during design stages, CS is now widely used in design decision-making. They are expected to provide relevant information in building performance evaluation, which is used by designers to assess feasibility or check against regulations.

2.1.2. Crowd simulation models and tools for evacuation

"Modelling amounts to finding an abstract representation of a real-world system that simplifies and assumes as much as possible about the system, and, while retaining the system's essential relationships, omits unnecessary details." (Druzdzel and Flynn 1999)

Crowd Simulation Models (CSMs) are intended to mimic realistic behaviour of people within certain environments by representing each person as an individual agent. Each agent is able to interact with the environment and other agents. CSMs are practically applied within software tools, commonly referred to as Crowd Simulation Tools (CSTs). The term CSM and CST is often used interchangeably. They are used in various situations: virtual crowds for computer games or films, training purposes for emergency situations, urban planning and for building evacuation design. Due to the rise in world population, CS methods will become invaluable to future infrastructure modelling (Zhan et al. 2008) (Khan et al. 2014).

There are several comprehensive reviews within the field of CS, which offer critical analysis regarding methodologies used (Gwynne et al. 1999, Kuligowski 2005) (Kuligowski 2016a), application domains (Kuligowski 2005), scale (Zhou et al. 2010), degree of realism (Duives et al. 2013) and high-rise buildings focused (Ronchi and Nilsson 2013). The afore-mentioned authors agree that there is no comprehensive model which can simulate all the complexities of human behaviour. Such a model would not be practical because as the complexity of the model grows, so does the computation time. Kuligowski (2005) advises that each model should be used for very specific purposes and users should be aware of each model's practical application and limitations. Ronchi and Nilsson (2013) mention that for a more comprehensive view, several models can be considered at the same time, as they might reveal more information from different perspectives. Zhou et al. (2010) and Duives et al. (2013) agree that models can be divided into microscopic models (small population) which have high precision, and macroscopic (large population) models with lower precision. Investigations carried out by Zheng et al. (2009) and Duives et al. (2013) suggest the Cellular Automata (CA), Social Forces (SF) and the Nomad models are the best methods for replicating reality. However, there is limited crowd heterogeneity route and destination choices when in the context of larger models, as these models were not calibrated to work realistically under large-scale conditions. The aforementioned authors confirm that one of the most used applications of these models is for fire evacuation, and that a fast computation time is not a critical issue as long as they are reliable and can accurately simulate route choices and destinations. They also suggest that more comprehensive models should be chosen over simpler ones, where possible.

An example of a limitation relates to the representation of the environment within the model. Some models represent the environment by dividing the surfaces into 2D arrays of cells (the CA approach), with agents being able to cross on cell at a time (Gwynne et al. 2001, FSEG 2018). Another common representation is the use of a more continuous surface, where agents have more freedom, thus effectively moving on very fine meshes (Musse and Thalmann 2001, Oasys Limited 2018a). The difference between the two is that the first one is faster to compute, while the second is able to represent more realistic human movement. On the other hand, a simpler calculation could be better suited for larger scale models. Regardless, this still does not guarantee that one tool or another is representing reality more accurately.

When comparing simulation models with reality Duives et al. (2013) argue that current models can be split into two categories: ones that mimic reality, and ones that try to be reality, with the later ones currently not being practical from a technical perspective as they would require significant computation capability, whilst still not guaranteeing better results. One of the major breakthroughs in the crowd simulation field is the Social Forces (SF) model (Helbing and Molnár 1995) which is widely used today to simulate the complex interactions between computer agents. Based on this model, future research has investigated several other factors, such as psychological factors, or simulating personal space forces based on particle interactions (Hesham and Wainer 2016). Fang et al. (2016) developed a model which simulates the concept of interpersonal relationships and attachments, influencing the way in which groups of agents interact during evacuations with regard to the decisions they make as a group. Other models focus on setting out the differences between individual and group behaviours, considering that when part of a group, individuals act differently (Raupp Musse et al. 2006, Li et al. 2015). Khan et al. (2014) present various methods for scene analysis of crowd behaviours which can be used to test the realism of existing simulation models and tools which rely on crowd data from real-life environments. This field has been of increasing interest due to a rise in population (Zhan et al. 2008) and a need to deal with emergency situations.

When comparing live drills with simulation results, it is hard to argue which is more representative of the truth, mainly due to the human factors. "*Repeated experiments on evacuation will never give the same outcomes because of the human factor, even when the same people are tested. Thus one experiment is never enough to prove a certain factor. Usually a distribution of several simulations is required.*" (Gwynne et al. 1999)

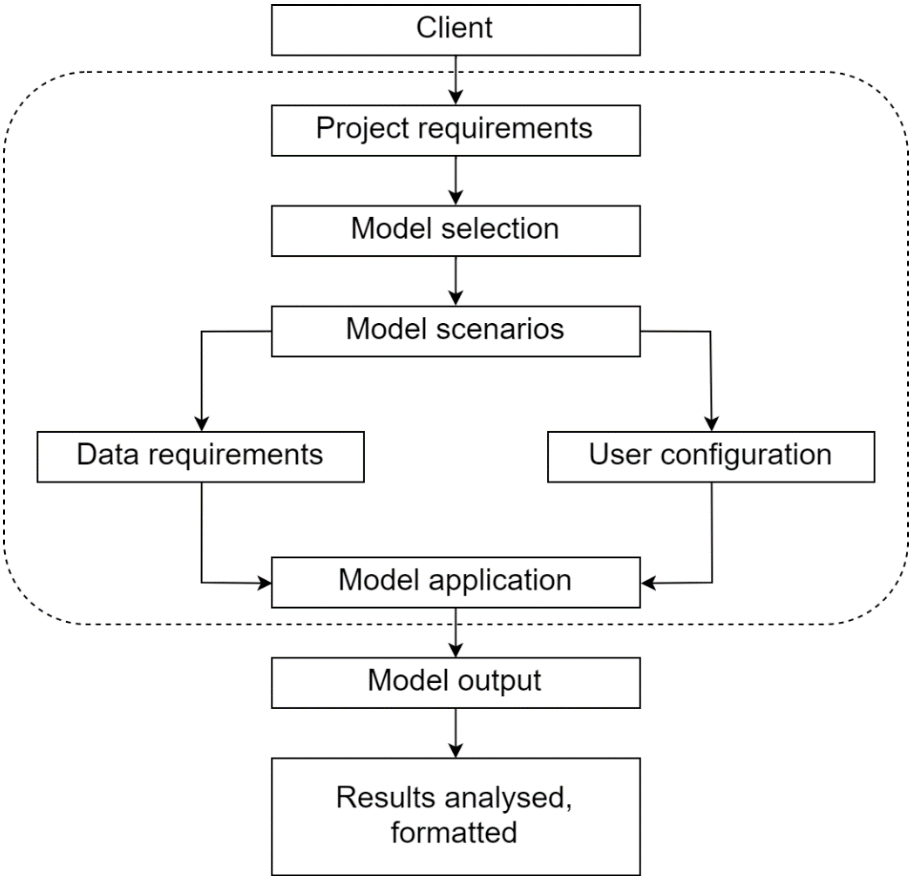


Figure 2-3. The computer modelling process for evacuation design, adapted from Kuligowski (2016a)

The process of modelling a crowd simulation scenario is best described by Kuligowski (2016a) and is shown in Figure 2-3. The entire process is heavily influenced by user input and follows 3 well defined steps:

1) Project requirements – client needs to assess the scope and context of the modelling process and what is expected to be gained from it;
2) Model selection – the tool which best meets the requirements should be chosen, considering its benefits, limitations and costs;
3) Model scenarios – users need to define all the boundary conditions of each model by considering:
    a. building configuration – defining the geometry, layout, exits, etc.;
    b. population configuration – defining agent numbers, positions, specified behaviours, etc.; level of sophistication may vary greatly;
    c. procedural configuration – defining routes of agents, flows and counter flows of groups, etc.;
    d. incident information – environmental conditions, such as the place of a fire.

The third step above represent the main model inputs and assumptions made about each scenario, which require several sources of information about the building occupancy and expected behaviour. These are assumptions which provide a different context to every scenario, influencing the final simulation results. An application (CST) then runs the models and provides outputs which are processed and interpreted during an analysis stage. It is not always clear how relevant the simulation output is, as it is dependent on large number of parameters (Hopfe and Hensen 2011, Kuligowski 2016a). To compensate for this limitation, it is often required to conduct several simulations, with several different assumptions and scenarios. This quickly becomes overwhelming when in the context of several design iterations, making it a highly inefficient process. This suggests the need to integrate and automate the process with de-facto BIM processes and standards.

2.2. Building Information Modelling for collaboration

2.2.1. BIM level interoperability

In the context of building design, industry currently relies on Building Information Modelling (BIM) processes and technologies. BIM has undergone a number of changes over the last decades, now encompassing multiple design domains and it is expected to extend further. There are several definitions of the concept, with a more recent definition of BIM being given by Crotty (2013) which sees the concept as an approach over several steps:

"A reference model of the building is created using one or more parametric component-based, 3D modelling systems. These systems exchange information about the building in one or more agreed standard file formats, with each other and with other systems which conform to the agreed formats. These exchanges are regulated by a set of protocols which establish the particular types of information to be exchanged between different members of the team, at different points in the project life-cycle."

A more simplified but widely accepted definition is given by Hardin and McCool (2015):

"BIM is a digital representation of the building process to facilitate exchange and interoperability of information in digital format."

Due to being an attractive concept in research and industry, BIM has developed several meanings and scopes, as the acronym 'BIM' can mean both 'Building Information Modelling' and 'Building Information Model'. This can generate confusion, associating the concept to buildings alone, when in fact the term can refer to the act of 'building' as a verb, extending the concept to infrastructure or other engineering structures or activities. According to Bos (2012), a mix-up of the term in the industry sector arose due to people confusing the concept of a 'shared data model', with the instance of a building model. BIM as a concept has a different meaning than BIM as an instance of a specific project model, but the two terms are used today interchangeably. For example: people will talk about the BIM model, referring to the digital model being projected using software tools, and they may also refer to the implementation of BIM processes, whereby project stakeholders work collaboratively using BIM standards and practices.

According to Eastman et al. (2011) and Bos (2012), early industry initiatives regarded central data models as an ideal means of storing and sharing data, where all project stakeholders would be able to store the data into a single model, a single 'point-of-truth'. Eastman et al. (2011) tries to distinguish between different software tools depending on the level of BIM implementation within an organisation, and the capabilities of such tools. The 3 main categories described by the authors also reflect three distinct levels of 'collaboration' within an organisation: (1) BIM tools, (2) BIM platforms and (3) BIM environments. The authors define the three concepts starting from the BIM tool as task-specific application toward a BIM environment as *"the data management of one or more information pipelines that integrate the applications (tools and platforms) within an organisation".* As the industry developed BIM processes further, the biggest barrier was the cultural change of openly sharing information, which was regarded with suspicion, as traditionally the construction industry works in separate information silos.

With time and along with technological advances, it was recognised that a common information format needed to be established, in order to be able to collaborate all relevant project model data consistently, without necessarily having to abandon the silo culture. This led to the development of the Industry Foundation Classes (IFCs) in the early 2000's which has evolved a great deal since. The IFC's soon became an international standard and is now widely adopted by professionals.

The Industry Foundation Classes (IFC, ISO/PAS 16739) schema was initially developed to ensure a standardisation of data transfers between different disciplines involved in the construction industry (Zhang et al. 2013). The schema has expanded over the last decades and has gone through several versions, being constantly adapted to industry needs worldwide. It is capable of capturing data concerning any building element type, and is a powerful tool in structuring building data and meeting industry interoperability needs (Berlo et al. 2015). The IFC is based on the EXPRESS language which was developed and standardised with the specific purpose of modelling product data concepts; it is a language of high expressivity which has enabled optimal storage capabilities. IFC offers a good degree of interoperability between design tools due to this, but it is also rather unique as the EXPRESS language is not used outside engineering domains.

Even though the industry is now technologically advanced, practitioners often feel overwhelmed by the diversity of tools being included under BIM, thus complicating the interoperability problem, as more and more design and knowledge domains are described and used digitally. Collaboration processes are time consuming due to the vast amount of information that needs to be created, analysed, checked for validation and delivered down the supply chain. This suggests an increasing need for automation of information processes and design protocols.

2.2.2. BIM-based evacuation design

Evacuation design using BIM has been a subject of significant attraction to the research community over the last decade, as it brings more dynamism to the model view of the data. Unlike static building elements, this extends the building context to human behaviour, bringing new ways in which digital models can contribute to the evacuation design problem. Several application domains for BIM-based evacuation were identified from the literature: virtual reality, path-finding, regulations checking and interoperability.

Virtual Reality (VR)

Rüppel and Schatz (2011) began investigation into fire-fighting virtual games, which imported and reconstructed a BIM within the gaming environment. A very similar methodology is adopted by Wang et al. (2013) and Wang et al. (2014), which simulates people using VR to evacuate the premises of a building and trying to track their behaviour, and also by Motamedi et al. (2016) trying to identify the best places for building sign placement within a design context. These developments however fail to implement the inclusion of realistic crowds, being simulated with human actors alone. The investigation of crowds and people’s movements using VR can be cheap to implement and can provide enhanced 3D environments. However, the main limitation of game environments is that they require validation to be accepted and used in realistic design scenarios (Kinateder et al. 2014). Additionally, a game environment is fundamentally different in how the model is computed. A CST will calculate the model mathematically at different time steps and replay the calculations, without the need for human interaction during the calculation process. By contrast, a computer-game will evaluate calculations at each frame time, and human interaction/input is often required as part of the process, which can also influence the progression of events in unknown ways. The main benefit of game environments is the experience of ‘real-time’, but it can be subjected to the engine capability and limits the environment to small-scale simulations. In addition to the factors above, VR applications are more concerned with visual and immersive experiences for the user side, often neglecting or being un-able to represent realistic human behaviour interactions with as much fidelity as validated CSTs.

Path-Finding (PF)

Chen and Huang (2015) developed a method for creating evacuation routes out of a BIM model. H. Lee et al. (2016) proposes an extension to the IFC schema with a 'path' concept for circulation purposes in a BIM design context. Chen and Chu (2016) developed a graph method for aiding evacuation in buildings by calculating the most efficient routes. Isikdag et al. (2013) presents a methodology to use BIM sematic level data for indoor navigation models considering IFC as the source of information. The calculation of the shortest path out of a building can be useful in design situations, but it does not consider the complexity of human behaviour, so these approaches do not allow for a realistic estimation of a travel time as defined in Section 2.1.1.

Regulations checking

Malsane et al. (2015) try to identify the requirements of integrating simulation safety tools and regulations. The scope of the research is limited to regulation in England and Wales, but it discusses in detail the level of knowledge formalisation and concludes that there is no overall consistency on how fire sub-system rules are addressed. Fire design is a very complex problem to solve due to the multitude of sub-systems that require audit and their inter-dependencies. The authors further state that with the use of the IFC standard, regulation formalisation should be more object-oriented, and thus more specific and easier to assess. However, due to the complex nature of describing regulations, IFC alone cannot encapsulate all the necessary information for valid performance and rules-compliance audit. These sort of methodologies have existed for some time, although not employing regulations from the UK. Lee (2010) created a framework for evaluating circulation rules a specific building type, using IFC concepts. Choi et al. (2014) adapts a model for high-rise regulation checking for prescriptive evacuation rules. Kannala (2005) proposed a similar way to assess building regulations using Solibri Model Checker plug-ins, based on IFC models. They use several algorithms to identify spaces and their connectivity. The studies above present methodologies limited to prescriptive rules, not incorporating CSTs.

Dimyadi et al. (2016) presents a system which relies on IFC model data and user input, which is compared against a Regulatory Knowledge Model consisting of the design rules applied to the process. The research checks output from multiple tools to assess fire safety performance of building designs and is IFC focused. Although a good step in the right direction, the process of integrating the information is not collaborative enough for more holistic design views or across the BIM lifecycle stages. These limitations are also mentioned by the same authors in another study (Dimyadi et al. 2015), where they recommend using ontology languages to express regulatory knowledge, due to higher expressivity and interoperability.

BIM interoperability

A number of studies are focused on integrating crowd simulation tools into various systems: Jalali et al. (2011) integrate three different domain tools together for fire evacuation management scenarios; Wang et al. (2015) use BIM platforms to provide building environment information into a system that performs calculations of escape routes and connects to a fire simulator; the authors present a sophisticated system using several tools to compare results across different design perspectives. There is no consensus on data exchange formats in these studies, but they regard BIM as the source of information. However, no use of IFC is mentioned, and the BIM data imported is limited to geometry.

With a clear focus on IFC, Wang and Wainer (2015) developed a cloud service evacuation design tool which uses different algorithms to calculate movement of agents. Marzouk and Al Daoor (2016) present a case study and analysis of using the MassMotion CST which simulates the evacuation of workers on site during a construction stage. The study also outlines a framework of using BIM information and tools in the process, mainly through using the IFC format. However, the use of IFC is limited to geometric components.

Many of the studies discussed above rely heavily on IFC, but still face difficulties when expressing rules and regulations on top of building models when trying to evaluate the performance of a design. Despite these attempts, a gap in the interoperability layer between BIM tools and fire safety tools is evident, with no common methodology or information transfer protocols, as is also pointed out by Wang and Wainer (2015). While IFC is the best option for storing structured data, it is less likely to meet the needs for inter-disciplinary design processes, when in the context of performance assessment for fire evacuation. In addition to that, the studies have expressed less interest in conceptualising and representing the factors which are the indicators of fire design performance or how they can be used in the context of automation. Very few studies have attempted to explore or extend the interoperability with commercial CSTs used in industry, preferring to develop their own tools instead, due to cost related issues. On the other hand, many CSTs used in practice offer very good IFC import, thus making them BIM compatible. Finally, no study investigated the interoperability with BIM beyond geometric information, which is insufficient for CS purposes, considering that valid simulation models require input from various other sources (contextual information), as was outlined in 2.1.2.

2.3. The Semantic Web and Linked Data paradigm

The term ‘Linked Data’ (LD) is a concept developed under the efforts of the World Wide Web Consortium (W3C) which enables the use of data more intelligently across the unstructured internet resources. With increased levels of semantics, LD represents a powerful tool towards increased ‘meaning’ of data on the Semantic Web (SW), as is shown in Figure 2-4.

“Semantics is a discipline dealing with the meaning of linguistics signs or symbols, that is, the words, expressions, and sentences of a language. […] In semantics, the language whose meaning is discussed is called the object language, while the language that is used to talk about the object language is the metalanguage. For example, in the sentence, ‘Snow is white’ is true, ‘Snow is white’ is in the object language, while the whole sentence is in the metalanguage” (Bunnin and Jiyuan 2004). The vision behind the semantic web is to create the next generation of the World Wide Web (WWW) where information is automated with the use of intelligent systems and software agents able to better interpret the data. LD is ‘machine interpretable’ and can be used by intelligent software systems to perform various operations on it, greatly increasing the capability of information retrieval. This can bring great benefit to design disciplines, with the primary condition being to express AEC relevant data into a semantic web language.

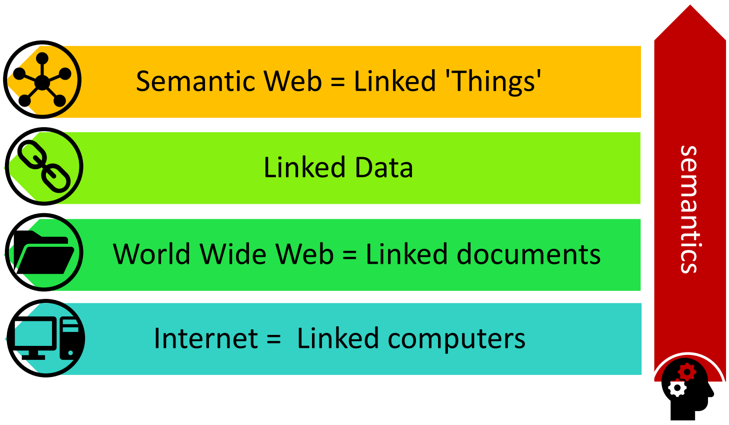

Semantic Web = Linked 'Things'
Linked Data
World Wide Web = Linked documents
Internet = Linked computers
semantics

Figure 2-4. Increasing levels of semantics for data on the Web

2.3.1. Semantic web languages – RDF to OWL

“The Resource Description Framework, or RDF, is a knowledge representation language for the Semantic Web, and is used to express knowledge about things both on and off the Web; RDF can be used to write metadata about web pages and to describe real-world objects with equal facility.” (Gibbins and Shadbolt 2009)

Information presented on the SW is represented as a graph of nodes and edges. Nodes represent things or values, while edges are properties which link two other nodes together. This represents the fundamental unit which is used for knowledge representation, and it is commonly referred to as a 'triple', which follows a pattern similarly to natural spoken languages: 'Subject -> Predicate -> Object' (SPO), as shown in Figure 2-5. The SW uses Uniform Resource Identifiers (URIs) (Masinter et al. 2005) to store and refer to 'things' and more specifically to RDF, Internationalized Resource Identifiers (IRIs), which act as an extension to URIs when defining namespaces (Dürst and Suignard 2004). This allows the definition of data concepts and knowledge on the Web using a standardised address to make it accessible and avoid conflicts.

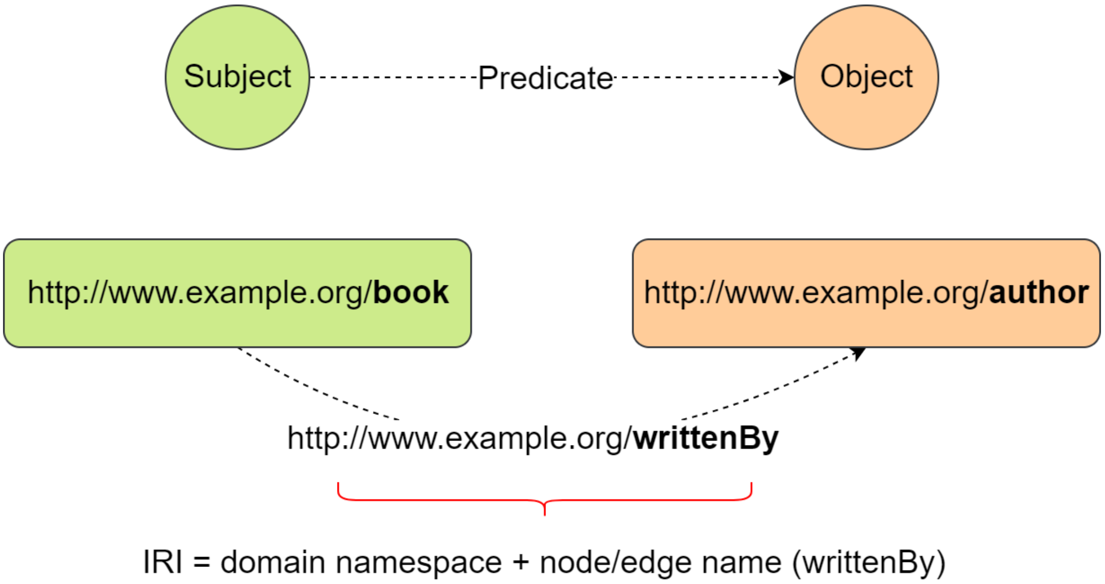

Subject
Predicate
Object
http://www.example.org/**book**
http://www.example.org/**author**
http://www.example.org/**writtenBy**
IRI = domain namespace + node/edge name (writtenBy)

Figure 2-5. Example of an RDF triple following the SPO pattern

RDF acts as the foundation stone for representing SW data with other semantic vocabularies being based on it, each with an increased level of semantic expressivity:

1) RDF Schema (RDFS)
2) Web Ontology Language (OWL)
3) Semantic rules: Sematic Web Rules Language (SWRL), Rule Interchange Format (RIF), SPARQL Inferencing Notation (SPIN).

SPIN is essentially a formalisation of rules based on the SPARQL Protocol and RDF Query Language (SPARQL) which is the preferred query language to access and manipulate RDF graphs. Detailed specifications on these concepts are available online at https://www.w3.org.

One of the most widely used ways to represent knowledge is through OWL ontologies, which are based on RDF but include many other logical operators and axioms, enabling very rich conceptualisation of data, information and knowledge alike. In this sense, an OWL model is able to work with basic data types, such as integers and strings, which are given more context when part of information models. These in turn can express relationships between different concepts (as shown in Figure 2-5), which can also achieve the conceptualisation of a knowledge field in an ontological sense. The term 'ontology' comes from ancient Greek philosophy and it *"deals with the essential characteristics of being itself (of Aristotle's being qua being), and asks questions such as 'What is or what exists?,' 'What kind of thing exist primarily?' and 'How are different kinds of being related to one another?'"* (Bunnin and Jiyuan 2004). In general terms, ontologies define 'things' which exist, while semantics characterise the relationships between these 'things' and or how to describe them.

The Oxford dictionary of Computer Science defines a programming ontology as *"a description of some concepts and their relationships, for the purpose of defining the ideas sufficiently to allow a computer to represent them and reason about them. Thus an agent's ontology specifies the basic building blocks of knowledge that defines what it can perceive and reason about. This is a kind of model and, as such, is very useful to define what agents or learning programs can know and what they can communicate. Ontologies are usually compiled for a particular 'domain', e.g. the domains of wind engineering, medical diagnosis, or office interior navigation, but they are more formal than domain knowledge."* (Butterfield and Ngondi 2016)

Once model concepts have been described, their inter-relationships need to be defined, which give a comprehensive representation of the model, not just semantically, but also ontologically. An *"axiomatization process aims at enriching ontologies semantics by the definition of axioms and rules between different entities. It is processed manually by experts of the domain. The axiomatization can be applied between entities of the same ontology, intra-ontology, or belonging to various ontologies, inter-ontology. Moreover, axioms can be defined for concepts and properties. However, the axiomatization process is performed through the high level expressiveness of OWL and the use of SWRL to define formally more complex relationships"* (Abdul-Ghafour et al. 2014). The relationships expressed in OWL are on a higher level than those in RDF or RDFS, and can define very specific terms which act as necessary requirements, restrictions over model concepts due to the use of Description Logics (DLs) (Gibbins and Shadbolt 2009).

Because of DLs, OWL is widely applied for practical purposes in various knowledge domains where data is categorised and analysed logically such as biology, medicine, geography, astronomy, agriculture, computer science, etc. (Motik and Rosati 2010). The applications usually deal with large datasets which require classification and conceptualisation of knowledge. This is also valid for the AEC sector, where multiple knowledge domains interact frequently. It does, however, pose challenges because construction projects have multiple organisations involved, which tend to collaborate for short periods of time (while the project lasts). When referring to large organisations, Hay (2006) mentions that they have begun to see the value in the semantics of all their systems and information, in that semantics allows people and software systems to better communicate with each other.

When in the context of BIM models, semantics lie at the core of all its objects, and in fact every object inside the model symbolises a real-life object, which eventually becomes a building component. The simplest example of these semantics is the properties which are attached to the programmatic objects: ‘Wall of 3000mm length’ or ‘Wall of concrete material’. Unlike IFC which is at its core structured model data, an OWL format adds more expressivity to the data. The use of ontologies in the AEC has gradually increased, with application domains in cost estimation (Niknam and Karshenas 2014) and risks analysis (Fidan et al. 2011) or energy performance (Tomašević et al. 2015).

2.3.2. Ontologies for building design

Succar (2009) introduced a BIM ontology of intersecting knowledge domains in an attempt to define conceptually the main fields and lenses of the BIM paradigm. Abanda et al. (2013) offer an overview of ontology and semantic web linked data trends in research over the last decade. There is clear interest in the fields of risk analysis, project management knowledge sharing and energy performance analysis. The authors mention that SW and LD are seen as beneficial because they facilitate interoperability between the large spectrums of application domains involved in the construction sector. However, they point out that very few applications exist commercially which are using ontology support. This is likely due to complex requirements for ontology-based collaboration in the field of design and construction. The study also identifies several research applications in energy performance analysis and building sustainability in general, but there was no mention of fire design performance analysis. This suggests a low level of research and development in the area.

From IFC to IfcOwl

Pauwels et al. (2011) is one of the pilot studies investigating the capabilities of semantic web rule checking, applied to acoustic building design, closely tied to IFC concepts. They state that the limitations in the IFC schema expressivity of concepts are overcome by an ontology approach. Another pilot study on using ontology tools is by Scherer and Schapke (2011), which describes a framework for using ontologies as a means of integration on the project level, which can include multiple models and processes. Such approaches enable the rule checking process to go beyond the schema scope, thus allowing for more flexible model view definitions, which is crucial in including non-traditional design analysis under the BIM umbrella. Long before these developments, Rűppel et al. (2006) proposed an ontology model framework for fire safety design, integrating different databases. This study was limited at the time due to insufficient technologies in the AEC sector. However, many developments today rely on IFC, which is seen as an underlying schema for structuring data, and IfcOwl (Beetz et al. 2009, OpenBIMstandards 2017b), its ontology representation , which provides higher level interoperability and reasoning capabilities. Schevers and Drogemuller (2006) pioneered a mapping between IFC to OWL to extend its interoperability capabilities. As developments around this topic grew, it became apparent that ontology representations of the IFC schema allow for a flexible and more robust backbone for interoperability requirements, as concluded by Venugopal et al. (2015). The computer-interpretable features of ontologies allow for validation methods and easier extensibility of other disciplines into the design process. However, this presents serious limitations when querying geometry data due to the object-oriented nature of the IFC schema. Pauwels et al. (2017) investigate the optimisation issues around its representation in terms of geometry retrieval of the data. Farias et al. (2015) also mention that the IFC STEP file was created for optimal information compression, but its object-oriented nature does not really align the same way semantically when represented in an ontology. Terkaj and Šojić (2015) also aim to improve the semantics of the IfcOwl, to make it more adaptable and robust over different application domains. The IfcOwl is currently under the process of becoming an international standard (BuildingSMART 2017), which would open the way towards more Web reliant BIMs.

Building regulations checking

Some studies represent certain regulations into ontology concepts and logical rules in order to facilitate a fast and automatic environment. Beach et al. (2015) is one of the more recent studies which applies regulation checking using ontology representations due to it being easier to manage and having a more interoperable environment compared to traditional software tools. The study focuses on presenting a more viable way to quickly convert textual rules and procedures into valid ontology representations and checking. The study was applied in the context of BREAM assessment, which is a good example of multi-disciplinary and multi-domain design decision making. The authors mention that when the SWRL rules are executed, the rules check only for failure case, thus suggesting to the users why it failed. This is a limitation of the Open World Assumptions (OWA). The users also have to complement missing data with input in many situations. A step further from this, Zhou and El-Gohary (2017) present a method which semi-automatically extracts information from design codes in order to facilitate the code-compliance schema against which models should be checked. However, this study is limited to the energy analysis domain. This could really speed up the process of interpreting design rules and regulations for automatic information retrieval.

Design applied research

Some good examples of developments using SW tools are presented by Lee et al. (2014) and Niknam and Karshenas (2014) for cost estimation and management of data; the latter uses SPARQL queries to integrate data over the web, such as industry suppliers cost data. Zhang et al. (2015) developed an ontology for hazards and safety, also using SWRL rules for more effective safety planning within the context of automation.

Another example is the development proposed by Grover and Froese (2016) to manage knowledge about buildings via a social platform. Although this study does not mention an integration with SW tools, it shows the direction of the industry towards smart homes and cities, where managing and exploitation of data is a requirement to truly benefit from it (Howell 2017).

Crowd simulation and human behaviour

The use of ontologies for human behaviour was explored by several studies in attempts to conceptualise realistic behaviours and were used on virtual agents, rather than CSTs (Vieira et al. 2005, Yoke et al. 2007). These studies, however, are not focused on design or evacuation scenarios, and have quickly become replaced by improved artificial intelligence agents within virtual game environments.

Kuligowski (2016b) aims to conceptualise the complexity of human behaviour and the types of actions they may take in real cases. Although these cannot be fully represented by any CSTs to date, they can be captured in ontology models. Trento et al. (2012) present a methodology to incorporate human behaviour in assessing building performance and usage by capturing this in an ontology. However, this is beyond the rules and regulations for design compliance and does not address the requirements for using BIMs in practice.

Onorati et al. (2014) is an example of using ontology methods for aiding the evacuation process, whereby ontology and semantic web technologies are used in the building operation stage. Damrongrat et al. (2013) proposes an ontological representation of the building plans, according to different functionalities so that evacuation events can be represented more comprehensively. Poveda et al. (2014) uses ontologies and ambient intelligence to gather knowledge about how evacuations progress in a building. Kraus et al. (2011) is an example of using building information defined in ontologies for an airport, with sensors. Mustapha and Frayret (2016) developed a framework to simulate agents paths in healthcare buildings for optimising building usage. Luo et al. (2016) present a methodology of using ontologies to manage events during fire, creating and using a knowledge base about the building in conjunction with BIM models. All the studies above are heavily focused on the building operation stage monitoring and simulating human activities, many seeing BIM as a source of geometry, but not applied in a BIM supported design context which collaborate with CSMs or CSTs, or follow validated design procedures.

2.4. Summary of literature findings

This chapter introduced the basic concepts used as part of the conducted work and offered an overview of the status-quo of research into the fields of CS, BIM and ontologies. The estimation of evacuation time is a complex process which relies on CS models and tools in practice (Section 2.1). This brings forth the first findings:

1) Section 2.1.1. outlined that each CSM and CST is different and consequently may output different results. Using multiple models and tools or selecting the most appropriate for each situation is recommended in light of each tool's limitations.
2) One simulation is often not sufficient to evaluate the evacuation performance. Creating relevant scenarios involves a lot of effort from safety engineers, requiring multiple information sources about the environment and the population, following validated procedures. This is a time-consuming process which requires specialised expertise, as was mentioned in Sections 2.1.1 and 2.2.2.

There have been many attempts to speed up the process using BIM models on various levels across multiple fire safety related fields. The level of interoperability between CS applications and BIM has been summarised over several fields including virtual reality, path-finding, regulations checking, and interoperability focused. However, this brings forth the following findings, outlined in Section 2.2:

3) There is no consensus on the information exchange requirements from BIM to the CS field. Most of the developments in research are limited to importing or re-constructing geometry from BIM, with no consensus on a format.
4) Additionally, geometry is insufficient to provide all the necessary information when creating a simulation model with regard to context (population capacities, placements, incidents, etc.), which needs to be constructed manually by designers, as was evident from section 2.2.2. This is also caused by the various distributed sources of information which can contribute to the context of a CSM, also outlined in Section 2.1.2.

More recent developments suggest the involvement of OWL ontologies to express human behaviour and integrate various models. The more relevant research in this field was presented, including the development of IfcOwl, which allows for extending digital building model data to other design fields. Section 2.3. outlined that:

5) The Semantic Web and ontologies provide a robust environment for integrating information from distributed sources, which has already seen significant development around the BIM field, clearly showing their potential for improving automation of design using intelligent systems and logical inferencing or reasoning.

Some ontologies have been developed within the field of human behaviour in a BIM setting. However, as outlined in Section 2.3.2:

6) No ontologies with a focus on CSM or following validated workflows or procedures exist to date which can conceptualise simulation data and collaborate with BIMs.

These would greatly benefit the automation of design procedures and allow intelligent agents to find and reason over distributed resources on the web to facilitate fast and accurate construction of models, and the evaluation of building performance when considering larger data environments.

Chapter 3. System design and methodology

Building upon the research findings from the literature review, a novel system is proposed which enables the exploration of linked data and knowledge processing based on a BIM approach. The conceptual system framework is outlined, along with the adopted methodology for its development, coupled with the aim to pursue knowledge during the entire process.

The proposed system framework is based on principles of knowledge representation and mining. The proposed tools for representing knowledge are OWL ontologies, hence the name ONTOCS which stands for Ontology Crowd Simulation. Knowledge Mining is defined as *“a derivation of human-like knowledge from data and prior knowledge”* (Kaufman and Michalski 2005), which includes Databases, Knowledge bases and Operators, as outlined in Table 3-1.

Table 3-1. Knowledge mining components (Kaufman and Michalski 2005)

Component	Description
Databases	the raw data present across various sources of information
Knowledge bases	the representation of existing knowledge
Operators	logical expressions used to supplement additional knowledge from existing knowledge bases

Having adopted the recipe for a knowledge mining system as described by Kaufman and Michalski (2005), the conceptual framework components and workflow are shown in Figure 3-1. The workflow ensures correct user input (i), correct interpretation of the reasoning processes (ii) and that the users receive relevant feedback (iii). The scope of

ONTOCS is narrowed down to the field of crowd simulation evaluation in evacuation scenarios, its main aim being to enhance the performance design processes which rely on using evacuation models for decision-making. However, its extensibility to future needs or inclusion of additional design disciplines was also considered throughout its development.

Due to the complexity of fire safety design, the processes involved in CS construction and analysis were considered independently from other sub-systems. *"[…] to improve the quality of decisions is to decompose a decision problem into simpler components that are well defined and well understood. Studying a complex system built out of such components can be subsequently aided by a formal, theoretically sound technique."* (Druzdzel and Flynn 1999)

The decision-making process involved in CSM evaluation was investigated through existing design guidance and literature, the functionality that CSTs provide, and consultations with experts in the field on several occasions.

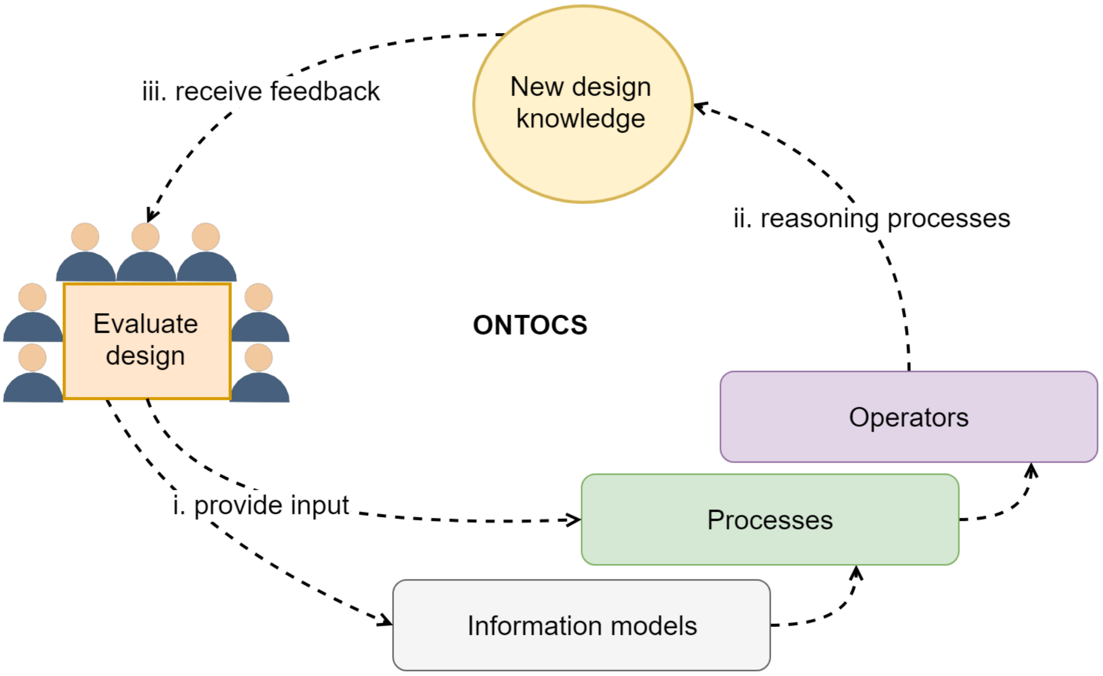


Figure 3-1. The envisaged processes for achieving knowledge mining using the proposed ONTOCS system

To further define the research direction with a focus on solving applied design and research problems, the ONTOCS system aims to achieve the following objectives:

1. The system must be able to interface with several tools and information sources, without the risk of being locked into a particular CST.

2. The system must enable automatic creation of simulation models using available data and information models, while also considering user input.
3. The system must enable feedback on design performance using simulation data and subjecting them to knowledge operators, whilst also considering user input.
4. The system must be fully functional and deployed in a practical use-case scenario for testing and validation.

The objectives for deploying a functional prototype for testing allows a parallel pursuit of knowledge about the adopted research approach, and this intersection of interests is outlined in Figure 3-2.

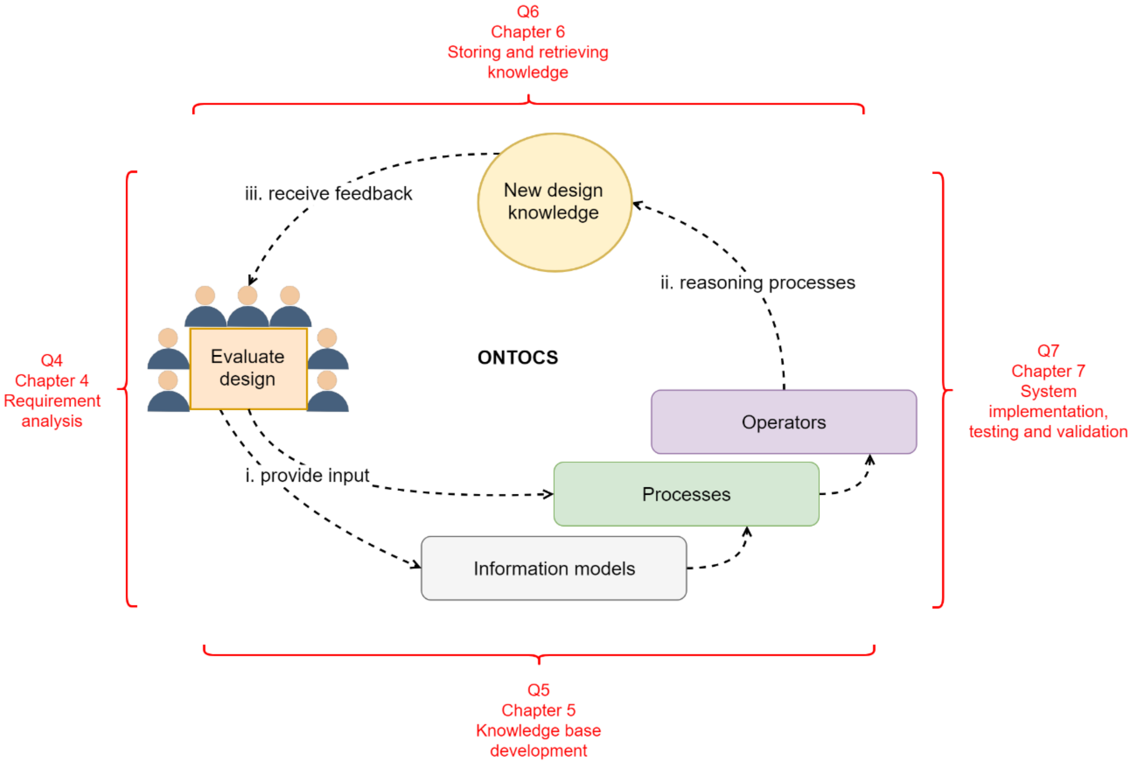


Figure 3-2. Pursuit of knowledge in parallel to system design and testing

The remainder of the research was split into four main sections, each focusing on a research question (Q4, Q5, Q6 and Q7). A practical approach was adopted, where existing tools and methods were investigated and tested in parallel with further literature surveys, and consultations with field experts where necessary.

Q4: ***What are the requirements for an intelligent system capable of integrating resources relevant to the CS field within the context of automation and analysis feedback, whilst considering practical deployment and future extensibility?***

The research methodology adopted for Chapter 4 is an extension of the literature whilst considering the envisaged proposed solution. Setting specific objectives for the desired functionality of the ONTOCS system also helped define the boundaries of the overall research. The requirements investigation considered review of relevant literature around design practice and industry tools for evacuation design. Understanding of practical design problems and methods used to tackle them was crucial in developing a functional system which follows well defined design protocols. A hands-on approach was taken in testing several tools and exploring their capabilities and limitations before deciding on which to incorporate into the prototype.

The methodology and rationale of this chapter breaks the research question in several parts, by following ONTOCS system development objectives:

i. The main objective is to be able to interface with several tools and information sources, without the risk of being locked into a particular CST. To ensure future interoperability and extensibility, a common taxonomy of concepts across CSTs was required. Several popular industry tools were chosen, their features and structures investigated and compared. This step was also important for the implementation of the CSS ontology in the next chapter;

ii. The second and third objectives of the ONTOCS system is to enable automatic creation of models (1) and feedback on design performance (2). Official published documentation from the UK was surveyed, in addition to academic papers and consultation with experts. This resulted in the definition of several information requirements for each stage. These factors were vital for the development of the ontologies from Chapter 5, and for the definition of the case study presented in Chapter 7.

iii. The final objective of the ONTOCS system is to allow its deployment in a practical use-case scenario. This required the consideration of tools used in industry for: crowd simulation modelling (MassMotion), using a digital building model as an information provider (IFC), a knowledge modelling and testing tool (Protégé) and hosting a knowledge management server (Stardog). These were initially investigated from an academic background and their features and capabilities tested to justify their inclusion into the system and into the research framework;

Q5: ***What are the challenges concerning information models and workflow processes being represented in a knowledge base, considering the requirements for integration and knowledge retrieval?***

The research methodology adopted in Chapter 5 involved an iterative ontology engineering approach, where the requirements for a knowledge-based system for CS performance design are developed into a knowledge base.

"1) There is no one correct way to model a domain – there are always viable alternatives. The best solution almost always depends on the application that you have in mind and the extensions you anticipate

2) *Ontology development is necessarily an iterative process.*

3) Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain"

Noy and McGuinness (2001)

Several main ontologies were developed which define the processes involved, and additional secondary ontologies for integrating external resources which contribute to model data. The ontology language chosen was OWL, working with the OWL2 version schema, to be able to give as much expressivity as possible. Each ontology was developed in parallel to the ONTOCS system and tested along the way. Adjustments were made in an iterative manner while also considering feedback from discussions with industry experts in the field on several occasions. The challenges for an integration in a fully functional knowledge base are presented in an attempt to align multiple knowledge domains. The alignment was done via widely accepted methods, such as matching concepts on the basis of their similarities (Euzenat and Valtchev 2004), or in some cases with the use of knowledge rules, where datasets were too large for manual methods.

Q6: ***What needs to be considered for design knowledge storage and retrieval concerning building egress performance using evacuation models?***

The research methodology adopted in Chapter 6 involved an iterative process of creating and testing different knowledge operator types which were able to reason on top of the already developed knowledge bases. Due to the nature of a knowledge base being reliant on several distributed resources, this brings a certain degree of complexity. To tackle this, a practical approach was adopted, which was discussed in pursuit of insight on this topic. The main aim was to assess what resources are needed by the ONTOCS system in order to perform useful logical operations on model data, while considering user input, building geometry and scenario context. The nature of storage and retrieval of knowledge base information are strictly linked. Operators needed to compute the relevant data had to be defined in a very specific context to reason correctly. The SWRL rules were developed in their relevant ontologies and initially tested in the Protégé software. They were then implemented into the system which relied on the Stardog server service to process them. SPARQL queries were chosen to check the validity of the rules, by interrogating the knowledge base manually and later programmatically, with all the necessary resources being provided on one local RDF store (database). All operators were checked, tested and gradually improved, which were then employed for a more realistic case study to identify further limitations.

Q7: ***How reliable is a knowledge-based system in understanding the building model and other linked data resources in facilitating correct and efficient design support?***

The research methodology adopted in Chapter 7 employs several use case scenarios where the developed system and knowledge base were tested, with the aim of validating them. The case studies were developed on a digital model of a real building in its operation stage. Data about the use and occupancy of the building was gathered, which was used to create a realistic design scenario and then simulated using MassMotion. The testing considered each stage independently:

i. Stage I, Scenario generation – the first use case compares the manually created scenario to those generated automatically by the ONTOCS system, in an attempt to assess if the system is able to correctly interpret the ontology resources using the proposed workflow and knowledge operators defined in Chapter 6. The differences between geometry and context creation were outlined concerning the model structure and comparing simulation results across several scenarios. Referring to methods on CST validations (Thalmann et al. 2007) states that *"Quantitative verification involves comparing model predictions with reliable data generated from evacuation demonstrations. Galea's work highlights (Galea 1998) two kinds of quantitative validation: historic and prediction-based validation. In the first case, the user knows the results from previous simulations and real exercises. The second case involves using the model to perform predictive simulations prior to having experimental results."* It is important to mention that the methodology here is not targeted at validating the CST, but at validating that the models generated by ONTOCS, which are then executed by a CST, resemble sufficient similarity to conventional ways of model creation. This is done by looking at the inputs and output results by comparing and contrasting their differences;

ii. Stage II, Analysis feedback – the second use-case tests the system on dealing with a large-scale of scenario models. A session running 36 models was used to validate that the rules and queries are able to correctly inform the designers about building performance. The results provided by the queries were checked against those present in the simulation files. The number of 36 scenarios was considered sufficient for a real-case design situation, simulating the building in incremental steps regarding its assumed population from 30% to 200% according to design practice.

During the cases above, the execution speed of knowledge operators was measured for both use cases by considering query time, thereby assessing the efficiency of the system. Additional scalability tests were carried out to identify limitations of the developed knowledge operators. Several dozens of measurements were taken to account for oscillation of query times and averages were plotted and discussed.

To assess the reliability of the use of secondary linked data resources, for each use case above, scenarios based on design guides capacities (using the UKSOC and Uniclass2015 ontologies) were also included. These are contrasted and compared to models which rely on real case data.

Chapter 4. Requirement analysis

This chapter outlines the technological and information requirements for delivering a functional system fit for practical deployment, following the vision of the proposed conceptual framework. The chapter is divided into three distinct sections, following the rationalised methodology, with the aim of answering Q4.

4.1. Design of crowd simulation tools

Crowd simulation models have been introduced in Section 2.1.2, along with their capabilities and limitations. From a research perspective, the last decades saw the development of mathematical models which can mimic human behaviour in various circumstances. These models were gradually adopted by the industry for practical uses. The most common use is in evacuation design, where scenarios are created to represent the act of building egress. It is important to understand that while the basic underlying mechanics of these models have changed in small proportion, the functions they perform and the way they are implemented have evolved in order to satisfy industry needs. The more recent AEC industry needs have been around a BIM-centric way of working, which is in line with this research' aim to include CSTs into the BIM paradigm. As such, tools with high interoperability with BIM models were considered more valuable.

A number of CSTs are available in industry and research, with various features that they provide to users. It was mentioned in Chapter 3 that in order to be able to interface several CSTs to interact with the ONTOCS framework, a survey of their basic functionality and features is required. This can then be used to establish a baseline of common functionalities and concepts used in the field of CS. Several CSTs which are widely used in industry were investigated and are these are shown in Table 4-1. All the investigated tools have been in development and improvement for the last decade, each receiving significant feedback from their users. Additionally, each tool was validated using commonly accepted validation techniques (Thalmann et al. 2007, Duives et al. 2013), and have been used on real-life projects on many occasions. Experts consider the validation process an ongoing one across the lifecycle of the software tool. Duives et al. (2013) mention that calibration of the model to represent a scenario in detail has a greater impact on the realism and behaviour than other methods of improving the tool, second only to the mathematical calibration.

It has been observed, as will be made clearer in Section 4.1.2, that the underlying topology of such models can be clearly distinguished across all of the investigated tools, regardless of the mathematical model used.

Table 4-1. List of CSTs investigated

Tool	Developer	Reference
Exodus	GUEL, University of Greenwich	FSEG (2018)
STEPS	Mott MacDonald	Mott MacDonald (2018)
MassMotion	Oasys, Arup	Oasys Limited (2018a)
Pedestrian Dynamics	INCONTROL Simulation Solutions	INCONTROL Simulation Solutions (2018)
Simulex	IES	Integrated Environmental Solutions Limited (2018)

4.1.1. Features and capabilities

The overall purpose of CSMs is to accurately represent human movement and complex behaviours. However, as these models becomes part of software tools, each tool comes with several features which extend the functionality of their underlying models. By investigating the CSTs in Table 4-1, several common features have been identified, which are summarised in Table 4-2.

Kang et al. (1990) have undertaken a study which defines and categorises how a software tool is able to incorporate features which can solve user problems. In this case, they define a software feature as: *“a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems”.* In essence, a software program is able to incorporate such features by applying code in very specific ways, used to solve common problems or to achieve a specific task. When considering the program itself, Batory (2005) cited in Apel and Kästner (2009) describes features as *“an increment of* *program functionality",* thus allowing one particular piece of software to achieve several functions, as mentioned above.

Table 4-2. List of main CST feature categories

Feature domain	Description
Geometry	Allows users to create geometric representations of the environment using basic types (lines, points, etc) or place pre-defined objects and edit them (floors, walls, etc). Various 3D or 2D model formats can be imported directly.
Agent	Allows users to define agents within the model, usually relating to their physical or behavioural properties.
Event	Allows users to specify the location, time and number of agents within the model. Additionally, users need to specify the routes or other agent actions at specific times during a simulation. Events govern the dynamics of a simulation by defining agent actions, routes and behaviours, which need to be pre-inputted by users.
Analysis	Allows users to work with output data and perform analysis and decisions on the performance of simulations. Includes several ways in which agent movement in the environment is tracked and presented, such as graphs, tables or density map overlays.
Visualisation	Includes several types of features which enhance the visual components of objects but have no impact on the simulation calculations. Common features also allow users to visualise the simulation as an animation or video.

The importance of identifying CSTs features is owed to the fact that they were deliberately created to achieve well defined functions. Apel et al. (2008), cited in Apel and Kästner (2009) also defines features as *“a structure that extends and modifies the structure of a given program in order to satisfy a stakeholder’s requirement, to implement and encapsulate a design decision, and to offer a configuration option”*. Program features implemented in these tools showcase the wider needs of the industry, and therefore reflect the way the industry uses them. As an example, the addition of density maps to CSTs is a feature which allows experts to analyse the flows of people through a building with more objectivity. Thus, their implementation and inclusion as a feature showcases its role in the design process.

Table 4-2 shows the common types of features encountered by testing and investigating the manuals of the 5 tools mentioned in Table 4-1. These are sufficient to define any situation within a built environment, with very specific types of populations, allowing a realistic representation of real life events. Each CST offers different interpretations of the same concept, in many cases under different names, but these can be objectively categorised by defining their functionality into one of the 5 main groups.

It should be noted that although ***evacuation*** design is the most applied case, CTS have various features to simulate other use-cases as well, such as a ***circulation*** mode. In ***evacuation*** mode, agents are meant to immediately head for the exit. For a more realistic scenario, some groups of agents can be programmed to delay this action, or react differently when an evacuation event triggers. The software investigated also provide a ***circulation*** mode, where agents are given certain routes or itineraries to follow. Circulation modes are used to model an expected daily use of the building. This use-case is considered out of scope for the current research.

Concerning interoperability, all tools can import geometric models from various formats, making them BIM compatible. However, they present serious limitations, primarily due to the differences between the BIM and CS domains and the concepts they use. Characteristically, CSTs compute the events on 2D environments, but are presented in 2D and 3D for the users for better visualisation options. In the cases of Exodus and Pedestrian Dynamics, any import from an IFC model for example would only require a cross section plan view of the model. In this case all semantic information and inter-relationships between the initial IFC model objects are lost, as they are effectively converted into lines or points. In contrast, MassMotion and STEPS can maintain the 3D representations of the models, but none of the semantics or relationships between these objects is used or even required. Thus, they mostly fulfil a role for visualisation. This makes any attempts for round-tripping through a CST tool impractical at the moment.

4.1.2. Taxonomy of model concepts

In order to understand what lies at the basis of the crowd simulation domain, a more in-depth analysis was required. In the attempt to establish a base taxonomy of concepts used by these tools in the CS domain, an investigation of the classes of objects that the tools use was carried out, each belonging to one of the five generic features. A taxonomy of things in the field is important to create a topology of the knowledge domain for crowd simulation and analysis. This is the basis for a more comprehensive ontological representation of the domain, presented the next chapter (Section 5.1.1).

"Taxonomy is the science of classification. Originally, it referred only to the classification of organisms. Now, it is often used in a more general way, referring to the classification of things or concepts, as well the schemes underlying such a classification. In addition, taxonomy normally has some hierarchical relationship embedded in its classifications." (Yu 2014)

In relation to creating taxonomies in the field of CS, there are several studies which categorise crowds and audiences types (Vieira et al. 2005, Durupinar 2010, Zhou et al. 2010), specific behaviours of crowds or groups of people within models (Duives et al. 2013, Trento and Fioravanti 2016), or for fire-fighting (Moreno et al. 2011). It is important to bear in mind that these definitions are all heavily focused on the classification of agent behaviour from a research perspective, rather than on a classification schema for a CS model, with many concepts which define a CSM being omitted, but many overlaps exist relating to agent concepts. Thus, an investigation into the implementation of software tools applied in CS represents a good starting point to identify and defining these missing concepts. Considering the main concepts around the CS knowledge domain, they are usually presented in the form of features. From a programming perspective, these are represented as objects, due to the object-oriented nature of the tools. Not all features are represented in single objects. Also, it was impossible to establish whether the source code implements these features as objects or not, due to the lack of access to it. Features and objects were identified from software testing, reviewing the provided documentation, and on some occasions through consolations with tool developers and vendors. Figure 4-1 below provides a summary of the common features present in the investigated tools (full tables on these are provided in Appendix A). Some concepts apply functions in more than one category, however only the primary one is shown in Figure 4-1.

Rahm and Bernstein (2001) and Abdul-Ghafour et al. (2014) present ways for schema matching by finding similarities in software structures. "Similarity consists in computing a distance between two entities by comparing their components, i.e. all the features involved in their definition. These components reflect heterogeneities at different levels: syntactic, structural and semantic" (Rahm and Bernstein 2001). As opposed to this methodology, a more generic top-level view taxonomy is required for this research. This is in line with the aim to provide large-scale simulation capability, where a number of CSTs can be used to facilitate similar analysis needs in parallel. The reasoning behind this is to compensate for the limitation of each CST being inherently different (Gwynne et al. 1999, Ronchi et al. 2013), thus potentially outputting different results.

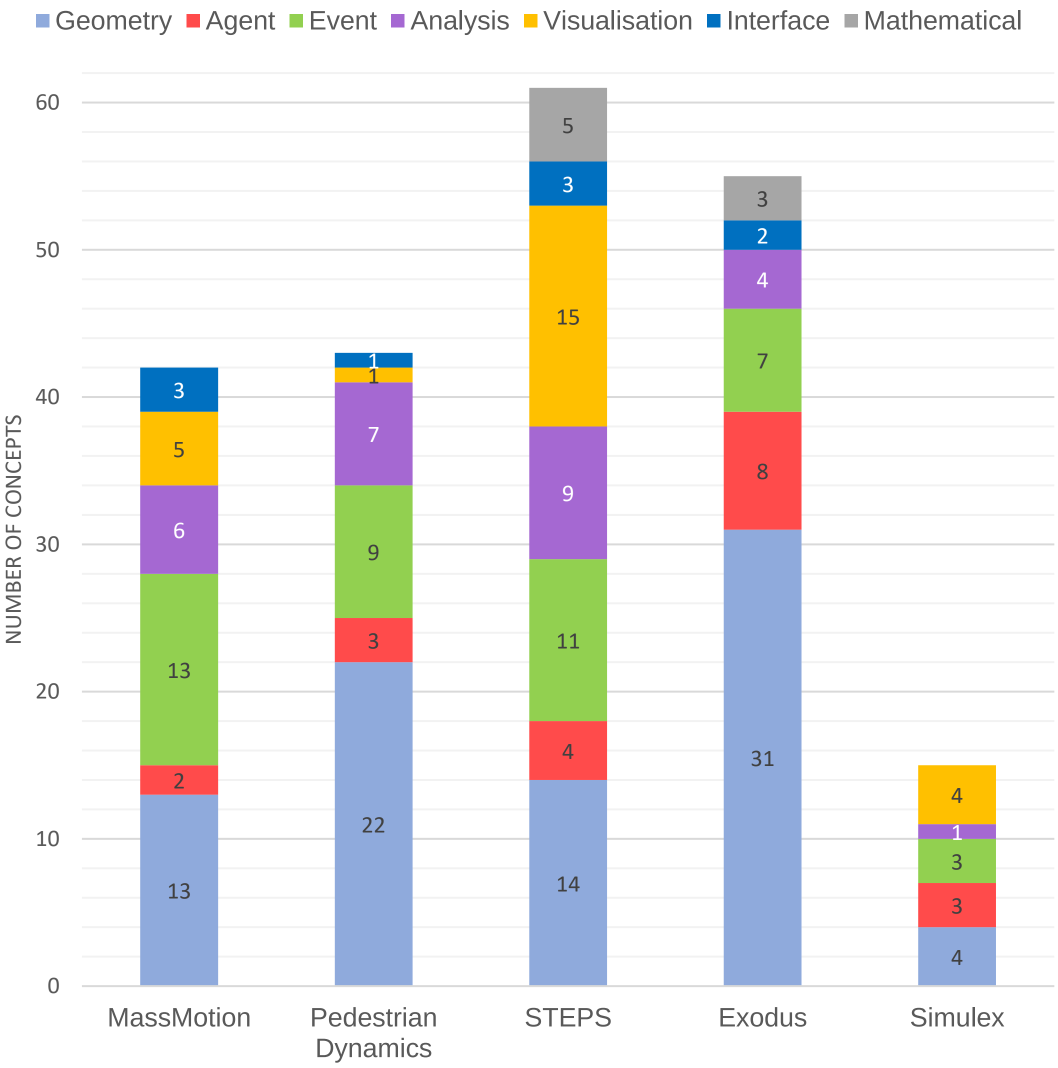


Figure 4-1. CST concepts by domain feature categories (plot data in Appendix A)

The Simulex tool stands out in Figure 4-1 with far less features than the rest. Being a module in the Integrated Environmental Solutions toolkit rather than a standalone application, Simulex offers limited range of features and objects, expressing only essential objects within a model. Nevertheless, its components are clearly distinguished within the categories, as expected.

All the tools have a consistent proportion of concepts, with exception of the ***Visualisation*** category, which includes features designed to enhance the view of the model and can therefore be more dependent on the level of tool development than other features. The

Analysis and ***Visualisation*** categories overlap in many cases (see Appendices A1-A5, fifth columns), due to the fact that the act of analysing a model involves users being able to visualise the output in various ways. The important factor which distinguishes the two is that ***Analysis*** concepts make express use of output data, whereas ***Visualisation*** concepts purely facilitate on geometric representation of model objects from all the other categories.

In addition to the categories outlined initially in Table 4-2, two more were identified: ***Interface*** and ***Mathematical***. ***Interface*** features are like Visualisation, dependent on level of tool development and having no impact on simulation output. The ***Mathematical*** category refers to features which allow users to alter the behaviour of the model or specify certain mathematical input which affect the simulation results. Not all tools give users this option, and so can be considered out of bounds of the common baseline of concepts. Additionally, these features are more concerned with the calibration of the underlying simulation model.

Considering the categories of ***Visualisation***, ***Interface*** and ***Mathematical*** as platform specific, the remaining four categories of concepts can be used to form a taxonomy around: ***Geometry***, ***Agent***, ***Event*** and ***Analysis***.

Geometry

Geometry concepts are used to define the building environment. These usually represent the surfaces on which people can walk, obstructions such as furniture or walls, and connections between surfaces such as stairs or doors. As shown in Figure 4-1, Exodus stands out with significantly more concepts from this category because it includes basic geometric types like points and lines in addition to complex shapes. The more complex object types have 3D representations able to define the environment very accurately. However, 3D geometry has no role in simulation calculations, as these are done on 2D surfaces, as mentioned previously. ***Geometry*** concepts can range from static building objects to dynamic lifts and in some cases even vehicles, which are able to move within the environment. Some concepts from the other categories also use geometric representations, such as avatars for agents, or overlay density maps for analysis objects. The most consistent types of concepts across all tools is the representation of static building objects.

Agent

Agent concepts are used to represent people in a simulation. This includes the definition of people’s physical properties such as: walking speeds, body size, bias towards turning left or right, etc. Most tools have predefined agent profiles which are based on research and are used to represent different types of people in real life. For example, some profiles can represent disabled people, with larger radius and slower movement speeds. Each profile needs to be calibrated to the specific tool so that it mimics reality to the fullest extent.

Other common concepts include the definition of entire groups of agents with similar characteristics, or the way in which to customise the population numbers using mathematical functions. BuildingEXODUS has more ***Agent*** concepts than the other tools and allows a more specific customisation of the population, possibly due to its extensive academic background. This allows users to tweak the agent profiles in more detail, which are otherwise pre-defined for its competitors.

It has been observed across the investigated tools that agents usually have various levels of freedom in terms of their behaviour. Agents are able to 'decide' which exit is best suited for them to take, depending on their familiarity with the layout or the level of knowledge they are provided about the routes and exits. Thus, the question on which exits are taken by agents becomes difficult to answer, as it may vary across multiple scenarios. Agents are also able to simulate a 'change of mind' and can decide to take an alternative route if the current one is blocked or too crowded. As such, a number of complex situations may arise as a result of these behaviours, which need to be considered for analysis purposes.

Event

The key characteristic about ***Event*** concepts is time. As they are dynamic actions which take place only during simulation calculations, events are triggered at specified times. These actions can be visualised using model animations. The most common event concepts rely on creating agents, thus populating the model at certain points in time. ***Events*** also describe the movement of people, such as moving from the origin point to the nearest available exit. Others act as triggers for changing agent behaviours, such as ticket gates, or dynamic obstructions for agents to avoid.

Analysis

Analysis concepts represent ways in which the model output data is compiled and viewed by the user. They allow users to understand the output and make decisions on the performance of the model. As shown in Figure 4-1, these concepts are quite similar in number, but will mostly rely on three main basic types: table, map and graph. The most common analysis objects are overlay maps, showing the density or congestion of agents in various areas of the model. Other, more special analysis types may include objects which keep track of certain events, such as the number of agents exiting via a certain door. Output data is recorded and computed into an analysis object. Due to large amount of data, not all of it can be outputted at the same time on screen, especially when considering the time dimension.

Considering the categories discussed above, Table 4-3 below lists all the common concepts across the investigated CSTs, allowing for the construction of a taxonomy of ‘things’ which are able to describe the Crowd Simulation knowledge domain.

Table 4-3. Taxonomy of common concepts for a crowd simulation analysis domain

Concept		Definition	Synonym
Geometry	Space	Walkable surfaces for agents.	Floor, Area
	Barrier	Surfaces which obstruct agent movement.	Obstruction
	Link	Connection between two walkable surfaces.	Transfer
	Portal	Entry and/or exit points for agents.	Entry or Exit Point
Agent	Agent	Representation of a building inhabitant.	Occupant
	Group	A collection of agents.	
	Profile	A definition of agent characteristics.	
Event	Journey	The act of describing agent movement from A to B.	Route
	Circulation	The act of agents following a route of waypoints.	Itinerary
	Evacuation	The act of agents exiting to nearest available exit.	Egress
Analysis	Graph	Simulation data plotted on a graph for user analysis.	
	Map	Simulation data plotted on a map, overlaid on the model.	
	Table	Simulation data in tabular format.	

All of the concepts mentioned above represent some form of model input or assumption. ***Geometry*** is modelled or imported from another tool. ***Agent*** numbers and profiles are assumed. ***Analysis*** requires the designer to select which output is relevant for the purposes of the investigation. These three categories require minimum user input. On the other hand, ***Event*** concepts are more complicated because they require the designers to define a specific context, requiring some form of ***‘expected’ behaviour input***. For example, designers need to state that certain groups of people will move towards a certain exit point, meaning that the event used to model the movement of people from A to B needs to be explicitly defined by the user. Concepts with required user input are highly dependent on circumstances, thus making each simulation scenario unique. Some tools have predefined behaviours for quick event deployment, but these usually require customisation from the users in order to increase the realism of scenarios. For more information, column 6 from the tables in Appendix A marks the features and objects which require customised user input related to their ‘expected’ behaviour.

4.2. Scenario automation requirements

In alignment with the requirements concerning automation, this section aims to bring forward the necessary level of information for complete, functional and realistic simulation scenarios and on how to deal with analysis output. Each of these stages is outlined separately.

4.2.1. Creating valid models

When considering the creation of valid simulation scenarios, two main categories of information input have been identified, as shown in Figure 4-2:

1) Geometric – information which defines the building environment within a simulation; this is provided from the ***Geometry*** category of concepts identified in Section 4.1;
2) Contextual – information which defines the circumstances of the simulated environment, such as: numbers of inhabitants, exit choices, agent characteristics, etc. This is provided from the categories of ***Agent*** and ***Event*** type categories from Section 4.1.

Khan et al. (2014) state that CSTs require user input, and therefore significant time for calibrating scenarios. For these to represent reality in an accurate way, contextual information is required such as the numbers and positions of agents. Cassol et al. (2016) developed a system consisting of configuration modules which specifically deal with:

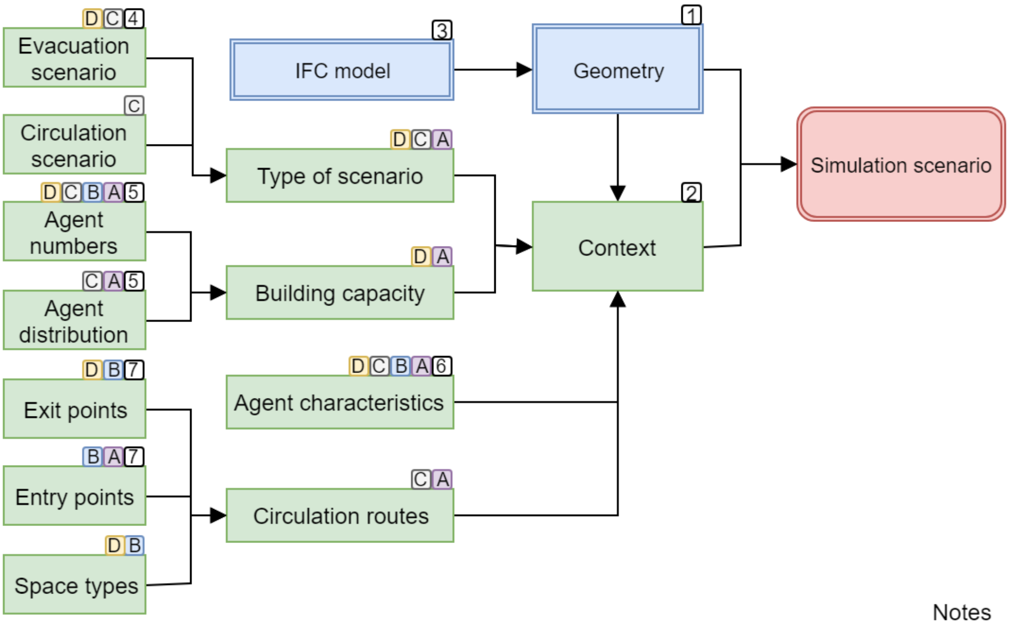
1) creating the geometry/environment,

2) creating population and,

3) creating the events.

This methodology follows similar steps to the one described by Kuligowski (2016a), already presented in Section 2.1.2. For the purposes of this research, the ***Agent*** and ***Event*** concepts are considered part of the context. The reason for this is that agents and events are usually defined together when a scenario is created. In addition to that, the geometry of the model is static in nature, with little variation across several simulations.

From the literature it was concluded that CSTs are often used to import from or work with BIM model data. In time this has also pushed most CST developers to facilitate various digital model import capabilities. However, as was evident from the literature and from the investigation outlined in the previous sections, this is mostly limited to geometry with no methodologies to retrieve a context about the simulation. Building digital model data is limited to geometry as most of the actual context information is not present explicitly. As opposed to geometry, context information provides important assumptions about each scenario and directly influences ***Agent*** and ***Event*** entities. Due to its various sources, contextual information can be hard to retrieve automatically without intelligent procedures in place. In practice, this information is usually provided by expert designers, who manually construct scenarios in accordance design analysis needs or predicted building use. This process is dependent on designer knowledge and experience and available published documentation on design procedures or regulations, which offer guidance on best practices, such as the UK PD 7974 (2004).



Notes

1 Geometric information is the same for multiple scenarios (static)

2 Information from this category is dependent on user input preferences (dynamic)

3 IFC schema objects individually mapped to crowd simulation object types

4 According to design guidance documentation and/or designer judgement

5 Dependent on time, spaces and scenario preferences

6 Agent walking speed, turn bias, etc.

7 Specified by designers, or implicitly from space types

Figure 4-2. Crowd simulation scenario information requirements (categories which contribute to context are in Figure 4-3)

The minimum requirements for a functional crowd simulation scenario have been identified from consulting the literature, guides and first-hand experience with CSTs (outlined in Figure 4-2). The primary requirement for each scenario is determining its population capacity. Additionally, Nilsson and Fahy (2016) mention it is imperative for designers to identify several factors which contribute to the context of a fire scenario: circulation paths, main exits/entrances and important waypoints.

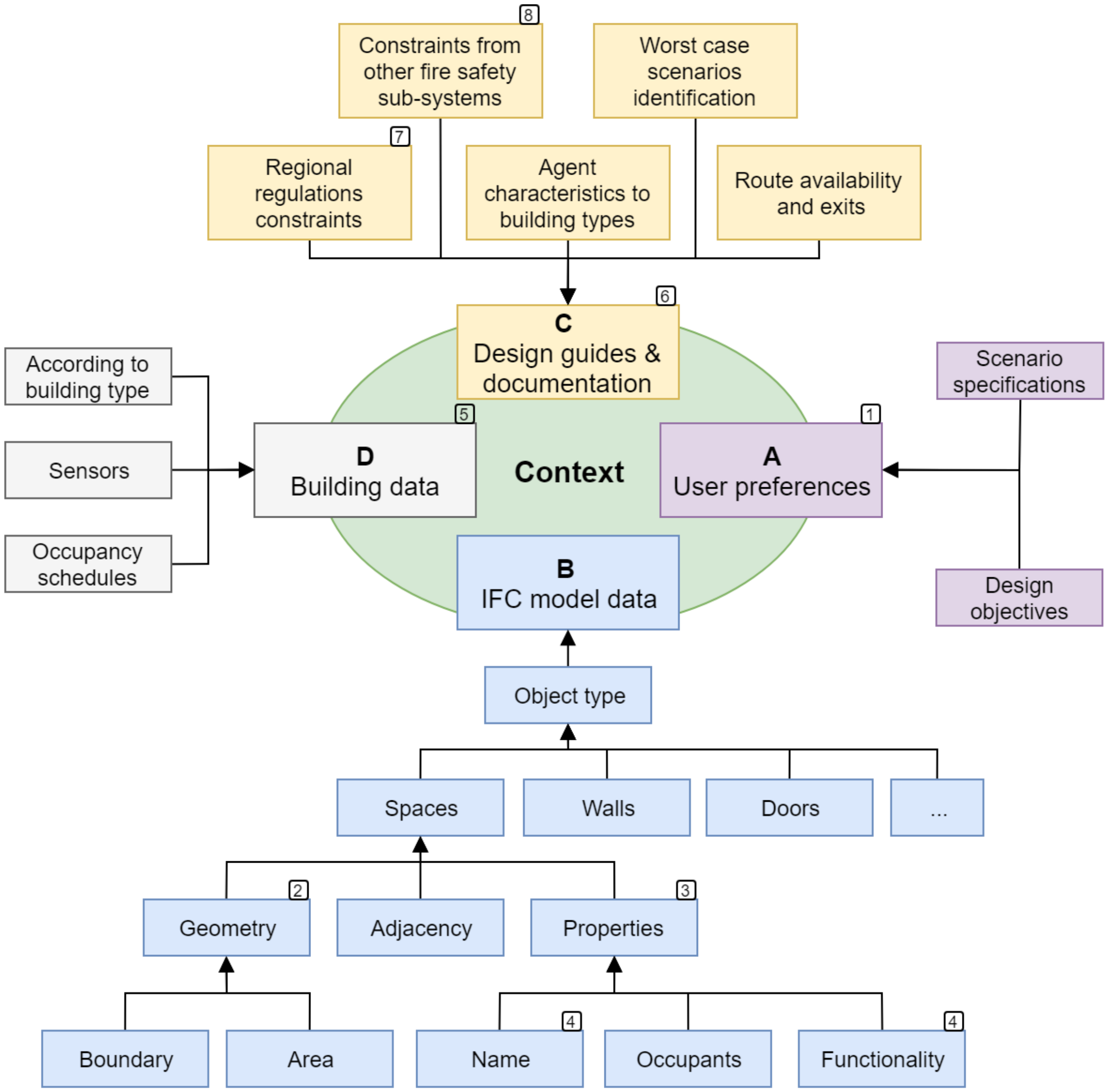
With the aim in identifying the sources of information which could be used in automation of context generation, four principal domains which can provide information input emerge, as shown in Figure 4-3:

A. ***User preferences*** **–** refers to the choices that the designer makes to generate a variety of scenarios which are relevant to the situation. For example, the designer should specify what type of scenario is chosen, what is the desired simulated building capacity, or which data sources are imported or used;

B. ***IFC model data*** – provides relevant building data, from geometric to contextual information. The data should be stated explicitly through specific properties. There are no defined standards for crowd simulation purposes, but the IFC schema allows the custom creation of properties at object level;

C. ***Design guides and documentation*** – when it comes to scenario assumptions, a variety of documentation guides and published documents can provide an overview of the factors to be considered. However, due to their indicative nature, much of the information is highly interpretable and circumstantial. The available information is spread across several documents. For instance, the PD 7974 (2004) part 6 is one of 7 documents published in the UK which were used to gather knowledge about the domain. However, information concerning occupant densities was vague, so local official regulation documents were required (The Building Regulations 2015);

D. ***Building data*** **–** live or historical data which refers to occupant traffic that might be relevant to the simulated building environment, e.g. data recorded from sensors, traffic cameras or exact numbers of occupants per space within a facility.


8
Constraints from other fire safety sub-systems
Worst case scenarios identification
7
Regional regulations constraints
Agent characteristics to building types
Route availability and exits
6
C
Design guides & documentation
According to building type
Sensors
Occupancy schedules
5
D
Building data
Context
1
A
User preferences
Scenario specifications
Design objectives
B
IFC model data
Object type
Spaces
Walls
Doors
...
2
Geometry
Adjacency
3
Properties
Boundary
Area
4
Name
Occupants
4
Functionality

Notes

1 User input can affect all other categories in B, C and D, which differ from one scenario to another

2 Object geometry can provide contextual information implicitly or is in conjunction with other data

3 Properties specific to building occupancy are not defined in the IFC schema, but custom ones can be used to provide explicit data

4 Names of specified functionality of a space can be used to decide route selection, entry and exit points with the correct rules and reasoning

5 Where available, historical or live data about the building can improve the accuracy and realism of simulation scenarios

6 Due to the indicative nature of design guides, many assumptions are flexible, vague and subject to designer personal judgement

7 Certain regulations vary with region, such as occupancy loads per space type

8 Due to the fire safety design complexity, other sub-systems can reflect the assumptions for crowd simulation scenarios

Figure 4-3. Information sources contributing to simulation scenario context

4.2.2. Creating analysis feedback

The second part in representing the knowledge processes required an investigation into what data and information are relevant to the analysis process. CSTs can generate a lot of data from every simulation created. Designers make use of imbedded tool ***Analysis*** and ***Visualisation*** features to be able to interpret the data and make decisions. To assess design performance objectively, certain Performance Indicators (PIs) need to be established. These need to allow not only human decision-makers, but also intelligent systems to distinguish between different scenarios and assess which data is most relevant to each situation, being a primary requirement for conceptualising machine-interpretable knowledge processes. The most important identified PIs are listed in Table 4-4.

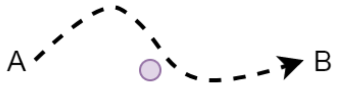
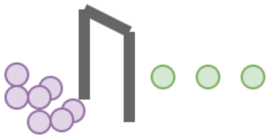
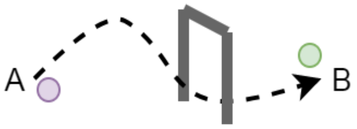
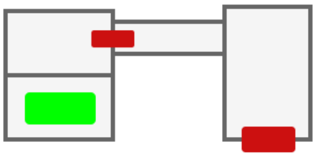
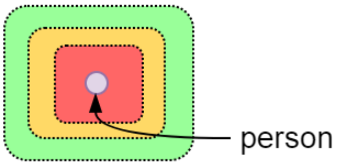
Nelson (2002) cited in PD 7974 (2004) modelled the effects of high traffic density on agent walking speeds, estimating that where a density is greater than 3.8 agent/m^2 the movement is completely halted. The identification of areas with high traffic density is therefore very important and is also coupled with the finding the occurrence in time of such events.

These are conventionally identified using density maps and simulation animations. The problem with relying on these is that it has to be evaluated by the engineer manually. However, with the use of a grading system this could also be done automatically. The evaluation of occupant densities can be done more automatically by adopting a scale such as Fruin's Level of Service (LOS) (Fruin 1992), which grades areas with different densities experienced over time. Figure 4-4 shows an example of this scale created using the MassMotion software.

Different PIs have been identified from the literature and from software capabilities. However, as mentioned previously, most of these factors are expected to be checked and analysed by humans.

When in the context of automation, certain logic rules and algorithms have to ensure the correct retrieval and processing of such data. In addition to that, analysis output needs to be considered in an object-oriented way, referring data to things like spaces, doors or agent objects.

Table 4-4. Identified PIs for building performance assessment during evacuation scenarios

	PI	Description	Visual representation	Source
1	Travel time	the time for agents to reach a destination point from a specific origin in the environment.		BS7974, Expert advice
2	Exit flow capacities	the flow capacity of a corridor, door or exit portal		
3	Escape time	the total time required by agents to reach a safe point		
4	Population density	density factor at a specific point in time, in a specific area of the environment		
5	Fruin's Levels of Service (LOS)	a way to quantify traffic density, describing the service state of a specific area in the environment		Simulation tools
6	Other PIs	situational or ad-hoc factors	N/A	N/A

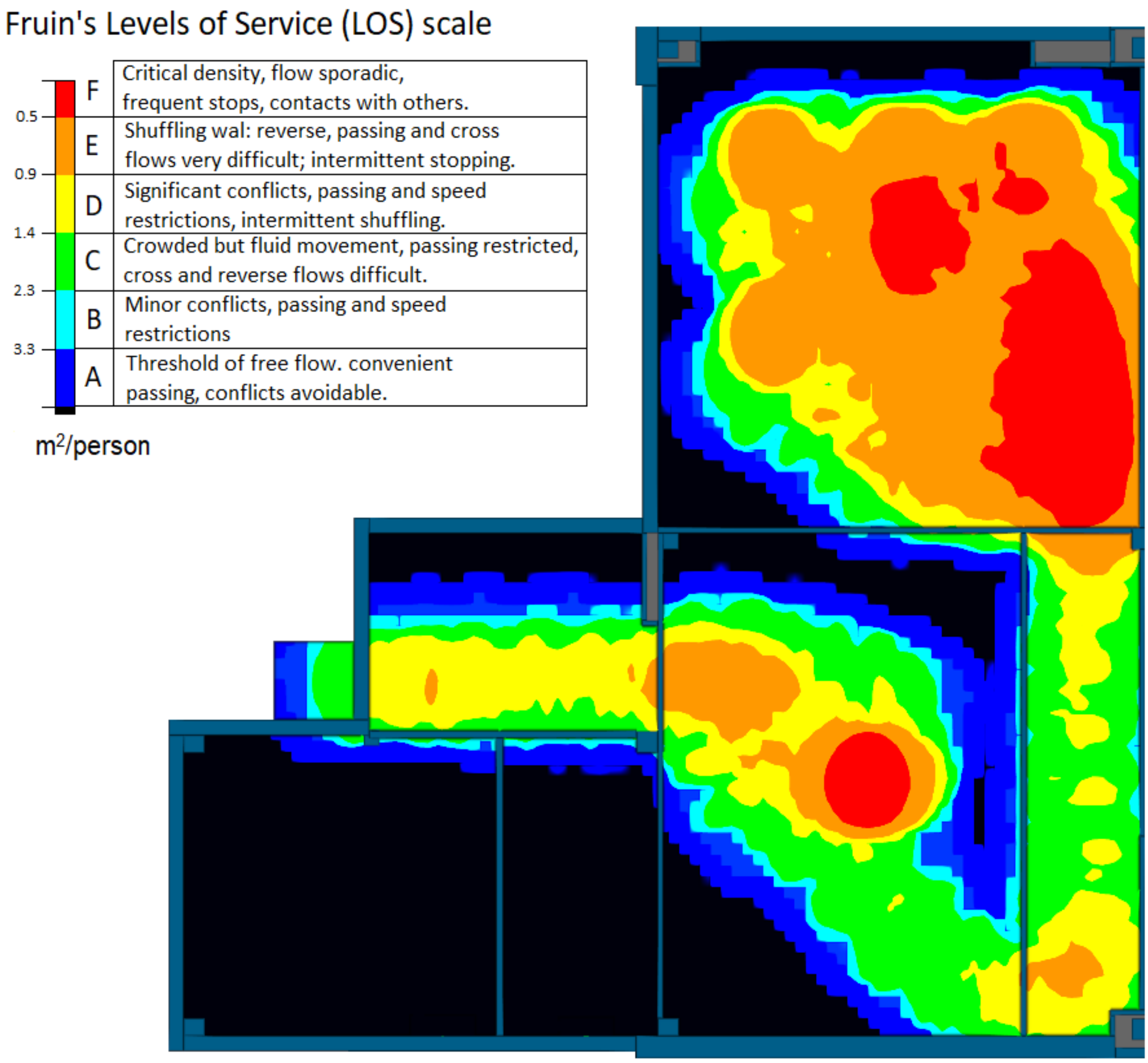


Figure 4-4. Example model with a plotted Fruin's LOS map

4.3. Technologies and tools requirements

This section was designed in mind with the practical deployment of the ONTOCS system as a prototype for testing and validation of the research hypothesis. Therefore, the inclusion of system components is justified here.

4.3.1. Crowd simulation model. MassMotion

Having analysed the underpinnings of CSTs and having taken into consideration their features, functionality and levels of interoperability with BIM, working with one tool was decided as the best choice for simplicity in delivering the system development in a speedy manner. Out of the tools investigated, ***MassMotion*** was the most thoroughly tested and was suited best for the purposes of this research. This was done mostly for its very good IFC import capabilities and the early involvement of the tools' developers with this research. However, a future aim is to be able to include several other similar tools within the large-scale simulation paradigm.

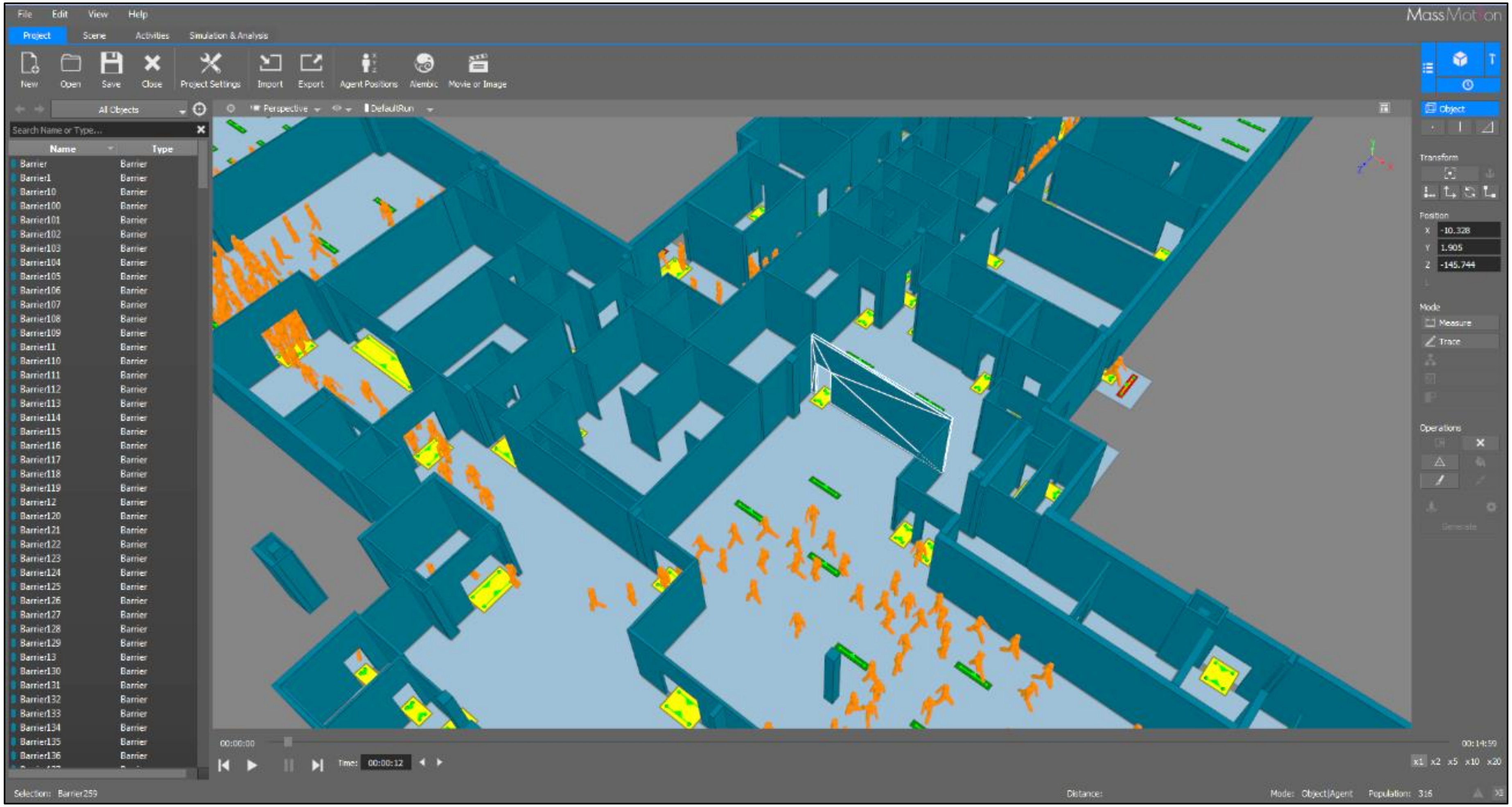
MassMotion is a crowd simulation tool developed by Oasys (Oasys Limited 2018a). According to the developers, the tool is aimed at professionals for testing and analysing the movement of people in a number of situations. The tool is used around the world by professionals on real world projects for pedestrian modelling (Oasys Limited 2018b).

The methodology used by MassMotion is a hybrid CA model (for the environment and movement) and represents Agents with full behavioural individuality. It is therefore able to simulate very realistic human behaviour making good use of agent characteristics, including the SF model (Helbing and Molnár 1995). Each agent within the simulation is able to calculate its own path using a cost function in terms of time. The longer it takes an agent to reach a destination the costlier it is. Each agent aims to achieve a low value cost during a simulation. Agents are therefore able to calculate their movement cost at each mathematical iteration and can decide to change their routes dynamically.

The software has been validated by Arup (Kinsey et al. 2015) as a professional tool and has been investigated by several studies (Thalmann et al. 2007, Kuligowski 2016a, Mashhadawi 2016). The latter study tests the software in several standard benchmarking tests and concludes that the default pre-set settings of the MassMotion software can deviate from real results by up to 60%. However, if the parameters are calibrated to fit the real-life scenario more closely, this can be as low as 13%. Typical calibration techniques involve the representation of reality as closely as possible, for example an accurate definition of agent physical properties to each individual person in reality or predicting exact starting locations of agents at simulation start. Kuligowski (2013) mentions that no simulation software can perfectly simulate reality, and also when it comes to real-life evacuations each is different due to human behavioural uncertainty (Kobes et al. 2010).

MassMotion offers very good interoperability with the IFC format, correctly importing complex geometric models, along with the ability to transform IFC objects into corresponding MassMotion objects. This is due to MassMotion being natively expressed in 3D, with every vertex having three coordinates. Another important aspect is the relatively simple and logical hierarchy for its taxonomy of concepts and a very object-oriented nature, as well as the XML format for simulation files, making MassMotion an open tool from an interoperability point of view. Lastly, the tool interface is user friendly and easy to learn (see Figure 4-5).

Due to these reasons, the MassMotion tool was chosen to be integrated with the developed system for the research prototype (workflow and implementation in section 7.1).



Name	Type
Barrier	Barrier
Barrier1	Barrier
Barrier10	Barrier
Barrier100	Barrier
Barrier101	Barrier
Barrier102	Barrier
Barrier103	Barrier
Barrier104	Barrier
Barrier105	Barrier
Barrier106	Barrier
Barrier107	Barrier
Barrier108	Barrier
Barrier109	Barrier
Barrier11	Barrier
Barrier110	Barrier
Barrier111	Barrier
Barrier112	Barrier
Barrier113	Barrier
Barrier114	Barrier
Barrier115	Barrier
Barrier116	Barrier
Barrier117	Barrier
Barrier118	Barrier
Barrier119	Barrier
Barrier12	Barrier
Barrier120	Barrier
Barrier121	Barrier
Barrier122	Barrier
Barrier123	Barrier
Barrier124	Barrier
Barrier125	Barrier
Barrier126	Barrier
Barrier127	Barrier
Barrier128	Barrier
Barrier129	Barrier
Barrier13	Barrier
Barrier130	Barrier
Barrier131	Barrier
Barrier132	Barrier
Barrier133	Barrier
Barrier134	Barrier
Barrier135	Barrier
Barrier136	Barrier

X -10.328
Y 1.905
Z -145.744

Time: 00:00:12

Selection: Barrier259

Mode: Object|Agent

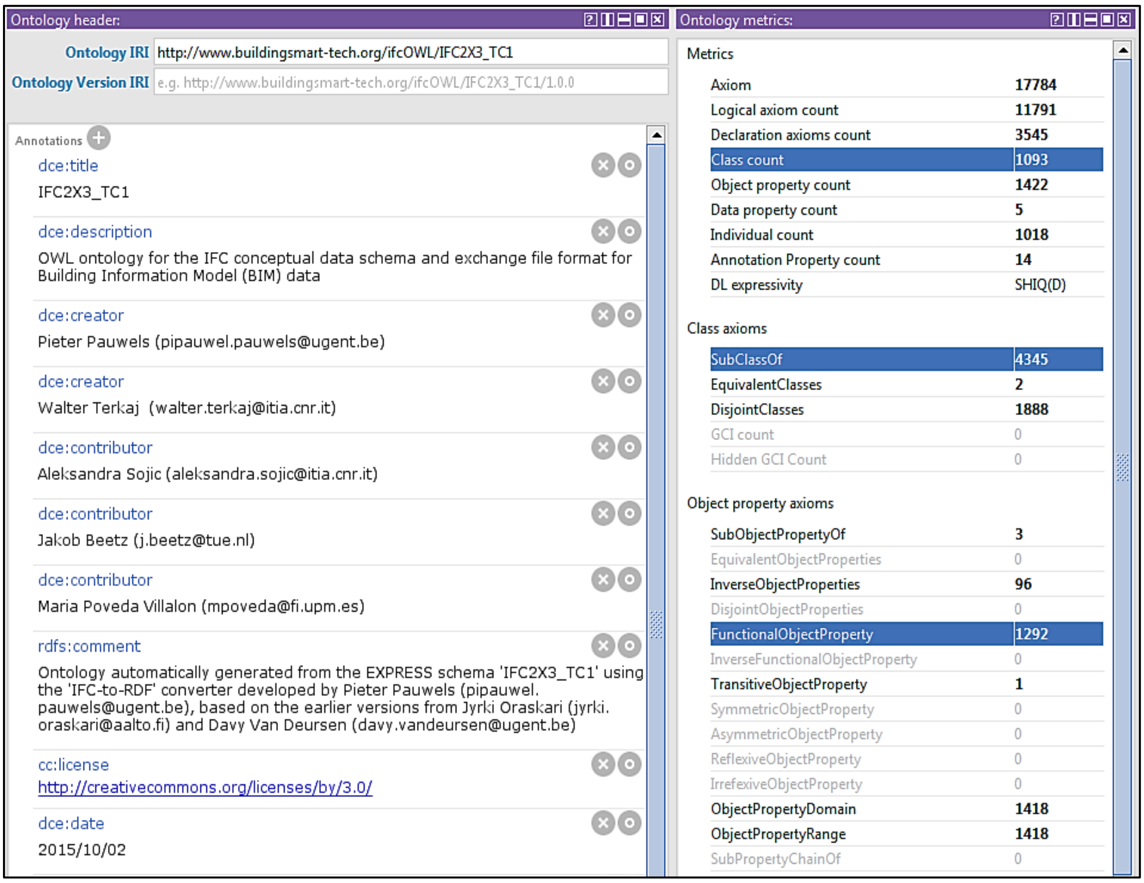
Population: 316

Figure 4-5. MassMotion interface

4.3.2. Digital building model. IFC and IfcOwl

The role of the IFC schema and format within the BIM paradigm has been thoroughly discussed in Chapter 2. IFC has been the debate of many academic papers since its inception and continues to evolve and deliver vendor-free interoperability between the AEC disciplines. Its sheer size of concepts and ability to represent any building model comprehensively, while also including design project data, and more recently energy modelling concepts make this the best candidate for a BIM data source. Due to its popularity in academia and industry, many CSTs have developed IFC importers, with some investigated in the previous chapter. Additionally, Section 4.2.1 already concluded that its structure can provide not only geometry, but other contextual information precisely because of its comprehensive nature.

Because the scope of this research and proposed system consider a knowledge level representation and mining, the IFC is the only format with an existing OWL representation, making it IfcOwl the only reasonable and reliable choice as a source of design model with web representation capabilities. IfcOwl is large, with a summary of the ontology in Figure 4-6; the one used for this research is based on the IFC2x3_TC1 release.



Ontology header:

Ontology IRI http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1

Ontology Version IRI e.g. http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1/1.0.0

Annotations

dce:title
IFC2X3_TC1

dce:description
OWL ontology for the IFC conceptual data schema and exchange file format for Building Information Model (BIM) data

dce:creator
Pieter Pauwels (pipauwel.pauwels@ugent.be)

dce:creator
Walter Terkaj (walter.terkaj@itia.cnr.it)

dce:contributor
Aleksandra Sojic (aleksandra.sojic@itia.cnr.it)

dce:contributor
Jakob Beetz (j.beetz@tue.nl)

dce:contributor
Maria Poveda Villalon (mpoveda@fi.upm.es)

rdfs:comment
Ontology automatically generated from the EXPRESS schema 'IFC2X3_TC1' using the 'IFC-to-RDF' converter developed by Pieter Pauwels (pipauwel.pauwels@ugent.be), based on the earlier versions from Jyrki Oraskari (jyrki.oraskari@aalto.fi) and Davy Van Deursen (davy.vandeursen@ugent.be)

cc:license
http://creativecommons.org/licenses/by/3.0/

dce:date
2015/10/02

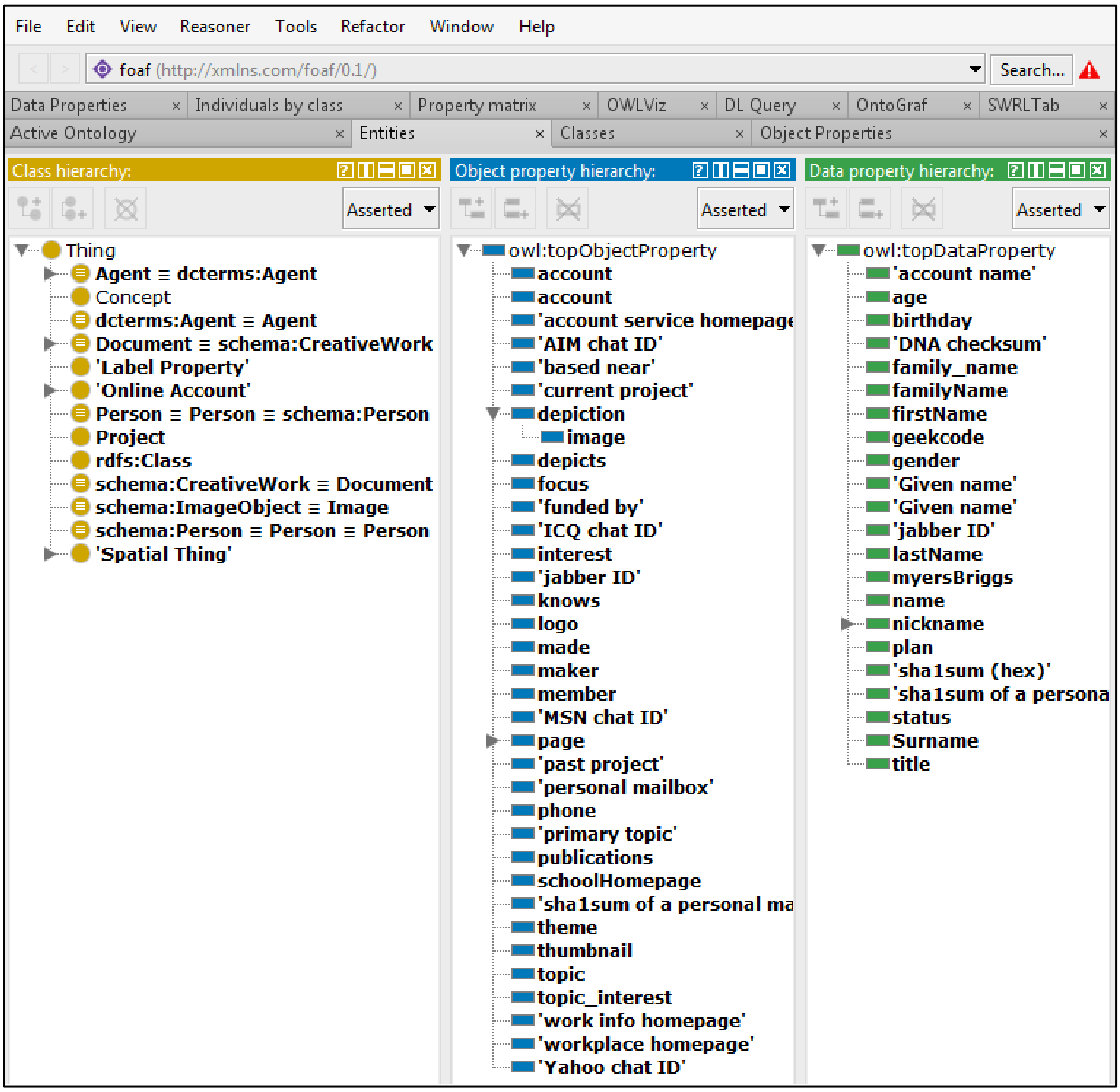
Ontology metrics:

Metrics	
Axiom	17784
Logical axiom count	11791
Declaration axioms count	3545
Class count	1093
Object property count	1422
Data property count	5
Individual count	1018
Annotation Property count	14
DL expressivity	SHIQ(D)
Class axioms	
SubClassOf	4345
EquivalentClasses	2
DisjointClasses	1888
GCI count	0
Hidden GCI Count	0
Object property axioms	
SubObjectPropertyOf	3
EquivalentObjectProperties	0
InverseObjectProperties	96
DisjointObjectProperties	0
FunctionalObjectProperty	1292
InverseFunctionalObjectProperty	0
TransitiveObjectProperty	1
SymmetricObjectProperty	0
AsymmetricObjectProperty	0
ReflexiveObjectProperty	0
IrrefexiveObjectProperty	0
ObjectPropertyDomain	1418
ObjectPropertyRange	1418
SubPropertyChainOf	0

Figure 4-6. Summary view of the IfcOwl ontology

4.3.3. Knowledge modelling tool. Protégé

Although OWL ontologies can be written as code, knowledge engineers make use of editor tools which allows for fast ontology creation. ***Protégé*** (Stanford University 2018) is a very popular ontology editing tool, initially developed in academia (Noy et al. 2003), which is free to use and with a large library of plug-ins developed by its community. The interface in Figure 4-7 shows a graphical view of the classes, object and data properties belonging to the popular Friend of a Friend (FOAF) ontology which is used to represent data about people on the Web with the scope of linking them on social network platforms. The software comes with embedded reasoners used to check the correctness (the ontology concepts should be correctly defined or stated) and consistency (the ontology should not contain conflicting information) of ontologies. Additionally, it allows the creation of rules and queries, which can be used to test the reasoning capabilities of ontologies. Finally, it offers ways to dynamically visualise ontologies in graphs.



Class hierarchy:

- Thing
  - Agent ≡ dcterms:Agent
  - Concept
  - dcterms:Agent ≡ Agent
  - Document ≡ schema:CreativeWork
  - 'Label Property'
  - 'Online Account'
  - Person ≡ Person ≡ schema:Person
  - Project
  - rdfs:Class
  - schema:CreativeWork ≡ Document
  - schema:ImageObject ≡ Image
  - schema:Person ≡ Person ≡ Person
  - 'Spatial Thing'

Object property hierarchy:

- owl:topObjectProperty
  - account
  - account
  - 'account service homepage
  - 'AIM chat ID'
  - 'based near'
  - 'current project'
  - depiction
    - image
  - depicts
  - focus
  - 'funded by'
  - 'ICQ chat ID'
  - interest
  - 'jabber ID'
  - knows
  - logo
  - made
  - maker
  - member
  - 'MSN chat ID'
  - page
  - 'past project'
  - 'personal mailbox'
  - phone
  - 'primary topic'
  - publications
  - schoolHomepage
  - 'sha1sum of a personal ma
  - theme
  - thumbnail
  - topic
  - topic_interest
  - 'work info homepage'
  - 'workplace homepage'
  - 'Yahoo chat ID'

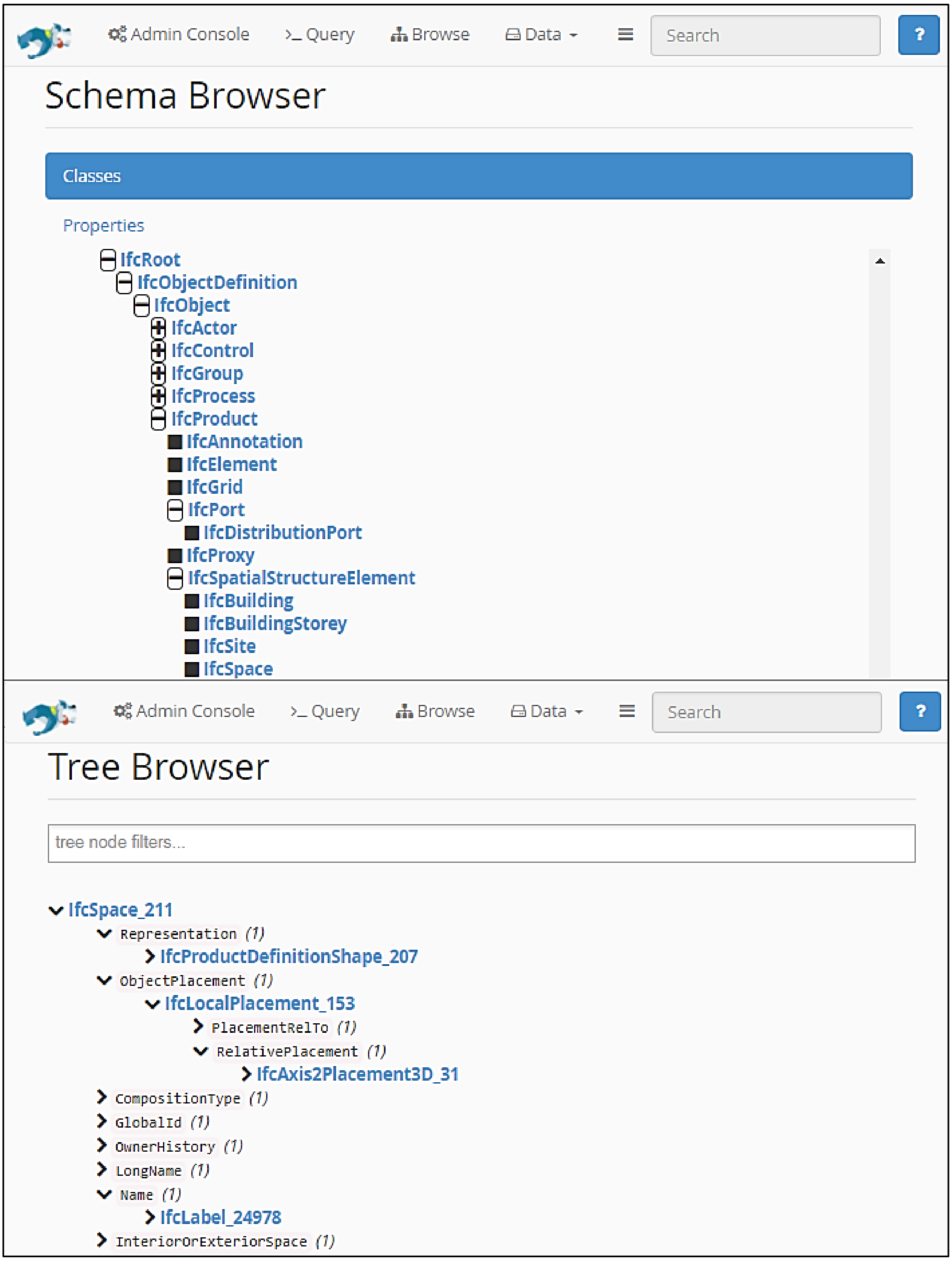
Data property hierarchy:

- owl:topDataProperty
  - 'account name'
  - age
  - birthday
  - 'DNA checksum'
  - family_name
  - familyName
  - firstName
  - geekcode
  - gender
  - 'Given name'
  - 'Given name'
  - 'jabber ID'
  - lastName
  - myersBriggs
  - name
  - nickname
  - plan
  - 'sha1sum (hex)'
  - 'sha1sum of a persona
  - status
  - Surname
  - title

Figure 4-7. Entities view tab of the FOAF ontology in the Protégé tool

4.3.4. Knowledge management server. Stardog

In the context of this research, a knowledge management server is required to storing OWL schemas and RDF resources graphs, and retrieve embedded knowledge. This is also termed a ‘triple store’, which is similar to a relational database, except that it stores and manipulates data in RDF using SPO patterns, or ‘triples’.



```
⊟ IfcRoot
  ⊟ IfcObjectDefinition
    ⊟ IfcObject
      ⊞ IfcActor
      ⊞ IfcControl
      ⊞ IfcGroup
      ⊞ IfcProcess
      ⊟ IfcProduct
        ■ IfcAnnotation
        ■ IfcElement
        ■ IfcGrid
        ⊟ IfcPort
          ■ IfcDistributionPort
        ■ IfcProxy
        ⊟ IfcSpatialStructureElement
          ■ IfcBuilding
          ■ IfcBuildingStorey
          ■ IfcSite
          ■ IfcSpace
```

```
⌄ IfcSpace_211
    ⌄ Representation (1)
        › IfcProductDefinitionShape_207
    ⌄ ObjectPlacement (1)
        ⌄ IfcLocalPlacement_153
            › PlacementRelTo (1)
            ⌄ RelativePlacement (1)
                › IfcAxis2Placement3D_31
    › CompositionType (1)
    › GlobalId (1)
    › OwnerHistory (1)
    › LongName (1)
    ⌄ Name (1)
        › IfcLabel_24978
    › InteriorOrExteriorSpace (1)
```

Figure 4-8. Stardog web interface pages for schema and tree browsers

The chosen server tool for deployment was ***Stardog*** (Stardog Union 2018), a popular RDF store used more commonly in industries (including NASA, Samsung, eBay and others), but has also been used in academia in important related studies around BIM (Pauwels et al. 2016, Farias et al. 2015, Pauwels and Terkaj 2016).

Stardog offers excellent OWL and OWL2 reasoning, also supporting the reasoning over SWRL rules and efficient querying using SPARQL 1.1. The system was developed in Java but integrates with several other programming languages via API packages and libraries. It interfaces over HTTP and SNARL protocols over the web. Additionally, the tool is very well suited for large scale triple databases, which can work from physical disk or memory storage.

The most important factor was its capability to support different levels of reasoning levels, as this research employs a combination of OWL2 syntax with many SWRL rules and SPARQ 1.1.

The developers offer a free community version which is limited to 25 million edges and nodes databases. Finally, its sophisticated browser interface (Figure 4-8) allows very convenient ontology schema and instances browsing, as well as querying and database management.

4.4. Summary

This chapter presented the requirements to enable a framework for an intelligent knowledge-based system for automatic simulation scenario creation in a building evacuation context, as was required by the methodology proposed in Chapter 3. The section presented an in-depth view of typical CST concepts and established a taxonomy to use for defining the CS domain conceptually. Then it presented the information requirements for automatic model creation from various sources and PIs for facilitating an objective feedback process for design decision-making. Finally it presented the tools used throughout the development of this research: the MassMotion tool, the IfcOwl as a BIM source, the Protégé tool for OWL ontologies creation and testing, Stardog RDF server for deployment.

Chapter 5. Knowledge base development

This chapter outlines one of the core contributions of this research, the development of several ontologies within the CS knowledge domain. The conceptual system design introduced in Chapter 3 defines several components required for knowledge formalisation and mining processes to occur. In this chapter, the ontology representations of the information models and the processes behind crowd simulation analysis are introduced. The contents are divided into three main sections.

The first part (Section 5.1) outlines the overall ontology development efforts. The Crowd Simulation Scenario (CSS) ontology defines the domain and semantics between the concepts identified in Chapter 4. The CSS ontology describes the crowd simulation model, but it lacks the concepts to be able to provide meaningful feedback to the design process. Thus, a separate ontology was developed for this purpose, the Feedback Analysis (FBA) ontology, presented in Section 5.1.2. The FBA ontology considers the design process of analysing the performance of a crowd simulation model, based on the requirements identified in Section 4.2.2. To showcase the integration of the design to other information models, an ontology was developed which links design code occupancy factors with space types, and was made to work together with an existing Uniclass ontology dataset, presented in Section 5.1.3.

The second part of this chapter (Section 5.2) aims to provide clarity and cohesively align the developments above with other important ontologies within this design context. The most important is the alignment with the existing IFC ontology, which acts as the central provider for geometry in a BIM-oriented fashion. The alignment between the IFC and CSS ontologies provides a greater understanding about the interoperability between these two knowledge domains (presented in Section 5.2.1). The alignment between the CSS and FBA ontologies is introduced next, showing the common concepts and how they work together in Section 5.2.2. The secondary ontologies mentioned above are also aligned in Section 5.2.3.

The final part of this chapter outlines various validation and consistency checks which were carried during the ontology engineering process.

5.1. Ontology development

Ontologies are introduced in Section 2.3.1 as a means to define knowledge domains which are currently used in the semantic web. Unlike vocabularies and taxonomies, ontologies offer a greater level of knowledge representation with more semantics and syntax. This is due to their ability to represent relationships of all types between concepts. Noy and McGuinness (2001) mention that there is no one correct way of defining an ontology and that this is usually an iterative process which depends on the knowledge domain and how the ontology is applied. The basic principle is to be able to represent knowledge about the building design evaluation process, and to primarily work with information models from these domains.

The ontologies developed during this research are listed below and their metrics are summarised in Table 5-1. They have all undergone several iterations and have been tested in various forms using the developed system. There are two main ontologies developed for describing the crowd simulation analysis domain:

1) Crowd Simulation Scenario (CSS) ontology – describes a crowd simulation model along with its output results;
2) Feedback Analysis (FBA) ontology – describes the use and generation of knowledge from CSS instance data and analysis objectives.

Two secondary ontologies were also developed:

1) MassMotion ontology (MM) – describes the MassMotion software structure;
2) UK Spaces Occupant Capacities (UKSOC) ontology – describes categories of spaces and their respective occupancy factors as per UK design guidance.

The developed ontologies are presented in the following sections, and their structure and reasoning are discussed considering how they are applied in practice. Some of the concepts are based on example competency questions which were used to help define the scope and applicability of each ontology. Before this, however, an overall picture is presented on the interactions of the developed ontologies, along with others which are used in defining the ONTOCS framework and system.

Table 5-1. Metrics for developed ontologies

	CSS	FBA	MM	UKSOC
Axiom	311	163	974	219
Classes	56	32	247	75
Object properties	15	10	38	1
Data properties	22	8	4	1
Individuals	0	0	0	12
DL expressivity	ALCHIF(D)	ALCHF(D)	ALCHF(D)	ALF(D)

5.1.1. ONTOCS overall alignment configuration

Kaufman and Michalski (2005) identified several ways of retrieving data and knowledge, depending on the origins of the data, its completeness, or whether its availability is dependent on time. Within this context, considering the overall alignment of the ontologies, the ONTOCS system falls in the category of *"learning from distributed data".*

A top-level view of the linking between resources is provided here, based on the developed workflow of the ONTOCS framework. Figure 5-1 shows the CSS ontology at the centre of the system, being responsible conceptualising model objects, results data and user assumptions, collaborating with several other ontologies for each of these purposes. Under it are the CST ontologies, with the MM ontology in this particular case. A CST ontology is responsible with representing and storing any tool specific information, and it represents data in its own internal structure. This is in line with the aim to use the CSS ontology as a generalised schema above individual CSTs, and therefore enabling the inclusion of several tools in the knowledge mining processes.

The IfcOwl ontology on the left of Figure 5-1 conceptualises the digital building model, under the BIM knowledge domain. The BIM is a source not just of geometry objects, but also any other relevant data for design or analysis which can exist explicitly via object properties. With additional resources in place, like given the examples of design codes and standard classification system, BIM model data can be leveraged to provide additional contextual information for the CSS ontology. Other Semantic Web Resources can be considered, as suggested in Section 4.2.1. However, due to time constraints not all could be investigated and implemented in practice during this research.

Finally, the FBA ontology conceptualises a design analysis knowledge domain, where data from the CSS is used in conjunction with user objectives to validate and test simulation results.

The main challenge was to correctly align the several knowledge domains without any conflicts or redundancies. Secondly, the available resources must allow for convenient access by the system working with as few knowledge domains as possible in order to optimise its performance. Ideally, each ontology must be self-reliant and modular, applying rules without depending on other external resources or other ontologies. The FBA ontology is unable to function on its own because it relies on the representation of the results in the CSS ontology.

Another challenge was to decide the best location of ontology rules, given that certain rules depend on more than one specific ontology, when resources need to be connected across multiple domains. This had an impact of where alignments had to take place in the first place. More on rules construction is discussed in Chapter 6.

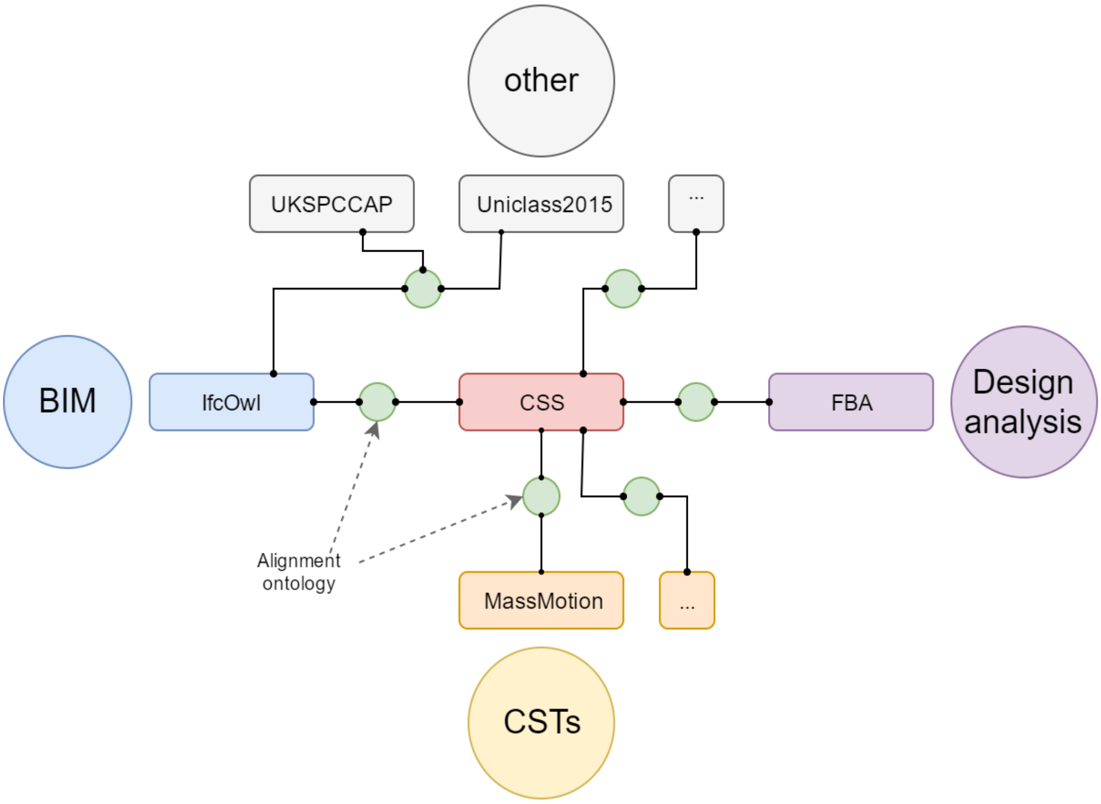


Figure 5-1. Alignment of ontologies for the ONTCS system and potential extensions to nearby knowledge domains

5.1.2. Representing the crowd simulation model

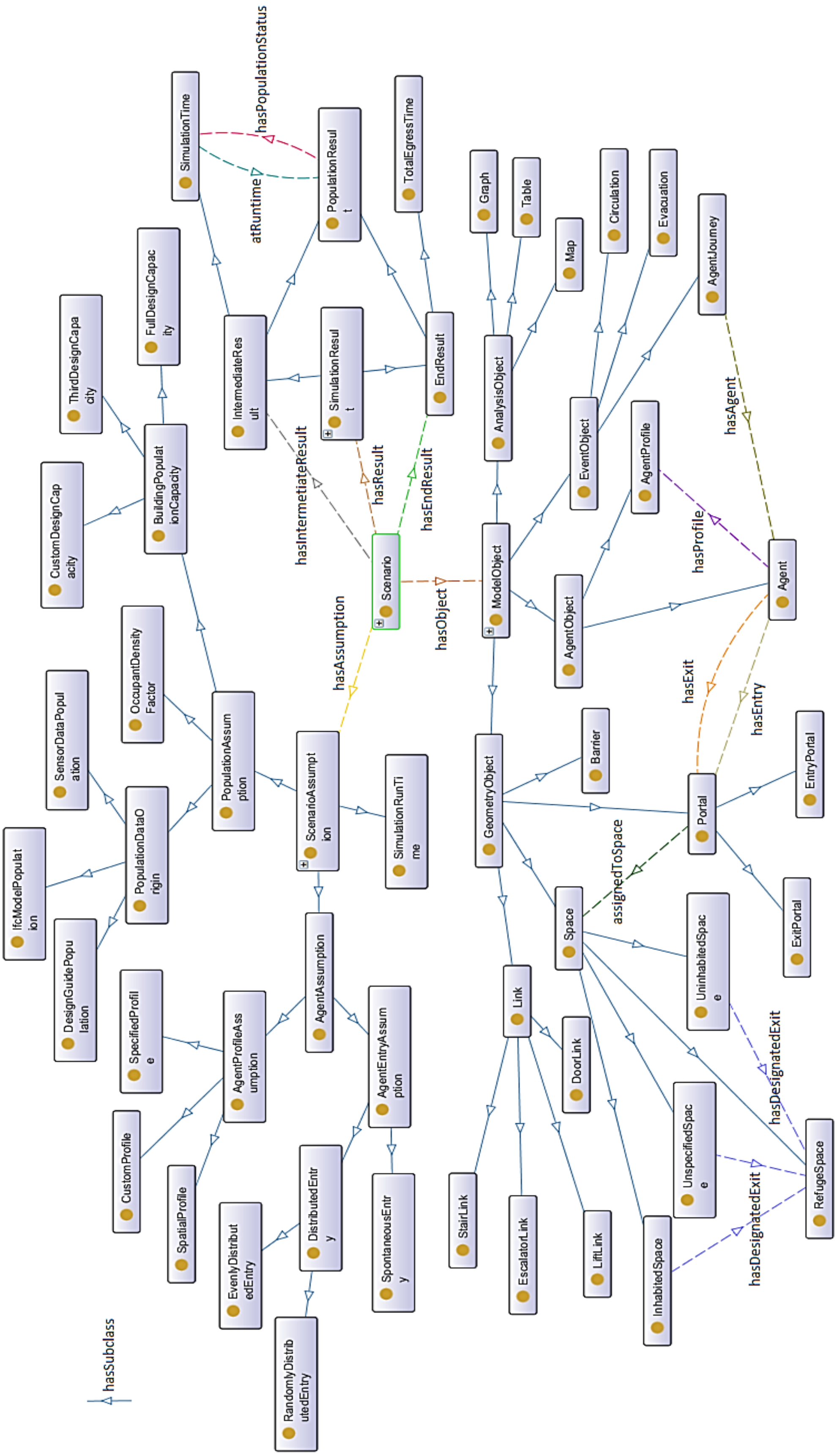


Figure 5-2. The Crowd Simulation Scenario (CSS) ontology

After investigating several crowd simulation tools, common concepts were identified across the knowledge domain. A generic 'Crowd Simulation Scenario' (CSS) ontology was developed which describes the necessary concepts, actions and data in a simulation environment, with a full view in Figure 5-2, and a full description on metrics and properties in Appendix B. The CSS ontology builds upon the already identified taxonomy from Section 4.1.2, adding additional relationships between concepts, as is presented in Figure 5-3. The taxonomy alone represented a hierarchy of concepts. *"All features of taxonomies, thesauri and Topic Maps can be expressed in ontologies" (Ullrich 2003).* The ontology brings additional 'meaning' for 'things' using object and data properties, equivalencies, instances and other logical relationships.

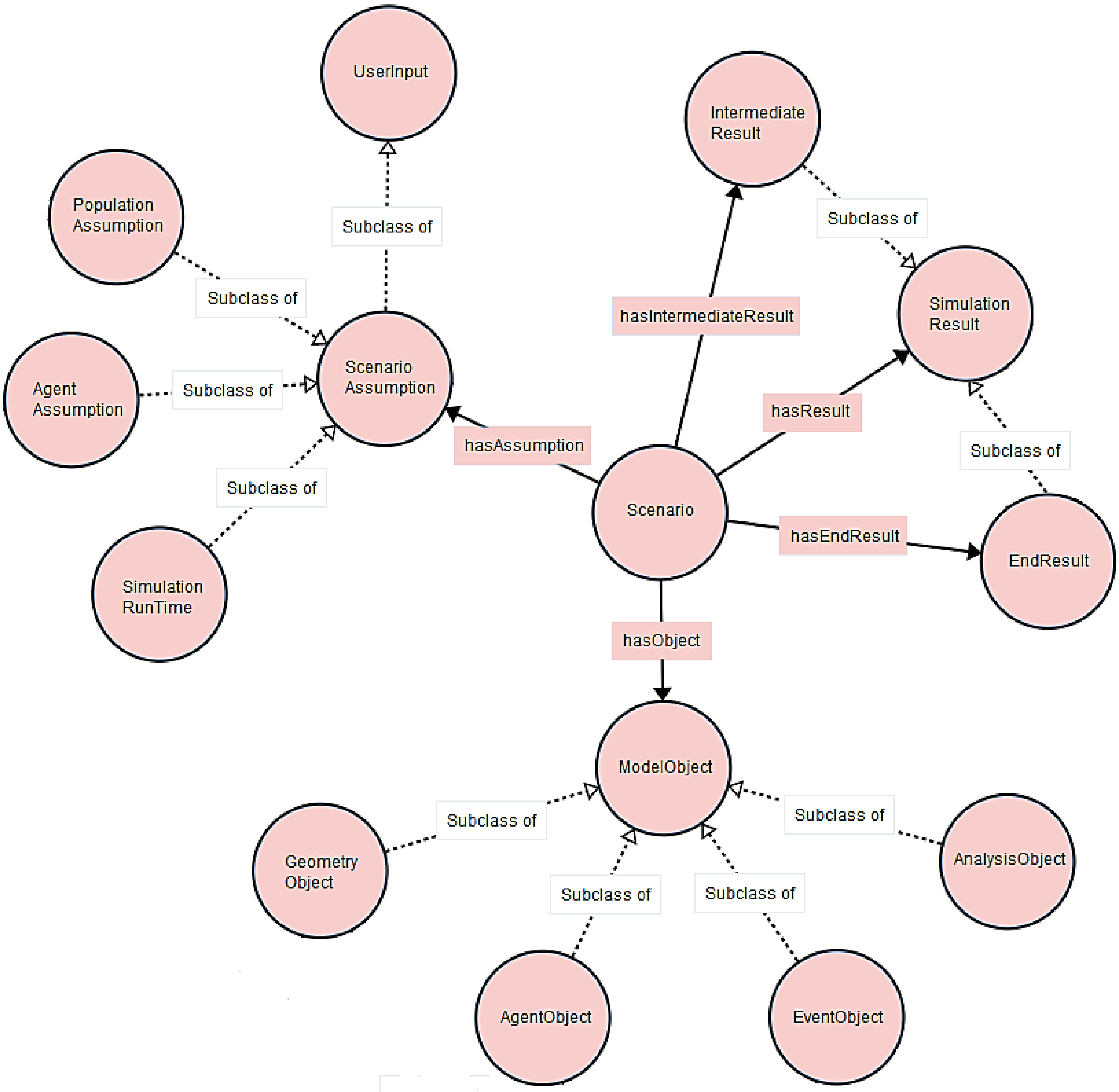


Figure 5-3. Main CSS classes in direct relationship with the 'Scenario' class

The core competency question relevant to the CSS ontology is:

- *What types of 'things' does a simulation scenario have?*

Figures 5-2 and 5-3 depict the ***Scenario*** class at the core of the ontology. In addition to model objects, which mirror the already defined taxonomy, there are other concepts which relate to the required user inputs identified in Section 4.2:

- Assumptions - conceptualising design choices via the main ***ScenarioAssumption*** class, which are part of the required input, thus a subclass of ***UserInput***;
- Results – conceptualising results about each simulation run, via the ***SimulationResult*** class. This is required for the analysis of the performance stage.

The reason for including these additional classes is to allow the CSS ontology to generalise model information on top a CST model or ontology, thus providing future extensibility. This is in line with the vision of the ONTOCS framework.

Model objects - competency question:

- *Which types of objects must exist within a simulation model?*

The ***ModelObject*** class specifically includes concepts which are present within the model. These closely resemble the features and objects present in a typical CST, with its four distinct categories: ***GeometryObject***, ***EventObject***, ***AgentObject*** and ***AnalysisObject***, as shown in Figure 5-4. Some of the main differences to Table 4-3 is the inclusion of a more generic ***Link*** class which is used to represent a connection between multiple types of surfaces. Therefore, its subclasses deal with representing a specific link type. For example, ***DoorLink*** is being used to model a doorway connection between two spaces on the same level, whereas a ***StairLink*** models a connection between two spaces which are on different levels. The geometry between a door and staircase is very different, but in a CST model representation, they are still effectively walkable surfaces which apply certain restrictions to alter agent movement.

The most characteristic object type for all CSTs is the one defining the walkable surfaces - ***Space***, which allows agents to effectively exist and act within the model. They are represented virtually within a model as surfaces without a 3D component. The name was chosen as they effectively refer to spaces in real buildings. Additionally, when considering a design scenario, a building environment is split by levels and spaces, so designers have an easier time identifying regions within a model. This conceptualisation is also in alignment with IfcSpace, presented in Section 5.2.1. The functionality of a space was required in order to refer to spaces in other specific circumstances, hence its subclasses from Figure 5-4.

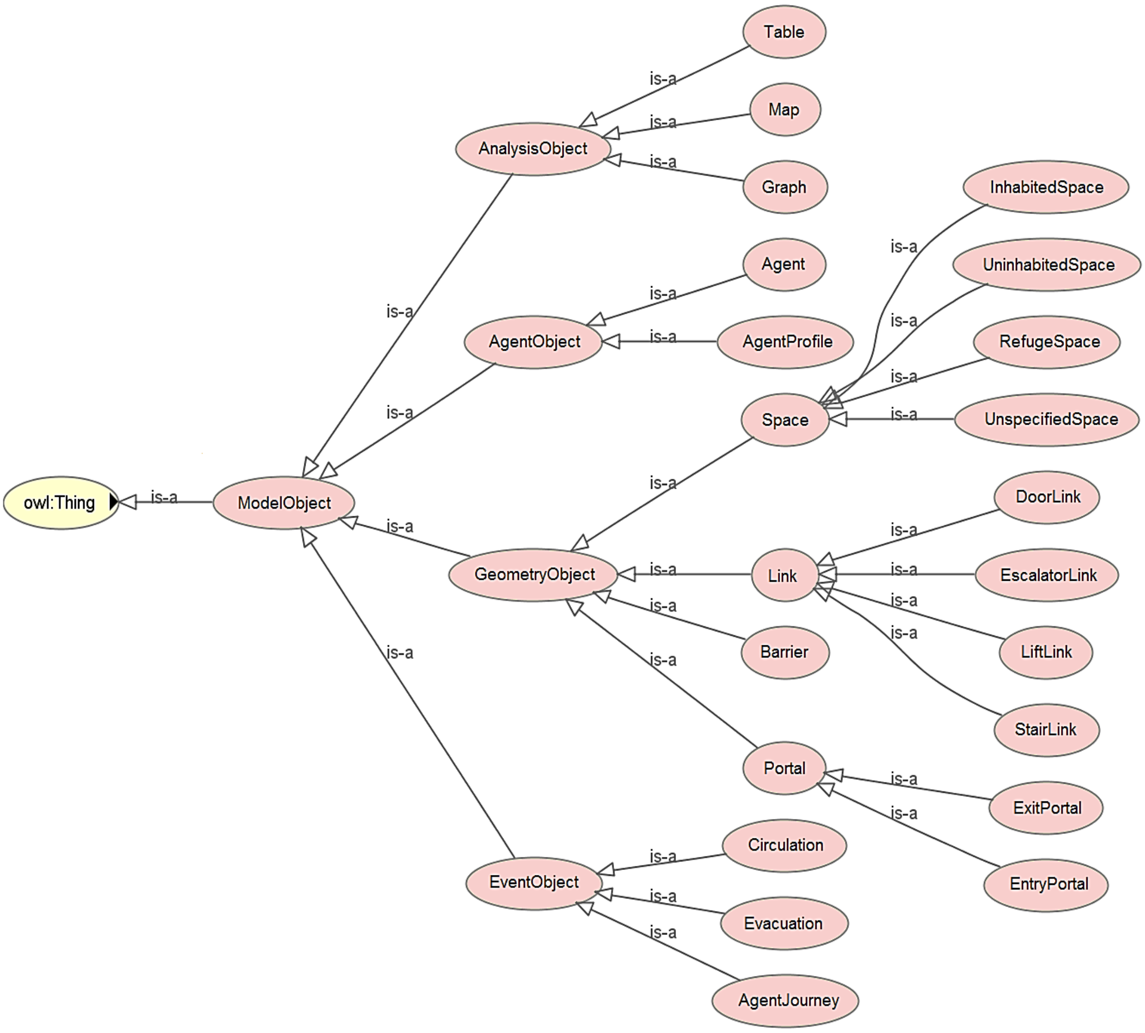


Figure 5-4. CSS ontology ‘ModelObject’ class hierarchy

- *What are the types of spaces within a building when evaluating an evacuation plan?*

An ***InhabitedSpace*** for example refers to a ***Space*** which has agents assigned to it, and it is considered inhabited in reality. A ***RefugeSpace*** designates the function for a space to act as a destination point for agents in an evacuation scenario. These add context to the model, as well as a means for automation allowing ontology reasoning to ‘understand’ the building environment.

Scenario assumptions – competency question:

- *Which types of assumptions must a simulation have?*

Figure 5-5 shows all the concepts classified as assumptions. These refer to concepts which are supposed to keep track of the assumed scenario context and are usually in relationship with ***EventObject*** or ***AgentObject*** classes. The ***ScenarioAssumption*** class therefore conceptualises questions such as:

- *What population data source is assumed?*
- *What population capacity is assumed?*
- *What agent profiles are assumed?*
- *What length of simulation time is assumed?*

Each of these assumptions can yield different results and influences the behaviour of agents and therefore the performance of the design. Within CSTs, as concluded in Section 4.1, these are usually user input assumptions. Each CST has several pre-set values for these inputs, such as different types of agent profiles.

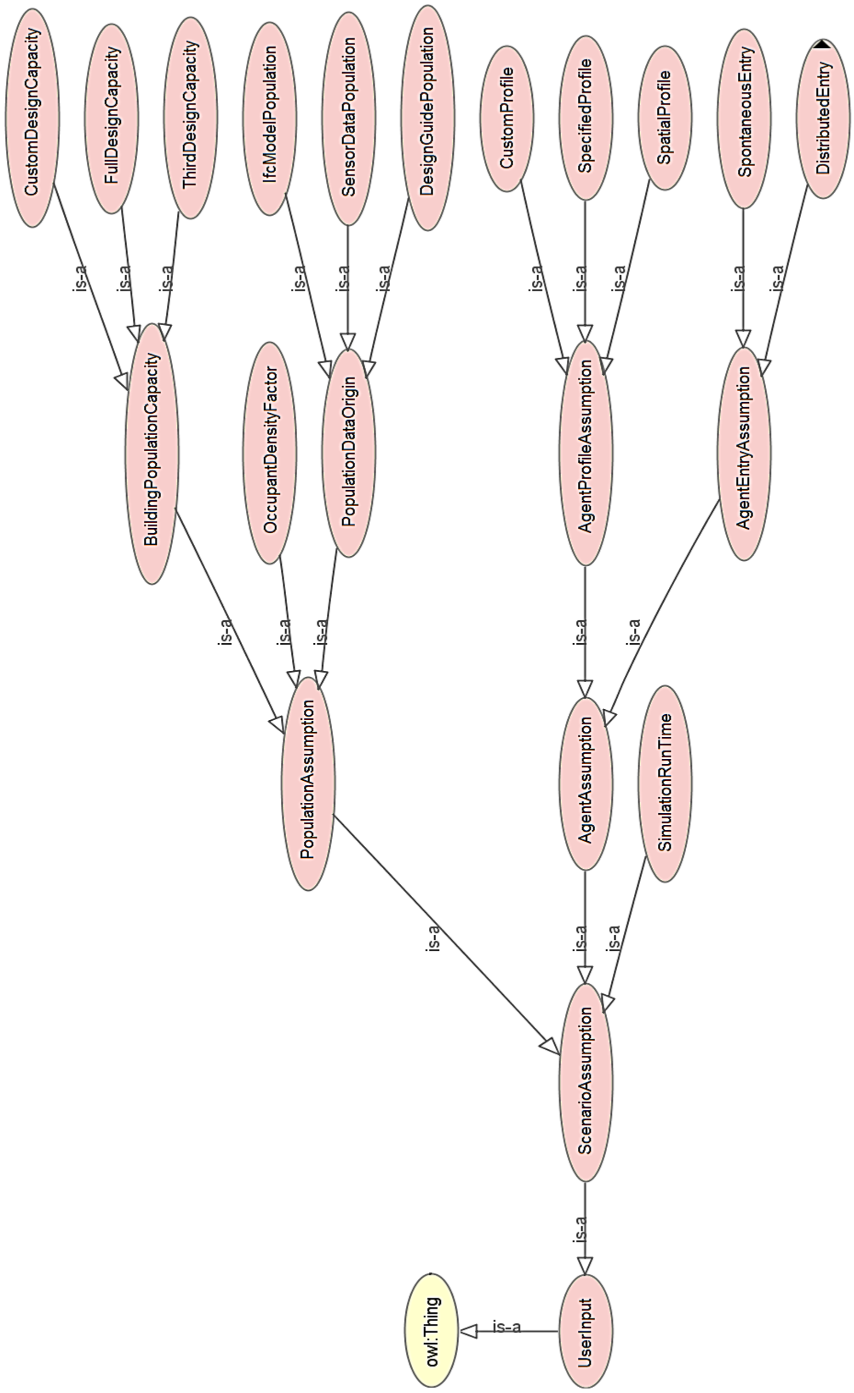


Figure 5-5. CSS ontology 'ScenarioAssumption' class hierarchy

- ➢ *Where is the population data coming from?*

None of the CSTs to date offer any capability of automatically populating a model with agents on a realistic premise. This is largely due to each building design being different and assumed building occupancy factors changing with region. However, as pointed out in Section 4.2.1, there are several viable resources where population data can be retrieved, hence the subclasses for the ***PopulationDataOrigin*** keeping track of them.

Another important factor in the context of automation and feedback, is the ability to represent where these assumptions are coming from, therefore differentiating between scenarios on larger scale of simulations, or when mining the resources for data at a future date, in different creative contexts.

Simulation results – competency question:

- ➢ *What types of outputs can a simulation have?*

It has been established in Section 4.1.2 that simulation outputs are usually presented to users via several analysis features such as tables or overlay maps. These have already been defined as ***AnalysisObjects*** in their own right. However, the data which they use is recorded in memory or databases, which are retrieved on user demand. The ***SimulationResult*** class (Figure 5-6) conceptualises the storage of relevant data which is retrieved on demand by the user. Its two main subclasses aim to differentiate between results at different points in time. Thus, ***EndResult*** encompasses definitive outputs, which are retrieved at the end of the simulation. For example, the ***TotalEgressTime*** is the time when all agents have safely evacuated the model, which is computed at the end of a simulation run. The ***IntermediateResult*** is meant store data ad-hoc, according to user objectives, and to provide data at certain times during a simulation. This is a special requirement for crowd simulation data as events and agent movement relates to ***SimulationTime***. Additionally, the performance of the design is monitored over time, thus being important for the analysis stage.

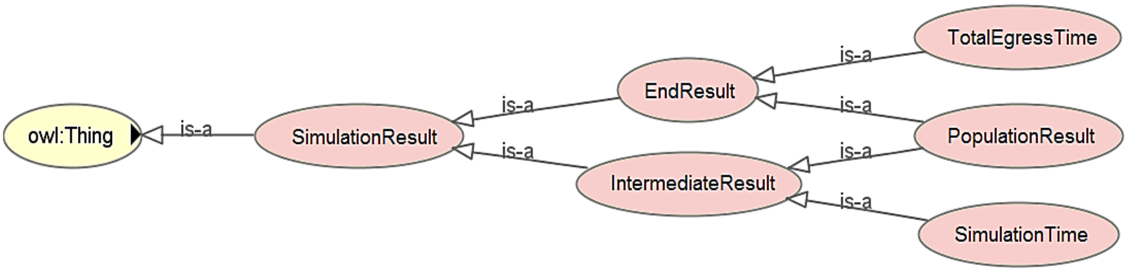


Figure 5-6. CSS ontology 'SimulationResult' class hierarchy

It is worth noting that this part of CSS ontology can be improved by conceptualising several other result types, depending on use case. However, it is highly dependent on the capabilities of the CST used, and the user requirements for analysing the results. More investigation into the matter is required for future work.

Agent relationships

Apart from the hierarchies presented above, there are several explicit semantic relationships between the defined classes. Full lists of data and object properties are provided Appendix B, along with their domains, ranges and other syntax constructs where necessary.

Within a simulation, the most dynamic objects are those describing agents. Figure 5-7 shows an example of the ***Agent*** class relationships to other sibling classes within the CSS ontology.

- *How are an agent's attributes defined?*

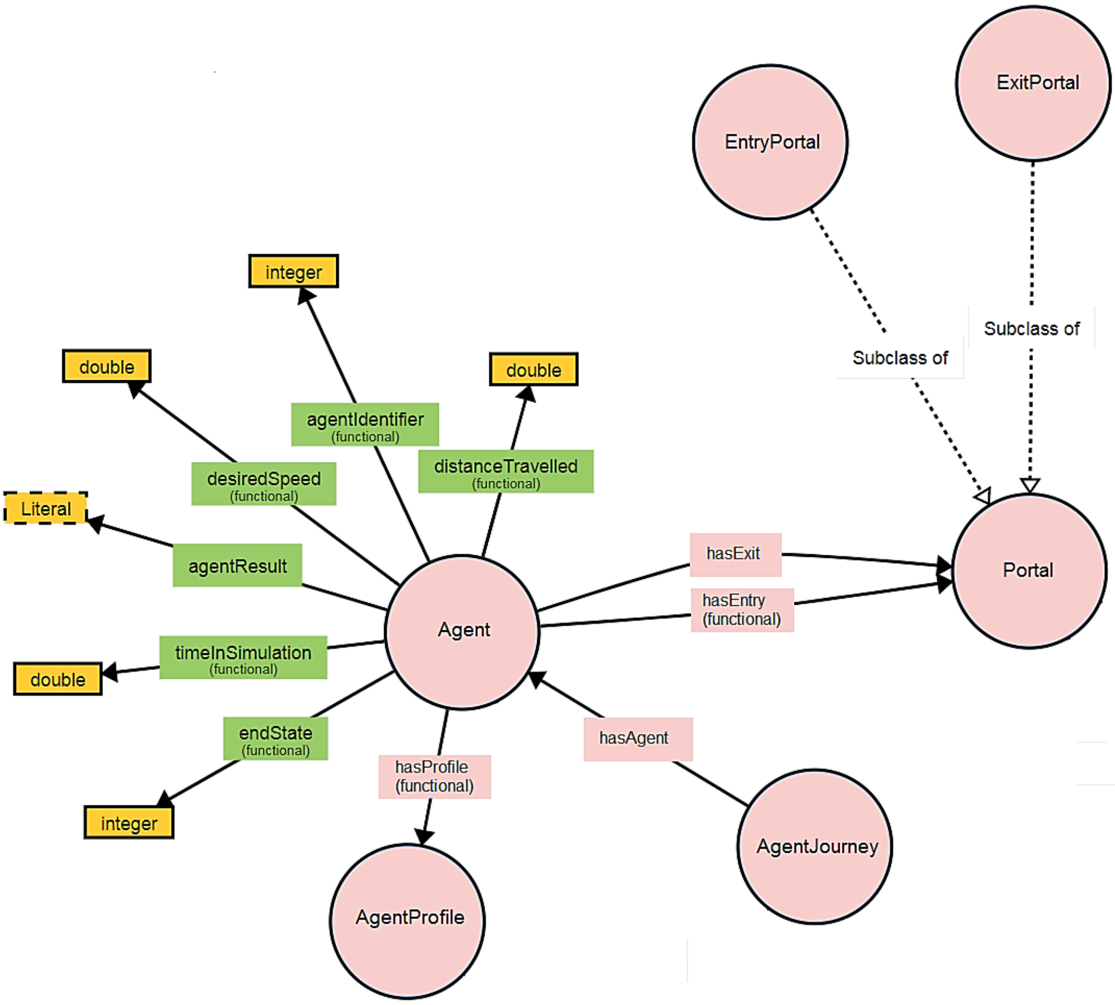


Figure 5-7. CSS ontology 'Agent' class relationships

An ***Agent*** individual has certain traits which are defined by ***AgentProfile class***, where its physical attributes such as movement speed and radius are stored.

- *Where does an agent enter the simulation and where does it leave it?*

Each agent has an entry and exit point within the model, which is done through ***Portal*** objects. Each agent must have at least one Portal as and entry point within the simulation, as described by ***hasEntryn (functional).*** The Agent can be allowed to use multiple exit points, described through the ***hasExit*** property. It is not excluded for an ***Agent*** to use the same ***Portal*** for both entering and exiting the model. Therefore, the properties between the ***Agent*** relating to portal objects are generalised at the ***Portal*** class level, as opposed to its two subtypes.

The properties shown in yellow in Figure 5-7 store agent specific data about its identity and behaviour within a model. These qualify as results data, recorded after a simulation execution, which are different from the assumed data already described by the ***AgentProfile*** class. For example, the AgentProfile assumes a speed of 1.2 m/s for each Agent at the start of a simulation, but the desiredSpeed stores 0.9 m/s; this is because although the Agent individual was trying to achieve the upper threshold, it may have been impeded by obstacles. Example competency questions on ***Agent*** data properties include:

- *Has an agent managed to exit the simulation safely?*
- *How much distance has an agent travelled until reaching the exit?*

The level of expressivity developed within the CSS ontology considers a detailed interaction between individuals which resemble programming objects, based on the taxonomy previously identified. Assumptions and results exist explicitly and relate to specific model objects, as well as to the overall scenario, through the use of property definitions. These conceptualisations can be leveraged to perform different knowledge mining techniques deployed in Chapter 6.

5.1.3. Representing the analysis feedback process

The previous chapter emphasised that CST outputs are provided in the form of a playback animations, graphs, density maps or tables, for user convenience. These were all conceptualised in the CSS ontology as individual types. However, to be able to find new knowledge about the design, certain feedback processes regarding output data needed to be defined. It was established in Section 4.2.2 that PIs are preferred when assessing model performance. However not all PIs have well defined threshold which objectively rank performance. Fruin’s LOS is one such a case, based on repeated research and observations. This is not the case for evacuation times. Design guides recommend certain evacuation values be decided by safety engineers (PD 7974 2004).

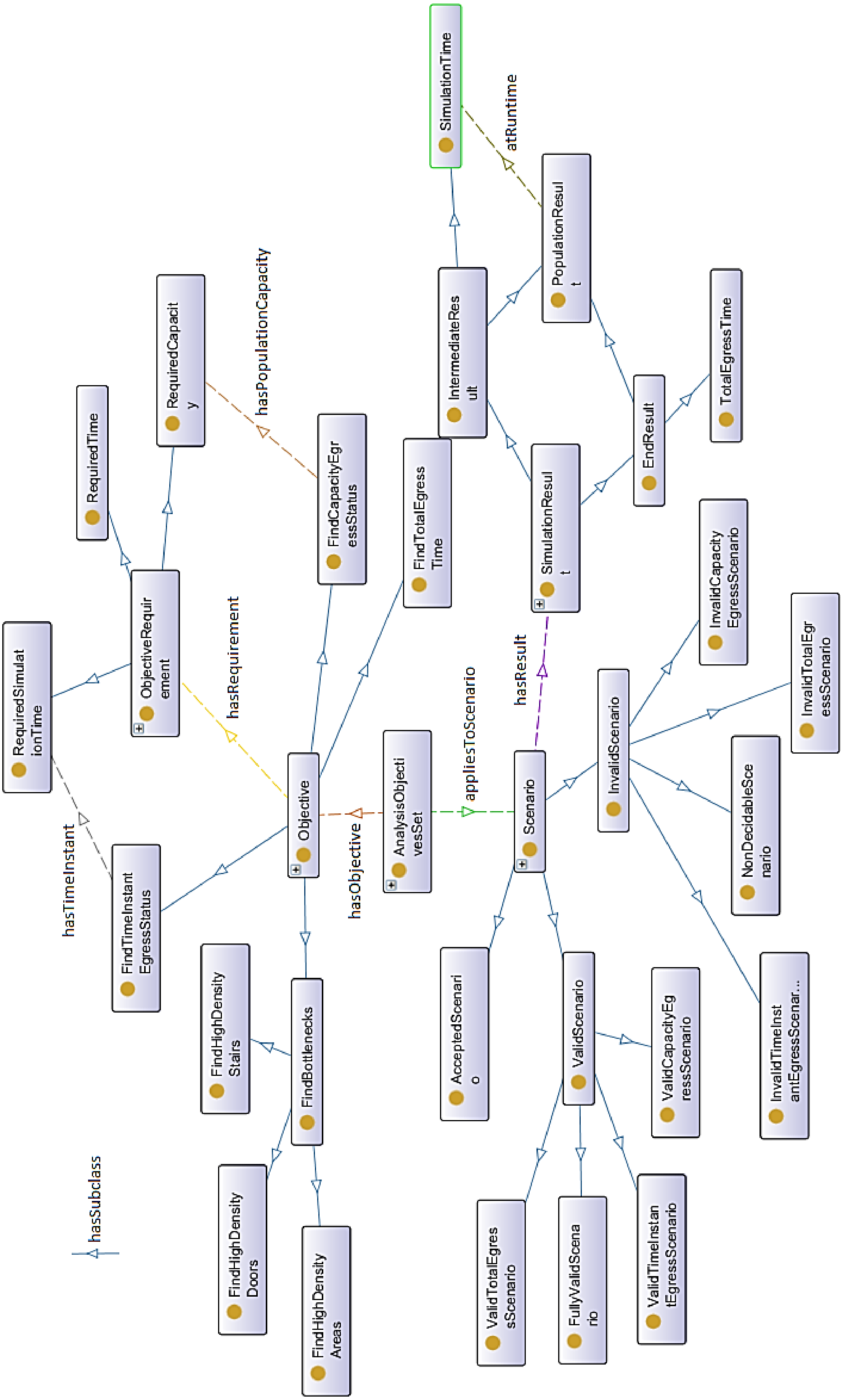


Figure 5-8. The Feedback Analysis (FBA) ontology

Ultimately, designers need to prove that the design is safe through accepted means (Shields and Silcock 1987). The Feedback Analysis (FBA) ontology was developed with these requirements in mind, with its main concepts shown in Figure 5-8. A full view of the FBA ontology, along with its metrics, object and data properties is provided in Appendix B.

The feedback process needs to be able to analyse results according to design objectives requested by the user:

- *How are user objectives and their requirements captured by the ontology?*

The ***Objective*** class conceptualises user analysis objectives, with user inputted threshold values for each instance stored by the ***ObjectiveRequirement*** class, connected through the ***hasRequirement*** property (see Figure 5-9).

- *What is the scope of one or multiple objectives in terms of scenarios to which it is applied?*
- *How can objectives be applied to scenarios on a large scale?*

The ***AnalysisObjectivesSet*** class (Figure 5-9) conceptualises a set of requirements from the user side which are applied to several models at the same time. ***AnalysisObjectivesSet*** can have multiple ***Objective*** individuals, each with its own ***ObjectiveRequirement***, allowing the definition of several PIs which can be simultaneously refer to multiple ***Scenario*** individuals. Thus, both properties ***hasObjective*** and ***appliesToScenario*** have 'one-to-many' relationship directed outwards to its relatives.

- *How are the results involved in the analysis process?*

The feedback process must access the simulation data and apply reasoning within given contexts. ***SimulationResult*** class conceptualises any generic results which belongs to specific ***Scenario*** individuals. The generic conceptualisation of the dependency relationships involved in the process are best described by the association of the two sets of triples:

'AnalysisObjectiveSet -> appliesToScenario -> Scenario' (1) AND

'Scenario -> hasResult -> SimulationResult' (2).

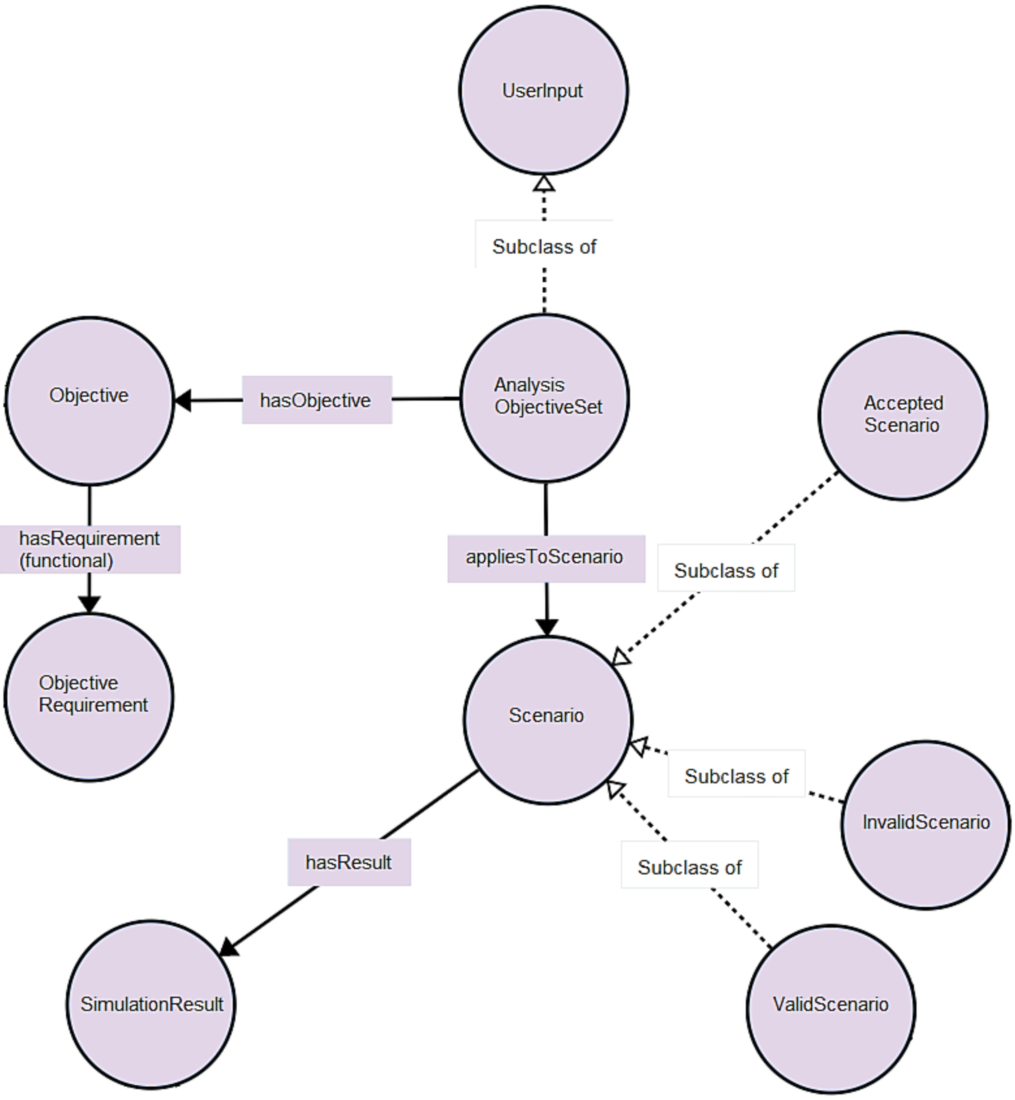


Figure 5-9. Main FBA classes in direct relationship with the 'Scenario' class

- *How can scenario models be classified and differentiated?*

The final concepts shown in Figure 5-9 are the subclasses ***InvalidScenario*** and ***ValidScenario*** for the ***Scenario*** class. These classes have been specifically implemented to deal with the knowledge mining process, where user objectives and scenario results are reasoned. The logical inferencing engine categorises each ***Scenario*** individual according to the result of the rule. Due to the requirements of the reasoning process, multiple subclasses for ***ValidScenario*** or ***InvalidScenario*** have been implemented, each corresponding to the ***TRUE*** or ***FALSE*** rule results. Figure 5-10 shows the properties defining the objectives and their requirements for two use cases which were developed and tested. The ***FindCapacityEgressStatus*** class for example, is used to query the status at a certain time in a simulation when a certain specified population percentage has been evacuated. The requirement is expressed via the ***RequiredCapacity*** class in this context, with its relationships shown in Figure 5-10.

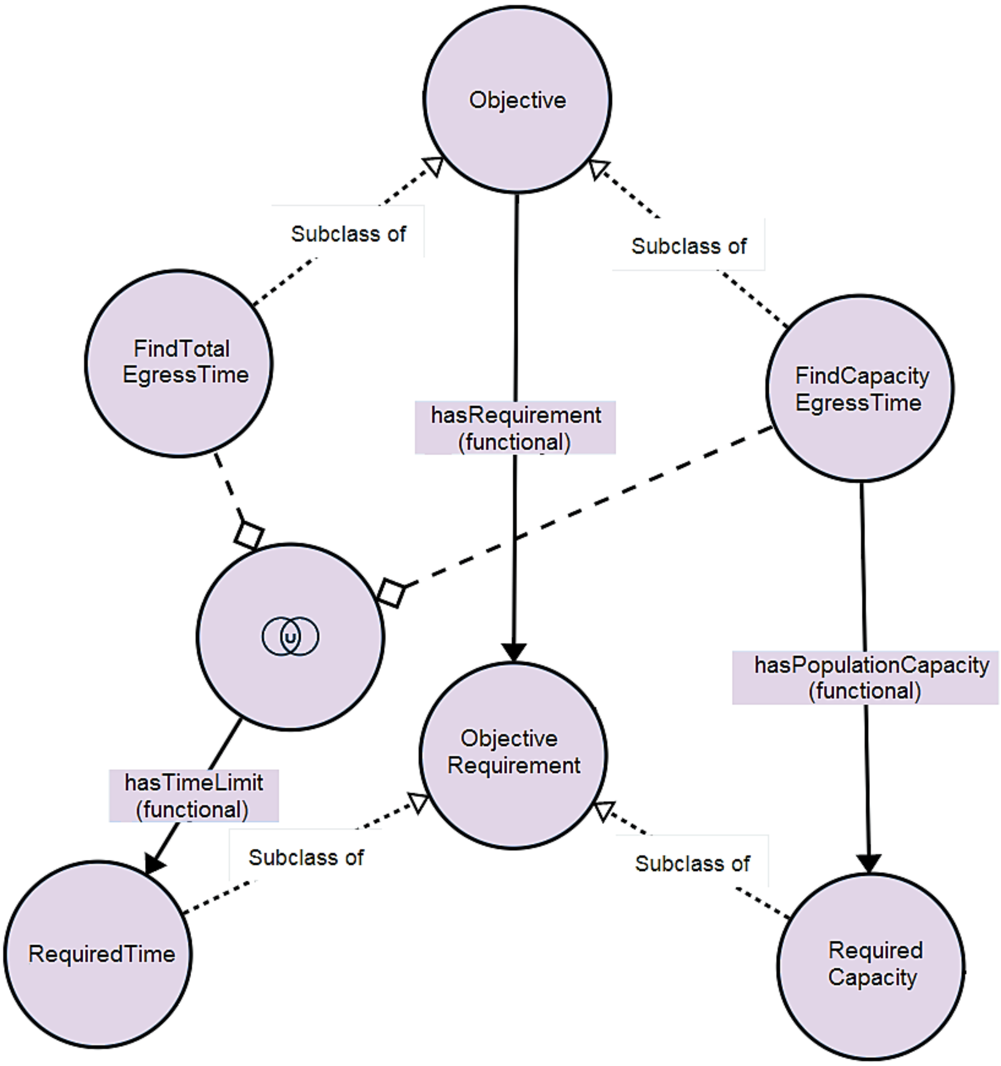


Figure 5-10. FBA ontology concepts capturing user objectives and requirements

Thus, the development of concepts relies on user objectives to be used. Only the use-case which have been used in testing have been developed for the FBA ontology, following the generic concepts described above. It is acknowledged that more consultation with engineers and designers is required to establish a full list of objectives and their requirements as they are used in practice.

5.1.4. Representing design codes

In Section 4.2.1 several required factors were identified in order to make a valid simulation scenario. One vital factor relates to the number of occupants within the model. This can be done several ways, as suggested in Figure 4-2. Design guidance recommends the most representative way of reality to be used (PD 7974 2004). If specific design data on occupancy is not available, such as in early design stages, there are alternative means to estimate occupancy using design regulations. The UK Approved Documents (The Building Regulations 2015) series provide means of measurement for occupancy densities.

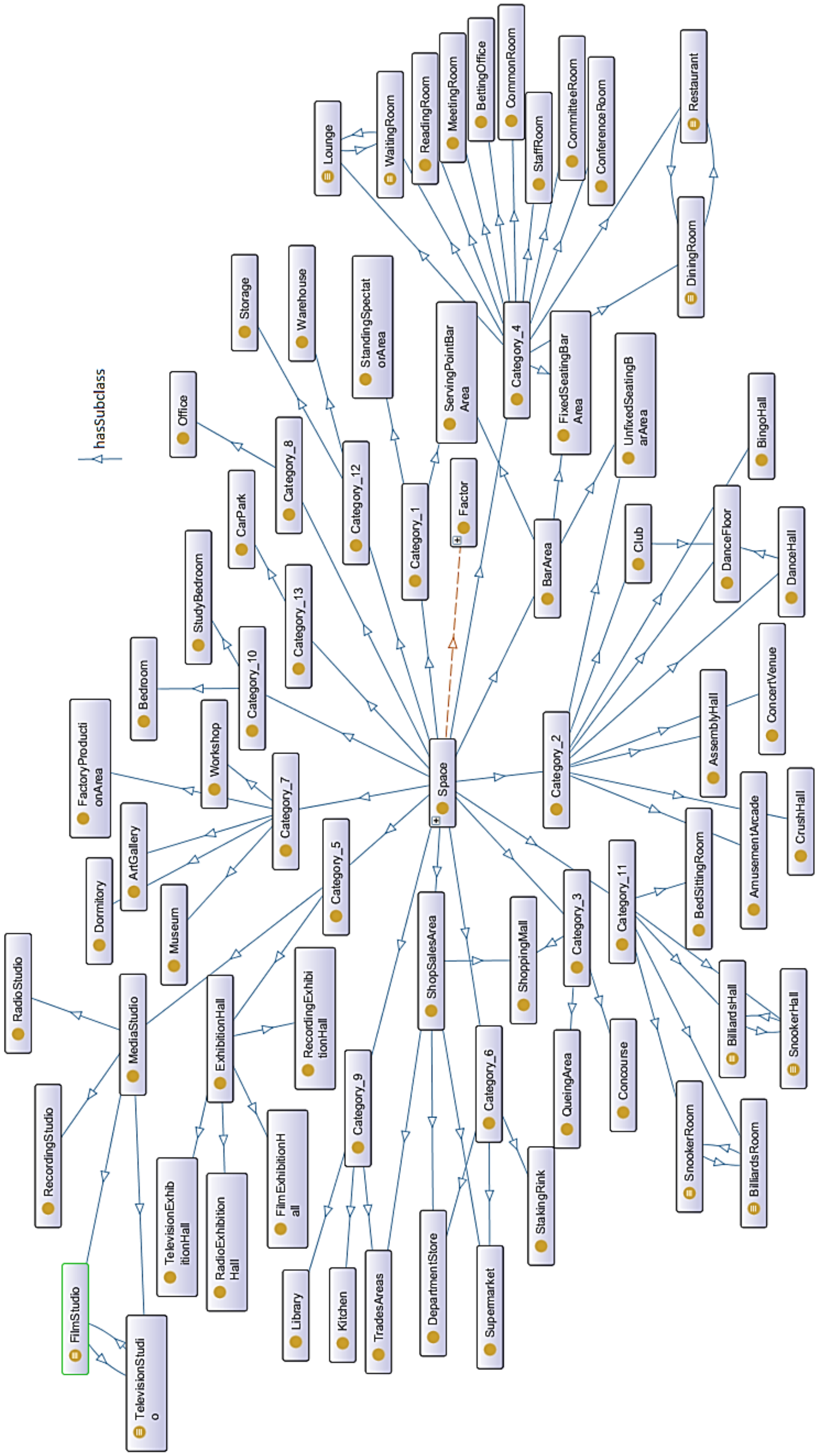


Figure 5-11. The UK Spaces Occupant Capacities (UKSOC) ontology

To showcase the capabilities of semantic web data integration as part of the overall research goal, an ontology was developed which expresses UK Spaces Occupant Capacities (UKSOC), with its full classes and individuals shown in Figure 5-11. A full view of the ontology is provided in Appendix B, along with metrics, object and data properties.

The ontology is a representation of design knowledge from Table 5-2, which refers to spaces and regions within a facility. Although this represents a simple table for types of spaces and their associated factors, the exact ontological representation of this was required to correctly assume the density factors for each space type.

During its development, it was observed that certain elements were vague, or in some cases redundant. Certain categories had the same factors, without any evident motive. At the same time, some types from the same category were marked with certain exceptions. Another serious remark is that the table itself is inconsistent as it fails to distinguish between spaces, zones or regions. For example, a ‘dance floor’ space can be very different from a ‘dance hall’. Additionally, a ‘dance hall’ can include a ‘dance floor’ space.

The finalised UKSOC ontology represents a more concise categorisation of the spaces, based on the available information. Any ambiguity left is a result of the initial table’s inconsistencies mentioned. From the 15 categories of spaces, 12 emerged in the ontology, with each category being assigned a different factor for population density, ranging from 0.5 to 30 m^2/person.

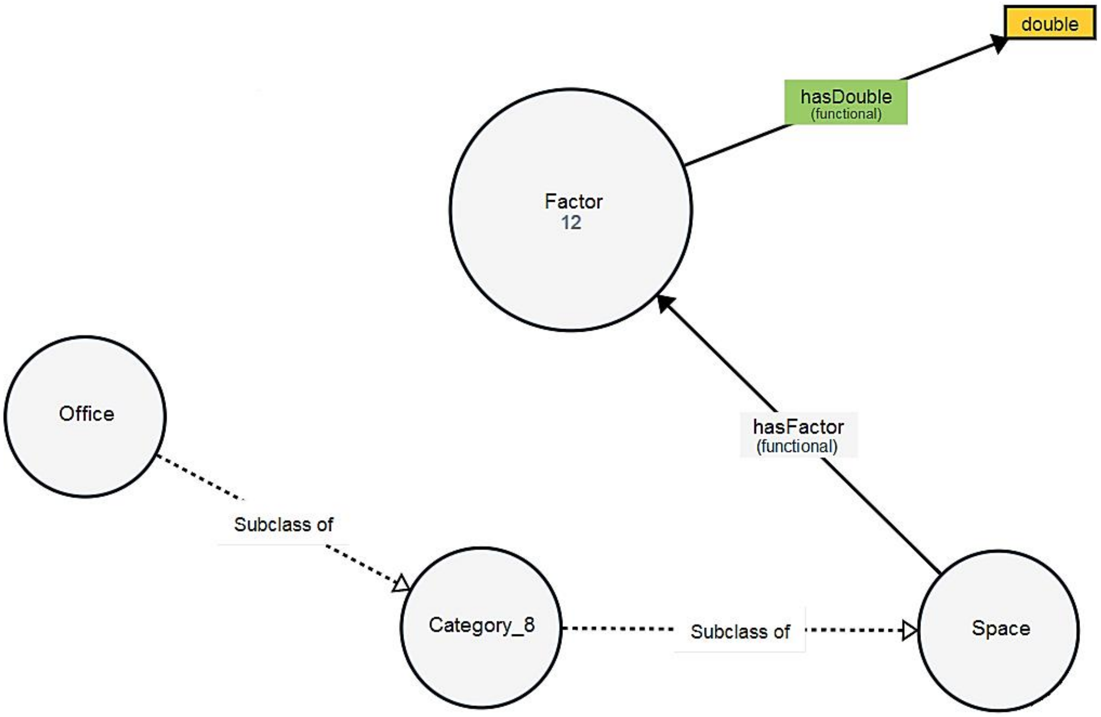
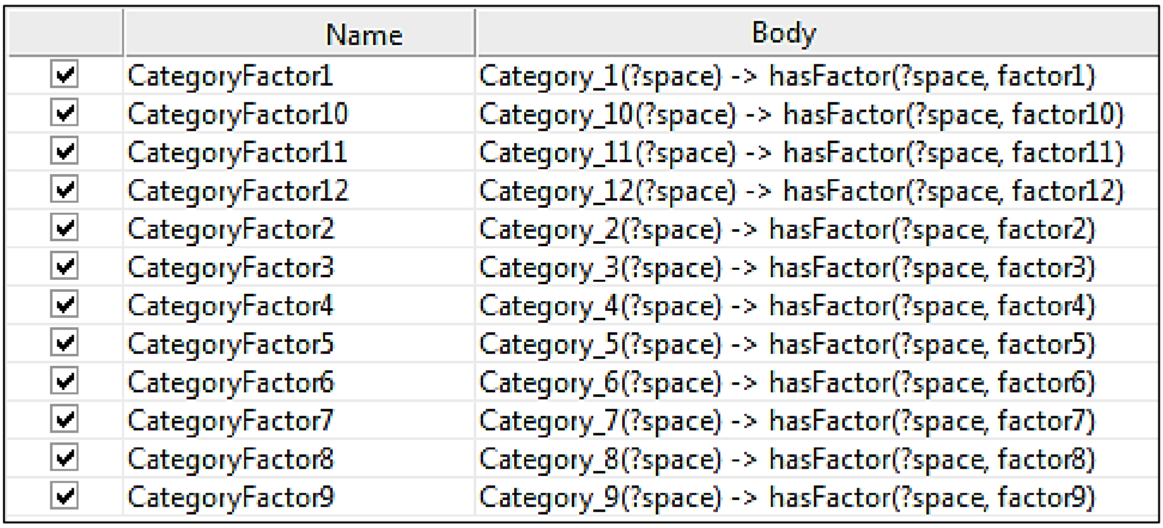


Figure 5-12. UKSOC ontology ‘Space’ class and its relationships

The basic relationships present are shown in Figure 5-12, where each category of space has a specific factor assigned. The factors were represented as ontology individuals for double type numbers. For simplicity, instead of mapping each factor to each space type, a set of SWRL rules were constructed (Figure 5-13), each rule attributing a certain factor for each ontology individual from its respective category.



	Name	Body
☑	CategoryFactor1	Category_1(?space) -> hasFactor(?space, factor1)
☑	CategoryFactor10	Category_10(?space) -> hasFactor(?space, factor10)
☑	CategoryFactor11	Category_11(?space) -> hasFactor(?space, factor11)
☑	CategoryFactor12	Category_12(?space) -> hasFactor(?space, factor12)
☑	CategoryFactor2	Category_2(?space) -> hasFactor(?space, factor2)
☑	CategoryFactor3	Category_3(?space) -> hasFactor(?space, factor3)
☑	CategoryFactor4	Category_4(?space) -> hasFactor(?space, factor4)
☑	CategoryFactor5	Category_5(?space) -> hasFactor(?space, factor5)
☑	CategoryFactor6	Category_6(?space) -> hasFactor(?space, factor6)
☑	CategoryFactor7	Category_7(?space) -> hasFactor(?space, factor7)
☑	CategoryFactor8	Category_8(?space) -> hasFactor(?space, factor8)
☑	CategoryFactor9	Category_9(?space) -> hasFactor(?space, factor9)

Figure 5-13. UKSOC ontology SWRL rules

The reliance of using design codes such as occupancy factors remains unclear, and is investigated in Chapter 7, along with several test case studies measuring the efficiency of the ontology and its rules above.

Table 5-2. Spaces occupant capacities (adapted from The Building Regulations (2015) Annex C3 – Methods of measurement)

Table C1 Floor space factors (1)		
Type of accommodation (2)(3)		Factor (m^2/pers)
1	Standing spectator areas, bar areas (within 2m of serving point) similar refreshment areas	0.3
2	Amusement arcade, assembly hall (including a general purpose place of assembly), bingo hall, club, crush hall, dance floor or hall, venue for pop concert and similar events and bar areas without fixed seating	0.5
3	Concourse, queuing area or shopping mall (4)(5)	0.7
4	Committee room, common room, conference room, dining room, licensed betting office (public area), lounge or bar (other than in 1 above), meeting room, reading room, restaurant, staff room or waiting room (6)	1
5	Exhibition hall or studio (film, radio, television, recording)	1.5
6	Skating rink	2
7	Shop sales area (7)	2
8	Art gallery, dormitory, factory production area, museum or workshop	5
9	Office	6
10	Shop sales area (8)	7
11	Kitchen or library	7
12	Bedroom or study-bedroom	8
13	Bed-sitting room, billiards or snooker room or hall	10
14	Storage and warehousing	30
15	Car park	2/pers
Notes	1. As an alternative to using the values in the table, the floor space factor may be determined by reference to actual data taken from similar premises. Where appropriate, the data should reflect the average occupant density at a peak trading time of year.	
	2. Where accommodation is not directly covered by the descriptions given, a reasonable value based on a similar use may be selected.	
	3. Where any part of the building is to be used for more than one type of accommodation, the most onerous factor(s) should be applied. Where the building contains different types of accommodation, the occupancy of each different area should be calculated using the relevant space factor.	
	4. Refer to section 5 of BS 5588-10:1991 Code of practice for shopping complexes for detailed guidance on the calculation of occupancy in common public areas in shopping complexes.	
	5. For detailed guidance on appropriate floor space factors for concourses in sports grounds refer to “Concourses” published by the Football. Licensing Authority (ISBN: 0 95462 932 9).	
	6. Alternatively the occupant capacity may be taken as the number of fixed seats provided, if the occupants will normally be seated.	
	7. Shops excluding those under item 10, but including - supermarkets and department stores (main sales areas), shops for personal services such as hairdressing and shops for the delivery or collection of goods for cleaning, repair or other treatment or for members of the public themselves carrying out such cleaning, repair or other treatment.	
	8. Shops (excluding those in covered shopping complexes but including department stores) trading predominantly in furniture, floor coverings, cycles, prams, large domestic appliances or other bulky goods, or trading on a wholesale self-selection basis (cash and carry).	

5.1.5. Representing a crowd simulation tool

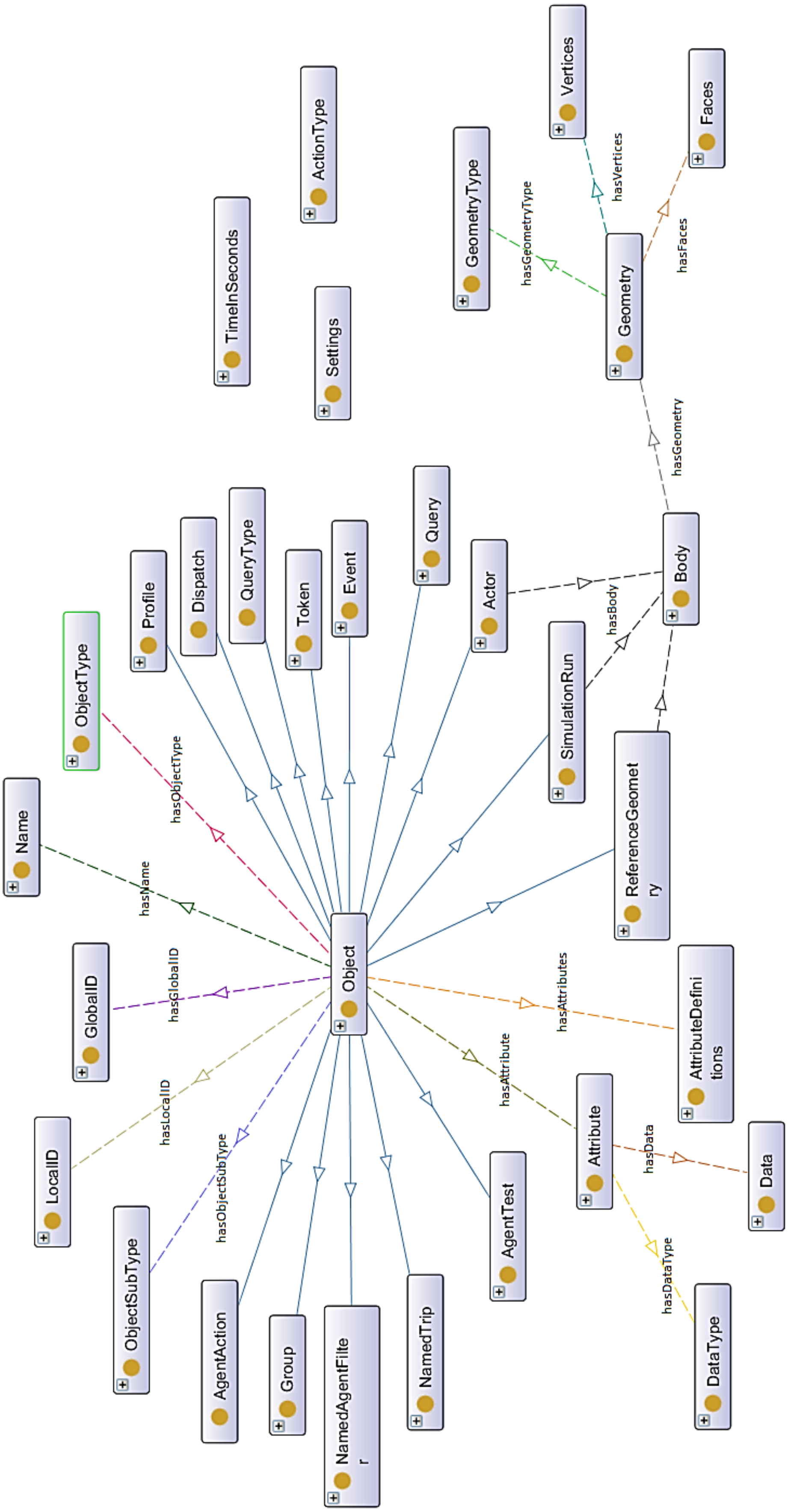


Figure 5-14. The MassMotion (MM) ontology

The previous ontologies represent the crowd simulation domains and analysis workflows around them for high-level knowledge storage and retrieval processes. However, in practice, each model has to run on specific tools, thus it must be able to connect to other semantic web resources. The MassMotion (MM) ontology was developed (Figure 5-14), which represents MassMotion (Oasys Limited 2018a) concepts in a very object-oriented nature, describing the structure of the tool. The concepts were developed over several iterations from testing the software capabilities and are based on the structure of the MassMotion simulation files. The XML format of the files have a clear hierarchy of objects, with fully labelled properties. A full view of the MM ontology is provided in Appendix B.

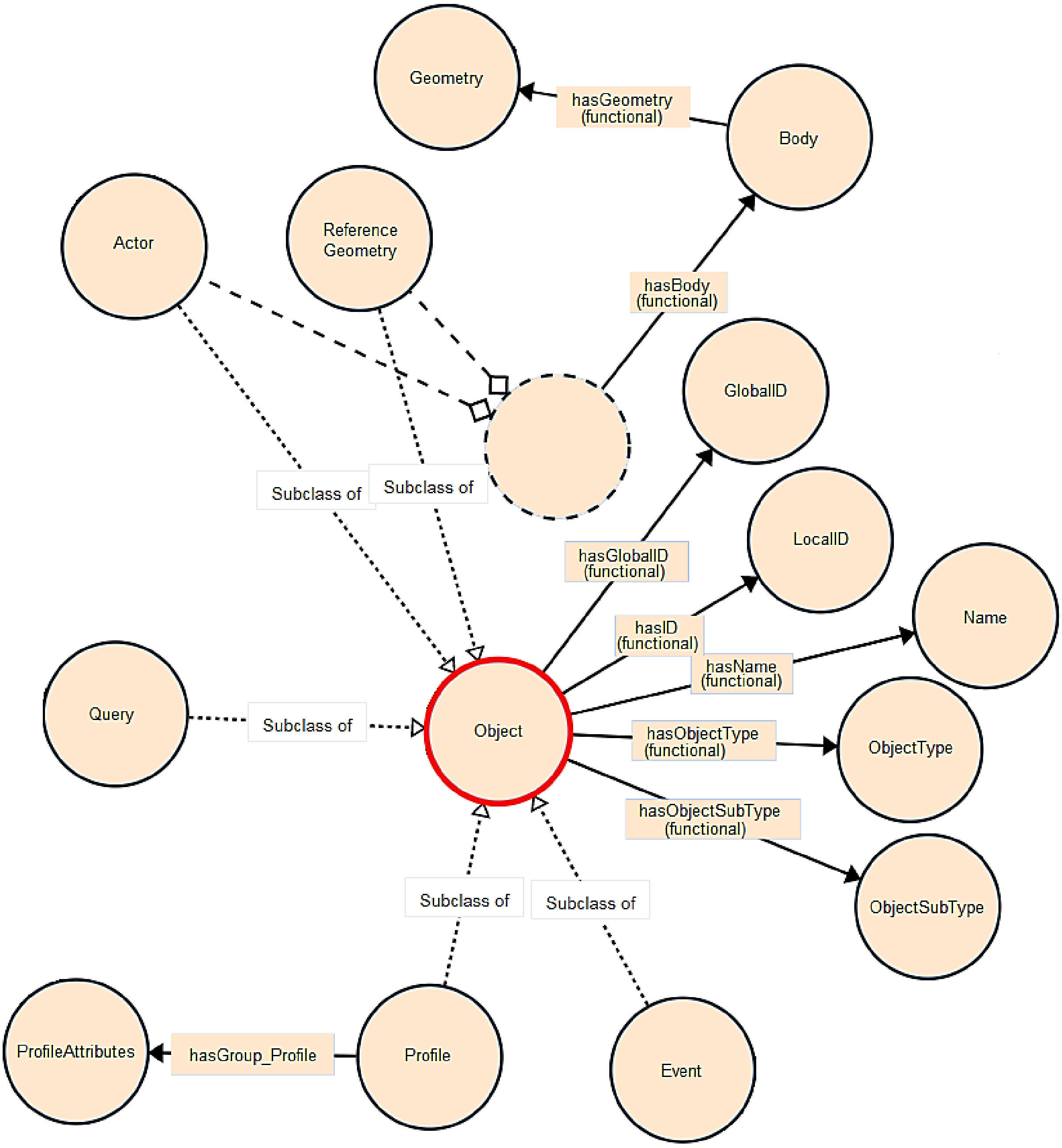


Figure 5-15. MM ontology “Object” class with important subclasses and properties

Due to the object-oriented nature of the software, the ontology expresses 247 classes, many of which describe data and nested objects. It was observed that the common concepts closely resemble the taxonomy hierarchy concluded in 4.1.2, and the CSS ontology. This is also evident from the example in Figure 5-15. The ***Object*** class has some generic properties related to identity data. Its main subclasses are:

1) ***Actor*** & ***Reference Geometry*** – all objects which have a 3D representation within the model, hence the ***hasBody*** relationship toward ***hasGeometry***;
2) ***Event*** – the superclass for all event types, describing actions during model execution
3) ***Profile*** – the object which contains agent properties
4) ***Query*** – the superclass in charge with defining analysis objects and their associated properties

It was observed that the ontology of a CST can differ significantly from its outlined features (Appendix A), as it is highly dependent of the design of the software itself, and on how the data is structured internally.

5.2. Ontological alignment of concepts

"Ontologies can be large, with tens, hundreds, or even thousands of classes and properties. Trying to take stock of such a complex framework of concepts can be daunting. There is active research into techniques to automate the process, but at this point, the task must ultimately be done by humans. While current tools can calculate class name and graph similarity metrics to try to give suggestions, they cannot yet consistently align ontologies automatically." (Hebeler et al. 2011)

Dibley (2011) mentions that there are multiple ways in which ontologies can be aligned, but one of the most reliable for the OWL language is to look at the similarities between concepts, specifically at the terminology and structures. OWL ontologies can be compared considering (Euzenat and Valtchev 2004):

- Terminology – comparing names of entities (including the use of a dictionary to identify equivalence);
- Internal structure – ranges and cardinality of attributes;
- External structure – comparing the relationships between concepts, such as hierarchies and their groupings.

5.2.1. Aligning digital building and crowd simulation models

Current engineering practice sees BIM at the core of the design process, being considered the central point of truth for all related information. The aim of this research is to maintain and extend this view to the CS knowledge domain. It was previously established that the IFC model provides all the necessary data about the building environment. The two schemas have been connected allowing the CS and IFC knowledge domains to collaborate. Ontology representations of the schemas were mapped across two very different hierarchies of classes. Nevertheless, several common concepts were found, shown in Figure 5-16.

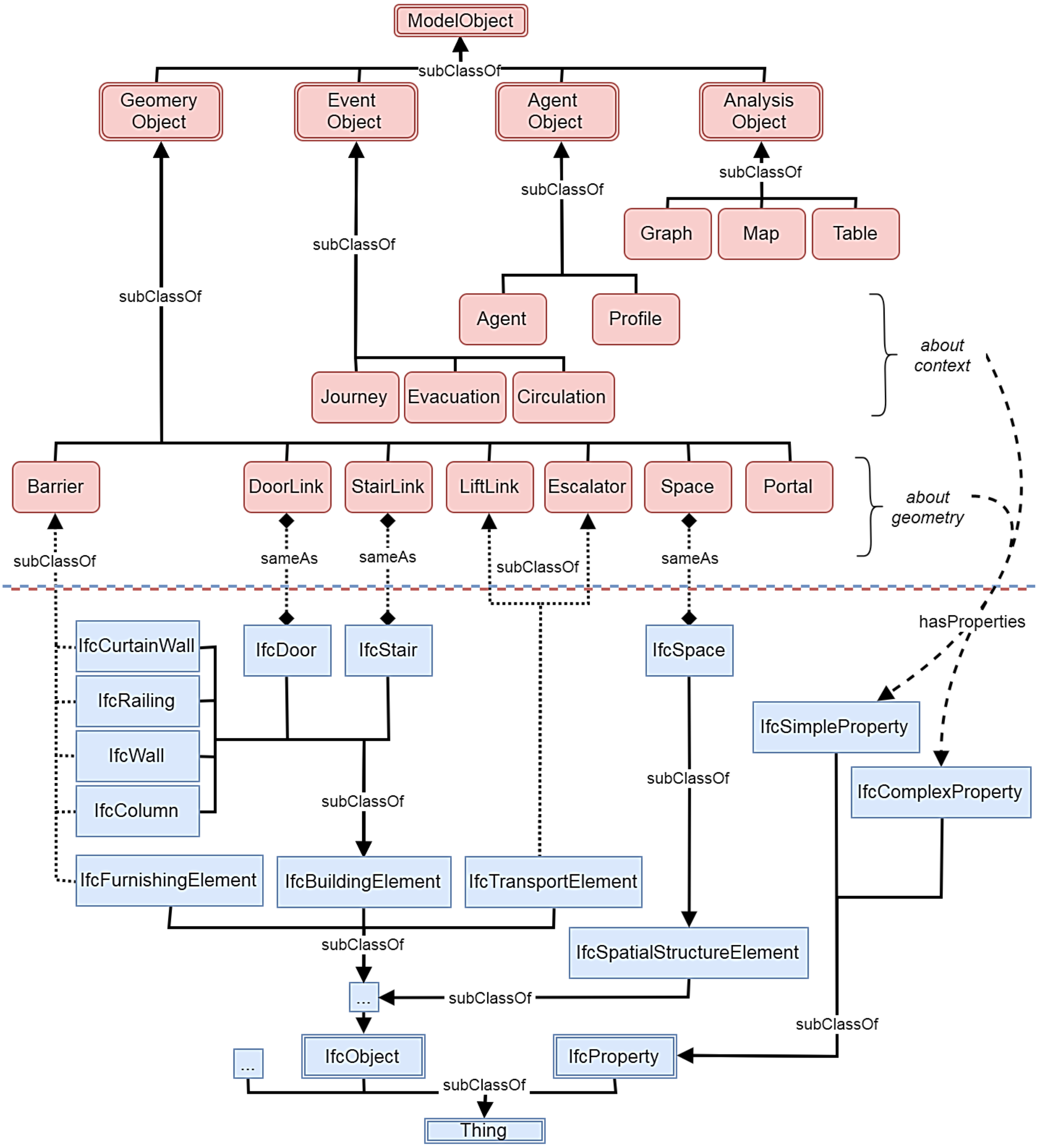


Figure 5-16. Alignment of classes between the CSS and IfcOwl ontologies

Despite the IfcOwl containing over a thousand classes, a relatively small number of classes were directly aligned. These are mostly those describing objects with geometric representations. The classes for ***IfcWall***, ***IfcColumn***, etc. are classified as a ***subClass*** of ***Barrier***. Even though in the IFC domain they are distinct entities, they all fulfil the same role: blocking the movement of agents. The fact that there are multiple types of ***Barrier***, which are distinct in IfcOwl, means that the ***owl:sameAs*** axiom is not sufficient. Yu (2014) states that the ***owl:sameAs*** *"is often used to link one individual to another, indicating the two URI references actually refer to the same resource in the world."* The entities of ***IfcDoor***, ***IfcStair*** and ***IfcSpace*** were identified as the only reasonable cases of declaring equivalency, where there is very little ambiguity. This approach is also confirmed in part by crowd simulation tools which import the IFC format.

The hierarchy of entities represented in IfcOwl is very complex as it reflects the IFC schema which is object-oriented. This gives rise to some limitations when expressed in ontology formats, as it can make rules and alignment of data and individuals challenging, as well as slow for extraction. From practical experience whilst conducting the research, this is especially true when referring to the geometry data. This issue was identified and addressed by several studies in an attempt to improve query times and make the data within IfcOwl more accessible (Farias et al. 2015, Pauwels et al. 2017).

While the common objects are related to geometry, there can be major differences in how the geometry is represented. The most well-known crowd models, such as the CA models, rely on mesh geometry objects. In its current state, IFC and IfcOwl store geometry in a compact way, storing basic constructs which need to be extracted and used to generate more complex shapes. This makes the alignment of geometry constructs between IfcOwl and the CSS impossible via name matching, and impractical via knowledge rules, thus a more direct approach is recommended as a way around this limitation. The ONTOCS system was developed to simply retrieve geometry, convert it in memory and explicitly store any relevant geometry in the other ontologies. Retrieving contextual information using properties is outlined in more detail in Chapter 6, as this process has to rely on knowledge operators.

5.2.2. Aligning scenario and feedback analysis models

The CSS ontology views the model through the prism of 'what is?' and 'what is happening when?' by defining geometry, agents and events. On the other hand, the FBA ontology views the same model from the prism of 'why is?' and 'what is the cause of what is happening?' by working with the results data. The CSS ontology has the role of linking information explicitly, while the FBA ontology focuses on finding information from explicit and implicit relationships.

The Figure 5-17 shows an example of aligned classes and object properties. A full mapping is provided in Figure 5-18. The ***Scenario*** class is present in both ontologies referring to a specific simulation scenario of a model. In the CSS it has the role of including all the model instances which define the environment, events and results data, and any assumptions made for each particular case. In the FBA ontology, this the same concept has the function of evaluating whether a ***Scenario*** meets different objectives, which are evaluated over results. In this sense, the ***EndResult*** class is defined in both ontologies in the same way, and therefore all relationships linked to it are equivalent in both ontologies, as those marked in green or half-green in Figure 5-17.

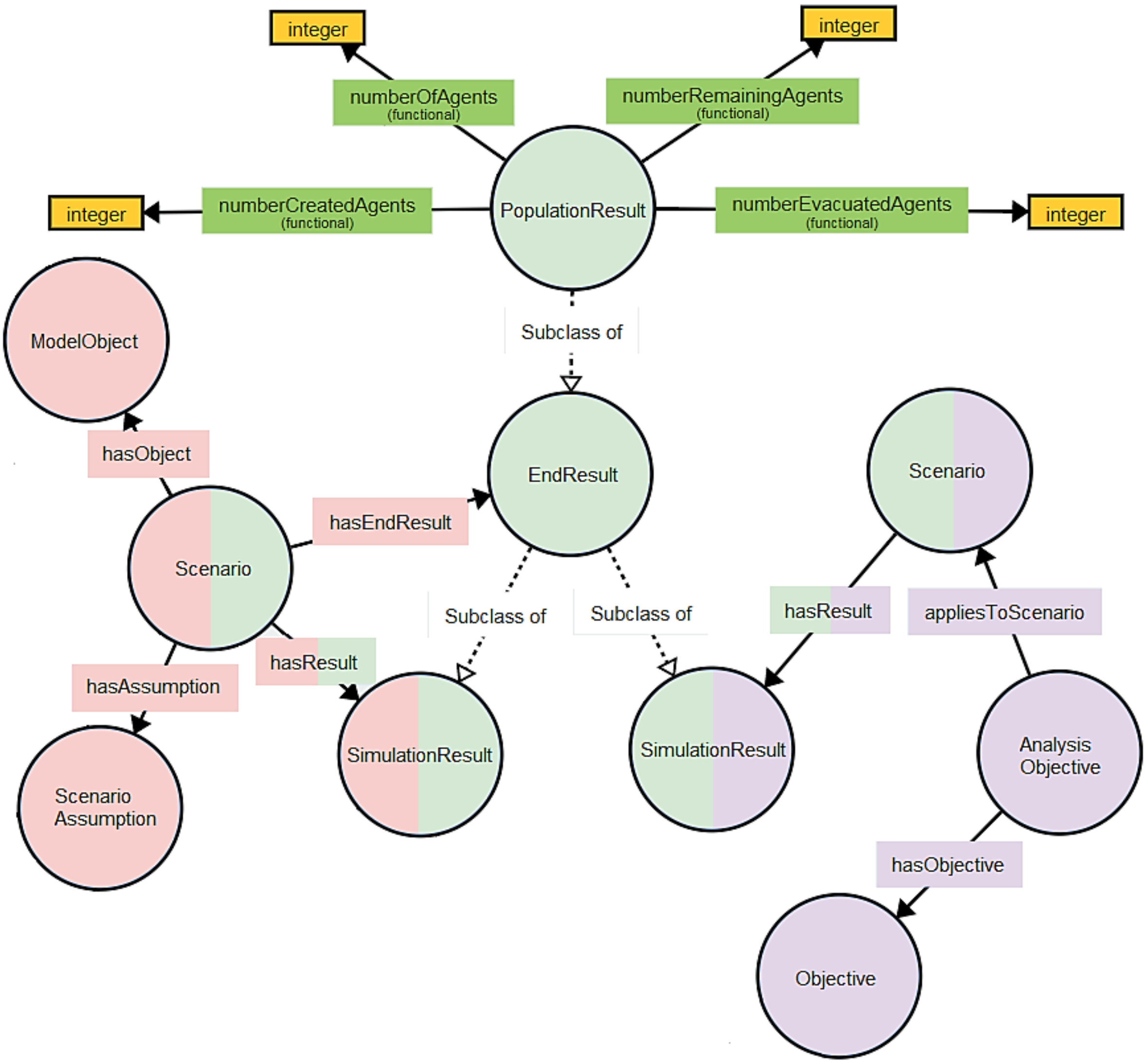
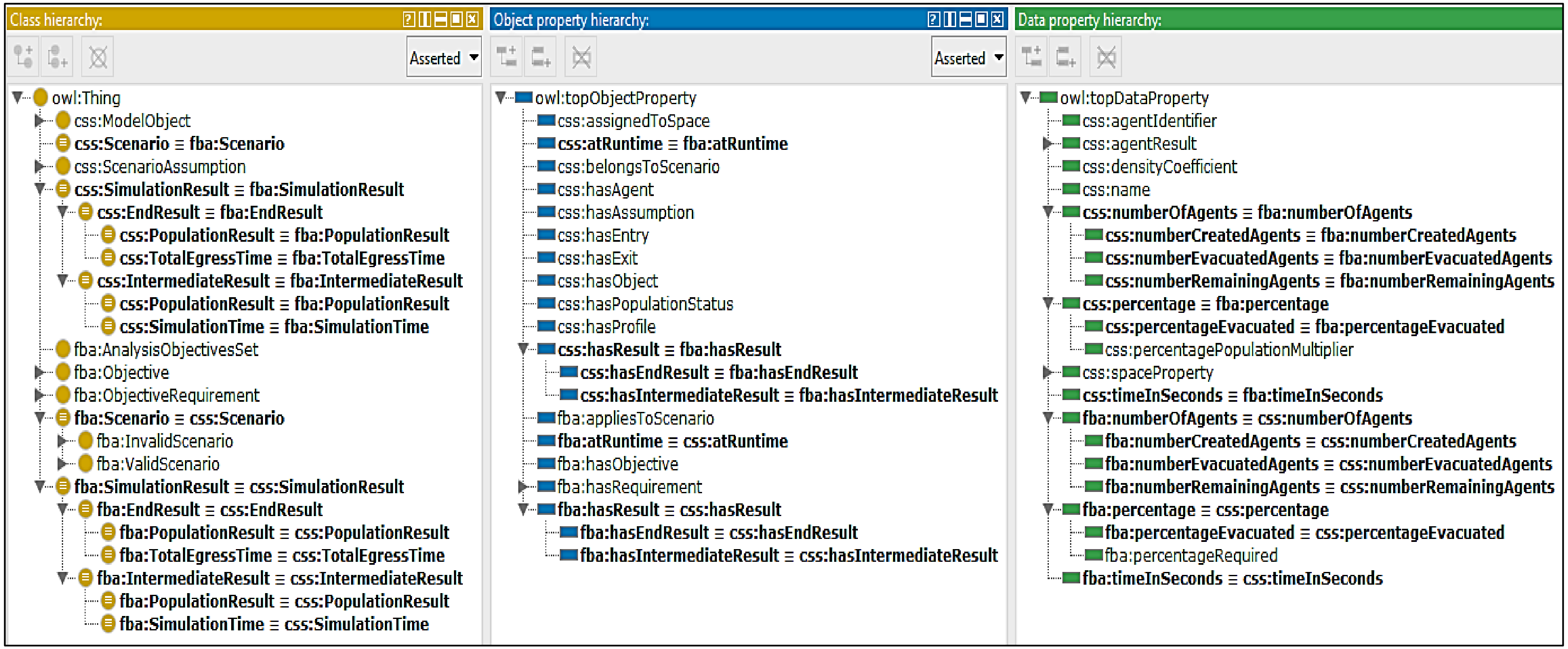


Figure 5-17. Alignment of main classes between the CSS with FBA ontology

The creation of objectives and results classes is an interlinked process because each objective depends on specific results. Therefore, these need to be created in each context and then mapped for the integration of the two models.



Class hierarchy:

- owl:Thing
  - css:ModelObject
  - css:Scenario ≡ fba:Scenario
  - css:ScenarioAssumption
  - css:SimulationResult ≡ fba:SimulationResult
    - css:EndResult ≡ fba:EndResult
      - css:PopulationResult ≡ fba:PopulationResult
      - css:TotalEgressTime ≡ fba:TotalEgressTime
    - css:IntermediateResult ≡ fba:IntermediateResult
      - css:PopulationResult ≡ fba:PopulationResult
      - css:SimulationTime ≡ fba:SimulationTime
  - fba:AnalysisObjectivesSet
  - fba:Objective
  - fba:ObjectiveRequirement
  - fba:Scenario ≡ css:Scenario
    - fba:InvalidScenario
    - fba:ValidScenario
  - fba:SimulationResult ≡ css:SimulationResult
    - fba:EndResult ≡ css:EndResult
      - fba:PopulationResult ≡ css:PopulationResult
      - fba:TotalEgressTime ≡ css:TotalEgressTime
    - fba:IntermediateResult ≡ css:IntermediateResult
      - fba:PopulationResult ≡ css:PopulationResult
      - fba:SimulationTime ≡ css:SimulationTime

Object property hierarchy:

- owl:topObjectProperty
  - css:assignedToSpace
  - css:atRuntime ≡ fba:atRuntime
  - css:belongsToScenario
  - css:hasAgent
  - css:hasAssumption
  - css:hasEntry
  - css:hasExit
  - css:hasObject
  - css:hasPopulationStatus
  - css:hasProfile
  - css:hasResult ≡ fba:hasResult
    - css:hasEndResult ≡ fba:hasEndResult
    - css:hasIntermediateResult ≡ fba:hasIntermediateResult
  - fba:appliesToScenario
  - fba:atRuntime ≡ css:atRuntime
  - fba:hasObjective
  - fba:hasRequirement
  - fba:hasResult ≡ css:hasResult
    - fba:hasEndResult ≡ css:hasEndResult
    - fba:hasIntermediateResult ≡ css:hasIntermediateResult

Data property hierarchy:

- owl:topDataProperty
  - css:agentIdentifier
  - css:agentResult
  - css:densityCoefficient
  - css:name
  - css:numberOfAgents ≡ fba:numberOfAgents
    - css:numberCreatedAgents ≡ fba:numberCreatedAgents
    - css:numberEvacuatedAgents ≡ fba:numberEvacuatedAgents
    - css:numberRemainingAgents ≡ fba:numberRemainingAgents
  - css:percentage ≡ fba:percentage
    - css:percentageEvacuated ≡ fba:percentageEvacuated
    - css:percentagePopulationMultiplier
  - css:spaceProperty
  - css:timeInSeconds ≡ fba:timeInSeconds
  - fba:numberOfAgents ≡ css:numberOfAgents
    - fba:numberCreatedAgents ≡ css:numberCreatedAgents
    - fba:numberEvacuatedAgents ≡ css:numberEvacuatedAgents
    - fba:numberRemainingAgents ≡ css:numberRemainingAgents
  - fba:percentage ≡ css:percentage
    - fba:percentageEvacuated ≡ css:percentageEvacuated
    - fba:percentageRequired
  - fba:timeInSeconds ≡ css:timeInSeconds

Figure 5-18. Alignment of all concepts between the CSS and FBA ontologies

5.2.3. Aligning design codes and classification criteria

The UKSOC ontology has limited applicability by itself but was used in gathering information about building models for testing. This was linked with the IfcOwl and an Uniclass RDF dataset currently in development (Bradley 2017). The latter is an RDF graph of the Uniclass 2015 classification tables, available at (NBS 2017). The scope of the Uniclass ontology is much broader as it aims to categorise all types of building elements, from which the UKSOC specifically aligns with spaces. The used Uniclass RDF dataset ontology was developed as part of another research project at Cardiff University and was only used for testing for the purposes of this research, but no contribution to its development is claimed.

There are over 500 space types present in the Uniclass2015, but only 70 were successfully identified and mapped directly to the UKSOC categories, with example of the mappings done for the first two UKSOC categories in Table 5-3 (see full list in Appendix B). Because design codes refer to spaces in a very generic manner, including entire sub-type categories from Uniclass2015 into the UKSOC was required.

Table 5-3. Example of aligned concepts between the UKSOC and Uniclass

Category	No	UKSOC	UNICLASS 2015		
		Type of space	Uniclass equivalent		Uniclass categories
		Description	Code	Title	Sub-group Title
1	1	Standing spectator areas	SL_90_20_83	Spectator standing areas	Common spaces
	2	Bar areas (within 2m of serving point)	SL_40_20_06	Bars	Dining spaces
2	4	Amusement arcade	SL_40_05_03	Amusement arcades	Amusement spaces
	5	Assembly hall	SL_25_10_05	Assembly halls	Educational spaces
	6	Bingo hall	SL_40_05_43	Indoor play spaces	Amusement spaces
	7	Club	SL_40_60_21	Dance floors	Performing arts spaces
	8	Crush hall	SL_90_10_27	Entrance halls	Circulation spaces
	9	Dance floor	SL_40_60_21	Dance floors	Performing arts spaces
	10	Dance hall	SL_40_60_21	Dance floors	Performing arts spaces
	11	Venue for pop concert and similar events	SL_90_20_05	Audience standing areas	Common spaces
	12	Bar areas without fixed sitting	SL_40_20_06	Bars	Dining spaces

Spaces in the Uniclass2015 and UKSOC are differentiated by the function they achieve. In Uniclass, the spaces are categorised hierarchically, with semantics at the level of a taxonomy. For example, an office sub-heading includes several specific office types: postal office, admin office, etc. In the fire safety domain, and therefore in the UKSOC, spaces are viewed by their number of inhabitants. For example, a dance hall and a bar area, belong to the same category, because both types tend to have similar population densities. This presented a lot of issues when mapping between the two different domains, as is shown with the ***Bars*** types spaces in Table 5-3, where ambiguity exists. During the mapping process it was concluded that the design codes would need to be more concise and include several other types of spaces. However, as previously mentioned, the most accurate population data about a building is encouraged to be used (PD 7974 2004). As such, the most realistic source of information would be the specified number of inhabitants for each space as inputted by safety engineers.

Due to the difference between the two ontologies, several alignment options were considered. The first one considered including Uniclass2015 categories within UKSOC classes. Due to the large number of categories present in Uniclass, this was considered impractical. The second choice was to perform alignment at an instances level, using rules. 56 rules were developed, each mapping specific Uniclass identifiers to category factors, with 2 examples shown in Table 5-4. The rules are essentially the implementation of the extended version of Table 5-3 (in Appendix B) which aligns spaces based on their similarity in name and function.

Table 5-4. Example of UKSOC and Uniclass alignment rules (Appendix B)

NO	Rule name	SWRL code
24	CF-Category_4-13-BreakoutSpaces	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_90_20_08") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
25	CF-Category_5-01-ExhibitionHall	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:containsIgnoreCase(?id, "SL_25_50") -> uksoc:hasFactor(?spaceClass, uksoc:factor5)

The question remains on how reasonable it is to assume the context of a CSM based on the UKSOC design factors, which is investigated through testing in Chapter 7.

5.3. Ontology validation

All developed ontologies were created using the Protégé tool, which includes several built-in reasoners. Each ontology was checked for inconsistencies at several stages throughout its development, using these reasoners. Any inconsistencies were corrected before actual testing using the ONTOCS system and implicitly the Stardog RDF server. All RDF stores which are created dynamically on Stardog during ONTOCS testing sessions operate using SL reasoning capability (a combination of RDFS, QL, RL, and EL axioms, plus SWRL rules). This is the most advanced reasoning type provided by the server tool, allowing reasoning using SWRL rules, SWRL built-ins, and other axioms. This provided a very good basis for testing the ontologies and their performance in practice.

The previous sections outlined the ontology structures in parallel to the more relevant competency questions, which the ontologies are able to answer. This is further proven in Chapter 7, where all the ontologies function correctly within the boundaries of the ONTOCS system and correctly provide answers to over 33 SPARQL queries many of which operate in conjunction with SWRL rules. The SPARQL queries are the practical implementations of competency questions which were also discussed with several industry experts from the fields of crowd simulation modelling.

Although not familiar to the field of ontology engineering, the consulted experts provided some valuable feedback and commented on the correctness of the approach. The comments were not always related to the ontology's structure, but more about the nature of a crowd simulation model and how things are considered in design. Here are some paraphrased example comments:

1) When analysing a layout, each room would have designated exits.
2) In preliminary design, it's more common to look directly at occupancy factors;
3) Add 'AcceptedScenario' class, meaning that a designer is satisfied with its performance;

Concerning the first comment, the object property ***hasDesignatedExit*** was added between several subclasses of ***Space***, which have the ***RefugeSpace*** class type as an object; this now recognises that a ***RefugeSpace*** is not just an exit point for an ***Agent***, but also a desired exit appointed to a ***Space***. A set of rules could be implemented to force agents to follow this appointed exit; however, this was not implemented as it will restrict the level of freedom for agents.

Although not related to the ontology correctness, a test-case was raised and investigated in Chapter 7 concerning the second commend, aiming to identify how different design codes taken from ontologies are, in comparison to data from reality.

The third comment was addressed by adding the ***AcceptedScenario*** class to the FBA ontology, as a subclass of ***Scenario***. However, in its current state, the FBA ontology is unable to categorise any individual as this class, due to missing mechanisms to capture this part of user input. This will be addressed in future work.

5.4. Summary

This chapter introduced the knowledge based developed using OWL ontologies. These represent both the ‘information models’ and ‘processes’ as part of the overall system framework described in Chapter 3. The chapter first outlined a high-level view of the ontologies used by the ONTOCS system and how they relate, following an in-depth view of each developed ontology. The CSS ontology sits at the centre of the models and processes, fully conceptualising the CS domain, aided by the FBA ontology to formalise analysis and feedback mechanisms. The MM ontology sits under the CSS to allow a practical collaboration of information and data from tool level to SW level. The UKSOC ontology was developed from existing design guidance to allow estimation of model population on the fly. In the final section the alignment between several ontologies was introduced presenting their challenges and limitations. The ontologies are used to represent information and knowledge about the crowd simulation domain. These are the means to facilitate automation, knowledge retrieval and storage using various methods outlined in Chapter 6.

Chapter 6. Storing and retrieving knowledge

This chapter outlines the development of several knowledge operators within the CS knowledge domain which facilitate the retrieval of knowledge in two very specific contexts: (1) for automatic CSM generation from available information models and resources, by allowing the ONTOCS system to 'understand' them, and (2) for processing simulation results analysis on a large scale taking into consideration user input objectives. In parallel to this, this chapter aims to show the necessities for achieving these processes using ontologies and knowledge rules, thereby answering research question Q6. The chapter introduces the knowledge operators used (6.1), storage of knowledge methods (6.2), and knowledge retrieval methods (6.3) developed for automation of information within the CS domain.

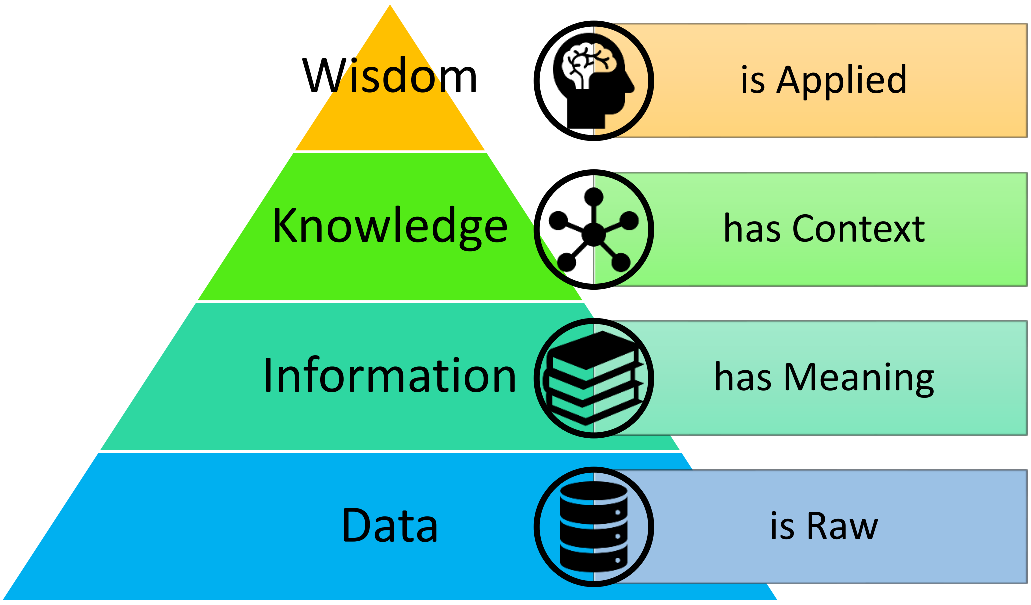


Figure 6-1. Pyramid with increasing levels of meaning

Yao et al. (2007) citing Bellinger et al. (2004) states that there are several layers that need to be considered when dealing with information and knowledge management systems, as shown in Figure 6-1. This is similar to the concepts of the Semantic Web and Linked Data defined in Section 2.3, where ‘meaning’ of data increases at each level. It is important to understand however that with each layer computation time to process the data also increases significantly.

Knowledge management systems have been neglected in the past in the AEC industry due to unclear benefits towards construction data management (Grover and Froese 2016). This is now having the opposite effect as the loose structure of the product chain in the AEC sector is breaking after every project and causing a lot of disruption when compared to other industries like plane manufacturing (Hardin 2009). Historically, the application of knowledge management tools has been limited on organisational level in terms of enterprise management with little implementations in practice when it concerns managing knowledge about a design process of a specific project. However, in light of the BIM lifecycle paradigm and increasing need for smart cities (Bejay Jayan 2016, Howell 2017), this may soon change, with a need to keep more knowledge about the building and its design for future uses.

6.1. Knowledge operator types

The previous chapter showcases how knowledge and information models can be expressed using ontology programming languages. The conceptualisation of knowledge is a process which involves humans expressing it in a computer understandable format, where semantic and logic rules are formalised and adhered to. This whole process of knowledge engineering is done manually, and not all knowledge is stated explicitly, so as to keep a focused scope of an ontology to a specific domain or use-case. However, ontologies are meant to be re-used in other domains, and designer need to find creative ways of retrieving knowledge models. To be able to formulate or retrieve more knowledge out of a system of this nature, Ullrich (2003) explains that there is a need for inferring and querying, acts which are able to exploit the rich expressivity of an ontology. Kaufman and Michalski (2005) already make use of inferencing using logical rules, which are considered operators over knowledge models. Logical rules are often referred to as ***reasoning rules***.

The next section (6.1.1) introduces the concepts of rules and their role in knowledge engineering. It is worth mentioning that for the purpose of this research, rules are regarded as operators for knowledge as introduced in Chapter 3. In addition to that, they are also able to represent knowledge in similar ways to ontologies. Rules are an attractive prospect in many applications since users find it easier to formulate knowledge, rather than go through more extensive ontological axioms (Krötzsch 2010). However, in the case of describing a complex system, such as ONTOCS, OWL ontologies alone are insufficient (Motik and Rosati 2010, Abdul-Ghafour et al. 2014) because of relying of distributed resources in many information formats. Additionally, Hebeler et al. (2011) mention that ontologies are not as flexible when working with data as rules are, and that they are most useful when there is a need to change the structure of the data from one knowledge domain to another. As such, ontologies are complemented by reasoning rules (Motik and Rosati 2010), and in this particular case they are used to imbed knowledge which is later retrieved using processes operating on model data.

6.1.1. Reasoning rules. SWRL

"A rule could be any statement which says that a certain conclusion must be valid whenever a certain premise is satisfied, i.e. any statement that could be read as a sentence of the form "if…then…" […] it is worth noting that the term 'rule' as such refers rather to a knowledge modelling paradigm than a particular formalism or language."

(Krötzsch 2010)

Many ways exist in which rules are defined in computer science. This is a field which has emerged since early computers, but they are all based on a logical inference operation.

"…the general form of a decision (or classification) rule is:

$$\mathit{CONSEQUENT} \Leftarrow \mathit{PREMISE} \;\lfloor\!\_\; \mathit{EXCEPTION}$$

where CONSEQUENT is a statement indicating a decision, a class, or a concept name to be assigned to an entity (an object or situation) that satisfies PREMISE, provided it does not satisfy EXCEPTION" *(Kaufman and Michalski 2005)*

By looking closely at the two definitions above, slight variations in a rule form exist. The second definition is different as it can express exceptions, whereas the first only expresses the condition. This is due to several types of rules which are applied in practice. Below are a few examples of different types of rules:

- Logical rules – example: "if (X) is true, then (Y) is also true". Uses logic implication, or inferencing, as described above; they act as an extension to a knowledge base and they are usually restricted by Open World Assumptions (OWA); they are ***declarative*** in nature;
- Procedural rules – example: "if (X), then do (Y); else do (Z)". A very explicit type of rules which makes them ***operational*** in nature as they are able to express the consequences of both "true" and "false" conditions; they are ***operational*** because they control the flow of action;

- Logic programming rules – example: “man(X) <- person(X), which is not a woman(X)”. These approximate logical semantics with procedural aspects from above types and are ***semi-declarative.***

As it can be seen, there are many ways of creating rules, however each type has its own limitations when it comes to practice, depending on several factors such as:

- Expressiveness – the degree to which it can express knowledge;
- Clarity of its semantics and syntax;
- Declarative vs operational in nature (as shown above);
- Performance in computation and reasoning;
- Level of support by software tools;
- Practical applicability;
- Ease of use, etc.

Ontologies were introduced in Section 2.3.1 as tools for a semantic web paradigm. It was established that their level of expressivity is high and that they are widely used. Previous chapters have also hinted to why ontologies are better suited for representing knowledge in terms of information models for evacuation design. With these justifications in mind and considering enumerated factors above the SWRL language was found the best suited for the aims of this research.

The overarching research methodology for the ONTOCS framework follows the direction employed by Kaufman and Michalski (2005) which was restricted at the time with applying operators on raw data with aims of identifying patterns. The conceptualisation of knowledge was implemented using inductive databases. It is worth mentioning that at that time, semantic web tools were just on the verge to become more popular in practice. Since then, a more common way to represent knowledge databases is with the use of RDF graphs and OWL ontologies. On another note, Krötzsch (2010) mentions that a large portion of knowledge modelling is strongly focused on using terminologies, thus resulting in a schema type model, due to Description Logics (DLs) having become more popular. This is possible in part because they can be used to describe things explicitly as they are in the real world, with various applications ranging from medicine, software engineering to language dictionaries. In contrast, rule languages are better suited for working with large datasets, allowing more flexibility and expressivity to the data (Krötzsch 2010).

SWRL stands for ‘Semantic Web Rule Language’ and was officially published by Horrocks et al. (2004) under W3C. The initial SWRL included several built-in functions, which can operate on datatypes, i.e. compare or add integers and strings. This gives it the possibility to manipulate and analyse data in its basic form, while working on top of

OWL classes and individuals. The main limitation in expressing ontologies using the early version of OWL is a good example on why rules are often required, as best explained by Horrocks (2005):

"[…] while the language includes a relatively rich set of class constructors, the language provided for talking about properties is much weaker. In particular, there is no composition constructor, so it is impossible to capture relationships between a composite property and another (possibly composite) property. The standard example here is the obvious relationship between the composition of the "parent" and "brother" properties and the "uncle" property."

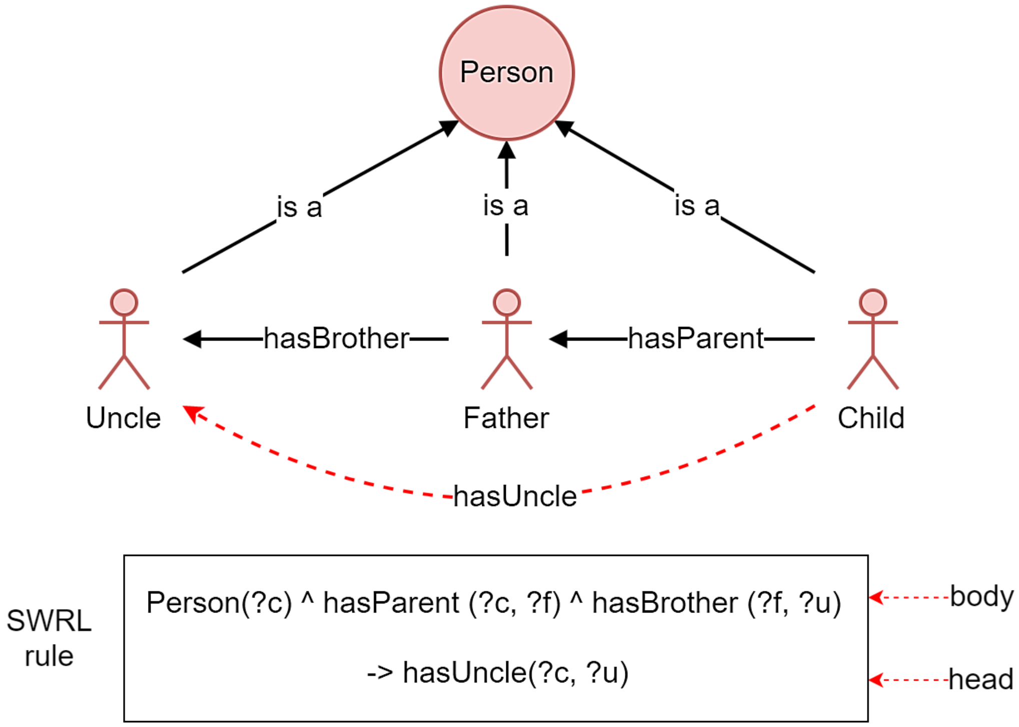
This example is shown in Figure 6-2, where the classes of ***Child***, ***Father*** and ***Uncle*** are all subclasses of the ***Person*** class, each with different object property relationships between them. The arrows pointing left from Child, can be seen to go in sequence toward Uncle. This represents a composite property and is summarised by the property ***hasUncle***, which allows a direct new relationship from ***Child*** to ***Uncle***. This sort of relationship is expressed in the box in Figure 6-2 in SWRL code. It would translate literally into:

If c is Person AND c hasParent f AND f hasBrother u

-> THEN c hasUncle u, where:

c, **f** and **u** are variables (ontology individuals), marked with **?**

^, **->** are **AND**, **THEN** respectively



```
Person(?c) ^ hasParent (?c, ?f) ^ hasBrother (?f, ?u)
-> hasUncle(?c, ?u)
```

Figure 6-2. Conceptualisation of a property chain using SWRL

In Figure 6-2 it can be observed that the rule can be split up in functional parts, which in practice are called ***atoms***. In this case, the ***body*** of the rule has three ***atoms***, each connected by the intersection operator (**^**). The ***head*** of the rule has one ***atom*** and can only infer one individual as part of the reasoning process.

While problems such as the example have been addressed with the release of OWL2, the importance of SWRL rules has not changed. However, with the assertion of such types of axioms is expected to make the reasoning and querying process slower (Krötzsch et al. 2011). At the same time, rules are better suited when working with large datasets. Adapting an ontology with SWRL rules makes that ontology 'undecidable' meaning that it is impossible to draw all logical conclusions from a knowledge base, even with unlimited time and resources. To account for this, DL-safe rules are SWRL rules restricted to known individuals. DL-safe rules are very expressive and decidable at the same time (Sikos 2015).

6.1.2. Queries. SPARQL

Several methods have been outlined by which knowledge can be represented and inferred. However, in order to access it in practical applications, this knowledge needs to be interrogated using programming queries.

A query language is *"a computer programming language used to retrieve information from a database"* (Slamecka and Hosch 2008).

The most common queries used in practice are based on SQL which operate on relational databases. Graph databases host RDF graphs and express data quite differently from relational databases, as they store SPO triples. To deal with querying graph databases, SPARQL was developed, which is a recursive acronym for "SPARQL Protocol and RDF Query Language".

"SPARQL is essentially a graph-matching query language. A SPARQL query is of the form H ← B, where B, the body of the query, is a complex RDF graph pattern expression that may include RDF triples with variables, conjunctions, disjunctions, optional parts, and constraints over the values of the variables, and H, the head of the query, is an expression that indicates how to construct the answer to the query." (Perez et al. 2006)

"Most forms of SPARQL query contain a set of triple patterns called a basic graph pattern. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable." (Prud and Seaborne 2006)

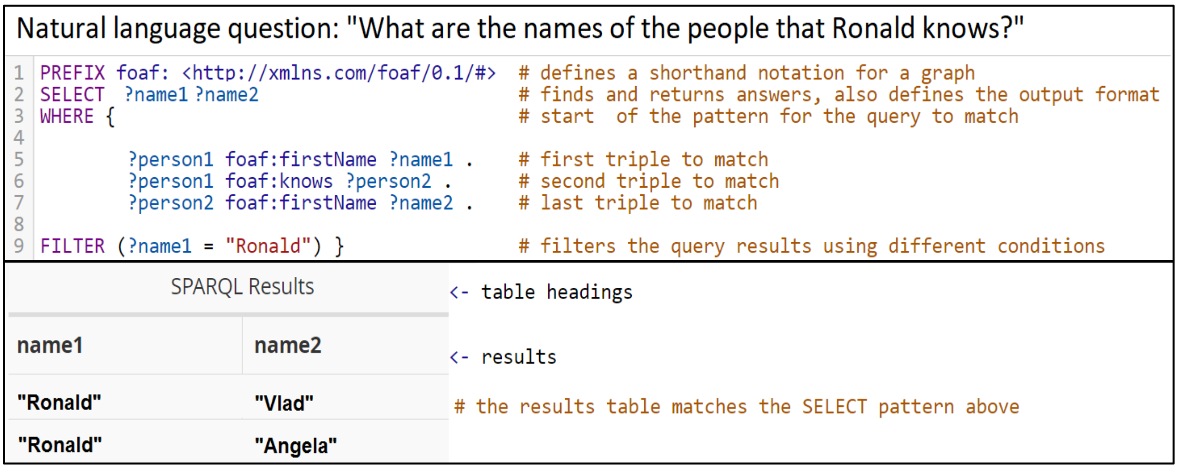
Several types of queries exist in SPARQL:

1) SELECT queries - the most useful operator, used to retrieve data from graphs. Returns a collection of query solutions;
2) CONSTRUCT queries – returns a new graph, as specified by the query triple pattern;
3) DESCRIBE queries – is used to find out the structure of the graph in question. Is useful when the format of various web resources is unknown to the user;
4) ASK queries – is used to answer a question with TRUE or FALSE;

Several operators within queries are used to limit or manipulate the results, such as:

a) FILTER – narrows down the search to specific values or classes in question; can significantly improve performance;
b) OPTIONAL – sets certain parts of the query as optional; some results entries will return null values in those cases;
c) UNION – effectively splits a question into two smaller ones which are run independently and then the results are aggregated;
d) DISTINCT – forces a query to only return valid entries once, filtering repetition;
e) ORDER BY – orders the results by a specific variable;
f) LIMIT – limits the result sets to a specified number; beneficial with large datasets; increases performance;

There are also other operators and syntax elements which can help customise a query from a very generic to a very specific type of question. Nested queries are also possible, if the engines running it support it. Full details on the SPARQL implementation has been published and available online (Prud and Seaborne 2006).



Natural language question: "What are the names of the people that Ronald knows?"

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/#>  # defines a shorthand notation for a graph
SELECT  ?name1 ?name2                        # finds and returns answers, also defines the output format
WHERE {                                      # start  of the pattern for the query to match

        ?person1 foaf:firstName ?name1 .     # first triple to match
        ?person1 foaf:knows ?person2 .       # second triple to match
        ?person2 foaf:firstName ?name2 .     # last triple to match

FILTER (?name1 = "Ronald") }                 # filters the query results using different conditions
```

SPARQL Results <- table headings

name1	name2
"Ronald"	"Vlad"
"Ronald"	"Angela"

<- results

the results table matches the SELECT pattern above

Figure 6-3. Example SPARQL query and results

The structure of SPARQL mimics the SPO triple structure, since this is the pattern it has to match and find. The difference is that an OWL statement will proclaim something exists, or is valid, whereas a query asks if something exists with a specific SPO composition. Figure 6-3 shows an example of a SPRAQL query where the FOAF ontology is queried finding the first names of people which know each other. The graph URL shortened to ***foaf*** for convenience using the ***PREFIX*** keyword. This graph is expected to have some named individuals, hence the ***SELECT*** query is looking to find the names of these individuals. These are all part of the head of the query, which describe its scope via ***PREFIX***, and its type from the keyword ***SELECT*** in this case. The body of the query starts at line 3, specifying the condition of the query using the ***WHERE*** operator. The pattern to match is in between the curly brackets and is made out of 3 triples: the first looks to find a certain ***?person1*** variable which has a first name, described by the object property ***foaf:firstName***; the second looks to find which other person is known to ***?person1*** via the ***foaf:knows*** property, with the ***?person2*** variable in the object position; the last is similar to the first triple, but the subject is now ***?person2***. Finally, the query results would be restricted so that ***?name1*** complies with a specified secondary condition. A query of this sort can be very close to natural languages, because of the SPO pattern. SPARQL queries are often used to test an ontology, thereby validating its competency questions posed during its development.

The most used types of queries throughout this research project were the ***INSERT*** and ***SELECT*** type. The former was used for injecting data resources within information models, whereas the latter was used for retrieving data, information and knowledge.

This section was concerned with the introduction of the two main types of knowledge operators: rules and queries. Their basic functionalities were described with examples, showing the basic principles on which knowledge is gained by the ONTOCS system. For the rest of this chapter, various types of rules and queries will be introduced, specifically concerned with crowd simulation analysis concepts.

6.2. Storing information and knowledge

The knowledge base developed and presented in Chapter 5 is a schema of concepts which describe the various processes involved in an integrated multi-disciplinary system - ONTOCS. The system is expected to dynamically work with data according to the schemas it includes. There are several information requirements which have to be provided from various inputs. These need to provide all the relevant ***instance*** data involved in the design process in order for the system to function correctly.

Understanding how instances work with RDF graphs is important to help distinguish between the schema model and the data model. In essence, a schema represents hierarchies, workflow, processes and imbedded knowledge; the instances represent the data and information used in processes and knowledge generation.

The Oxford Dictionary defines an instance as “*an example or single occurrence of something*” (Stevenson 2011) and it is used in object-oriented programming to describe concrete objects belonging to a specific class. In ontology terms, ***instances*** are termed ***individuals***, and they are the occurrences of one or multiple classes within an ontology model. McGuinness and Van Harmelen (2004) in the OWL release report define that ***individuals*** “*are instances of classes, and properties may be used to relate one individual to another*”. Without instance data, knowledge operators would have nothing to process. Figure 6-4 below outlines the differences and connection between an ***instance*** (***individual***) and a ***concept*** (***class***) in programming and in ontologies.

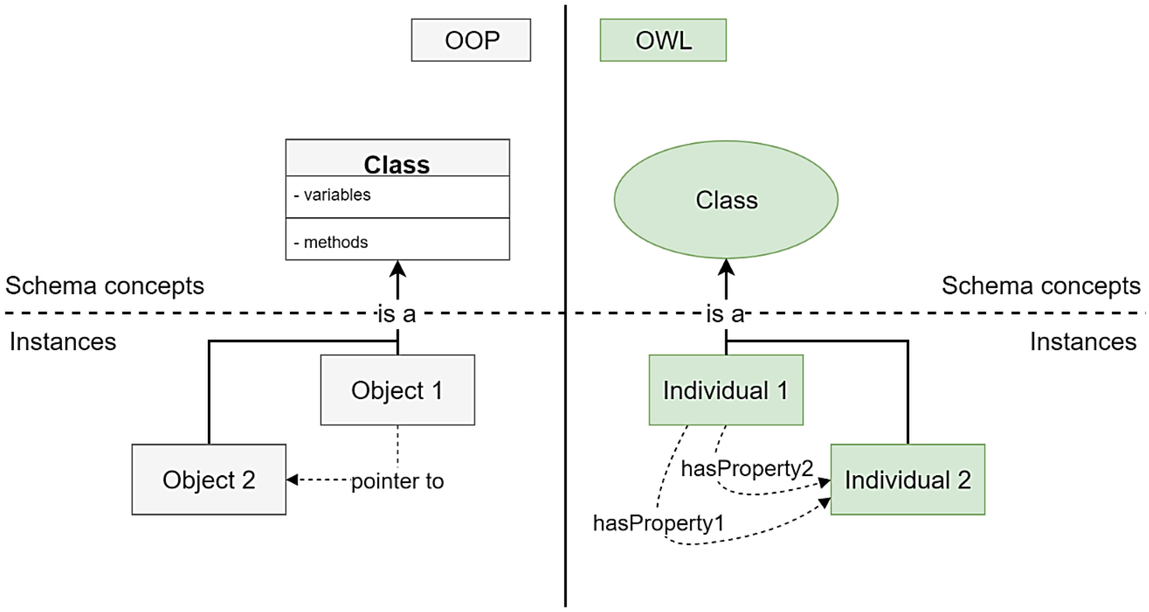


Figure 6-4. Example of comparison between OOP instances and OWL individuals

The main difference between instances from the two domains is the way they behave. An object-oriented programming instance usually belongs to a specific class, taking all its attached variables and methods, and most importantly, it is a declared and initialised as a valid instance. An ontology individual on the other hand, can be used anywhere within the ontology without declaration being a requirement (Yu 2014). Although an open way of using instances can be convenient, it can lead to problems on validating the instance model data. To account for this limitation, the OWL2 syntax introduced the ***entity declaration*** notion, where “*each class, property, or individual is supposed to be declared in an ontology, and then it can be used in that ontology and ontologies that import that ontology*” (Hitzler et al. 2009). This notion helps deal with ontology model management data and allows for clear definition of individuals and concepts across several ontologies. Figure 6-4 suggests that each instance must be defined as belonging to a class, however when it comes to defining the properties of that instance, OWL allows a much greater flexibility in defining individuals, allowing to express any relationship to any other instance, without programming constrictions. Additionally, instances in OOP are usually kept in memory as a program runs, whereas ontology individuals make use of IRIs which can be accessed on the Web and can be used over several simulation iterations and across several knowledge domains.

Instances can be created in several ways: using API to parse OOP objects into RDF individuals, by using INSERT query types, or manually added in ontology editors. The developments for the ONTOCS system mostly utilised API packages to programmatically populate ontology information models.

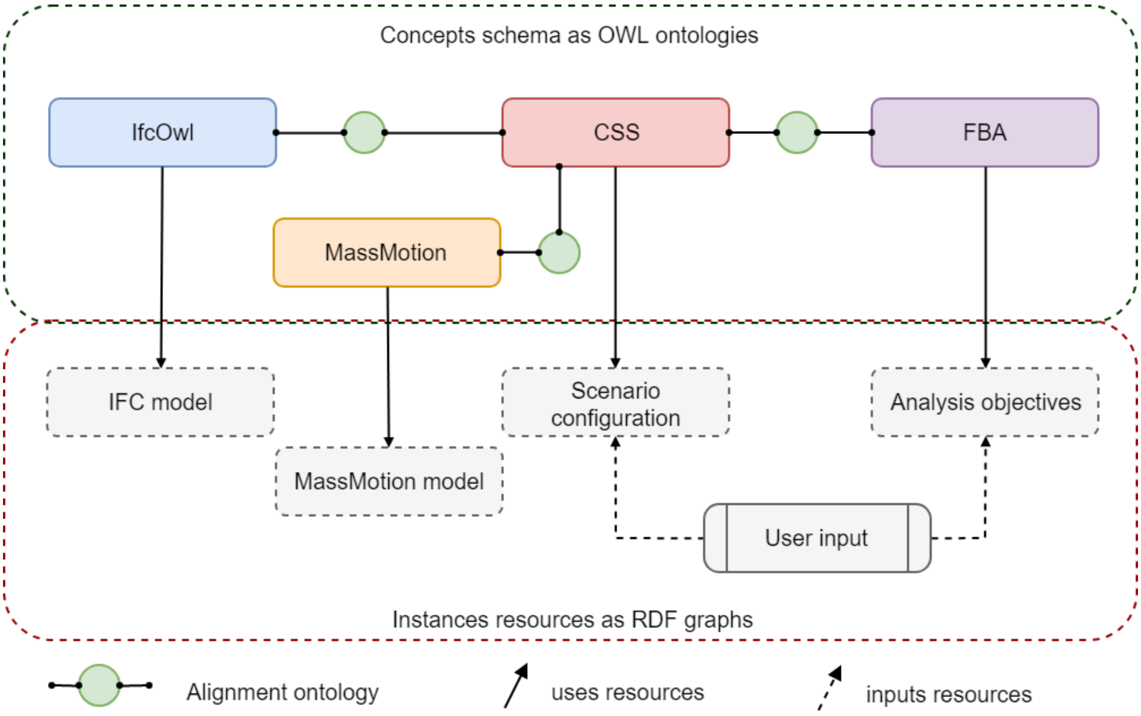


Figure 6-5. ONTOCS ontologies using RDF graphs as resources

Figure 6-5 outlines the basic resources which add context to the ONTOCS knowledge base. The OWL ontologies represent knowledge as schema concepts, while their individuals are provided in separate RDF graphs with unique date stamped IRIs. The mapping between the resources is governed by the ontology alignment, SWRL rules and in some cases reasoning queries are used programmatically. The next section of this chapter expands on the requirements, methods and benefits for storing these resources by looking at the process from two perspectives:

1) resources about the building models and
2) resources about the design process.

6.2.1. Storing building model data and information

The instances which refer to the building environment can be stored across several ontologies:

1) IFC model – instance model of the building to simulate and analyse;
2) CSS model – instance model of the scenario description, including all the relevant model objects;
3) CST model – tools specific instance model, for MassMotion in this particular case;

The three models above refer essentially to the same building, but from three different perspectives, each with a different structure of the data. Figure 6-6 shows an example of a ***Space*** object and its three perspectives. Each has explicitly defined object and data properties in its own domain, with some references to its other views denoted by the ***sameAs*** axiom. The three individuals from Figure 6-6 are in a relationship of equivalency, because they virtually refer to the same space in real life. Each individual belongs to its own resource RDF graph, but when in the context of the aligned schemas and rules, they are identified as the same logical instance. It can also be observed that each individual has different semantics in its own domain, and that there are few cases where the same data is present across all of them, and this is usually related to identity. Using logical inference, data need not be repeated as it is accessible across all domains and thus redundancies are avoided.

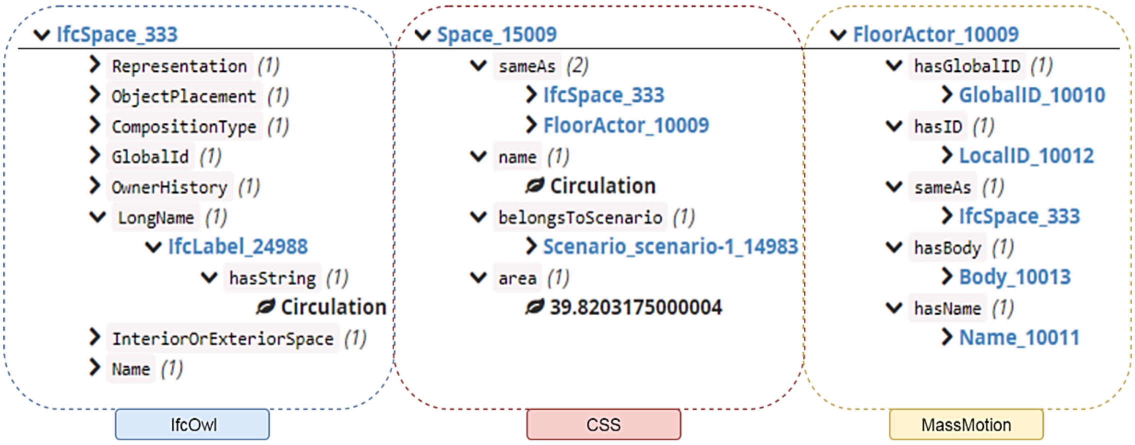


Figure 6-6. Example of a space instance which is represented by 3 equivalent ontology individuals across several knowledge domains

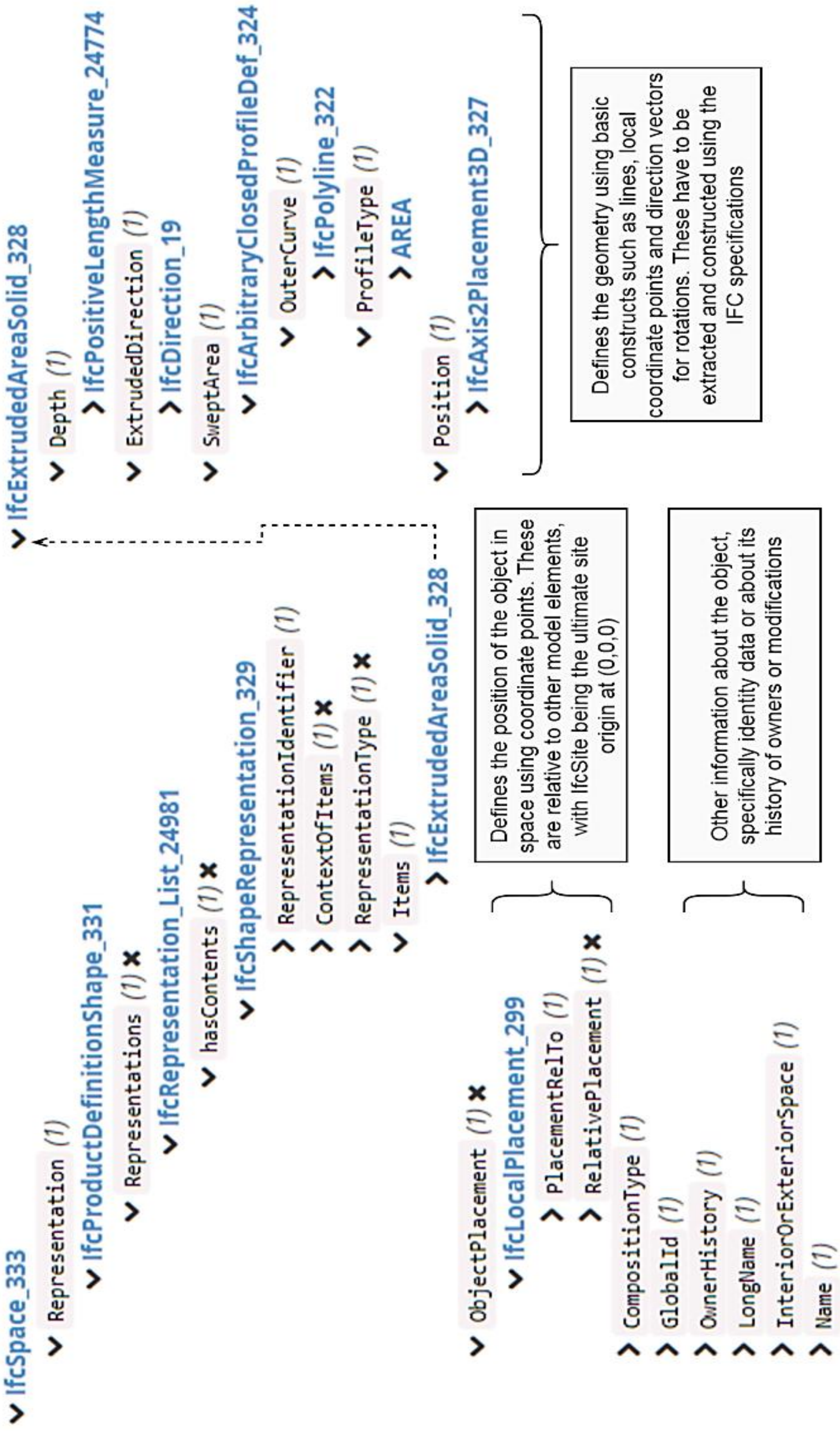
In a study about expressing building models from one CAD tool to another, Abdul-Ghafour et al. (2014) mention that *"The ontology Y is yet described at the terminological level having no instances. The aim is at creating instances in Y by finding for each instance in X the corresponding concept in Y. An ad hoc ontology, called mapping ontology, is created to store mapping axioms and rules between X and Y."* Ideally, this means that a representation of an IfcOwl individual could be fully converted into its equivalents in the neighbouring domains. This process however becomes increasingly complex as the gap difference between the domains is greater. The fundamentally different way in which geometry is expressed, means that performing this conversion using ontologies alone becomes even more arduous, as was also mentioned in Section 5.2.1 in the case of aligning concepts between CSS and IfcOwl. Additionally, when considering crowd models, it was established in Section 4.2.1 that even though all the geometric information can be extracted from the IFC, the contextual information is lacking.

For these reasons, a more in-depth look at the used tools and how they represent the semantics is required.

The IFC model

IfcOwl is the ontology representation of the IFC schema and was developed in such a way as to ensure it resembles its structure with great fidelity (Beetz et al. 2009). This has proved to be a double-edged sword, as the scale and complex structure of the IFC schema has made the ontology very large and the logical restrictions required by the OWL language has resulted in many relationships and classes. At the same time, it enables more creative ways in design and product data exchange or automation in the construction field, has some studies have done so already (Scherer and Schapke 2011, Pauwels et al. 2011, D.-Y. Lee et al. 2016). Since then, various tools have been developed which work with IfcOwl and are able to parse an IFC STEP file into an RDF graph.

Figure 6-7 shows an example of the structure of the ontology, with the names in bold blue representing class individuals, while the other lines are data and object properties. The ***IfcSpace_333*** individual is the parent on this particular hierarchy tree, being related to dozens of other individuals, each members of other classes which branches out towards the primitive data types such as integers or strings. The OOP nature of the schema is clearly reflected in its OWL representation as well, causing very deep trees and thereby making the data hard to access using SPARQL. This reduces the performance of retrieving data, but it also means that developments based on IfcOwl require experts with a high understanding of its structure in the first place. This has been a problem since its inception and has been discussed extensively (Pauwels et al. 2016), with a more recent suggestion to simplify it (Terkaj and Šojić 2015), especially concerning its geometry (Pauwels and Roxin 2016, Pauwels et al. 2016). It is important to note however, that the IFC schema was developed for optimal storage and exchange and not for performing more complex operations. Due to its compressed geometry, tools have to reconstruct in memory from basic geometric concepts.



Defines the geometry using basic constructs such as lines, local coordinate points and direction vectors for rotations. These have to be extracted and constructed using the IFC specifications

Defines the position of the object in space using coordinate points. These are relative to other model elements, with IfcSite being the ultimate site origin at (0,0,0)

Other information about the object, specifically identity data or about its history of owners or modifications

Figure 6-7. Tree hierarchy of an IfcSpace individual in IfcOwl.

This allows relatively evident mapping across to a crowd simulation model, as was done in Section 5.2.1. However, this was done so at a generic class level, without taking into account the complexities of geometric representation of elements, as it was considered impractical. This means that geometry for the CS domain had to be converted from IFC programmatically using mathematical implementations.

The main benefits of using IfcOwl is that it can contain all the necessary information required for a CS simulation, ranging from geometry, identities of objects to contextual information which can be present via object data properties. The problem lies in the fact that certain properties need to exist in the model in the first place. These need to be stored in the BIM platform at creation and explicitly exported to the IFC model.

The main limitation of using IfcOwl in the current context is its large schema, making it hard to work with, nested properties which need to be known to retrieve easily, and finally its geometric representation which needs to be extracted at every iteration and manipulated in memory.

The MassMotion model

The geometric representation in a CST can differ a great deal from one tool to another. In the case of MassMotion, geometry is represented by triangulation, meaning that each plane or face of a 3D object is divided into several triangles. The creation of the geometry is usually done manually or through various model import capabilities, which is then corrected and adjusted. In the case of the developed system, the geometry is retrieved from the IfcOwl model (Section 6.3.1), converted in memory and then the new MassMotion geometric objects are used to populate the MassMotion RDF resources graph. This was done for simplicity reasons, as it does very little to advance the concept of knowledge mining if done using ontology rules.

When it comes to simulating the environment, additional objects are created (***Events***, ***Agents***, etc.) taking into account user input and several other resources. These are created automatically in accordance to each scenario, as part of the first stage of the knowledge mining process, Stage I – Scenario generation, which is presented below in Section 6.3.1. Once the simulation is completed, a CST creates a large dataset which records events in time and space about what has occurred within the model. For example, it records every frame for an agent since its entry in the model until it has reached the exit, keeping track of its speed and waypoints list, including which spaces it traversed. MassMotion saves this data in a SQLite database.

Considering the vast amount of data and in particular its dependency on expressing it according to time frames, it was considered best to keep simulation data in its native format (SQL) and query these databases as required by the knowledge mining process. Thus, certain data about RDF individuals for the MassMotion and CSS ontologies are created ad-hoc, as shown in Figure 6-8. This is not true for ***Geometry*** data or identify data related ***Events*** and ***Agents.*** This approach was found more convenient when working with results data relevant to the analysis process. The initial Stage I data is recorded in RDF graphs regardless if simulation results are not. This allows for the future execution of the model should results data be lost.

The main benefit of using the MassMotion ontology is that it allows a model view of a simulation and stores all the necessary data to re-generate a simulation execution.

The limitation lies in the fact that it may hold redundant data from the IFC model or CSS models, as these are required by the program's programmatic objects. The second limitation is that although it can be used to store simulation data, it would be highly inefficient. Thus, conceptualisations of the relevant data should be stored on a higher level using the CSS model.

The Crowd Simulation Scenario (CSS) model

The CSS ontology model sits at the core of the knowledge mining process, as was indicated in Section 5.2.4. It was outlined previously that the CSS model acts as an integrator between BIMs and CSTs, as shown in Figure 6-8, where CSS individuals reference MassMotion individuals and IfcOwl equivalencies. This means that from the CSS perspective, only certain data is relevant, while particularities of the data is provided in the other aligned domains.

Firstly, model elements, especially those of a static nature (i.e. geometry) are stored in generic way recording identity data, certain properties used in the knowledge mining process and most importantly the domain of each model object when used in a particular scenario context. This presents a way in which to manage large data sets, where one IfcOwl instance can have several equivalent simulation instances for each scenario.

Secondly, model results are stored as required by the knowledge mining process, as is suggested in Figure 6-8. These are in line with the requirements inputted by the feedback analysis process and the FBA, as outlined in the next section.

Considering the limitations of the IfcOwl and MM models, the CSS is better suited for storing the context, which is what makes each simulation different. The IfcOwl is the source of geometry and some implicit contextual data, but the CSS can store it explicitly and it can account for user input. The CSS model should not be burdened with storing geometric representations as it would defeat its purpose of being generic in nature. Identity data of geometric objects and where these are in the other models is recommended. Storing simulation results ad-hoc also benefits from keeping the model relatively small in size and potentially improving processing speed for knowledge retrieval operations in the future. However, this makes the CSS ontology highly dependent on external resources.

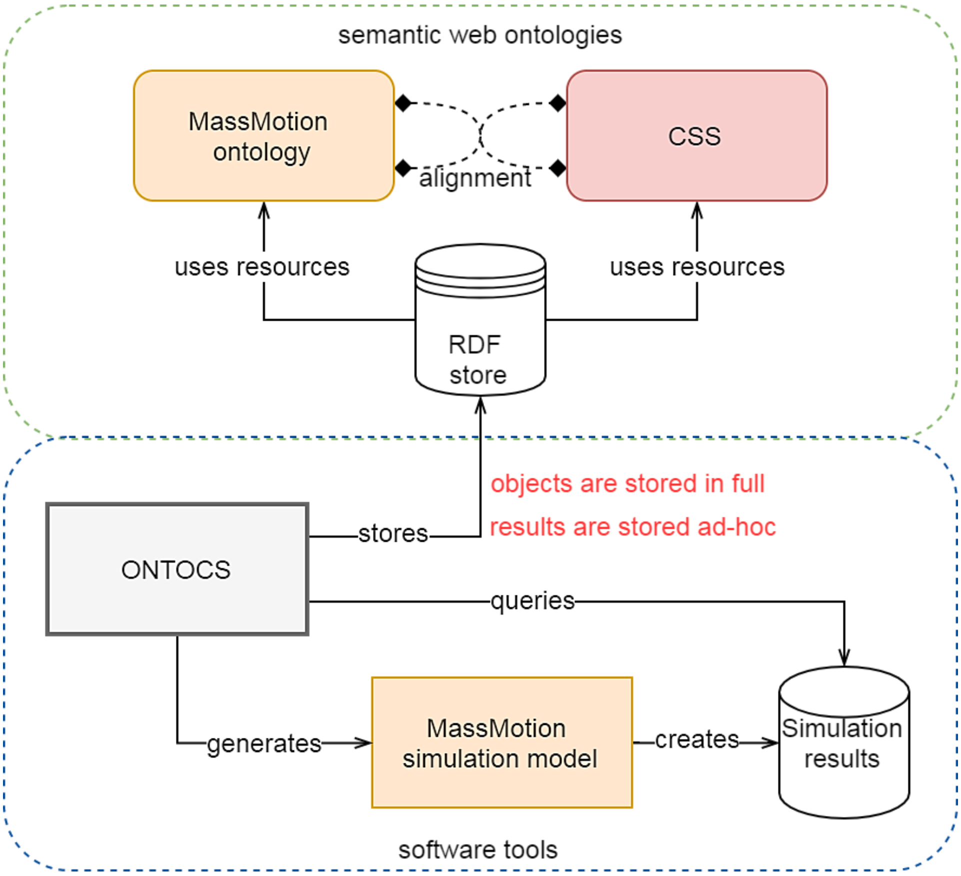


Figure 6-8. MassMotion and CSS storage of objects and data in RDF

6.2.2. Storing design process information and knowledge

Design is by nature an iterative process which needs to consider constraints imposed on the end-product. The collaborative processes required for achieving a building design are immense, with many stakeholders involved and because of this it is a process of compromise (Kvan 2000). This is because the decision-making process is complex and needs to consider several views of the model not just for structural integrity or costs, but also safety and environmental protection. Several studies have recognised the importance of decision-making during design, and have proposed a myriad of solutions on how to manage this process digitally (Plume and Mitchell 2007, Shafiq et al. 2012, Fernando et al. 2013, Oh et al. 2015, Zhang et al. 2015). The main problem on a macro level seems to stem from the fact that the industry is working in different silos of information, and each of these is updated in technology at a different rate. The CSTs are a good example for this, as their interoperability with BIM has emerged relatively late, compared to those of other design disciplines, as was concluded in Chapter 2.

Kalay (1998) in Plume and Mitchell (2007) points out that *"each specialist has their own view and set of objectives, arguing that design collaboration works best where those same specialists adopt what he called a 'super-paradigm', agreeing to a course of action to achieve a common goal for the whole project, rather than narrowly considering their own objectives in isolation."* Reaching a consensus on how to best manage the decision-making process has always been an issue in design. Using an ontology approach to keep track of the design-decision making process could bring the benefits of integration using semantic web resources, as is suggested by the methodology of this research. In addition to that "design intent should be captured and processed by intelligent systems" (Abdul-Ghafour et al. 2014). The exact methods on how to do this differ based on the systems proposed in industry and research.

Considering the above statements from the perspective on crowd simulation analysis, there needs to be a clear understanding of what information needs to be captured and how it should be stored for knowledge retrieval at a later stage. Due to the nature of how CST work, two separate factors have been identified in this research:

1. Storing scenario assumptions – using the CSS ontology;
2. Storing design intent – using the FBA ontology;

These two main factors were also considered when developing the ontologies, through asking the competency questions, as was discussed in Section 5.1. Apart from storing model data about its elements, they were also intended to store designer input, which participates in defining the context for each scenario. Section 5.1.1 emphasises that the CSS ***ScenarioAssumption*** class and its subclasses are used to capture the user input and try to retrieve the information required to construct the context of a scenario. This implicitly fulfils the role of storing these assumptions for future use, while also contextualising the knowledge retrieval process. For example, when searching for answers, scenarios can be grouped depending on their assumptions, to allow for a comparative analysis.

The large-scale simulation of building performance brings into consideration the way in which knowledge storing is managed. For example, a set of objectives can be applied to a multitude of simulation data models, and as such the relationships and rules must be implemented during a knowledge mining process. Figure 6-9 shows an example of how the data is linked in this context. The BIM produces many simulation models, each with a different configuration being saved in as CSS resources. At creation, the CSS models record the assumptions of the users.

During the analysis process, user objectives are also captured as FBA resource models using the ***AnalysisObjectivesSet*** class, and these can be applied to a specific set of scenarios allowing the definition of a scope when querying the knowledge models. This is the method through which design intent and decision-making is stored explicitly as knowledge about the design process, while also making it convenient to work with large data sets.

The input of design intent must be made explicit, and new ontology individuals have to be created programmatically for each design iteration. This method is therefore limited to the extent of the ontology concepts defined for objectives and assumptions, and by the software tool used to store this information from memory at the appropriate time.

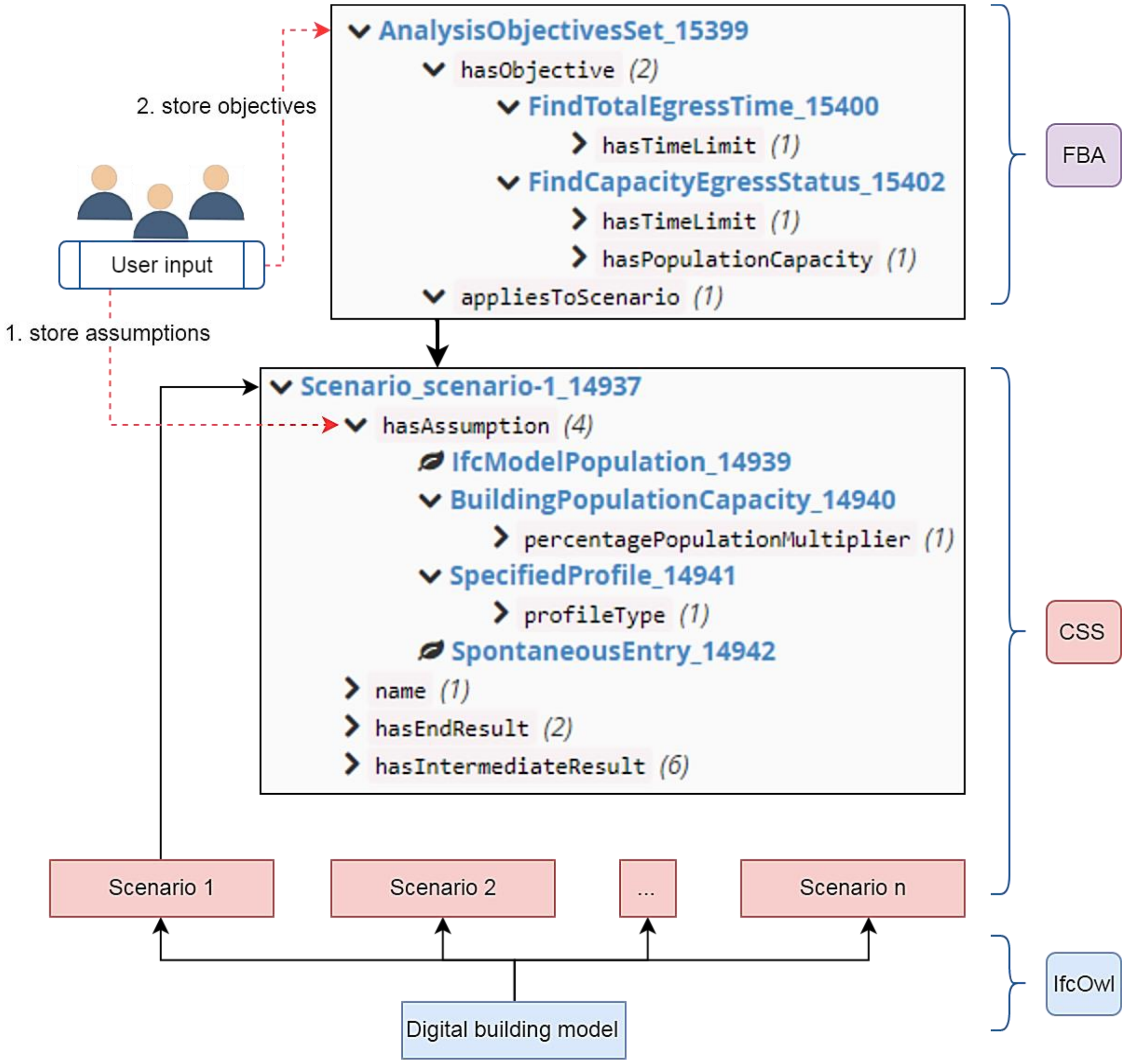


Figure 6-9. Storing scenario assumptions (1.) and user objectives (2.)

6.3. Retrieving information and knowledge

With all the mechanisms in place to store data, information and knowledge, as shown in the previous section, it is possible to carry out knowledge retrieval processes on available resources by applying the types of operators introduced in Section 6.1. The mechanisms developed and presented here follow the crowd simulation model construction and analysis of the process, in line with the ONTOCS platform functionality developed, and following the CSS and FBA ontologies during the process (workflow in Figure 6-10):

- Stage I, Scenario generation – is concerned with constructing a valid crowd simulation model with a specific scenario context using various sources of information (as identified in Section 4.2.1), while also considering user input along the way, in order to tailor each scenario to design expectations. This stage uses reasoning operators to gather resources across several models, and to identify important concepts. More importantly, it attempts to 'understand' the building and its circumstances and uses this knowledge to construct a realistic scenario.
- Stage II, Analysis feedback – is concerned with executing the scenarios generated in Stage I on a large scale, then retrieving results and comparing them to user design objectives. This stage uses reasoning operators to aggregate results in conjunction with applied knowledge rules to find answers about the performance of the scenario as a whole or the behaviour of certain model objects.

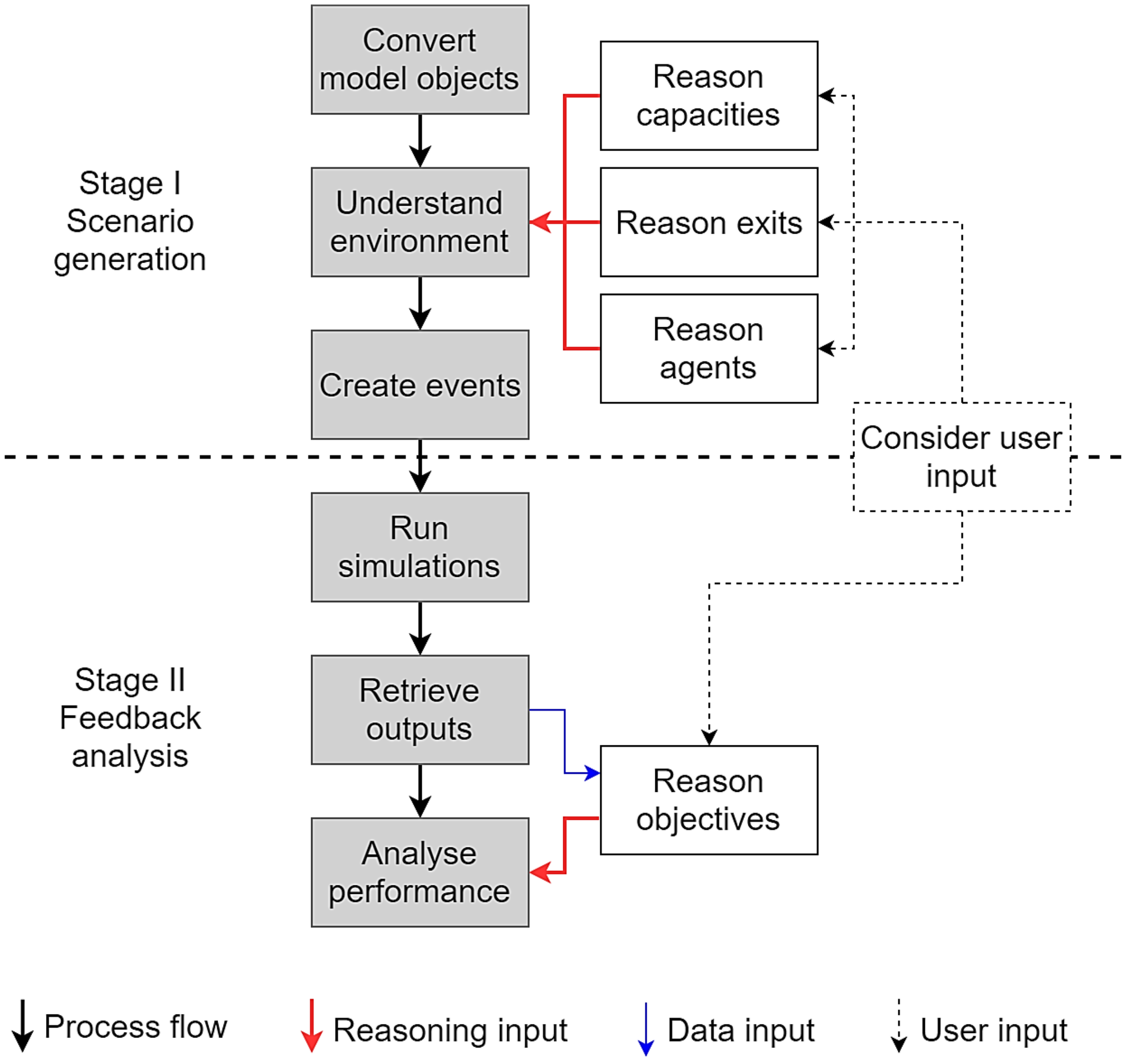


Figure 6-10. ONTOCS system process workflow through its two main stages

The two stages reflect the way in which humans conduct design starting with the creation of the environment (which carried out by using IFC model resources), and then creating the events which add context to each model. For the second stage, the analysis is applied using similar factors which are used in practice, except that they are expressed using operators which are then applied at the user's discretion. The main benefits to this process is the automation of the model creation and the ability to manage model data on a larger scale, allowing the analysis of multiple models at the same time. The limitation of this method is that designers conventionally rely on visualisation to understand the building and crowd simulation event, which in this case are quantified using algorithms and rules, thus eliminating the visualisation part of the process.

6.3.1. Stage I – Scenario generation

The knowledge mining process for the first stage of the process is about the system being able to 'understand' the building model. This is also outlined in Figure 6-10, where the 3 most important factors each answer a question about the assumed context of a scenario:

1. Capacity – How many people inhabit the environment?
2. Exits – Where are they supposed to evacuate?
3. Agents – What sort of people are assumed?

Answering these questions requires multiple steps. 26 SPARQL queries have been constructed and tested which are relevant to the first stage of the process, summarised in Table 6-1. These operate in conjunction with many SWRL rules across different ontologies (see Appendix C).

The first step resides in understanding the geometry of the model. This is different from re-constructing geometry using schema specifications and algorithm in the sense that the ontology is able to reason which types of geometric concepts are required from one domain into the other, whilst also 'understanding' their purpose in these application domains. This must be converted in a knowledge graph in the first place, which is to express the IFC model into IfcOwl. Knowledge operators then process this model and other resources and are used to identify the relevant objects and data which are used generate the files required by the CST. Several SPARQL queries were constructed for model conversion purposes to retrieve the geometry (Table 6-1). Figure 6-11 showcases the main categories of queries, with Figure 6-12 showing an example of the query which retrieves names and other identity data from the IfcOwl model (using query Q-IFC-2).

The geometry is defined from basic constructs such as lines, points and direction vectors. The ONTOCS platform extracts the geometric data from IfcOwl, converts it in memory and creates the equivalent MassMotion instances. The consequence the highly nested nature of IFC and IfcOwl resulted in long queries which had be constructed in order to get the most basic elements (such as coordinate points or lengths of rectangles). It was observed during testing that this is slower in performance than importing a model from IFC directly (tests and results are discussed in Chapter 7). This was addressed in several papers, and some rules have been developed by Pauwels et al. (2016) to simplify the structure. This sort of rules were used to allow faster identification of properties related to items, or to wrap primitive data types, which can considerably improve the querying times. They are expected to increase significantly with the size of the elements in the model.

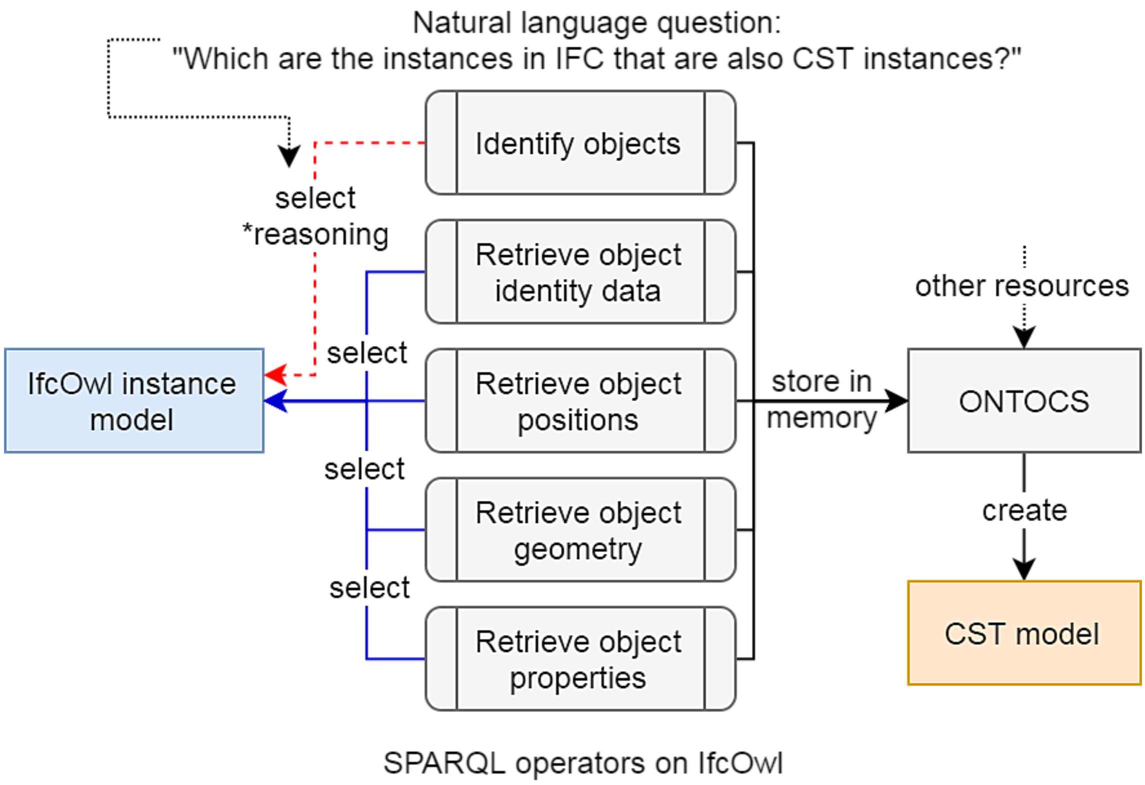
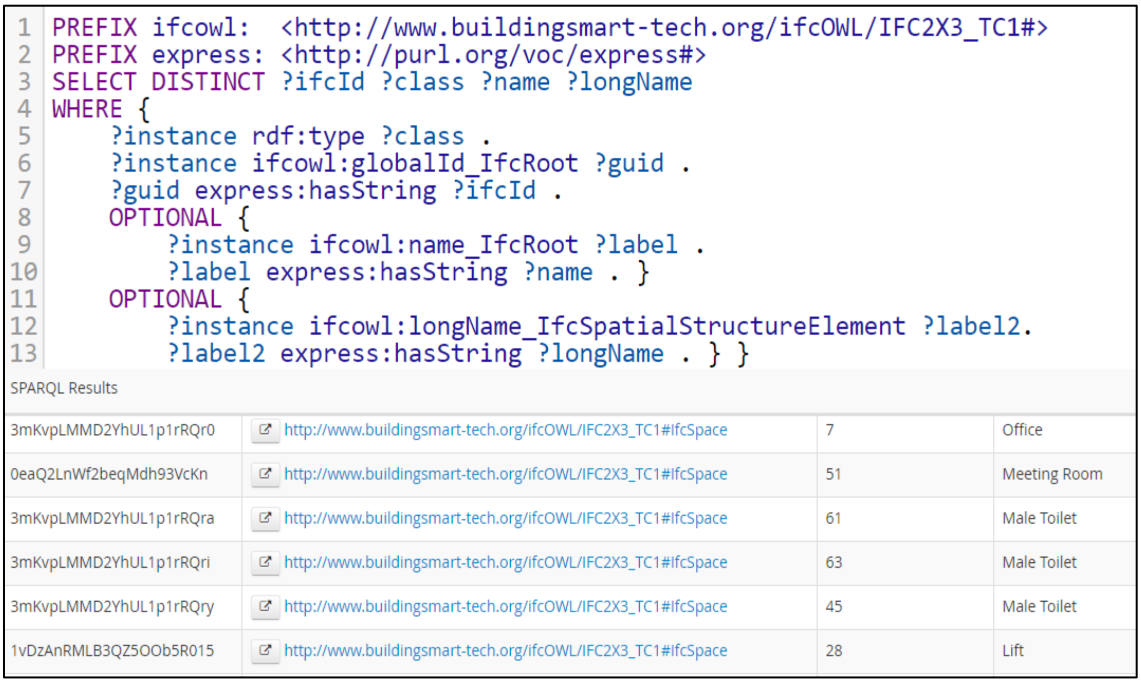

Natural language question:
"Which are the instances in IFC that are also CST instances?"
Identify objects
select
*reasoning
Retrieve object identity data
select
IfcOwl instance model
Retrieve object positions
select
Retrieve object geometry
select
Retrieve object properties
store in memory
other resources
ONTOCS
create
CST model
SPARQL operators on IfcOwl

Figure 6-11. SPARQL operators retrieving information from the IfcOwl model

However, before retrieving the geometry, the scope of a CST and what geometric objects it can use. This was established in the alignment between the CSS and IfcOwl ontologies in Section 5.2.1. The conceptual alignment is then applied here in conjunction with the reasoning query labelled ***Identify objects*** from Figure 6-11 (Q-FBA-1 in Table 6-1), which is summarised in natural language. An example of the query running on a test model is shown in Figure 6-13. This is the first step in filtering the vital objects which needs to be exchanged from the BIM to the CST, thus enabling knowledge already expressed in the ontologies to be applied and acting in a similar way to a Model View Definition (MVD) protocol. The secondary effect of this is that it implicitly leaves out all the information which is out of scope for a simulation scenario.

Table 6-1. List with developed queries for Stage I.

Query code	Name	Reasoning	Category
Q-IFC-1	Find objects	yes	Identify objects
Q-IFC-3	Get IFC Storeys	no	Identify objects
Q-RES-4	Find inhabited spaces	yes	Identify objects
Q-RES-5	Find exit spaces	yes	Identify objects
Q-IFC-2	Get IFC Types	no	retrieve identity data
Q-IFC-5	Get IFC Placements	no	retrieve object positions
Q-IFC-6	Get IFC Placements (spaces)	no	retrieve object positions
Q-IFC-7	Get IFC Placements (mapped)	no	retrieve object positions
Q-IFC-19	Get IFC Orientations	no	retrieve object positions
Q-IFC-4	Get IFC Shapes	no	retrieve object geometry
Q-IFC-8	Get IFC Rectangle shapes	no	retrieve object geometry
Q-IFC-9	Get IFC Rectangle shapes (mapped)	no	retrieve object geometry
Q-IFC-10	Get IFC Arbitrary shapes	no	retrieve object geometry
Q-IFC-11	Get Arbitrary shapes (mapped)	no	retrieve object geometry
Q-IFC-14	Get IFC BREP shapes	no	retrieve object geometry
Q-IFC-15	Get IFC BREP shapes (mapped)	no	retrieve object geometry
Q-IFC-17	Get IFC Extrusions	no	retrieve object geometry
Q-IFC-18	Get IFC Extrusions (mapped)	no	retrieve object geometry
Q-IFC-20	Get descriptions	no	retrieve object properties
Q-IFC-21	Get areas	no	retrieve object properties
Q-RES-1	Get occupancy	no	retrieve object properties
Q-RES-2	Get classifications	no	retrieve object properties
Q-RES-3	Match occupancy factors	yes	retrieve other resources
Note: A full description of the queries is available in Appendix C.			

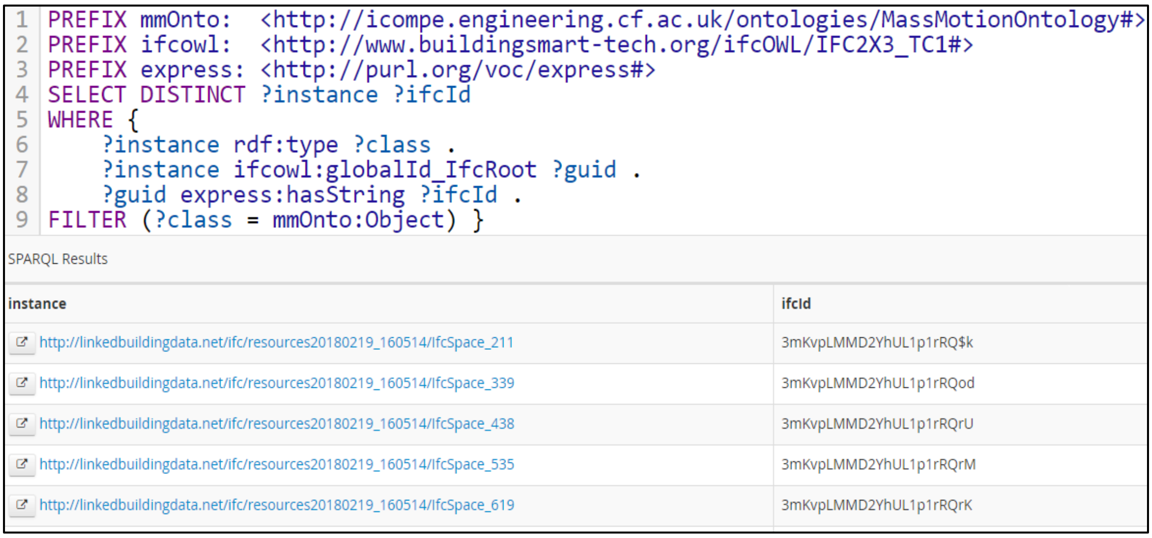


```
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
PREFIX express: <http://purl.org/voc/express#>
SELECT DISTINCT ?ifcId ?class ?name ?longName
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    OPTIONAL {
        ?instance ifcowl:name_IfcRoot ?label .
        ?label express:hasString ?name . }
    OPTIONAL {
        ?instance ifcowl:longName_IfcSpatialStructureElement ?label2.
        ?label2 express:hasString ?longName . } }
```

SPARQL Results

3mKvpLMMD2YhUL1p1rRQr0	http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcSpace	7	Office
0eaQ2LnWf2beqMdh93VcKn	http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcSpace	51	Meeting Room
3mKvpLMMD2YhUL1p1rRQra	http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcSpace	61	Male Toilet
3mKvpLMMD2YhUL1p1rRQri	http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcSpace	63	Male Toilet
3mKvpLMMD2YhUL1p1rRQry	http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcSpace	45	Male Toilet
1vDzAnRMLB3QZ5OOb5R015	http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcSpace	28	Lift

Figure 6-12. SPARQL query Q-IFC-2 operating on the IfcOwl model, retrieving data about instances with results shown below it



```
PREFIX mmOnto:  <http://icompe.engineering.cf.ac.uk/ontologies/MassMotionOntology#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
PREFIX express: <http://purl.org/voc/express#>
SELECT DISTINCT ?instance ?ifcId
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
FILTER (?class = mmOnto:Object) }
```

SPARQL Results

instance	ifcId
http://linkedbuildingdata.net/ifc/resources20180219_160514/IfcSpace_211	3mKvpLMMD2YhUL1p1rRQ$k
http://linkedbuildingdata.net/ifc/resources20180219_160514/IfcSpace_339	3mKvpLMMD2YhUL1p1rRQod
http://linkedbuildingdata.net/ifc/resources20180219_160514/IfcSpace_438	3mKvpLMMD2YhUL1p1rRQrU
http://linkedbuildingdata.net/ifc/resources20180219_160514/IfcSpace_535	3mKvpLMMD2YhUL1p1rRQrM
http://linkedbuildingdata.net/ifc/resources20180219_160514/IfcSpace_619	3mKvpLMMD2YhUL1p1rRQrK

Figure 6-13. SPARQL query Q-IFC-1 reasoning IfcOwl individuals which are also MassMotion ontology individuals according to ontology schema alignment

Once these objects are identified, they are stored, and further information is retrieved using various other ***SELECT*** queries, which are always matched in memory through the ***IfcIdentifier*** class to ensure the correct data is retrieved for each object. Due to the structure the IFC schema and the long nature of SPARQL queries prevents the efficient retrieval of all the data in one go. Filtering improves the performance of the queries, as Pauwels et al. (2016) have concluded over several tests in querying building model data in IfcOwl format, and it reduces the scope of the query to relevant data as well. Due to reasoning flags and depending on query complexity, it can sometimes yield very large datasets of results for generic queries, most of which are expected to be out of scope. If the query is too specific, it can yield no results. To account for this, some queries operating on IfcOwl ***FILTER*** down on specific class types. This resulted in some queries being very similar in structure, especially at the beginning on the code, only to differ slightly towards the end (see Appendix C for full query codes).

The second step involves understanding the context of the model. The information requirements which can contribute to the context of a simulation model were identified in Section 4.3.1. Not all of the identified sources are always required, and the process of defining the context of a scenario needs to be dictated by the designer, thus user input needs to be considered throughout the entire process. From consultations with industry experts, out of the geometric concepts, the spaces were identified to be the most important, as they can describe the building's functionality and the layout of the spaces has a significant impact on the behaviour of the inhabitants (Chitty and Fraser-Mitchell 2003, Kobes et al. 2010).

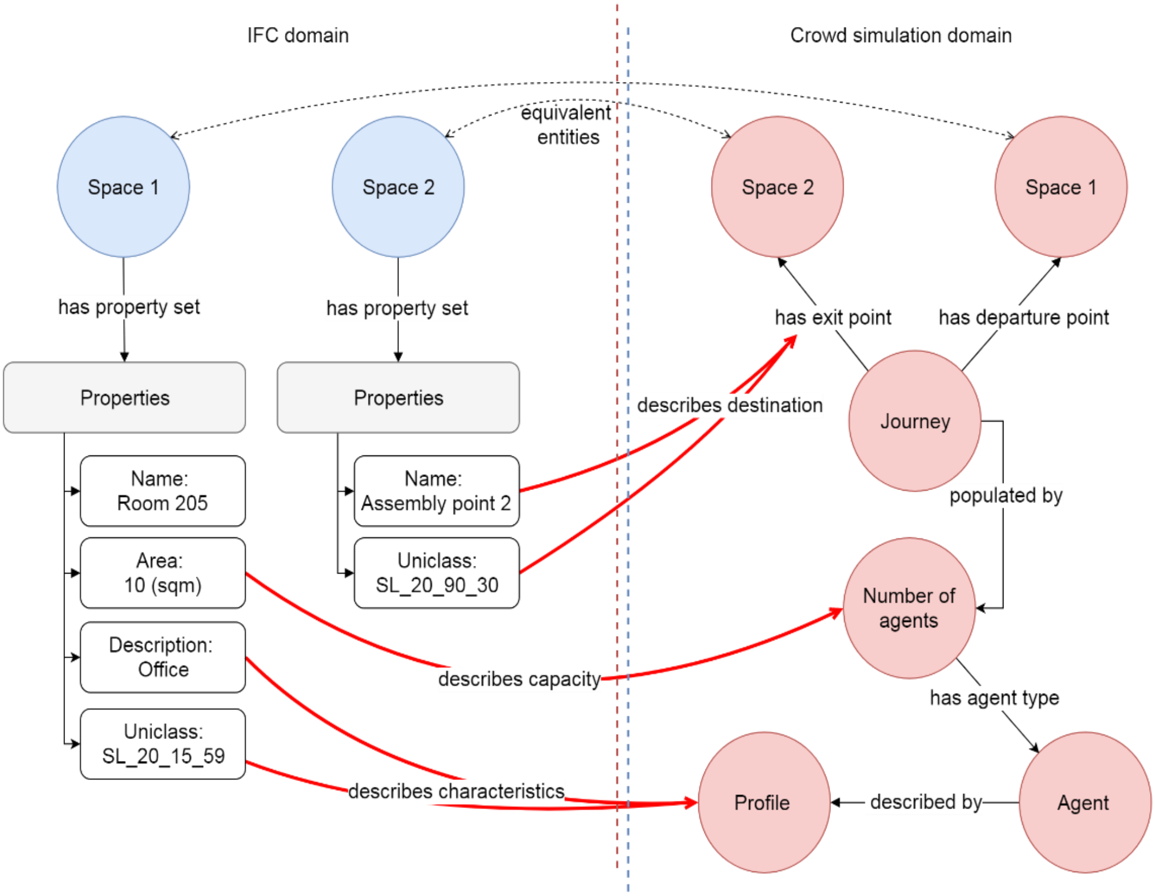


Figure 6-14. Example of contextual information being interpreted by rules from an IFC domain to provide context to a crowd simulation model

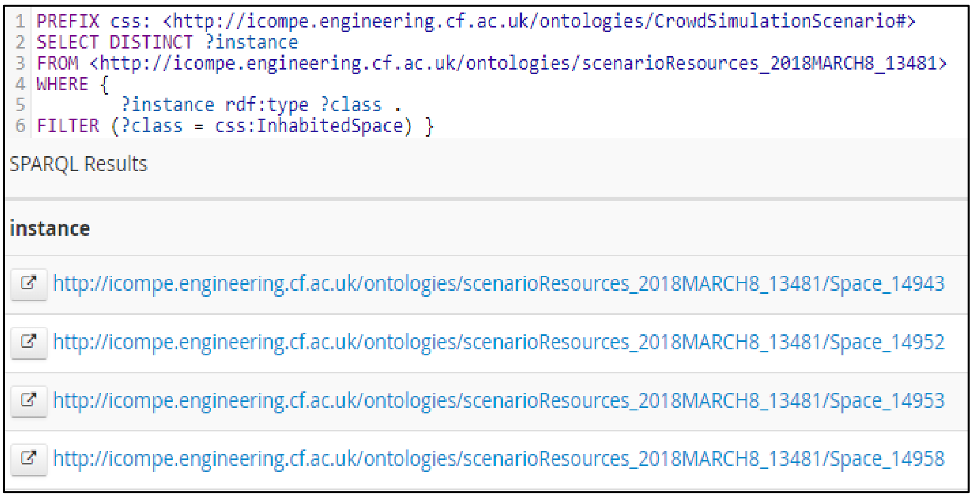
From design experience imbedded in guides and documentations and from expert consultations, two main categories of Space were identified as most relevant, which are represented in the CSS ontology:

- ***InhabitedSpace*** – any space which has a value of inhabitants at the start of a simulation which is not 0, thus a departure point for agents;
- ***RefugeSpace*** – any space which is a designated area for refuge from fire, thus a destination point for agents.

Several ways in which spaces can provide context to a simulation have been explored. Figure 6-14 shows an example of building data explicitly available in an IFC model, which can be used to describe things which are relevant to how spaces are assumed in a crowd simulation domain. These examples are possible answers to the previously asked questions about the simulation context.

In order to extract this knowledge, the following operators have been developed:

1) SPARQL queries which retrieve the relevant properties about spaces from the IfcOwl, such as areas (Q-IFC-21), or identity data related to names (Q-IFC-2) or design codes (Q-RES-2, Q-RES-3); this data is then explicitly stored within the scope of the CSS ontology as RDF resources for each specific scenario;
2) SWRL rules which operate on explicit resources from the CSS ontology and other resources such as the Uniclass2015 or the UKSOC factors; They fulfil the role of identifying which spaces are ***RefugeSpaces*** or ***InhabitedSpaces***, and what their capacities are; these are provided in Table 6-2 below.
3) SPARQL rules which operate on the CSS ontology used to retrieve the reasoned knowledge by the SWRL rules mentioned above; thus the system is able to interpret what the function is for every space object in a scenario context (example query and results in Figure 6-15).



```
PREFIX css: <http://icompe.engineering.cf.ac.uk/ontologies/CrowdSimulationScenario#>
SELECT DISTINCT ?instance
FROM <http://icompe.engineering.cf.ac.uk/ontologies/scenarioResources_2018MARCH8_13481>
WHERE {
        ?instance rdf:type ?class .
FILTER (?class = css:InhabitedSpace) }
```

SPARQL Results

instance
http://icompe.engineering.cf.ac.uk/ontologies/scenarioResources_2018MARCH8_13481/Space_14943
http://icompe.engineering.cf.ac.uk/ontologies/scenarioResources_2018MARCH8_13481/Space_14952
http://icompe.engineering.cf.ac.uk/ontologies/scenarioResources_2018MARCH8_13481/Space_14953
http://icompe.engineering.cf.ac.uk/ontologies/scenarioResources_2018MARCH8_13481/Space_14958

Figure 6-15. SPARQL query (Q-RES-4) reasoning individuals which are classified as ‘InhabitedSpace’ within the scope of the CSS ontology

The verification of the SPARQL queries and SWRL rules has been done by constant testing throughout the development of the ONTCS system by ensuring the correct information is retrieved from basic mock model data. Within the scope of the described resources here and in previous chapters, they function correctly. Their efficiency was tested, and a discussion is available in Section 7.2.

Table 6-2. List of SWRL rules for the CSS ontology classifying space types

SWRL CODE	DESCRIPTION
R-CSS-1, InhabitedSpace	
Space(?space) ^ occupants(?space, ?number) -> InhabitedSpace(?space)	A space with occupants is considered an occupied space
R-CSS-2, RefugeSpace	
Space(?space) ^ uniclassCode(?space, ?code) ^ swrlb:stringEqualIgnoreCase(?code, "SL_20_90_30") -> RefugeSpace(?space)	If a space has the specific Uniclass code for Refuge Space (SL_20_90_30), is considered a RefugeSpace class in the CSS ontology.
R-CSS-3, RefugeSpace	
Space(?space) ^ name(?space, ?text) ^ swrlb:containsIgnoreCase(?text, "exit") -> RefugeSpace(?space)	If a space has a name suggesting it is a fire refuge area, it is classified as RefugeSpace in the CSS ontology.
R-CSS-4, RefugeSpace	
Space(?space) ^ name(?space, ?text) ^ swrlb:containsIgnoreCase(?text, "refuge") -> RefugeSpace(?space)	If a space has a name suggesting it is a fire refuge area, it is classified as RefugeSpace in the CSS ontology.
R-CSS-5, RefugeSpace	
Space(?space) ^ description(?space, ?text) ^ swrlb:containsIgnoreCase(?text, "refuge") -> RefugeSpace(?space)	If a space has a description suggesting it is a fire refuge area, it is classified as RefugeSpace in the CSS ontology.
R-CSS-6, RefugeSpace	
Space(?space) ^ description(?space, ?text) ^ swrlb:containsIgnoreCase(?text, "exit") -> RefugeSpace(?space)	If a space has a description suggesting it is a fire refuge area, it is classified as RefugeSpace in the CSS ontology.

6.3.2. Stage II – Analysis feedback

The second stage of the knowledge mining process aims to correctly interpret simulation results on a large scale and provide feedback to safety engineers. The analysis feedback process is characterised by asking the system questions similar to those in natural language, like:

‘Which scenario evacuates agents in less than 1 minute?’

Figure 6-10 shows the involvement of users in this process. The ONTOCS system allows users to choose from a set of different predefined objectives, which query model data according to identified PIs in Section 4.3.2. The use of objectives allows the ontology reasoning to scope on specific tasks.

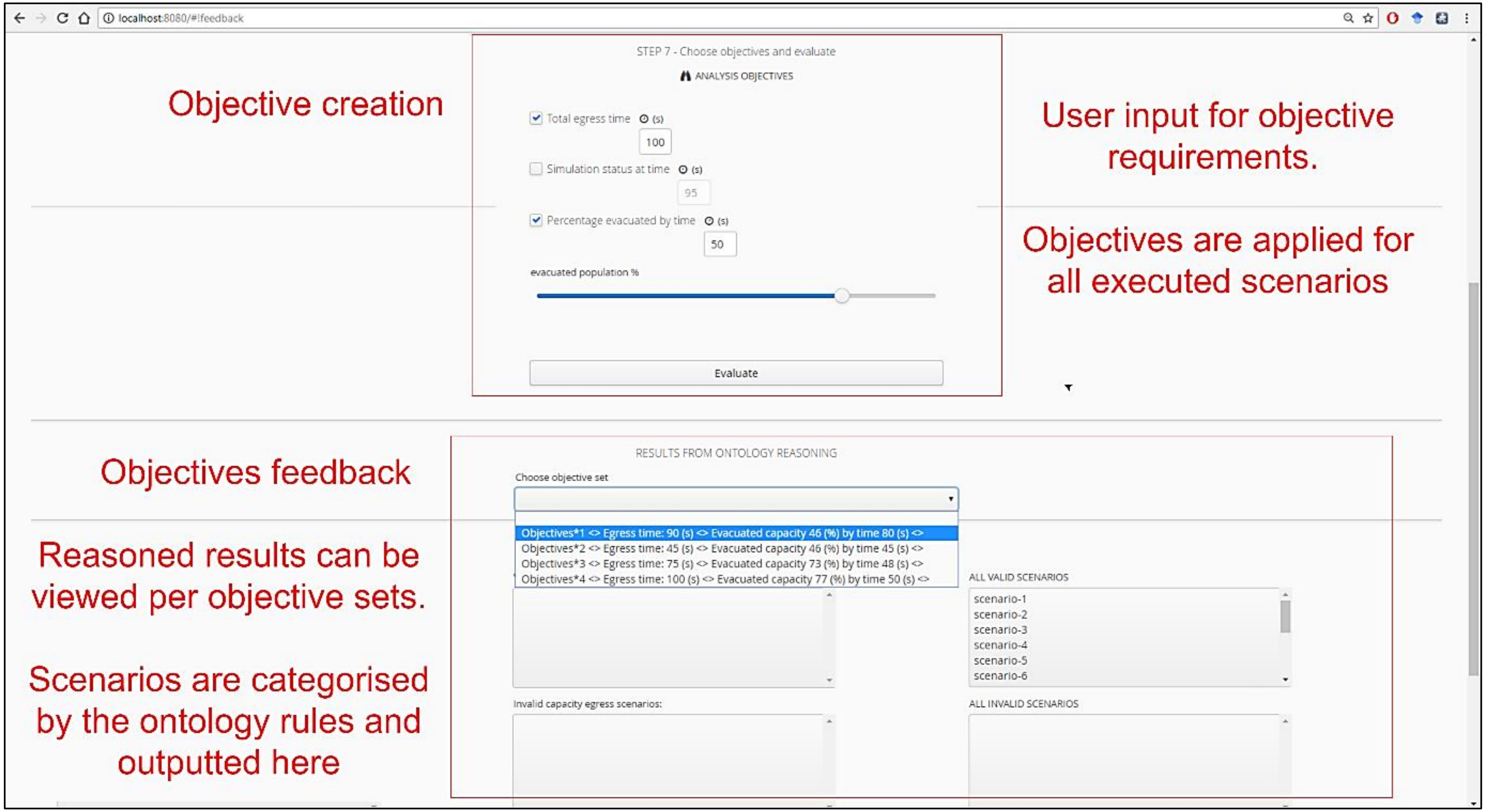
Table 6-3 shows examples of user objectives, and the operators they rely on. Each row in the table represents an instance of FBA ontology ***ObjectiveAnalysisSet*** class (as introduced in 5.1.2) A set includes two separate objectives, each answering specific questions:

a. Total egress time – what is the total time for all the agents to travel to the exits?
b. Capacity egress – by what time can x% of the population be evacuated?

These objectives have to be inputted by users when evaluating scenarios, as shown in Figure 6-16. By applying several rules, the system can provide answers for the sets of objectives chosen. The process time increases with the complexity of the rules in place, as well as with the number of tested scenarios, (see Chapter 7 detailed case study).

Table 6-3. Example of objective sets inputted by uses for the analysis stage

Objective set	a. Total egress time (s)	b. Capacity egress		Valid scenarios
		population (%)	time limit (s)	
1	90	50	45	1 to 9
2	90	75	45	1 to 5
3	120	75	60	1 to 10



Objective creation

STEP 7 - Choose objectives and evaluate

ANALYSIS OBJECTIVES

☑ Total egress time (s): 100

□ Simulation status at time (s): 95

☑ Percentage evacuated by time (s): 50

evacuated population %

User input for objective requirements.

Objectives are applied for all executed scenarios

Objectives feedback

RESULTS FROM ONTOLOGY REASONING

Choose objective set

Objectives*1 <> Egress time: 90 (s) <> Evacuated capacity 46 (%) by time 80 (s) <>
Objectives*2 <> Egress time: 45 (s) <> Evacuated capacity 46 (%) by time 45 (s) <>
Objectives*3 <> Egress time: 75 (s) <> Evacuated capacity 73 (%) by time 48 (s) <>
Objectives*4 <> Egress time: 100 (s) <> Evacuated capacity 77 (%) by time 50 (s) <>

ALL VALID SCENARIOS

scenario-1
scenario-2
scenario-3
scenario-4
scenario-5
scenario-6

Invalid capacity egress scenarios:

ALL INVALID SCENARIOS

Reasoned results can be viewed per objective sets.

Scenarios are categorised by the ontology rules and outputted here

Figure 6-16. ONTOCS interface for objective input page

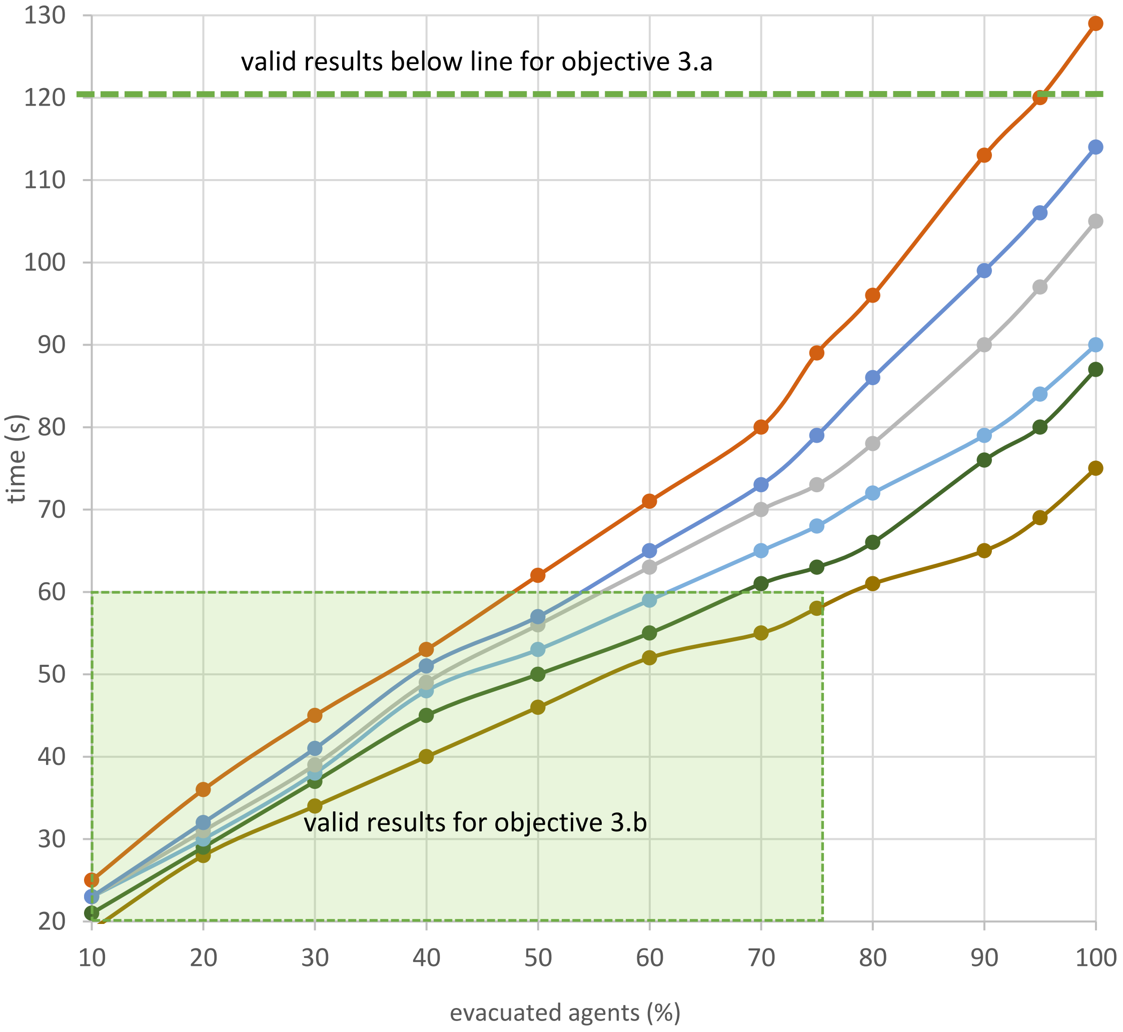


Figure 6-17. Plotted egress progression results for several example scenarios and objective set 3 from Table 6-3

The system is able to query data across various databases from simulations, and process the results with reasoning flags, categorising each scenario in accordance to user objectives. Figure 6-17 shows an example of egress progression plotted data from mock simulations in time vs agent percentages. The lines plotted represent agents leaving the premises of the building, the higher a point on the line the more time it takes to evacuate the more agents. Outlined in green, AnalysisObjectiveSet 3 shown the line (3.a) and the area (3.b) under which the scenarios are meeting user requirements.

The basic functionality lies in categorising scenarios in accordance to each objective, and thereby each rule overseeing it. The reasoning of the rules is retrieved by asking the ontology questions posed by the SPARQL queries in Table 6-4. When these queries are sent through, they effectively call their corresponding rules, listed in Table 6-5.

The inclusion of ***InvalidScenario*** class and its subclasses was required to assess which scenarios do not meet objective requirements, as the nature of a SWRL rule only allows one atom at the head of the rule, thus for every rule checking for ***ValidScenario***, its opposite exists for ***InvalidScenario***. With the current rules in place, ***ValidScenario*** and ***InvalidScenario*** classes are not mutually exclusive. This is because a scenario can satisfy one objective, such as 3.a from Table 6-3, and if evaluated true, be classified as a ***ValidTotalEgressTimeScenario,*** subclass of ***Valid Scenario,*** but fail another (like 3.b) and be classified as an ***InvalidCapacityEgressScenario***, and consequently a subclass of ***InvalidScenario***. This means that a scenario can be valid for one objective, but invalid for another objective and thus be categorised simultaneously as both valid and invalid. To mitigate this limitation, another rule is put in place which checks that all objectives are met at the same time, categorising it as a ***FullyValidScenario*** class within the developed FBA ontology. This class is used by a SWRL rule which effectively intersects the first two rules R-FBA-1 and R-FBA-2. The nature of SWRL rules reasoning and combined with OWL expressivity causes the rules to be very specific in nature and be applied on well-defined classes for them to process fast and correctly.

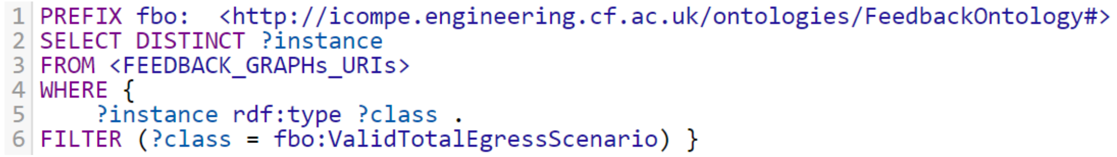
The knowledge mining process is undertaken by the intelligence imbedded within the developed ontologies and SWRL rules by following the imposed processes at a scenario level (using performance indicators such as egress/travel times). This method facilitates the finding of new knowledge about the performance of each specific scenarios stemming from a version of a building model (in IFC). The results are presented to users for further analysis and decision-making. The level of new knowledge is dependent on the expressed knowledge and developed processes to retrieve it.

Table 6-4. Developed queries retrieving knowledge from the FBA model

SPARQL QUERIES

Q-FBA-1, Find valid total egress scenarios

Question	Which scenarios are a ***ValidTotalEgressScenario*** class?

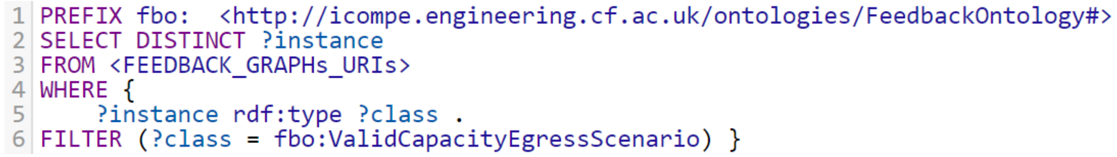


```
PREFIX fbo:  <http://icompe.engineering.cf.ac.uk/ontologies/FeedbackOntology#>
SELECT DISTINCT ?instance
FROM <FEEDBACK_GRAPHs_URIs>
WHERE {
    ?instance rdf:type ?class .
FILTER (?class = fbo:ValidTotalEgressScenario) }
```

Function	Finds the valid scenarios which evaluate true by applying rule R-FBA-1.
	Requires reasoning? YES

Q-FBA-2, Find valid capacity egress scenarios

Question	Which scenarios are a ***ValidCapacityEgressScenario*** class?

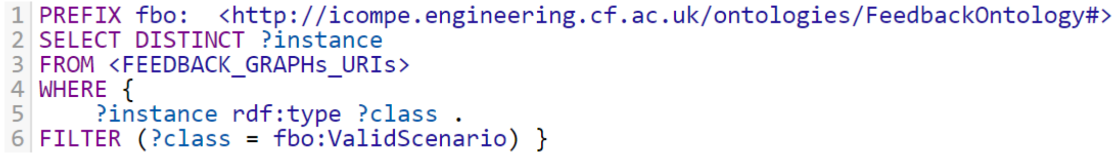


```
PREFIX fbo:  <http://icompe.engineering.cf.ac.uk/ontologies/FeedbackOntology#>
SELECT DISTINCT ?instance
FROM <FEEDBACK_GRAPHs_URIs>
WHERE {
    ?instance rdf:type ?class .
FILTER (?class = fbo:ValidCapacityEgressScenario) }
```

Function	Finds the valid scenarios which evaluate true by applying rule R-FBA-2.
	Requires reasoning? YES

Q-FBA-3, Find valid scenarios

Question	Which scenarios are a ***ValidScenario*** class?

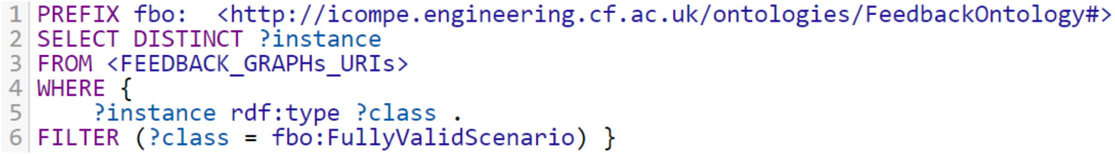


```
PREFIX fbo:  <http://icompe.engineering.cf.ac.uk/ontologies/FeedbackOntology#>
SELECT DISTINCT ?instance
FROM <FEEDBACK_GRAPHs_URIs>
WHERE {
    ?instance rdf:type ?class .
FILTER (?class = fbo:ValidScenario) }
```

Function	Finds scenarios which are valid from any rule applied. Computes the union valid subclasses (implicitly applies rules R-FBA-1 and R-FBA-2.
	Requires reasoning? YES

Q-FBA-4, Find fully valid scenarios

Question	Which scenarios are a ***FullyValidScenario*** class?

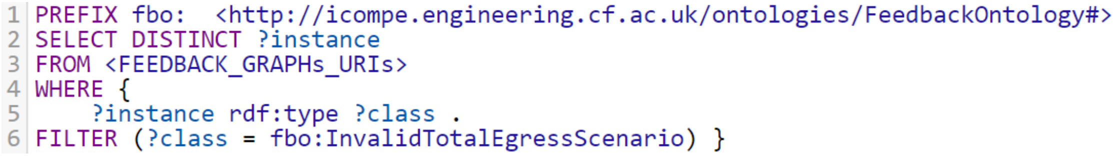


```
PREFIX fbo:  <http://icompe.engineering.cf.ac.uk/ontologies/FeedbackOntology#>
SELECT DISTINCT ?instance
FROM <FEEDBACK_GRAPHs_URIs>
WHERE {
     ?instance rdf:type ?class .
FILTER (?class = fbo:FullyValidScenario) }
```

Function	Find the valid scenarios which evaluate true by applying rule R-FBA-3.
	Requires reasoning? YES

Q-FBA-5, Find invalid total egress scenarios

Question	Which scenarios are an ***InvalidTotalEgresScenario*** class?

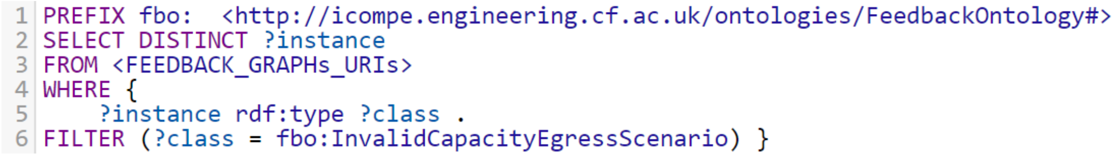


```
PREFIX fbo:  <http://icompe.engineering.cf.ac.uk/ontologies/FeedbackOntology#>
SELECT DISTINCT ?instance
FROM <FEEDBACK_GRAPHs_URIs>
WHERE {
     ?instance rdf:type ?class .
FILTER (?class = fbo:InvalidTotalEgressScenario) }
```

Function	Find the invalid scenarios which evaluate true by applying rules R-FBA-4 and R-FBA-5.
	Requires reasoning? YES

Q-FBA-6, Find invalid capacity egress scenarios

Question	Which scenarios are an ***InvalidCapacityEgresScenario*** class?

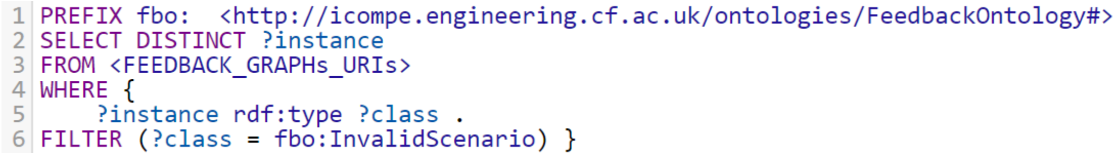


```
PREFIX fbo:  <http://icompe.engineering.cf.ac.uk/ontologies/FeedbackOntology#>
SELECT DISTINCT ?instance
FROM <FEEDBACK_GRAPHs_URIs>
WHERE {
     ?instance rdf:type ?class .
FILTER (?class = fbo:InvalidCapacityEgressScenario) }
```

Function	Find the invalid scenarios which evaluate true by applying rule R-FBA-6.
	Requires reasoning? YES

Q-FBA-7, Find invalid scenarios

Question	Which scenarios are an ***InvalidScenario*** class?



```
PREFIX fbo:  <http://icompe.engineering.cf.ac.uk/ontologies/FeedbackOntology#>
SELECT DISTINCT ?instance
FROM <FEEDBACK_GRAPHs_URIs>
WHERE {
     ?instance rdf:type ?class .
FILTER (?class = fbo:InvalidScenario) }
```

Function	Finds scenarios which are invalid from any rule applied. Computes the union of subclasses that are invalid (implicitly calls rules R-FBA-5 and 6)
	Requires reasoning? YES

Table 6-5. SWRL rules operating with the FBA ontology for classifying scenarios

SWRL RULES	
SWRL CODE	DESCRIPTION
R-FBA-1, ValidTotalEgressScenario	
hasObjective(?objectivesSet, ?objective) ^ FindTotalEgressTime(?objective) ^ hasTimeLimit(?objective, ?requirement) ^ timeInSeconds(?requirement, ?timeLimit) ^ appliesToScenario(?objectivesSet, ?scenario) ^ hasEndResult(?scenario, ?result) ^ TotalEgressTime(?result) ^ timeInSeconds(?result, ?timeResult) ^ swrlb:lessThanOrEqual(?timeResult, ?timeLimit) ^ hasResult(?scenario, ?popResult) ^ PopulationResult(?popResult) ^ numberRemainingAgents(?popResult, ?remainingAgents) ^ swrlb:equal("0"^^xsd:integer, ?remainingAgents) -> ValidTotalEgressScenario(?scenario)	If a simulation result is below the required time specified in the objectives AND there are no remaining agents within the simulation, then the scenario is classified as a valid ValidTotalEgressScenario in the FBA ontology. The opposite of rule R-FBA-4 and R-FBA-5
R-FBA-2, ValidCapacityEgressScenario	
hasObjective(?objectivesSet, ?objective) ^ FindCapacityEgressStatus(?objective) ^ hasTimeLimit(?objective, ?timeRequirement) ^ timeInSeconds(?timeRequirement, ?timeValue) ^ hasPopulationCapacity(?objective, ?percentageRequirement) ^ percentageRequired(?percentageRequirement, ?percentageValue) ^ appliesToScenario(?objectivesSet, ?scenario) ^ hasIntermediateResult(?scenario, ?simulationTimeResult) ^ SimulationTime(?simulationTimeResult) ^ timeInSeconds(?simulationTimeResult, ?timeResult) ^ swrlb:lessThanOrEqual(?timeResult, ?timeValue) ^ percentageEvacuated(?simulationTimeResult, ?percentageResult) ^ swrlb:equal(?percentageResult, ?percentageValue) -> ValidCapacityEgressScenario(?scenario)	If an intermediate result has a certain capacity of the population evacuated below a certain time, it is a valid scenario - ValidCapacityEgressScenario in the FBA ontology. It is the opposite rule for R-FBA-6
R-FBA-3, FullyValidScenario	
ValidTotalEgressScenario(?scenario) ^ ValidCapacityEgressScenario(?scenario) -> FullyValidScenario(?scenario)	A scenario satisfies multiple criteria objectives at the same time, it is a valid scenario – FullyValidScenario in the FBA ontology
R-FBA-4, InvalidTotalEgressScenario	
hasObjective(?objectivesSet, ?objective) ^ FindTotalEgressTime(?objective) ^ hasTimeLimit(?objective, ?requirement) ^ timeInSeconds(?requirement, ?timeLimit) ^ appliesToScenario(?objectivesSet, ?scenario) ^ hasEndResult(?scenario, ?result) ^ TotalEgressTime(?result) ^	If a simulation result is above the required time specified in the objectives, then the scenario is classified as invalid - InvalidTotalEgressScenario in the FBA ontology. The opposite of rule R-FBA-1, and will act in conjunction with R-FBA-5 to check agent

timeInSeconds(?result, ?timeResult) ^ swrlb:greaterThan(?timeResult, ?timeLimit) -> InvalidTotalEgressScenario(?scenario)	numbers remaining in the simulation.
R-FBA-5, InvalidTotalEgressScenario	
hasObjective(?objectuiveSet, ?objective) ^ FindTotalEgressTime(?objective) ^ appliesToScenario(?objectiveSet, ?scenario) ^ hasEndResult(?scenario, ?result) ^ PopulationResult(?result) ^ numberRemainingAgents(?result, ?remainingAgents) ^ swrlb:notEqual("0"^^xsd:integer, ?remainingAgents) -> InvalidTotalEgressScenario(?scenario)	If a simulation still has remaining agents which have not evacuated in time, then the scenario is classified as invalid - InvalidTotalEgressScenario in the FBA ontology. The opposite of rule R-FBA-1, and acts in conjunction with rule R-FBA-4 which checks the required evacuation time.
R-FBA-6, InvalidCapacityEgressScenario	
hasObjective(?objectivesSet, ?objective) ^ FindCapacityEgressStatus(?objective) ^ hasTimeLimit(?objective, ?timeRequirement) ^ timeInSeconds(?timeRequirement, ?timeValue) ^ hasPopulationCapacity(?objective, ?percentageRequirement) ^ percentageRequired(?percentageRequirement, ?percentageValue) ^ appliesToScenario(?objectivesSet, ?scenario) ^ hasIntermediateResult(?scenario, ?simulationTimeResult) ^ SimulationTime(?simulationTimeResult) ^ timeInSeconds(?simulationTimeResult, ?timeResult) ^ swrlb:greaterThan(?timeResult, ?timeValue) ^ percentageEvacuated(?simulationTimeResult, ?percentageResult) ^ swrlb:equal(?percentageResult, ?percentageValue) -> InvalidCapacityEgressScenario(?scenario)	If an intermediate result with a specified capacity of the population evacuated later than the required time, it is an invalid scenario - InvalidCapacityEgressScenario in the FBA ontology. It is the opposite rule for R-FBA-2
R-FBA-7, ValidTimeInstantEgressScenario	
hasObjective(?objectivesSet, ?objective) ^ hasTimeInstant(?objective, ?requirement) ^ timeInSeconds(?requirement, ?timeInstant) ^ appliesToScenario(?objectivesSet, ?scenario) ^ hasIntermediateResult(?scenario, ?popResult) ^ PopulationResult(?popResult) ^ atRuntime(?popResult, ?simulationTimeResult) ^ timeInSeconds(?simulationTimeResult, ?simulationTime) ^ swrlb:equal(?timeInstant, ?simulationTime) ^ numberRemainingAgents(?popResult, ?remainingAgents) ^ swrlb:equal("0"^^xsd:integer, ?remainingAgents) -> ValidTimeInstantEgressScenario(?scenario)	If an intermediate result has evacuated all agents at a specific time, it is a valid scenario – ValidTimeInstantEgressScenario in the FBA ontology

6.3.3. Linking models for future extensibility

From the two previous sections, methods to store and retrieve information and knowledge were presented in a specifically applied way, to meet system requirements. As resources are stored and retrieved throughout the design process, it is necessary to conceptualise the relationships between models over the BIM lifecycle and how CSS and FBA models converge for future extensibility to other design problems.

Concerning the way in which the system deals with scenarios on a large-scale, Figure 6-18 conceptualises the relationships between information and knowledge models over time. It is important to consider the progression of the BIM model over its lifecycle on the Y axis, where changes in building design or layout are expected. This effectively brings forth a new design problem with regard to the evacuation plan. On the X axis, the figure shows the change in scenario context, with each scenario assuming different things, and each performing differently. The scenario and feedback models are labelled as 'dynamic' because they refer to the 'static' models in different circumstances. The instances across all models eventually refer to the same 'things' in reality. The feedback analysis process is managed with the help of the feedback models, which link the scenarios their assumptions, results and user input together with the BIM design model for collective analysis.

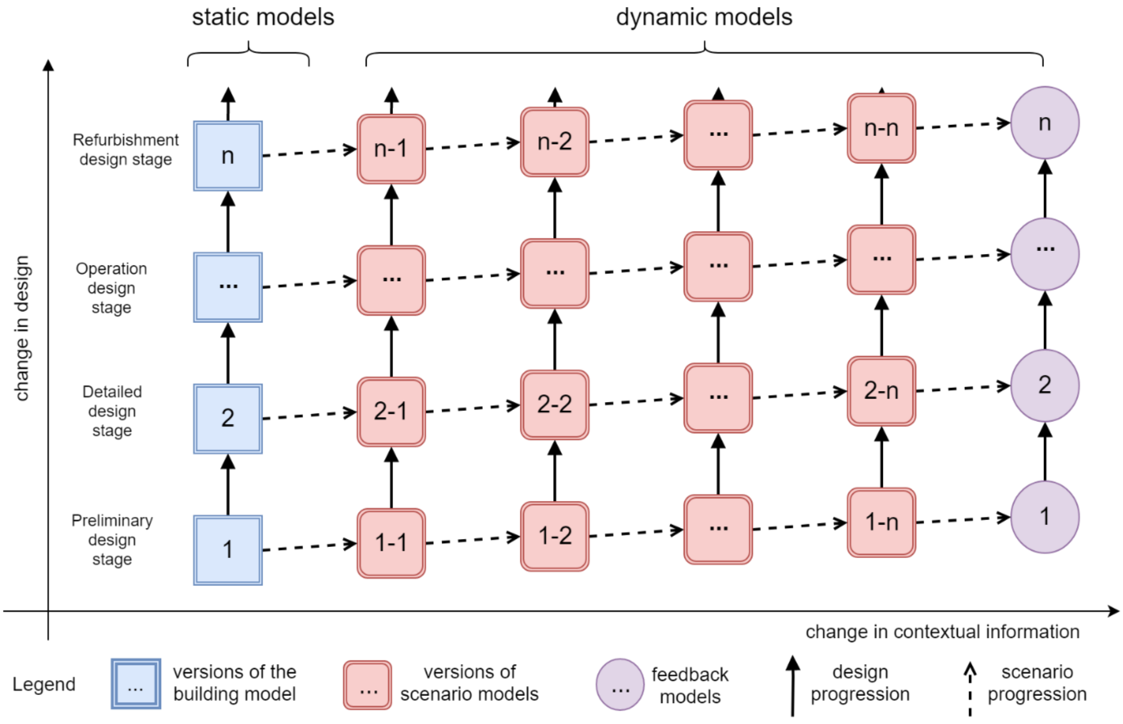


Figure 6-18. Static and dynamic information model progression considering changes in design and context

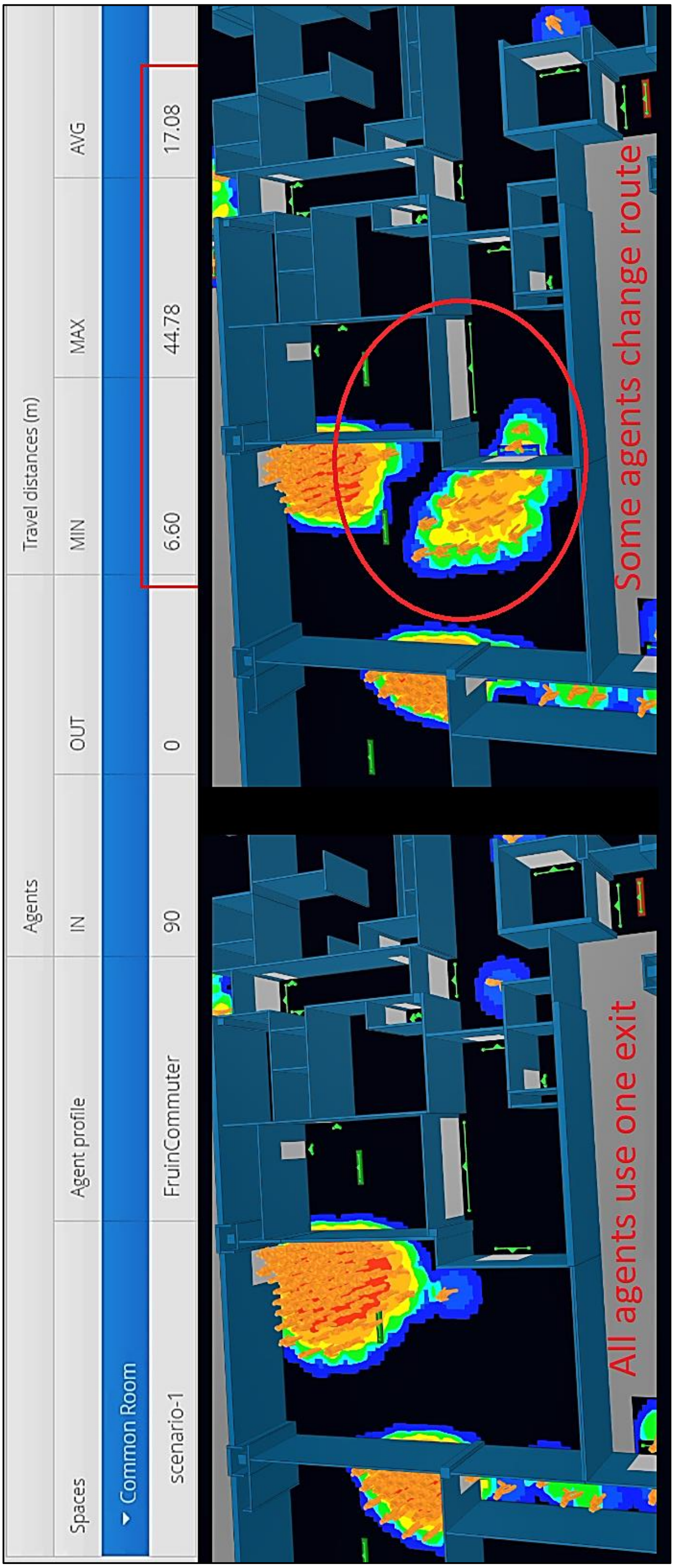
6.3.4. Scenario vs element view

The previous sections of this chapter introduced ways to reason about linked data on a scenario level and on an object level. However, when it comes to Stage II, knowledge operators were defined only for a scenario model level. The main cause of this is the lack of formalised knowledge from design guidance.

From discussion with industry experts, it was noted that besides the spaces, doors and stairs are the types of objects which can cause concerns at a design level. However, there are very few ways in which these are effectively measured and assessed. The most common way, as was also mentioned in Section 4.2.2 is to rely on travel time, which can relate to an entire scenario, a space or even individual agents. Conventionally, travel time and distances from spaces to nearest exits are assessed geometrically using layout plans.

When it comes to assessing this using CSTs, safety engineers rely on visualising the problem or on various tool features allowing them to track agents. Figure 6-19 shows an example of measured travel time and distances of agents, reported as an average in terms of the space of origin. This sort of information is aggregated together at an object level by the ONTOCS system making use of simulation data and cross-scenario linking of concepts, as shown in the previous section. From the room in question, over 90 agents evacuate using available exits. The shortest exit is only 6m away, whereas the longest is over 40m. A human observing these events in an animation can easily explain why that is, certain agents will evacuate on a different route, as CST have the ability to simulate such complex human behaviours. However, when attempting to imbed this sort of awareness into a knowledge system, things are more challenging.

Although agent movement can be tracked through model calculations, it can be hard to correctly interpret their behaviour. Due to the many assumptions present in each scenario and the large data provided by simulation outputs, explaining ‘why something is happening’ is the role adopted by the FBA ontology. Due to the complex interaction of concepts, it can become very complex to represent in an ontology, or undecidable when applying rules. A second example is conceptualised in Figure 6-20 showing that some factors may not explain the cause of certain results and their behaviour. As such, it is required to leverage the embedded knowledge and the relationships that exist between the different assumptions. Let’s consider the example of a forming bottleneck in a certain area in a building, like ***Space 3*** shown in Figure 6-20. High traffic density in certain areas is caused by the influx of agents provided by various origin points, i.e. ***Spaces 1 and 2***. However, determining which origin point has more impact in causing the bottleneck is a complicated problem, as it is dependent on many factors such as agent characteristics, geometry of the spaces, distribution of agents, etc.



Spaces	Agent profile	Agents		Travel distances (m)		
		IN	OUT	MIN	MAX	AVG
▾ Common Room						
scenario-1	FruinCommuter	90	0	6.60	44.78	17.08

Figure 6-19. Example of scenarios where a group of agents decide to change the evacuation route

Complex rules need to be put in place which can represent this in formalised ontological and semantical knowledge that could help determine the causes.

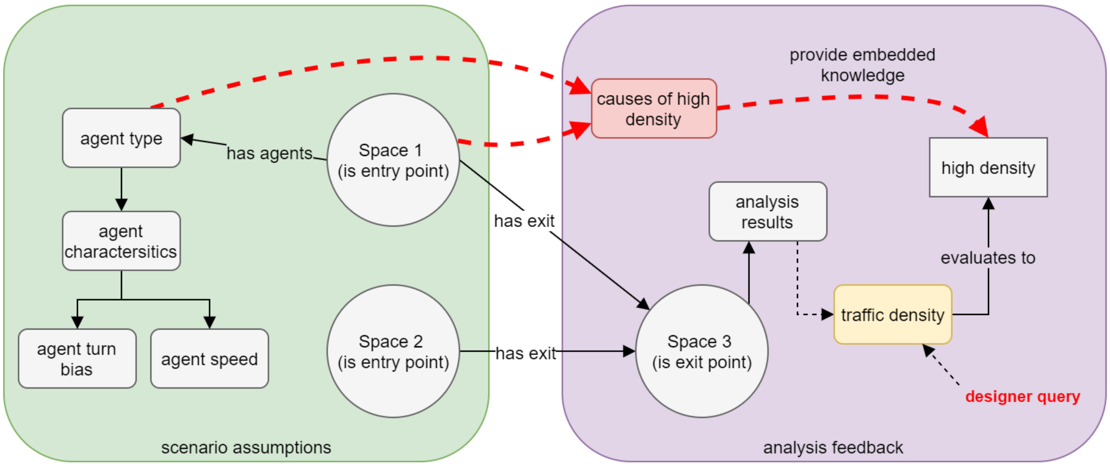


Figure 6-20. Example of objects and properties (contextual and geometric) influencing the final analysis result

When considering such rules, careful consideration is required along with validation of the rules. The retrieval of such knowledge is complex, and it is limited by the reasoning types that ontologies and SWRL rules can provide. Additionally, due to the OWA which governs ontology reasoning, where evaluation of rules can be TRUE or FALSE, but also UNKNOWN.

The easier alternative is to make use of linked data to leverage its connections of objects and present the results in an easy to interpret way by designers, as was shown in Figure 6-19.

6.4. Summary

This chapter presented ways in which knowledge about building design performance can be stored and retrieved intelligently, and how to make use of explicit and implicit knowledge which can be leverage by using semantic web ontologies, SWRL rules and SPARQL queries. The chapter began by showing which data and knowledge is worth storing throughout the process and how, while hinting at the limitations and challenges of such a method. The second part outlines ways in which implicit and explicit knowledge is mined using knowledge operators developed, discussing how they work and what the concerns are when constructing them. The knowledge mining is facilitated by ontologies reasoning in conjunction with rules, by categorising scenarios in accordance with design objectives. The chapter finished by showcasing how knowledge models interact on a higher level and mentioned the challenges and limitations of machine-interpretable rules at an object level feedback concerning crowd simulation-based evacuation models.

Chapter 7. System implementation, testing and validation

This chapter presents the development of the ONTOCS (Ontology Crowd Simulation) system and validation of the developed knowledge base from previous chapters. The first part (Section 7.1) outlines the system design. The second part (Section 7.2) presents a case study carried out on a real-life building. The tests are done following the overall methodology presented in Section 3.2, and each use-case carried out aims to prove that the system is functional and reliable when performing knowledge mining. The case study objectives, use-cases and rationale are presented in detail before showing and discussing the results.

7.1. Introduction to the ONTOCS system

ONTOCS follows the conceptual process of knowledge mining-based design iteration loop, as previously introduced in Chapter 3. The basic principles of knowledge mining and storage processes and their requirements have influenced the system architecture. Although the overall framework makes use of several other third-party tools, from the programming perspective, ONTOCS controls the processes and information exchange to facilitate a semi-automatic process of multi-scenario construction and analysis on a large scale.

The main independent tools which collaborate during the design process are:

1) ONTOCS – the main system packages developed in Java. It is responsible for controlling the entire process, connecting the different tools and models together, as well as providing the user interface. Its main class, ***SystemManager***, integrates all the packages;
2) Stardog – the RDF database server responsible for storing all the relevant ontologies and to provide reasoning in the back-end;
3) MassMotion – the CST (introduced in Section 4.3.1) which is responsible with running the scenarios on a large scale and provides raw data on the evacuation events within the simulated building environment;

4) IFC to RDF converter – converts IFC EXPRESS format files to RDF models to work within the schema of IfcOwl (OpenBIMstandards 2017a);
5) Jetty server (The Eclipse Foundation 2018) – provides the database support for hosting the entire ONTOCS application to run as a web service. It is used to store all simulation files and results databases in SQL, as part of each design session.

Although BIM tools were not envisaged as part of the system, they are expected to provide the building digital models. To ensure openness and interoperability, the IFC format was preferred. During testing and development, the building modelling was done using Autodesk Revit 2016 and 2017. The possibility of embedding the ONTOCS interface into a BIM tool was considered initially. However, this was omitted in order to avoid relying too much on a single BIM platform.

7.1.1. ONTOCS process workflow

The high-level interaction between the system components is shown in Figure 7-1. The components follow a specific workflow process which guide the users through the two main stages of the knowledge mining process. The arrows in the figure indicate the flow of information and the collaboration between the several tools and ontologies. The process starts with the acquisition of all the necessary information via ***Stage I input,*** which considers input from several sources: most importantly the IFC building models, which are the focal point around which knowledge is stored, but also the user input which is considered at each step. The ontologies developed and discussed in Chapter 5 are hosted on Stardog graph databases. The ontology schema models and resources are all part of the same graph store, where all the relevant data is linked, with no external resources having been used at the current stage. The ontologies are constantly queried, and data and knowledge resources are stored explicitly to facilitate a smooth flow of the process. ***Stage I inputs*** and resources are used to generate simulation scenarios - which effectively become ***Stage II inputs***. Finally, simulation results and user objectives are reasoned to provide performance analysis outputs back to the users via the interface. The entire process is defined by eight main processes of controlled data flow, as numbered in Figure 7-1:

1) Converting building from IFC to RDF - this is done automatically by the system when uploading a chosen IFC model. Alternatively, direct upload of an IfcOwl instance model is possible;
2) Uploading the IfcOwl digital building model on the RDF store - the IfcOwl schema ontology is also uploaded during this process which gives context to the model resources;
3) Processing of user input and additional resources – other schema ontologies, like the CSS and FBA are uploaded to the same RDF session store (additional

ontologies can also be uploaded); User input concerning scenario context is captured by the interface and processed by the ONTOCS packages, with relevant scenario assumptions being parsed into CSS ontology individuals as separate RDF resource graphs; all these resources are then used to carry out step 4).

4) Generating MassMotion scenario files – the number of scenarios is inputted by the user, with each scenario context being outputted into a separate file which is then executed at step 5). The CSS ontology and other relevant linked resources are queried to find the data required to create functional MassMotion scenario files (queries and rules applied at this step were described in Section 6.3.1);

5) Executing MassMotion scenario files – the ONTOCS ***SystemManager*** class passes the generated files to the MassMotion tool via the console; each scenario is executed by the tool which provides confirmation on the status of each run back to the ***SystemManager***;

6) Recording simulation results – MassMotion saves simulation results in SQLite databases, which are then kept on the Jetty server in specific session folders. These are accessed by the ONTOCS application ad-hoc for finding results which are related to user objectives;

7) FBA ontology reasoning processes – the inputted user objectives and scenario results are saved into the FBA resources graphs for rules processing (knowledge operators for this stage were described in Section 6.3.2);

8) Retrieving knowledge about the design – reasoning results from step 7) are retrieved by the system and outputted to the user interface in collections of results, making use of the ***AnalysisObjectivesSet*** class from the FBA ontology – essentially each set of objectives was reasoned and provided results within the scope of the investigated scenarios.

The steps described above present an overview of the process workflows of information and knowledge storage and retrieval, following the implementations from Chapter 6. As discussed in previous chapters, due to the complex nature of crowd simulation model and the input requirements, each step relies on the correct execution of previous ones. Additionally, knowledge management processes are highly dependent on the context of the information (Bates 2011). As such, formalising all the concepts need to be a prime concern in order to for the output to be correct and relevant.

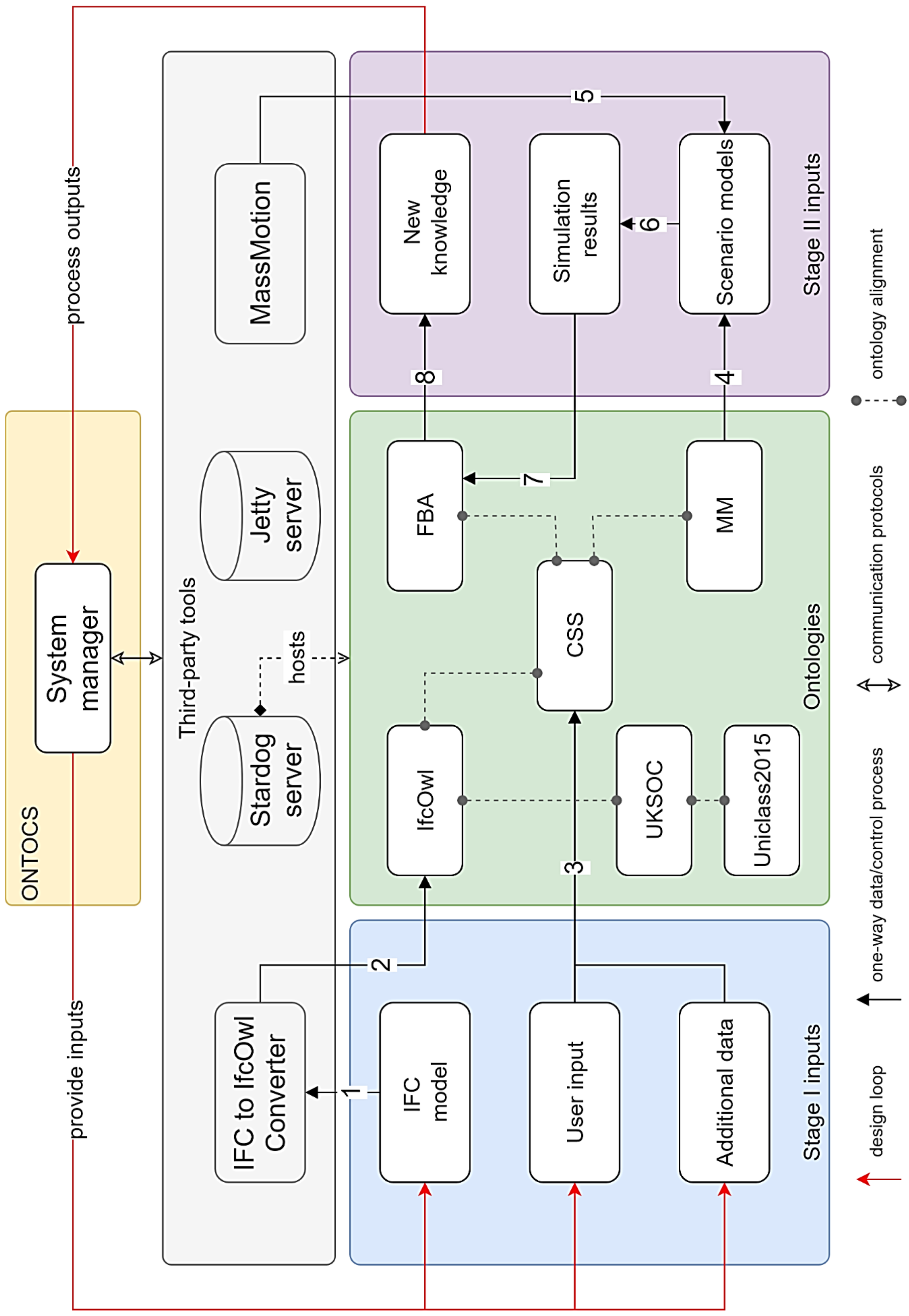


Figure 7-1. ONTOCS system components with numbered workflow steps

7.1.2. ONTOCS architecture

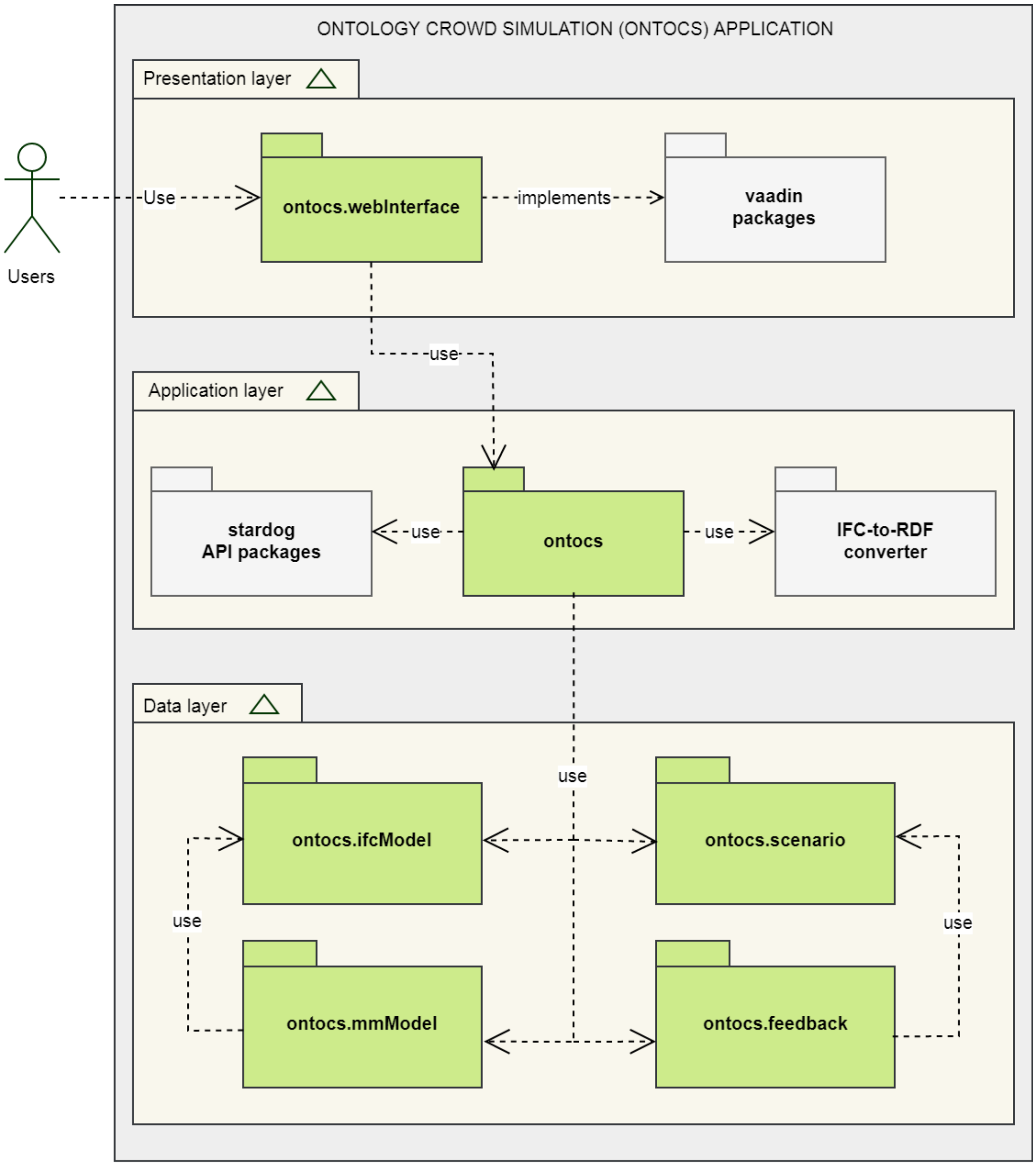


Figure 7-2. ONTOCS system package diagram across functional layers; packages in green represent developed code; packages in grey represent imported code

The system architecture is presented in Figure 7-2 using Unified Modelling Language (UML) diagrams (Pilone and Pitman 2005). The core package is ***ontocs***, which includes the ***SystemManager*** class. This class was designed according to different use case scenarios to facilitate the workflow of the entire process. It saves all the data in memory for each session and is the package which facilitates collaboration between different applications. It makes use of secondary APIs and third-party developed programs to communicate with the Stardog server, convert IFC to IfcOwl and to run MassMotion as a background service when executing the simulations. The ***ontocs*** package is reliant on

4 other sub-modular packages which deal with data extraction and manipulation as shown in Figure 7-2:

1) ***ontocs.ifcModel*** package – retrieves data from the IfcOwl RDF instance models by running SPARQL queries. Once the relevant data about model objects is retrieved, including identity data, geometry and properties, it is stored in memory for future manipulations and conversions to other models and tools; this package also includes methods to create the geometry from IFC schema concepts;
2) ***ontocs.mmModel*** package – its primary function is to facilitate programmatic conversion from IFC objects in memory to MassMotion objects. Additionally, it has methods for enabling communication with the external MassMotion simulation tool;
3) ***ontocs.scenario*** package – manages information about scenario set-up and user input assumptions; uses the CSS and other ontology resources to store explicit data and knowledge or to retrieve implicit information about a scenario. The package methods follow a specific workflow meant to facilitate correct creation of simulation scenarios. For example, it begins with identifying the geometry environment, then proceeds to create events according to ontology resources and user input;
4) ***ontocs.feedback*** package – uses the FBA ontology to store simulation results and user input, and to retrieve implicit information from the SWRL rules defined for the analysis stage. This package collaborates with the previous one which stores and manages the data about scenarios in memory.

Considering the prototype architecture of the ONTOCS system, several limitations were observed throughout development and testing.

Firstly, regarding the IFC model, special classes have been created to tackle not only the retrieval of basic IFC constructs from the IfcOwl ontology, but also the ability to generate the geometry from scratch. This requires significant development and upkeep for future implementations.

Secondly, the high-level generalisation of the CSS ontology is unable to provide all the details for constructing complete object data at a CST level. Using multiple CSTs to be part of the design loop might be beneficial when comparing different performance results, but this requires specifically tailored packages to integrate each CST with the system.

Thirdly, the scenario and feedback packages include classes that query the knowledge base, as well as classes that parse data from memory objects and populate the resource graphs for different ontology domains. This requires extensive knowledge of the used ontology structures and high upkeep costs to ensure data correctness.

Finally, the presentation layer of the system is required for processing user input and guiding the users throughout the design workflow. Although the system provides a high degree of automation concerning BIM data, the user input remains vital for constructing scenarios and knowledge which is relevant to the situation at hand. This means that the interface design must provide engineers with the necessary tools for correctly assessing the information and design knowledge returned by the system. The interface implemented for ONTOCS is provided in Appendix D.

7.2. ONTOCS Case study on Queen's Buildings

The second part of this chapter focuses on testing the ONTOCS platform though a case study carried out on a real-life building. The aim of the case study was to assess the viability of using an ontology-based system in a design context. Several objectives were defined to break down the research question Q7.

Firstly, the ability of the system to understand the building data and secondary resources correctly was considered vital. Thus, during development, the imbedded code and knowledge operators were constantly tested and improved. The case study is used to validate this assumption.

Secondly, the efficiency of the system to work with large datasets was considered important in showing the benefits of automation. This was assessed by comparing manually constructed models to automatically constructed ones for one use case. Query speed and scalability was tested by processing results on a set of 36 simulations for a second use case.

Thirdly, the reliability on building design codes to provide population data was investigated when other sources of data is not available. To assess this, simulation results of real building occupancy data were compared with simulation results of design codes' data.

The next sections introduce the building and outline the case study objectives and assumptions. The results are then provided in two separate sections, one for Stage I – Scenario generation (Section 7.2.3) and one for Stage II – Analysis Feedback (Section 7.2.4), each of the sections using a different use case on the same building model. Finally, the results are discussed in Section 7.2.5 following the defined objectives.

7.2.1. Building description

The building chosen for the case study is an academic environment building from Cardiff University. The building has several floors, however only the ground floor level was taken into consideration for simplicity and for a more in-depth analysis. The ground level has a good mix of spaces such as laboratories, dining areas, common rooms, offices, lecture rooms and many auxiliary spaces. The environment experiences a lot of traffic during the day, depending on the academic courses.

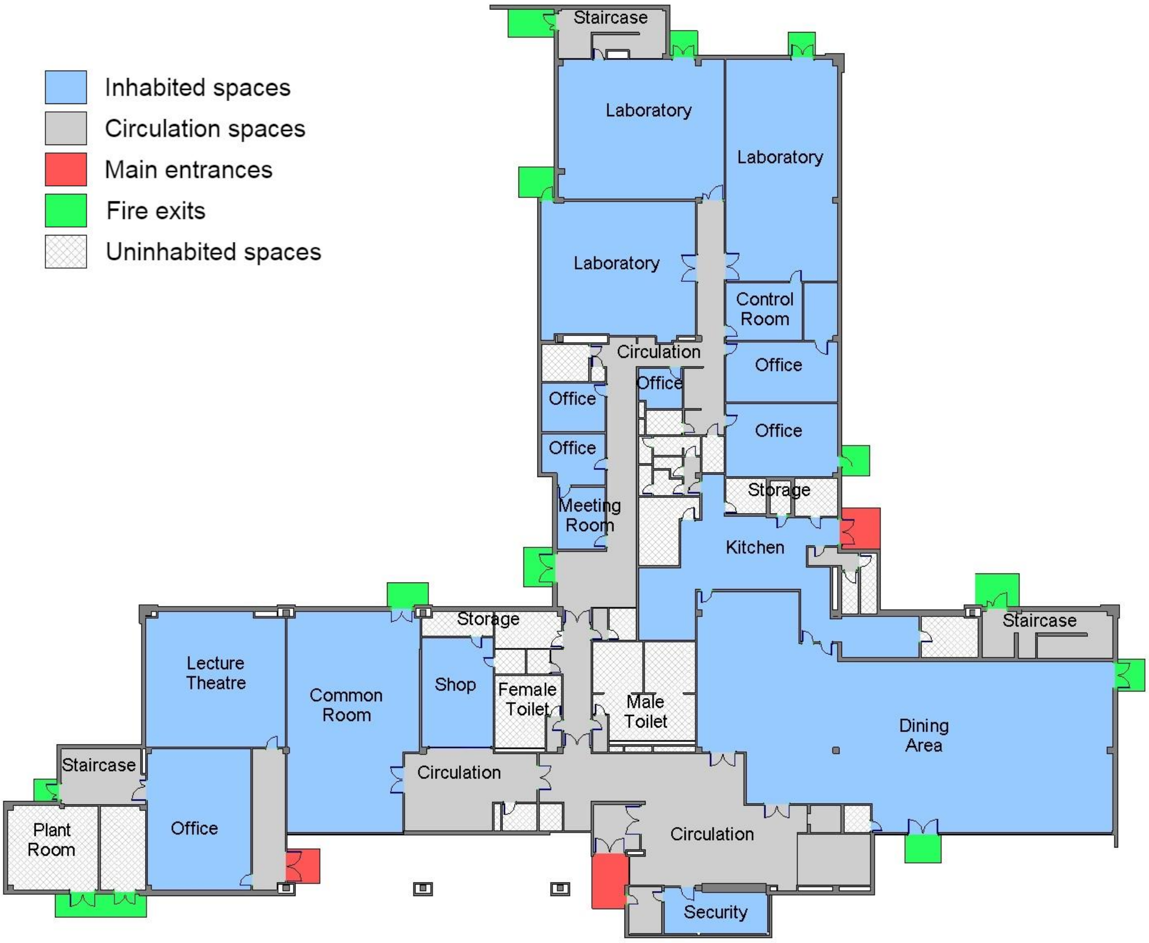


Figure 7-3. Case study building layout, ground floor

A survey of the use of the building was carried out, with the aim to identify the level of occupancy, available fire exits and routes. The layout shown in Figure 7-3 divides the spaces in several categories for simplicity, with a full description of each space provided in Appendix E. The ground floor has three main fire compartments, each with their respective entrances outlined in red. These main entrances experience heavy traffic on a daily basis. All available data regarding space occupancy and functionality was attached to objects during the modelling stage, and therefore is present explicitly in the IFC/IfcOwl models for fast processing by knowledge operators.

7.2.2. Use cases – objectives, assumptions and rationale

The following questions outline the objectives of the case study:

1) How fast is a scenario model generated, in terms of geometry and context?
2) How fast is the context created when relying on design guide resources?
3) What are the differences between a manually created simulations and those created by the ONTOCS system?
4) How reliable are the simulation results originating from the automatic process for future analysis consideration?
5) Is the system correctly interpreting the results according to user inputted design objectives?
6) How reliable are design occupancy factors in comparison to real building occupancy data?
7) How efficient is the reasoning process for evaluating user objectives on a large scale?
8) How do query times scale with increasing number of simulations?

Use case for Stage I – Scenario Generation

The first use case aims to answer objectives 1) to 4), which are directly targeted at the automatic scenario generation stage.

A number of scenarios were developed manually using the MassMotion simulation tool, based in the data gathered about the building. The results of this process were then compared to the results provided by scenarios created automatically by the ONTOCS system. The manually constructed scenarios are presented in Table 7-1, following the recommendations from the PD 7974 (2004) of simulating scenarios at 100% and 33% building capacities with and without the main entrances being available as evacuation exits. The different capacity percentages are expected to allow for the estimation of different evacuation times as was introduced in Section 2.1.1.

A pre-set MassMotion agent profile was used across all scenarios, which is in accordance to the PD7974 document. The agents were programmed to evacuate as soon as the simulation starts and to head to the nearest exits available to them. In the case of manually constructed models, agents were modelled to evacuate through the nearest exits, according to the real building fire plan, not being allowed to choose fire exits from other fire departments. On the other hand, the automatically generated scenarios assume that all agents are aware of all the exits, thus MassMotion will aim to optimise the flow of agents. The rationale behind this decision is that with assumptions being the same, the MassMotion tool will give nearly identical results. This also allows to assess if the available ontology information is enough to generate similar scenarios to those done manually.

The appearance of the agents within the simulation was set to be instantaneous and simultaneous, meaning that all agents assigned to a specific space appear at the start of the simulation and all at the same time. This simulates the act of people being already present in each inhabited space. This however creates high density areas in some parts of the model at simulation start. To increase the level of realism, scenarios SC5 and SC6 from table 7-1 have been manually developed with more distributed agent entries across larger rooms. For these two scenarios, several other portal objects were constructed for 7 out of the 17 inhabited spaces, allowing the generation of agents to be less dense at the start of the simulation. All the walls and columns inside the building shell are present, therefore simulating under applicable design conditions.

Table 7-1. List of constructed and analysed scenarios

Scenario	Capacity	Entrances	Exits	Agents	Profile
SC1	100%	available	available	373	PD7974
SC2	100%	blocked		373	
SC3	33%	available		124	
SC4	33%	blocked		124	
SC5*	100%	available		373	
SC6*	100%	blocked		373	

Note: Scenarios SC5 and SC6 were only constructed manually for arguing a more realistic distribution of agents across larger rooms. These types cannot be created by the ONTOCS system at this time.

Use case for Stage II – Analysis Feedback

The use case for the second stage was created to address questions 5) to 8), which are set in the context of a larger scale of simulations. A number of 36 simulation scenarios were inputted into ONTOCS.

These scenarios were divided in two categories depending on the population data source (Table 7-2). Each scenario assumes a different population capacity, as is recommended in PD 7974 (2004) when assessing the performance of a specific building layout. In a first series of scenarios (1 to 18), data is taken from the IFC model data, which is present explicitly within the constructed model via custom object properties (e.g. each space object has an assigned number of occupants). The data present in the IFC model reflects the surveyed maximum capacity of each space from the building in real life. The second series of scenarios (19 to 36), the IFC model is missing data about the population, which is reasoned by implicit means, where the type of space is identified from the Uniclass classification, which is then given an occupancy factor based on the UKSOC codes. This factor is then multiplied with the area of each space, giving an approximate number of occupants per space.

Most scenario assumptions are identical to those used in the previous use case, apart from the variation of the population capacity. Additionally, all scenarios in this use case assume the main entrances of the building to be blocked.

Table 7-2. Simulation scenarios created using the ONTOCS system for analysis

Scenario	Capacity	Population data	Entrances	Exits	Profile
1-18	30 – 200% (increments of 10%)	IFC model	blocked	available	PD7974
19-36		UKSOC			

Simulation results from all the simulations were manually checked against results returned by the system. Two sets of objectives (each with two sub-objectives) were assumed (Table 7-3), aimed at evaluating which scenarios satisfy them.

Table 7-3. Set of analysis objectives inputted into the system for evaluating the 36 simulation scenarios

Objective set	Objective	Objective		Valid scenarios
	(a) Total egress time (s)	(b) Capacity egress		
		population (%)	time limit (s)	
1	120	50	60	?
2	120	95	90	?

In order to assess the performance of the system on dealing with large data sets, a series of measurements were carried out when the system was evaluating the above objectives:

1) Single objective analysis – each objective from Table 7-3 was queried separately;
2) Multi-objective – both objectives from Table 7-3 were queried together.

To answer question 8) regarding system scalability, the steps above were repeated for multiple sessions, each with a different number of simulation scenarios stored in memory, from 1 to 36 in increments of 1 (e.g. one session ran 10 simulations, then a new session was created with 11 simulations, then 12, etc.). Each query in every case was tested 5 times to account for anomalies and average performance values were plotted. This resulted in each query being tested 180 times in total, and consequently some rules they depend on up to 720 times in total. The rationale behind querying objectives separately or together was to evaluate how rule reasoning time performs in each case, and thereby investigating if certain rules behave differently under multiple circumstances. The queries used to retrieve reasoning results were previously outlined in Chapter 6 and summarised in Table 7-4 below. Each query (with specific name and code) relies on one or multiple developed SWRL rules which are triggered when the RDF database is queried for new knowledge about the design. It is expected that each query execution time will differ based on the amounts of recorded results data from simulations, and on the number of inputted objectives from the user side. Thus, query times in these cases were measured and contrasted in the following sections.

Table 7-4. SPARQL queries and their respective SWRL rules operating within the FBA ontology

Query		Dependent rules	Objectives applied
Name	Code		
Find valid total egress scenarios	Q-FBA-1	R-FBA-1	single (a)
			multiple (a) & (b)
Find valid capacity egress scenarios	Q-FBA-2	R-FBA-2	single (b)
			multiple (a) & (b)
Find valid scenarios	Q-FBA-3	R-FBA-1 R-FBA-2	single (a)
			single (b)
			multiple (a) & (b)
Find fully valid scenarios	Q-FBA-4	R-FBA-3	single (a)
			single (b)
			multiple (a) & (b)
Find invalid total egress scenarios	Q-FBA-5	R-FBA-4 R-FBA-5	single (a)
			multiple (a) & (b)
Find invalid capacity egress scenarios	Q-FBA-6	R-FBA-6	single (b)
			multiple (a) & (b)
Find invalid scenarios	Q-FBA-7	R-FBA-4 R-FBA-5 R-FBA-6	single (a)
			single (b)
			multiple (a) & (b)

The following section presents the results from the testing, which are then discussed together at the end of this chapter.

7.2.3. Stage I – Scenario generation use case results

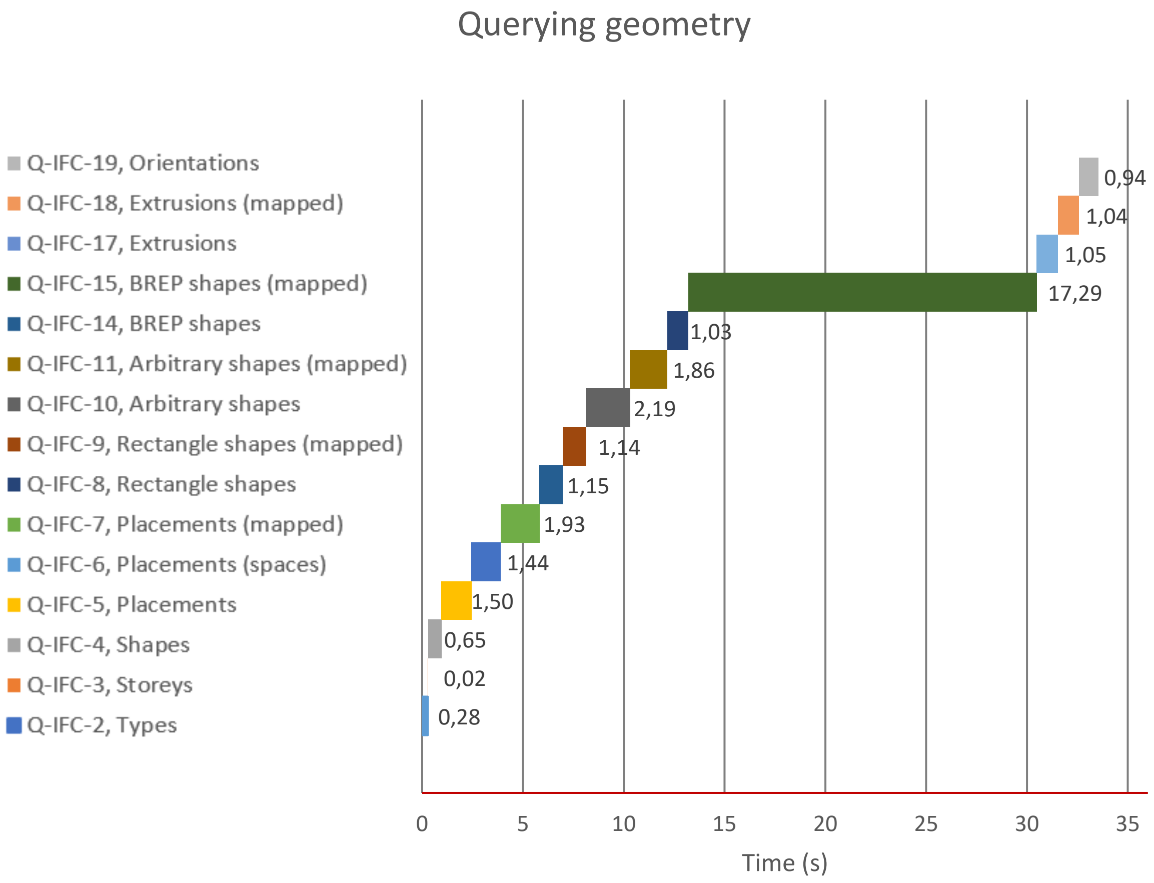


Figure 7-4. Average query times for geometry retrieval

Results on retrieving model information from IFC and additional resources are shown here. The query times for retrieving geometry is shown in Figure 7-4, while the context retrieval is shown in Figure 7-5.

The query times show an average retrieval time based on 10 measurements taken in total for each query (see Table E-2, Appendix E for full data). The retrieval times depend on the initial building model and the number of objects it contains.

The input model used is summarised in Table 7-5, showing its size in different formats, and the relevant model objects being found and converted from IFC.

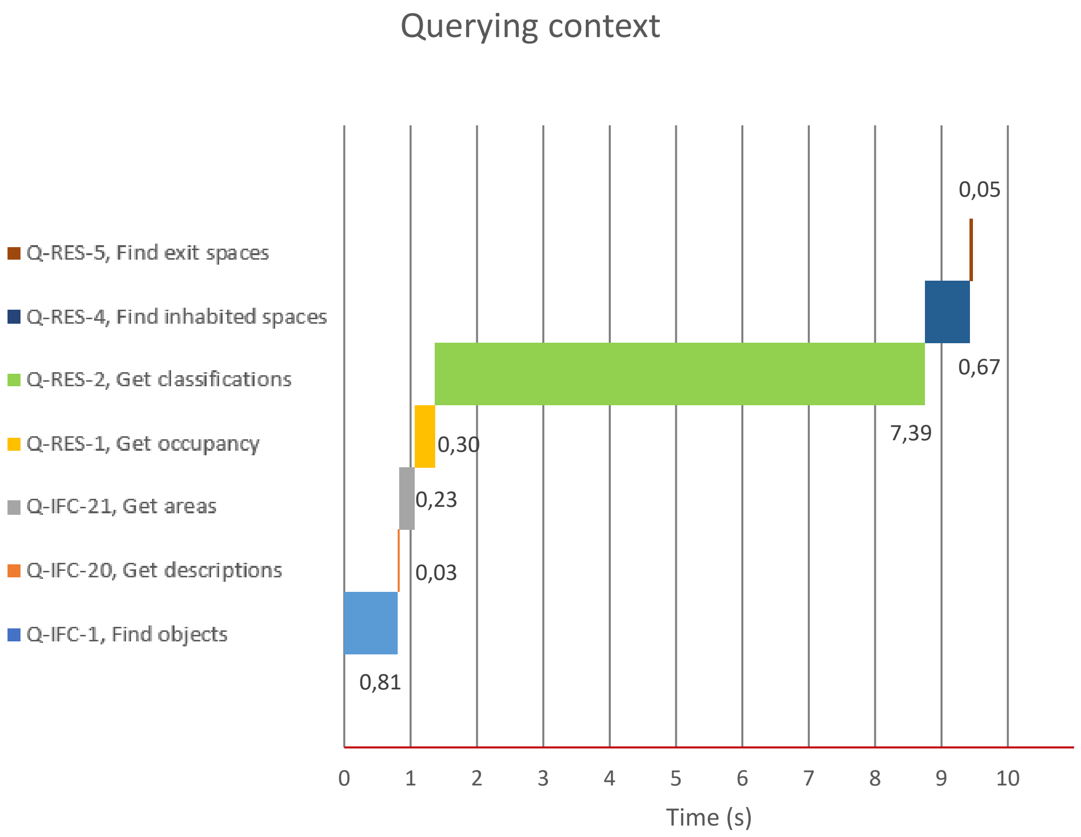


Figure 7-5. Average query times for context retrieval

Table 7-5. Input model size and ONTOCS conversion report

<table>
<tr><th colspan="4">Model size</th></tr>
<tr><th colspan="2">Revit 2018</th><th>IFC</th><th>IfcOwl (RDF)</th></tr>
<tr><td colspan="2">11.6 MB</td><td>1.7 MB</td><td>11.3 MB</td></tr>
<tr><th colspan="4">Model objects converted to MassMotion (and CSS) from IFC</th></tr>
<tr><th>Barriers/Walls</th><th>Barriers/Columns</th><th>Floors/Spaces</th><th>Links/Doors</th></tr>
<tr><td>254/254</td><td>41/41</td><td>84/84</td><td>77/77</td></tr>
</table>

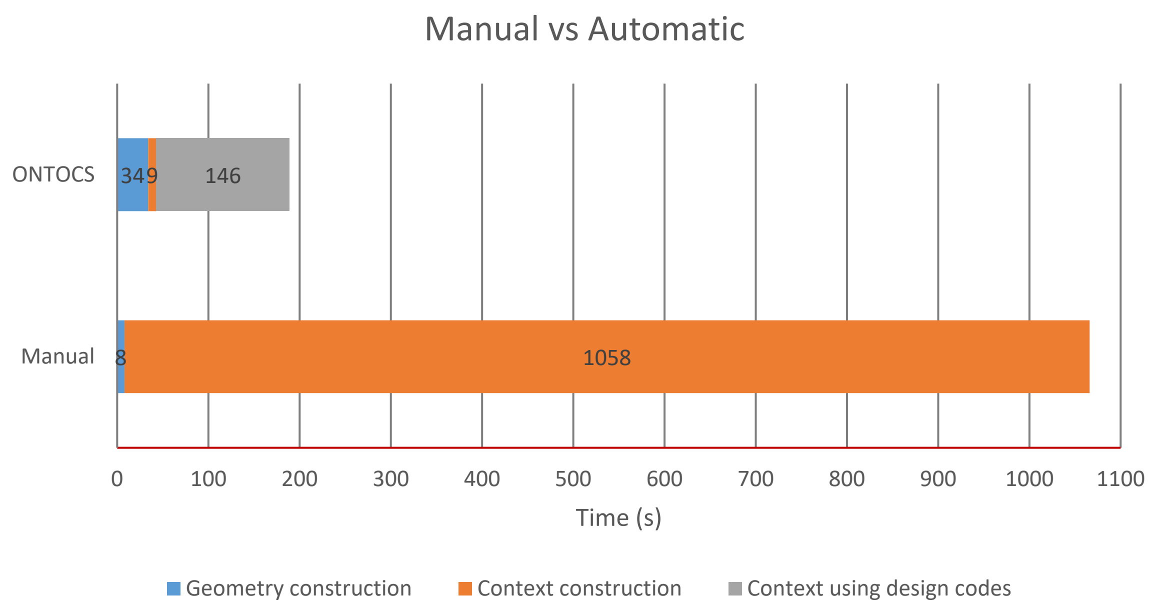


Figure 7-6. Comparison of manual versus automatic model creation time

Results on comparing the manual model construction to automatic model construction by ONTOCS is shown here. The first column in Figure 7-6 shows a scenario being constructed solely from the queries working in IFC model data and properties and additional context is constructed from using the UKSOC and Uniclass. The final column shows the time for manual construction using the data in Table 7-6. The data in the table assumes an expert level of MassMotion user.

Table 7-6. Time for manual construction actions of the model using MassMotion

	Action	Quantity	Time (s)	Total time (s)
1	Import IFC model	1	8	8
2	Convert objects	1	2	2
3	Discard unused objects	1	1	1
5	Correct Links	3	10	30
6	Create Portals	32	15	480
7	Create Journeys	17	25	425
8	Evaluate for errors	1	120	120

Figure 7-7 shows two merged models showing their differences. Additional geometric objects with no IFC equivalents (e.g. Portals) are constructed automatically based on IFC centroids of spaces, which can differ from positioning done manually.

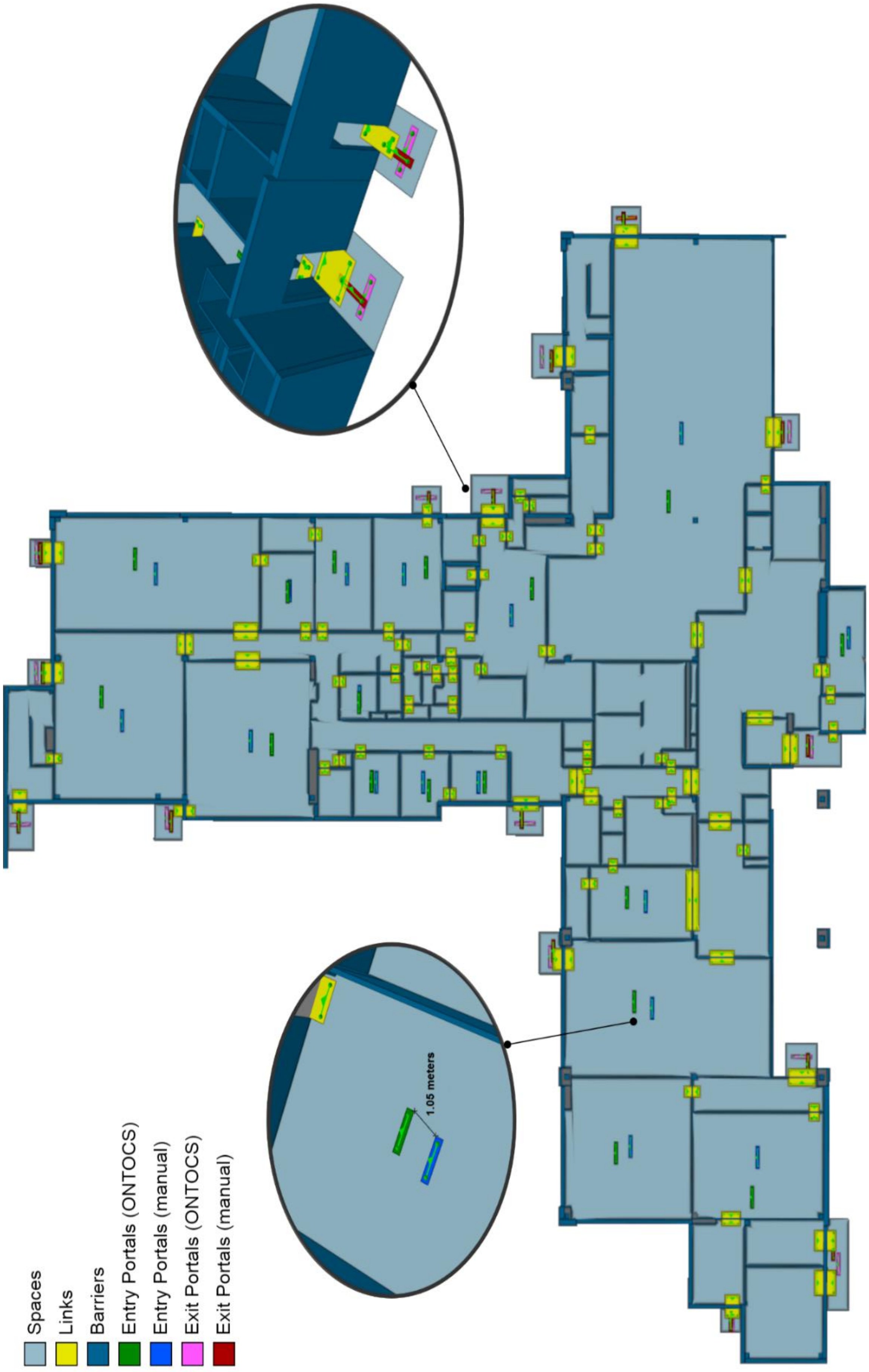


Figure 7-7. Merged manual and ONTOCS automatic models for object positions comparison

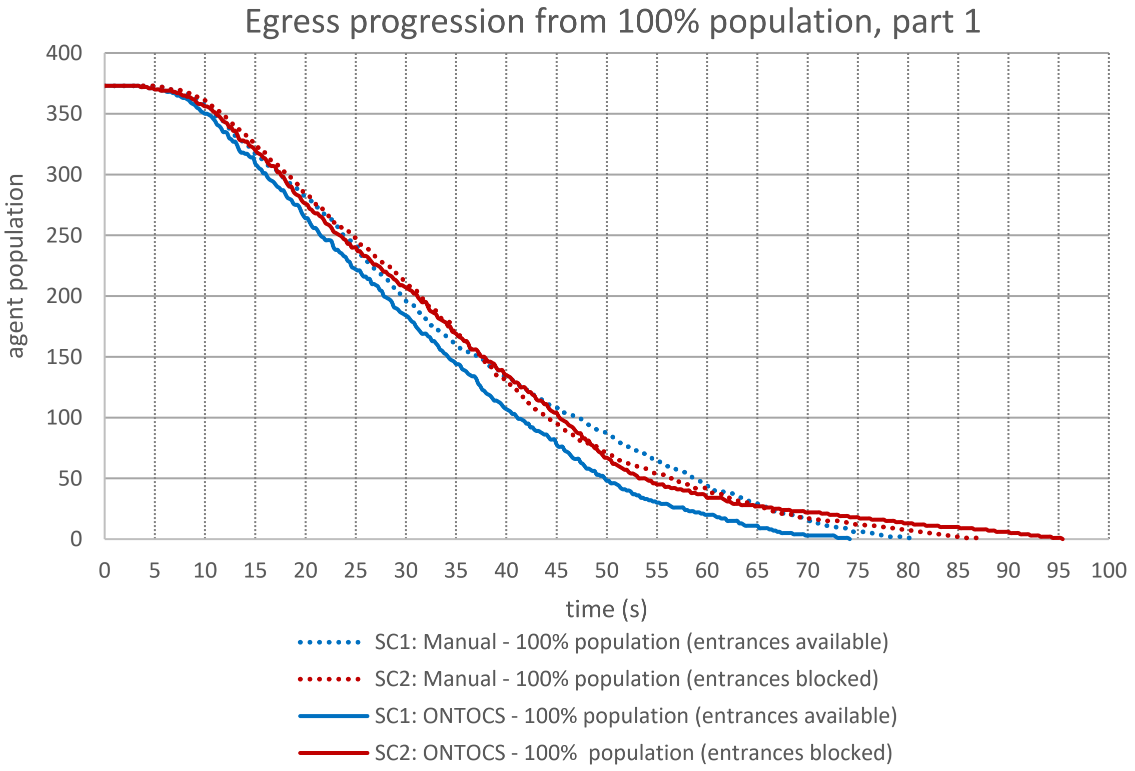


Figure 7-8. Plotted agent numbers versus time for scenarios SC1 (blue) and SC2 (red)

The results for the scenarios running at 100% population capacities is shown here, according to the assumptions stated in Table 7-1.

Figure 7-8 plots manual and ONTOCS generated models for scenarios cases SC1 and SC2, contrasting the two. Blue lines assume scenarios with main building entrances available, while the red assumes entrances blocked.

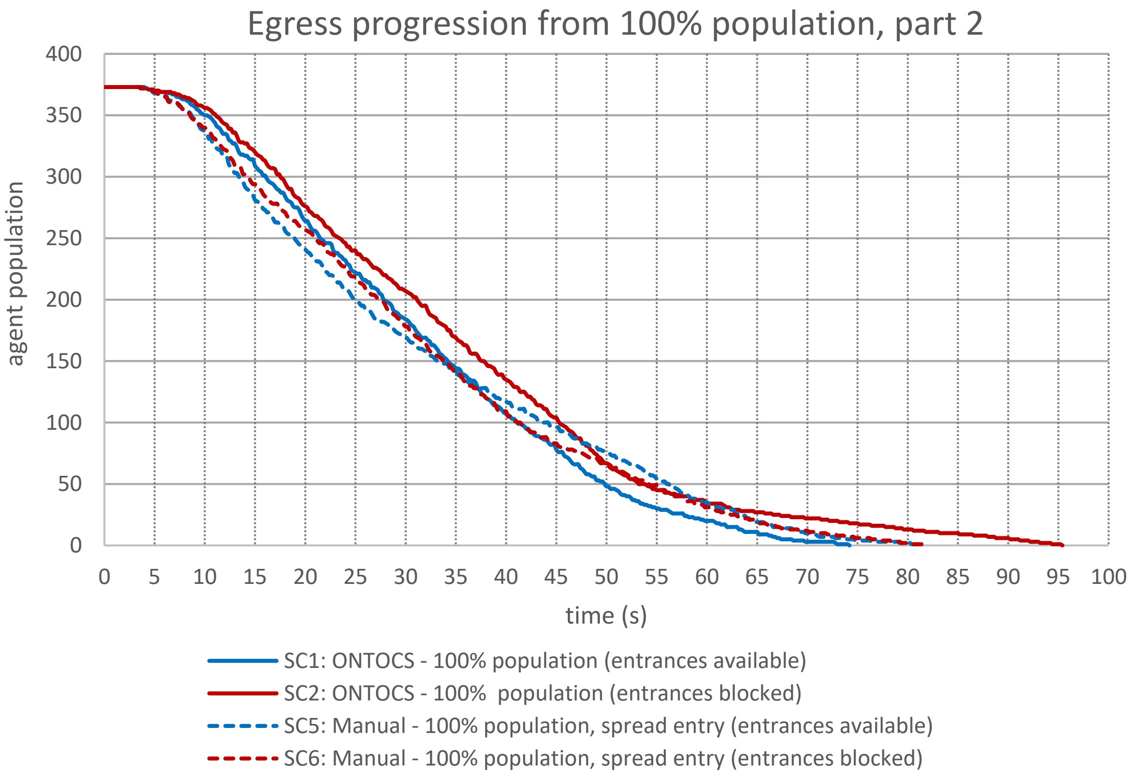


Figure 7-9. Plotted agent population versus time for scenarios 1 and 2 (ONTOCS), 5 and 6 (manual)

Figure 7-9 plots ONTOCS generated models for scenarios SC1 and SC2 and contrasts them with manually constructed models for scenarios SC5 and SC5. The last ones assume a more spread out entry for agents over the larger spaces.

For both Figures 7-8 and 7-9 it can be observed that some lines have several dips down the lines, especially for ONTOCS scenarios. The presence of those points suggest high density around exits at certain points in time. The deeper the curve, the higher the traffic.

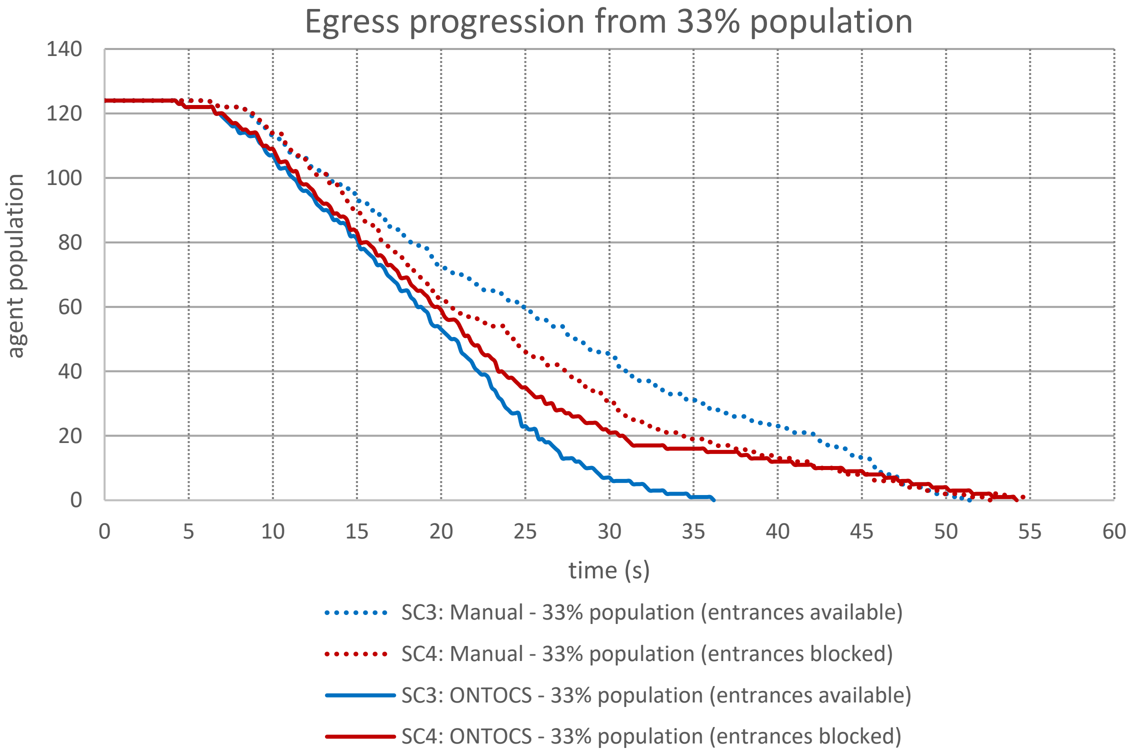
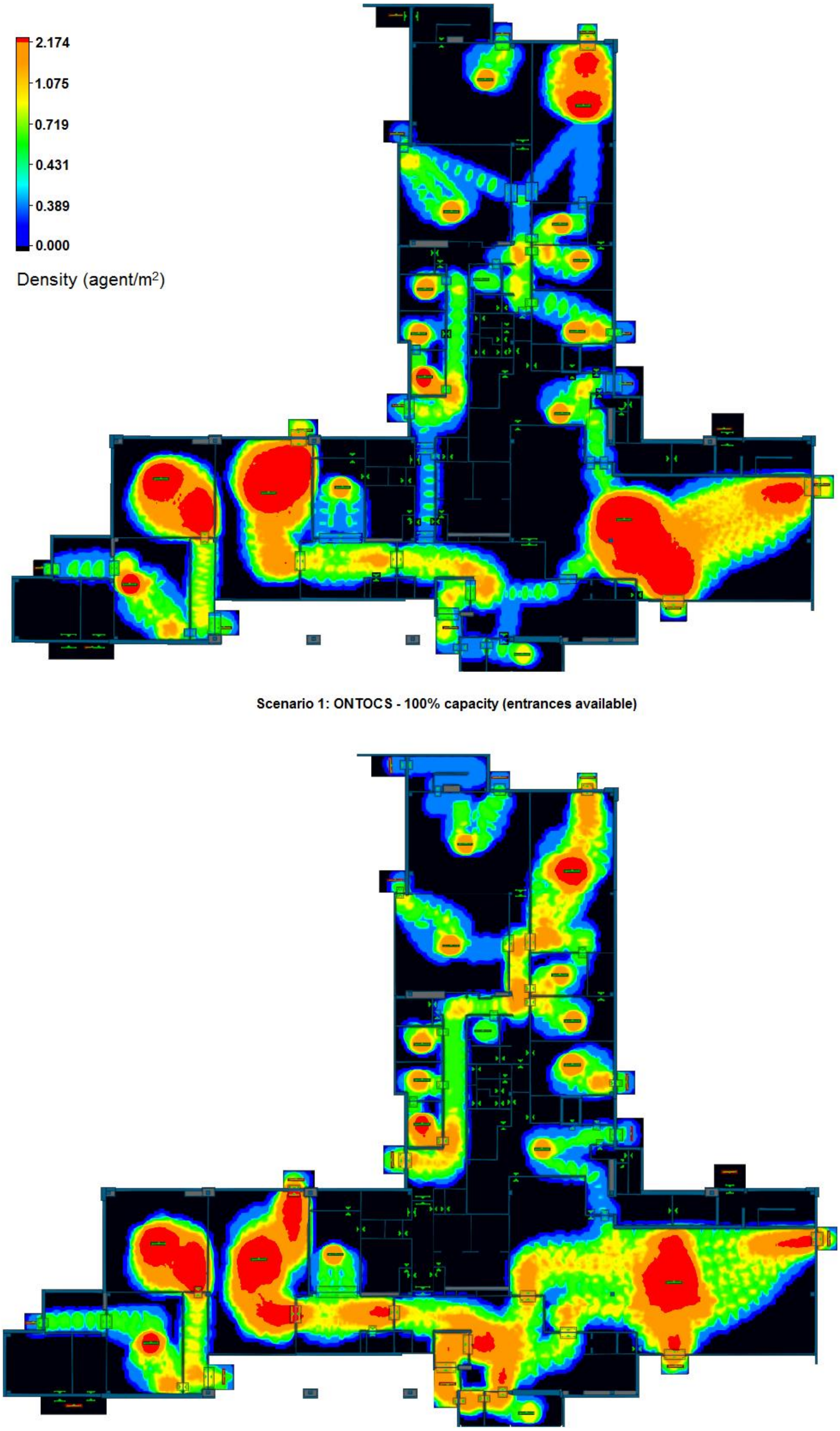


Figure 7-10. Plotted agent population versus time for scenarios SC3 (blue) and SC4 (red)

In a similar way to the previous one, this section shows plotted manual and automatic models, but at a 33% population capacity in Figure 7-10. It can be observed that the differences of the lines are much higher at a lower population case, than in Figures 7-8 and 7-9. This is because a lower population gives agents more freedom to move around the model.

Figure 7-11 on the right shows plotted density maps for the maximum experienced densities during simulations. Only scenario case SC1 was plotted as an example of the differences that can occur in agent movement between a manual model and an automatic model. This reflects the differences in model construction and assumptions.



Scenario 1: ONTOCS - 100% capacity (entrances available)

Scenario 1: Manual - 100% capacity (entrances available)

Figure 7-11. Plotted maximum density experienced during scenario SC1 for ONTOCS and manually constructed models

7.2.4. Stage II – Analysis feedback use case results

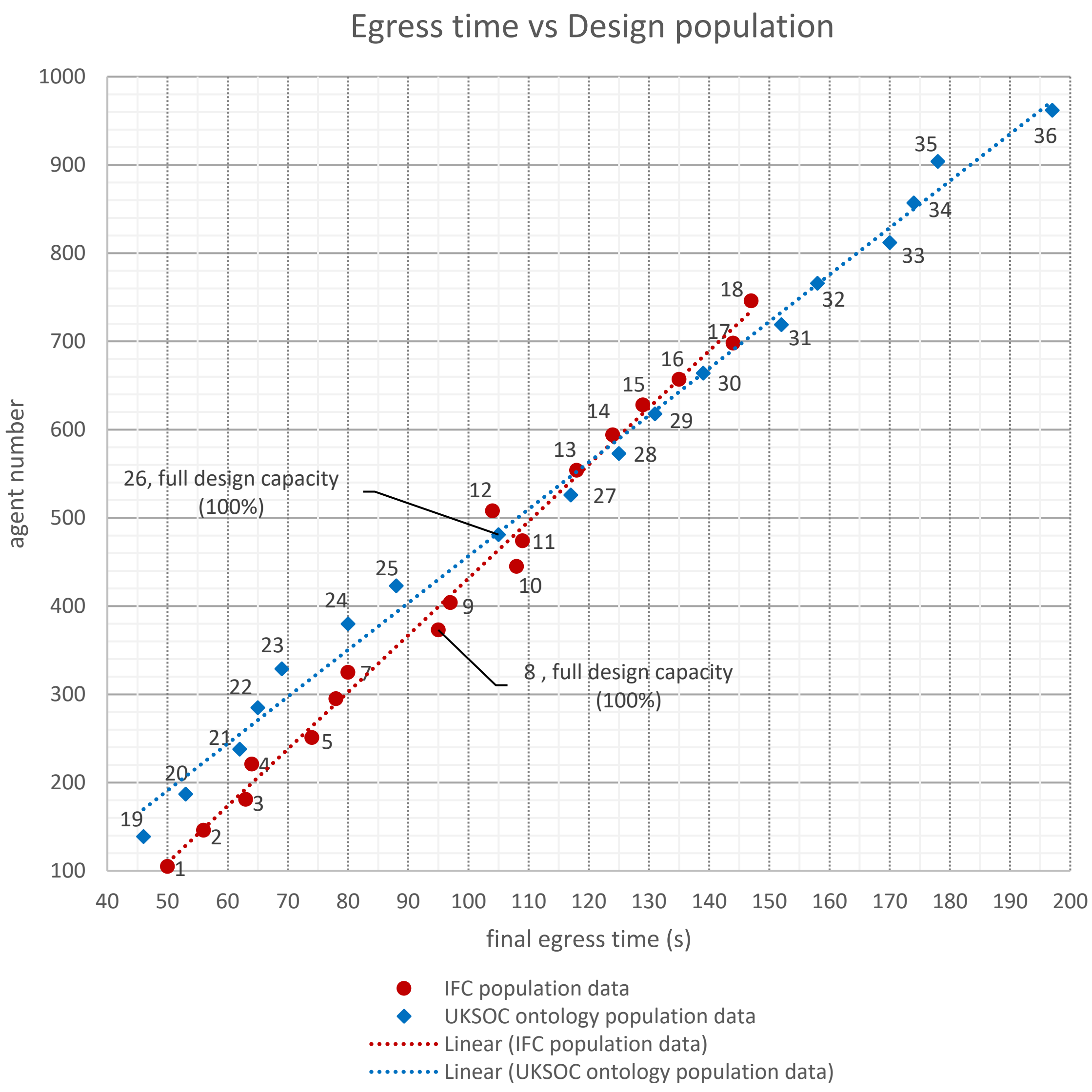


Figure 7-12. Plotted agent number versus final egress times for scenarios 1-36

The figure above shows a plot of final egress time for all the 36 scenarios for the second use case. The trend lines show the expected performance of the building with increasing population. The IFC model data has a steeper trendline. This is because it assumes some spaces to be much more populated than others (see Appendix Table E-1, Appendix E), compared to design codes which allow a more uniform spread of the population density per each space area value.

Table 7-7. Objective sets with ontology reasoning answers per query

Objective set		Valid scenarios				Invalid scenarios		
1		Q-FBA-1	Q-FBA-2	Q-FBA-3	Q-FBA-4	Q-FBA-5	Q-FBA-6	Q-FBA-7
	objective	(a)	(b)	(a) or (b)	(a) and (b)	(a)	(b)	(a) and (b)
	IFC	1-13	1-18	1-18	1-13	14-18		1-14
	UKSOC	19-27	20-30	19-30	20-27	28-36	31-36	28-36
2		Q-FBA-1	Q-FBA-2	Q-FBA-3	Q-FBA-4	Q-FBA-5	Q-FBA-6	Q-FBA-7
	objective	(a)	(b)	(a) or (b)	(a) and (b)	(a)	(b)	(a) and (b)
	IFC	1-13	1-13	1-13	1-13	14-18	14-18	14-18
	UKSOC	19-27	19-26	19-27	19-26	28-36	27-36	27-36

Note: Cells in yellow highlight an error, where scenario 19 is missing from the answers

The table above shows a summary of the answers provided by the FBA ontology and reasoning when objective sets were inputted. These can be checked against values plotted in Figures 7-12, 7-13 and 7-14.

Figures 7-13 and 7-14 show the progression of the evacuation event, as opposed to the final evacuation time vs population from Figure 7-12.

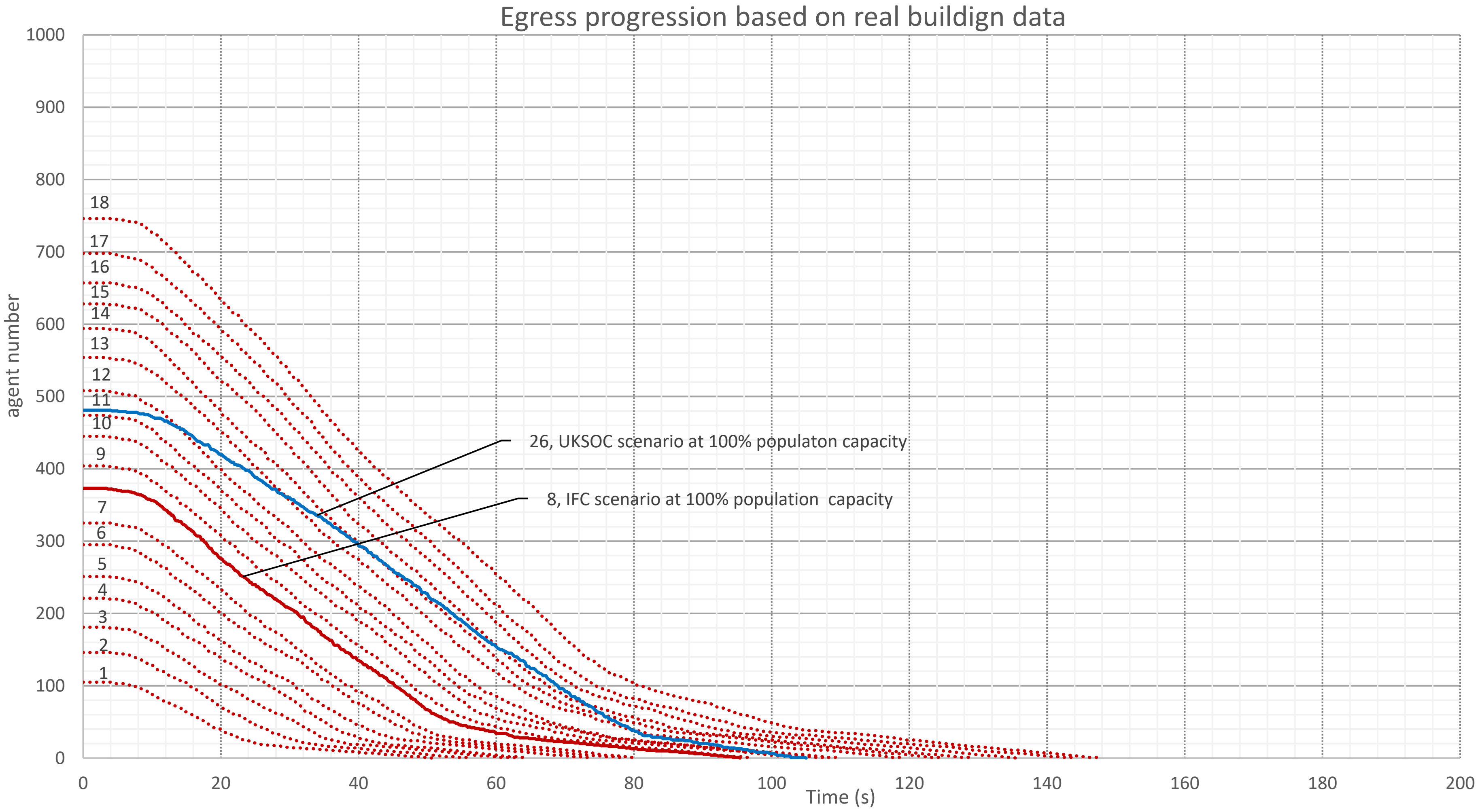


Figure 7-13. Plotted agent numbers versus egress progression in time for scenarios 1 to 18 which use real building data stored in the IFC model. UKSOC scenario 26 (blue) at 100% population capacity was added for comparison

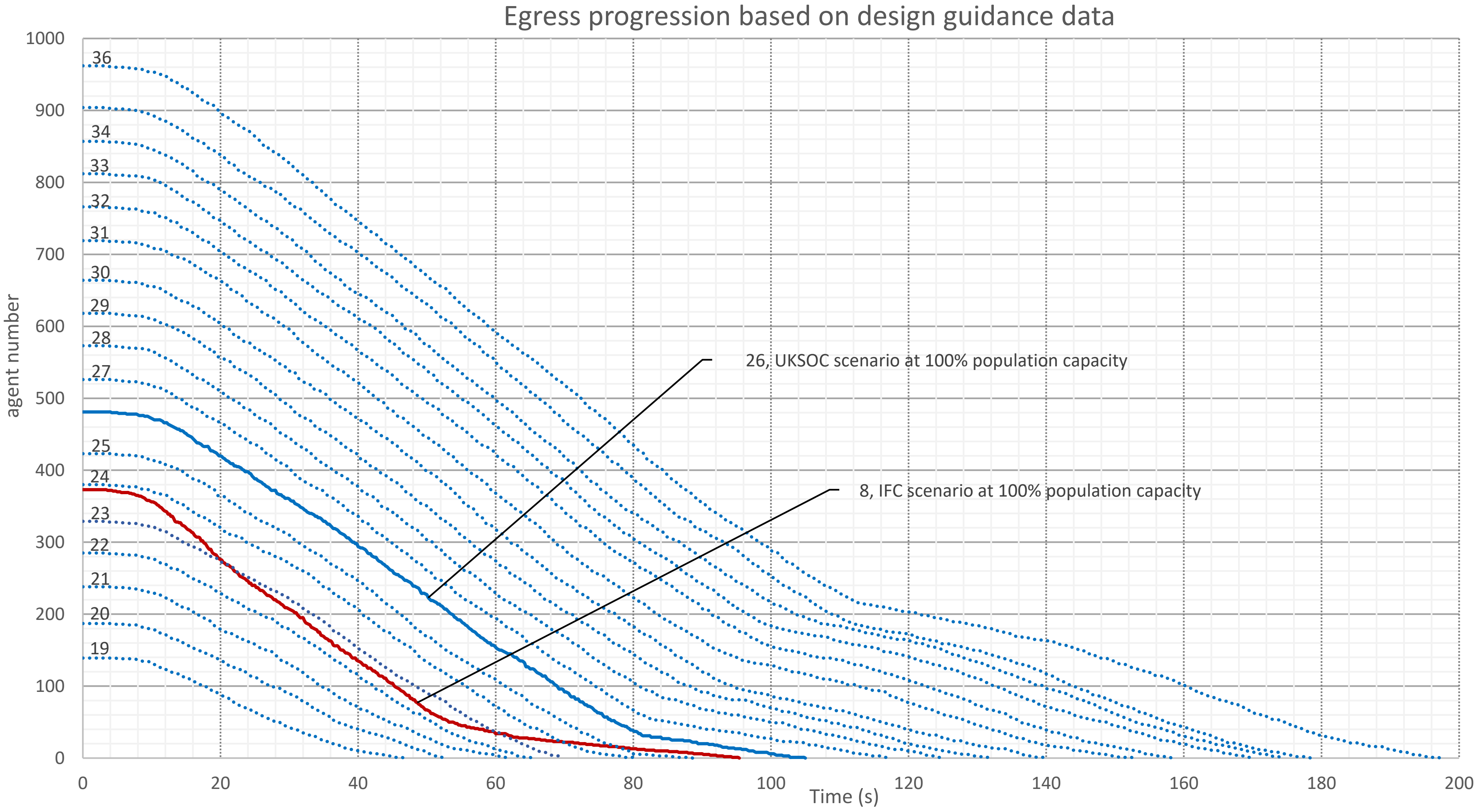


Figure 7-14. Plotted agent numbers versus egress progression in time for scenarios 19 to 36, which use factors reasoned from UKSOC. IFC scenario 8 (red) at 100% population capacity was added for comparison

From this point onwards, the results show query times and assess the efficiency of reasoning, and later the scalability is shown.

Table 7-8 summarises the query times of FBA reasoning queries when retrieving knowledge about design performance over a set of 36 simulation scenarios. An average of 10 measurements was taken for each query in every condition, with red cells being removed from the average calculations as they are considered anomalies.

Table 7-8. SPARQL query times measurements taken during the session with the 36 scenarios running on the ONTOCS system

Query	Dependent rules	Objectives applied	Time measurements (milliseconds)										AVG
			1	2	3	4	5	6	7	8	9	10	
Q-FBA-1	R-FBA-1	single (a)	1659	1230	1232	1275	1301	1307	1304	1286	1391	1349	**1297**
		multiple (a & b)	2211	3111	3581	3387	3613	3653	3533	3594	3578	3572	**3514**
Q-FBA-2	R-FBA-2	single (b)	709	760	722	866	968	865	699	678	683	833	**778**
		multiple (a & b)	1247	987	985	947	940	953	967	979	987	972	**969**
Q-FBA-3	R-FBA-1 R-FBA-2	single (a)	887	939	858	951	953	925	1063	939	1032	966	**951**
		single (b)	302	311	339	345	363	356	298	305	301	332	**325**
		multiple (a & b)	3741	3251	3306	3784	3655	3755	3673	3758	3720	3766	**3641**
Q-FBA-4	R-FBA-3	single (a)	11	7	8	10	7	8	8	6	9	12	**9**
		single (b)	7	7	7	21	107	10	5	6	6	7	**8**
		multiple (a & b)	33524	33320	33370	33722	33235	33259	33243	33296	33281	33191	**33344**
Q-FBA-5	R-FBA-4 R-FBA-5	single (a)	157	110	114	127	135	139	111	142	97	98	**123**
		multiple (a & b)	202	174	172	161	185	166	161	180	185	178	**176**
Q-FBA-6	R-FBA-6	single (b)	327	316	300	464	344	350	311	282	321	382	**340**
		multiple (a & b)	841	595	580	592	588	593	600	613	580	596	**593**
Q-FBA-7	R-FBA-4 R-FBA-5 R-FBA-6	single (a)	121	134	111	111	104	120	104	116	97	95	**111**
		single (b)	300	308	286	327	359	354	315	292	296	414	**325**
		multiple (a & b)	1073	719	850	723	746	742	738	738	751	736	**749**

Note: Cells in red were removed from average calculations. Each query was executed 10 times for each objective condition from Section 7.2.2

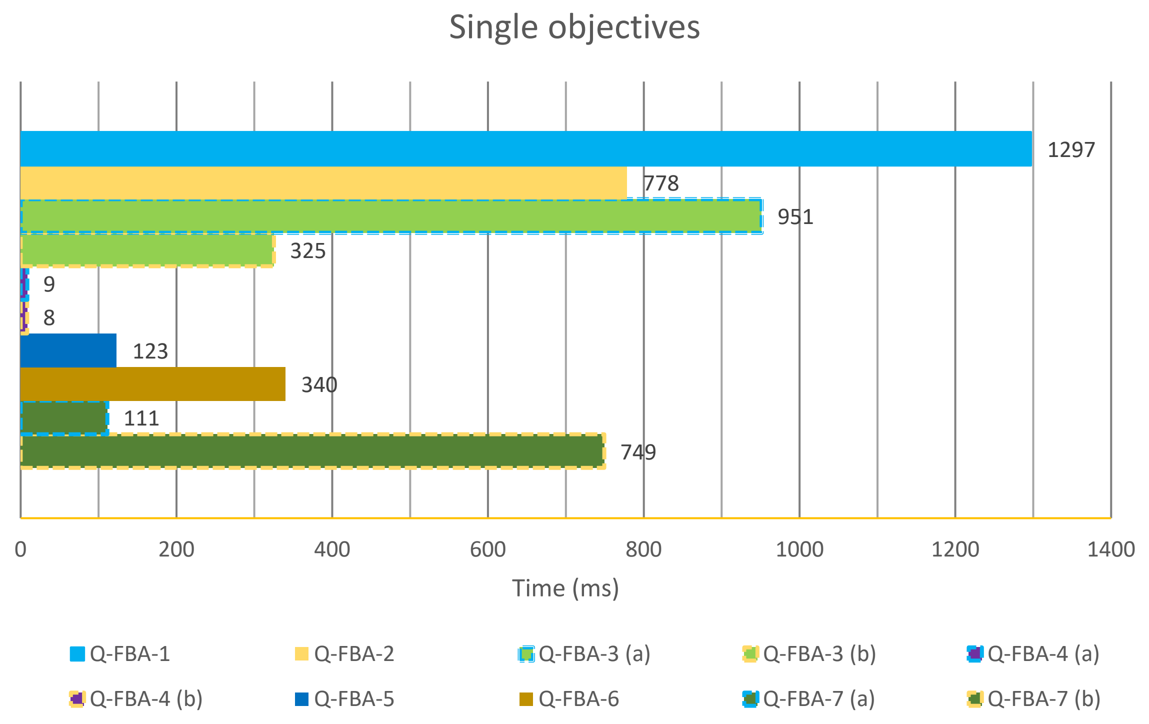


Figure 7-15. Plotted query reasoning times for single objective input

Taking data from Table 7-8, the figure above shows the results related to reasoning times when only one objective is inputted at a time form the use end. It can be seen that for each case, results are very different.

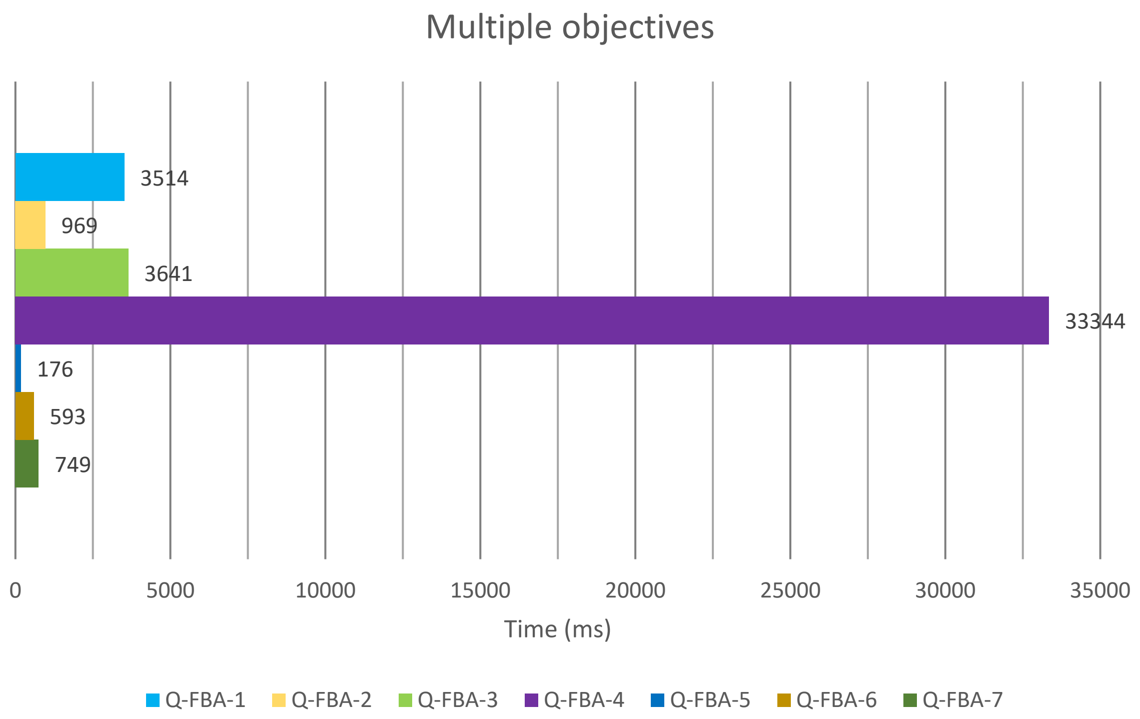


Figure 7-16. Plotted query reasoning times for multi-objective input

Taking data from Table 7-8, the figure above shows the results related to reasoning times when multiple objectives being inputted at the same time. This is in contrast with Figure 7-15.

The next charts show scalability results.

A list of full plot data is available in Table E-3, Appendix E.

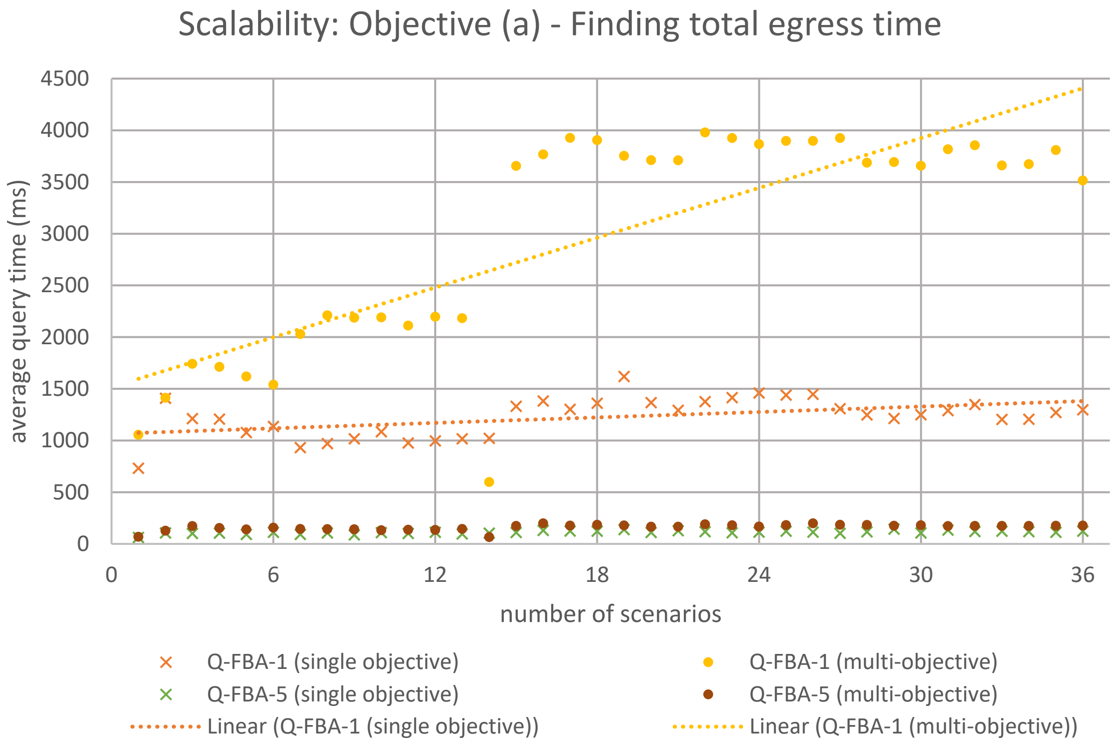


Figure 7-17. Scalability for objective (a) – Total egress time

The figure above shows the trend lines for queries in single and multiple objective cases for when rules which process objective (a) are applied.

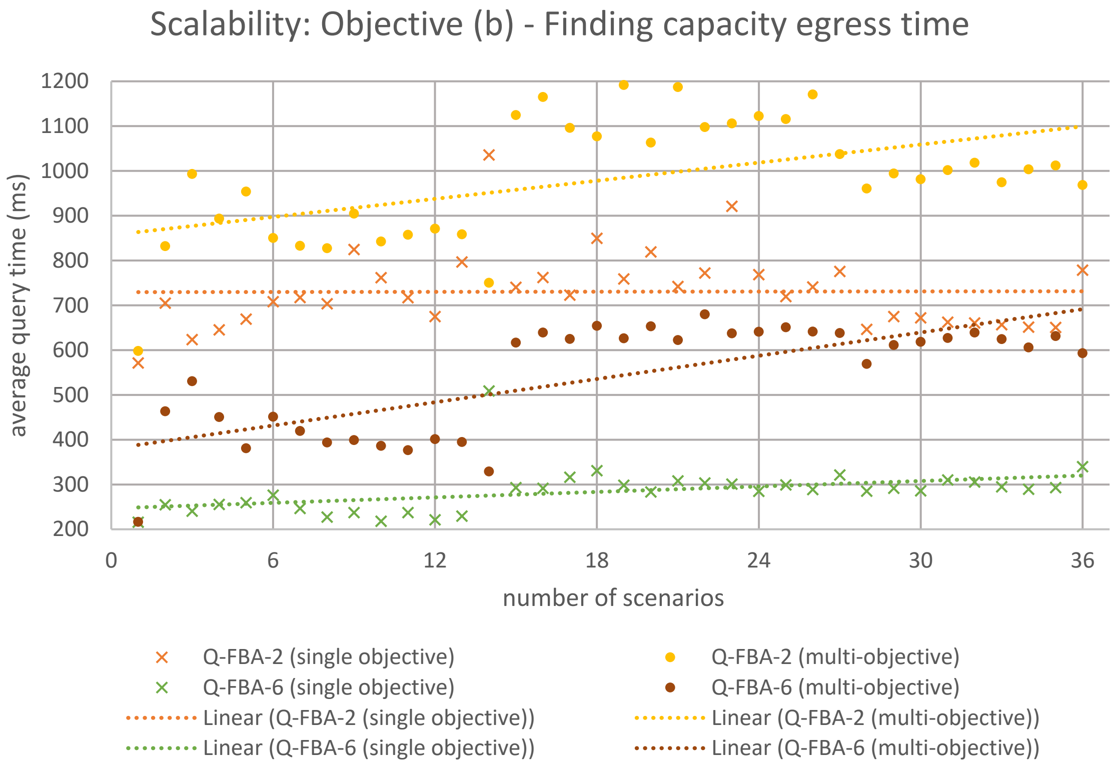


Figure 7-18. Scalability for objective (b) – Capacity egress time

The figure above shows the trend lines for queries in single and multiple objective cases for when rules which process objective (b) are applied.

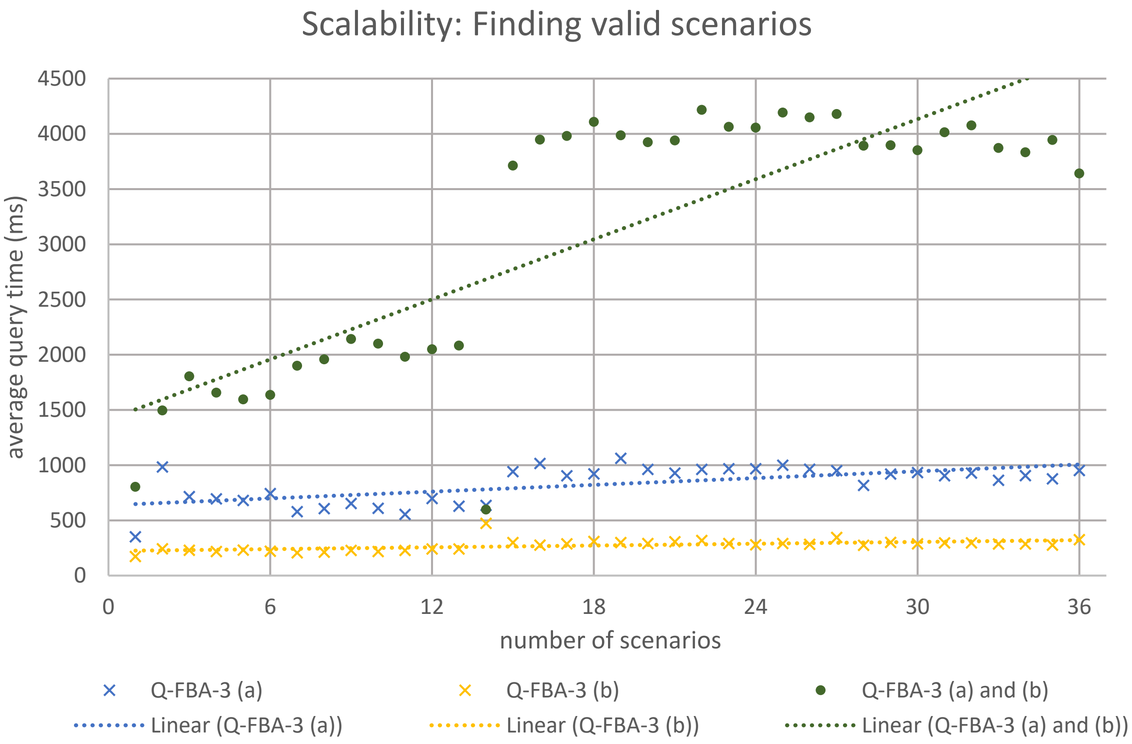


Figure 7-19. Scalability for finding valid scenarios

The figure above shows the scalability of query Q-FBA-3 in single and multi-objective cases when looking to find ***ValidScenario*** class individuals. The query performs a union of valid scenarios from both objectives (a) and (b). It therefore implicitly uses the rules which govern both objectives (R-FBA-1 and R-FBA-2)

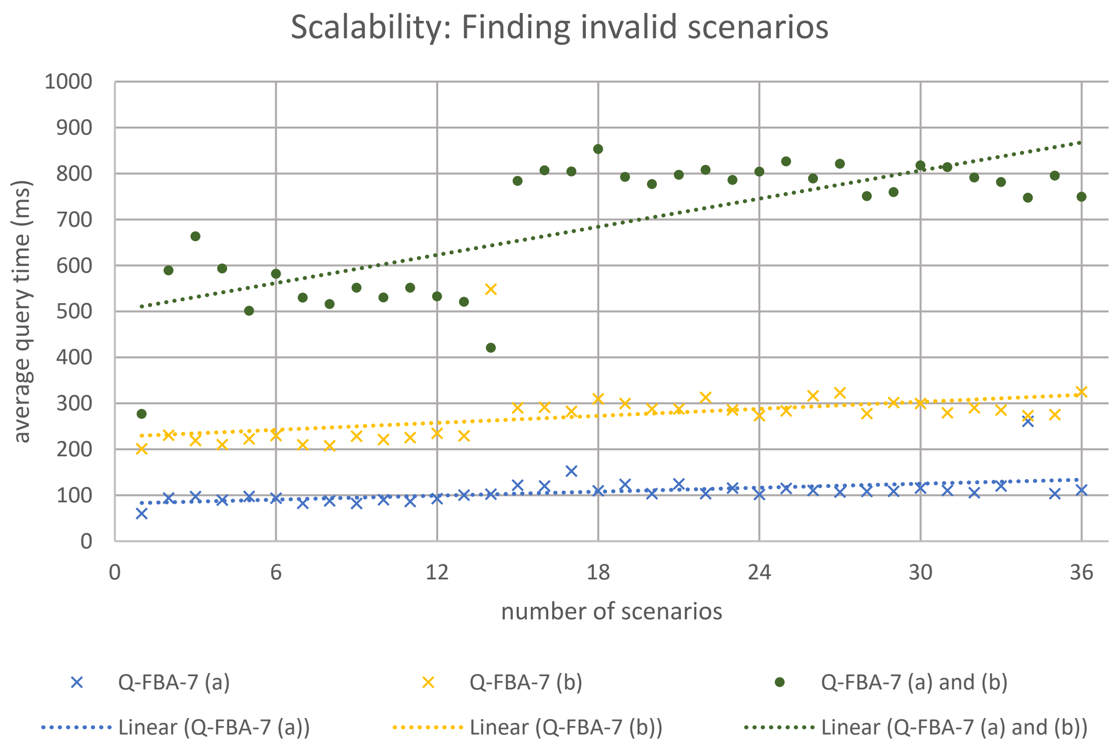


Figure 7-20. Scalability for finding invalid scenarios

The figure above shows the scalability of query Q-FBA-7 in single and multi-objective cases when looking to find ***InvalidScenario*** class individuals. The query performs a union of invalid scenarios from both objectives (a) and (b). It therefore implicitly uses the rules which govern both objectives (R-FBA-4, R-FBA-5 and R-FBA-6)

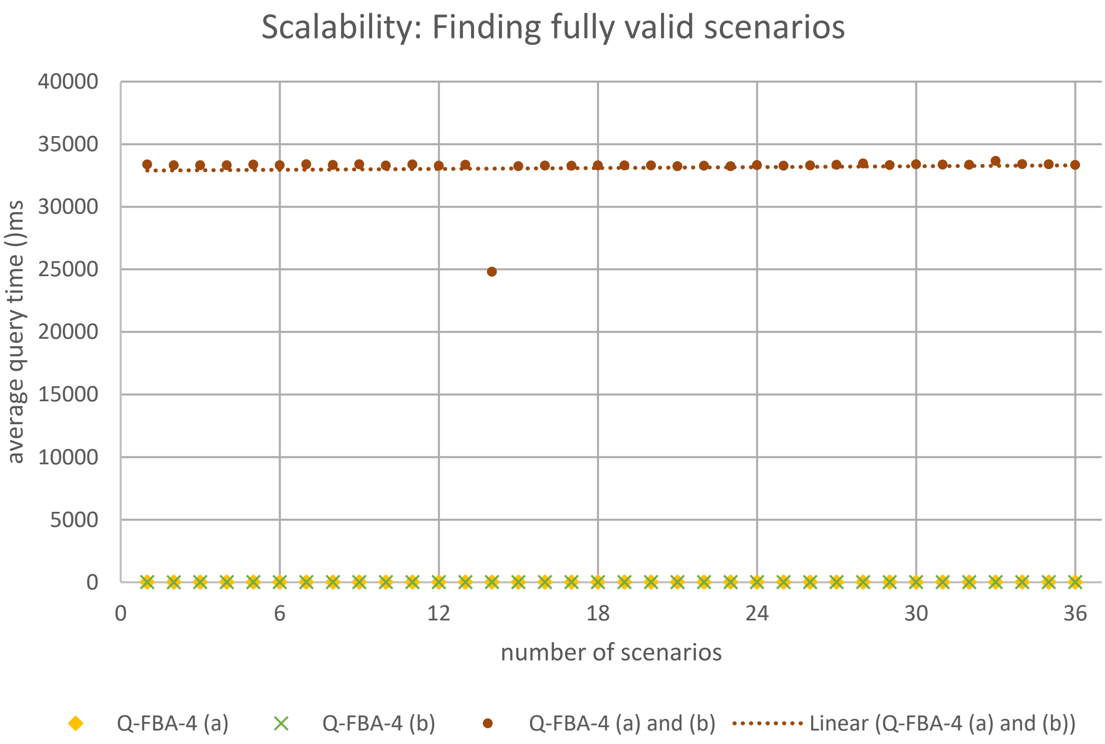


Figure 7-21. Scalability for finding fully valid scenarios

The figure above shows the scalability of query Q-FBA-4 in single and multi-objective cases when looking to find ***FullyValidScenario*** class individuals. The query performs an intersection of valid scenarios from both objectives (a) and (b).

It explicitly uses rule R-FBA-3, which is implicitly calling both rules for (a) – R-FBA-1 and (b) R-FBA-2. The intersection of nested rules results in a very high difference of query times. The lower values are near zero by comparison, because they do not have access to full data for reasoning, as R-FBA-3 needs to depend on both its implicit rules at the same time to actually function.

7.2.5. Discussion

Considering the assumptions presented in Section 7.2.2, the results for each use case are discussed question-by-question to outline the findings and limitations from the case study.

1) ***How fast is a scenario model generated, in terms of geometry and context?***

The efficiency for geometry and context information retrieval using the SPARQL queries is outlined in Figures 7-4 and 7-5 respectively. In terms of the geometry, querying from the IfcOwl instance model was on average around 34 seconds. The structure of the IfcOwl ontology needs long queries to retrieve basic measurement data types. Coupled with nature of the SELECT SPARQL queries, which loop through the results matching triple structures, the process becomes inefficient. Query time increases as the triple pattern grows. This is evident in Figure 7-4, where the longest time is for the BREP mapped shape query, which selects several loops of points for each complex geometric object. This would cause models with complex geometry such as steel columns and furniture to take a very long time to query using the current version of the IfcOwl. More testing would be required to assess the scalability of the developed queries on larger building models. In terms of context, the retrieval of explicit IFC object properties is significantly faster than geometry components, averaging around 9 seconds (Figure 7-5). It is worth noting that only a few properties are retrieved, but due to the shorter triple chains, they perform significantly faster compared to geometry retrieval queries.

2) ***How fast is the context created when relying on design guides resources?***

The context queries on explicit IFC model properties require no reasoning, making them relatively fast. However, the query labelled ***Q-RES-2, Get classifications*** in Figure 7-5 is by far the longest to process. This query triggers the ontology alignment to match the classification code attached to each space to its correspondent in the Uniclass2015 dataset graph. The reason for this delay is because the Uniclass2015 has thousands of classification codes on various levels. Once these are retrieved, the design occupancy factors for each space are reasoned. Figure 7-6 shows the query time required to retrieve the occupancy factors to average at 146 seconds. This delay is caused because the reasoning process has to evaluate the 56 SWRL rules which were used to correctly align the UKSOC and Uniclass2015 ontologies. Thus, retrieving implicit context information can take significantly longer than retrieving explicit data properties. However, as suggested by Figure 7-6, the time required to retrieve the context in this case is 7 times shorter than when constructing the model context manually. The figures show the average times experienced during testing (see Table 7-6), with the building environment being imported from the IFC model. Creating the building entirely within the software without using a digital BIM was not considered efficient, as it would increase the time significantly, along with the risk of representing the environment inaccurately in terms of element sizes. The automatic process is in every case significantly more efficient. Additionally, the process of knowledge mining building context needs to be done only once, which is then used to create multiple scenarios. It is expected that the more secondary resources are in use, the time to retrieve the context would increase and it would be dependent on the optimisation of the ontology mappings in place.

3) ***What are the differences between a manually created simulations and those created by the ONTOCS system?***

Although automatic processes will always be for faster than manual model construction, the automatic output needs to be correct. Figure 7-7 shows a merge between a manual and an automatic model. The construction of the geometry is nearly identical, as both processes imported and constructed the IFC model geometry correctly. However, the additional geometric objects which are constructed for the scenario context are positioned differently. Firstly, the placement of Portal objects within rooms is different. ONTOCS places each portal on top of the Space centroid as extracted from IFC, with each space having a corresponding portal. Spaces with a complex polygon shapes have shifted centroids, thus the difference between Portal positioning. For very large areas such as the ***Common Room*** or ***Restaurant*** in the building, a single portal is not representative enough of reality as it places all agents clustered together when simulations start. A more realistic scenario would have to distribute the agents across the entire area. However, it is difficult to correctly construct additional portals across an area, as it can easily end up outside it, or be obstructed by a barrier wall or furniture. The alternative would be to use a CST which is able to distribute agents across spaces without the use of Portal objects.

4) ***How reliable are the simulation results originating from the automatic process for future analysis consideration?***

Figures 7-8, 7-9 and 7-10 show plotted results of both manual and automatic scenario results. The biggest difference is of 16 seconds evacuation time between the ONTOCS and manual models at 33% with entrances available (Figure 7-10). In addition to this, the two evacuation curves have very different trends in comparison to other pairs of scenarios. By contrast, the closest results in terms of time between a manual and an automatic simulation are those at 33% capacity with blocked entrances, at a difference of 2 seconds. This suggest that regardless of the assumed exit routes or the difference in portal placement, scenarios with fewer available exits and fewer flow restrictions on doors will be the most similar. The same is true when considering the 100% capacities scenarios, where the red lines in Figure 7-8 and Figure 7-9 follow a more similar trend than the blue lines, although the final egress times differ by approximately 10 seconds in both figures. Manually constructed scenarios 5 and 6 modelled a more distributed agent entry yet resulted in very similar trendline and results to those assuming a single-entry portal. Curiously, both scenarios 5 and 6 have achieved very similar egress times, despite one having entrances blocked. This is explained by the fact that a more distributed agent placement has less of an impact on forming queues at doors, thus resulting in a faster evacuation.

Finally, Figure 7-11 outlines the maximum density measured for two different scenarios (one manual, one automatic) and it can be observed that the agents in the automatic model are using the closest exits, resulting in some areas experiencing higher traffic densities. This simulation evacuated the agents 6 seconds sooner than the manual one, aiming to maximise the evacuation regardless of the density at the exits, which might be prone to other safety risks in real life cases.

5) ***Is the system correctly interpreting the results according to user inputted design objectives?***

Table 7-7 shows the answers provided by each reasoning query which were checked against the plotted results in Figures 7-12,13 and 14. Most of the answers are correct, with the exception of those involving queries Q-FBA-2 and Q-FBA-6, which check against objective (b). Scenario 19 seems to be missing from the table of results, both for the valid check rule (R-FBA-2 used by Q-FBA-2) and the invalid scenario check rule (R-FBA-6 used by Q-FBA-6). An investigation into the results revealed that the data required for evaluating scenario 19 was missing from the ontology resources graph. This is because the algorithm fetching the SQL data looks for a specific percentage of population within a simulation and retrieves the simulation time. Due to its low population, scenario 19 does not record any data for percentages between 49-51%, as a larger group of agents are leaving the simulation within a very short time, skipping the 50th percent. This is not a limitation of the ontology reasoning, but rather one concerned with validation of the data available from simulations which is provided to ontology reasoning.

6) ***How reliable are design occupancy factors in comparison to real building occupancy data?***

As a secondary objective to evaluating the reasoning processes, it was established early on that other resources could be used to contribute to the context of the simulation. The aligned UKSOC and Uniclass2015 ontologies provide population data correctly, similar to the one present explicitly from the IFC model. However, in Figure 7-12 it can be observed that the populations assumed from UKSOC was over-estimated by a range of 28% to 32% when compared to the corresponding scenarios using real data. This is because a factor is multiplied by the area of a space and the number is rounded down to a whole. While the initial evacuation times are similar at first, as the population assumed increases the results start to vary. The final evacuation time of the UKSOC population was 40% higher than the one from the IFC data. Overall, the results are inconclusive, and more investigation would be required based on other building layouts.

7) ***How efficient is the reasoning process for evaluating user objectives on a large scale?***

Figures 7-15 and 7-16 plot the average query times for retrieving the validity of the scenarios, with full measurements in Table 7-8. As expected, single objective query times are shorter than the multi-objective case. In the latter, reasoning times increased by varying factors from 1.24 to 4000 times longer. Due to their dependency on multiple rules, they are discussed separately here:

- From queries operating on objectives (a) and (b), Q-FBA-2 performed the best and increased by 24%, whilst Q-FBA-1 is the least performant and increased by 270%. Although both rely on similar rules (13+1 atoms), Q-FBA-2 performs significantly better;
- From queries operating on multiple objectives implicitly, Q-FBA-3 performs least well when working on rules for objective (a), but significantly faster when working on rules from objective (b); this level of disparity does not exist for Q-FBA-7 which looks for invalid scenarios;
- Overall it can be observed that queries which look for TRUE answers, trying to identify valid scenarios for objectives, perform worse than their respective FALSE check queries;
- Finally, Q-FBA-4 increases in query time by a factor of 4000; this is explained by the fact that it relies on several rules, which are intersected implicitly by the R-FBA-3 rule to check for fully valid scenarios (scenarios TRUE for (a) and (b) at the same time). For single objective tests (Figure7-15), the query times are extremely low, suggesting that the query is not applied for reasoning. This is because the query graph domains (defined by the PREFIX keyword) restrict the rule (R-FBA-3) from information to evaluate the other two implicit rules it calls.

The results above suggest a need to optimise the rules for better processing time by improving query or ontology structure, and also a need to investigate how they perform with an increasing number of scenarios.

8) How do query times scale with increasing number of simulations?

Considering the high increase in processing time from the previous question, it was investigated whether the query times increase linearly with the number of scenarios. Figures 7-17 to 7-21 show the relevant results for this question. Although several measurements were taken for each average time, the overall performance seems to oscillate between sessions. The session with 14 scenarios recorded anomaly measurements across several queries. Several observations were made on the provided data plots:

- Concerning queries which satisfy objective (a) – Figure 7-17, there is some degree of linearity increase for the multi-objective case. This however is followed by a sharp increase starting with the 15 scenarios session, after which it stabilises. This appears to be connected to the low performance of query Q-FBA-1, as discussed previously. However, when the query is applied in a single objective context it has a very low gradient. Queries which check for invalid scenarios appear to remain constant.
- Concerning queries which satisfy objective (b) - Figure 7-18, there is a low linear gradient for the multi-objective case; this however is negated by the fact that they begin to stabilise for the last scenarios, suggesting they tend towards a constant. For the single objective case they remain constant throughout all measurements, despite their oscillations.
- Concerning queries which rely on multiple rules for both valid (Figure 7-19) and invalid scenarios (Figure 7-20) they show a very low gradient for the single-objective cases, and appear to show a steep gradient for the multi-objective case. However, as with the cases before, they stabilise towards the end points to almost no significant increase; these show very similar trends to those encountered at Q-FBA-1.
- Concerning query Q-FBA-4, which intersects multiple rules (Figure 7-21), the trends are constant. Although it shows a dramatic increase in query time from single to multiple objectives, it experiences no increase in terms of the number of scenarios inputted.

It can be concluded that the developed knowledge operators under the current system are able to scale well with increasing number of scenarios, showing no clear sign of a steep increase in query time. It has been observed that the query times oscillate frequently, and that there are certain steps of more significant increase after a certain number of scenarios, the most evident one starting at 15. To be able to establish the limits of this methodology for applying SPARQL and SWRL rules, a significantly higher number of scenarios would need to be tested. However, it is debatable if this would be a requirement in practice.

7.3. Summary

This chapter began with introducing the developed ONTOCS system, showing its process workflow and underlying architecture. The system was tested on a case study of a real building. Two use cases were defined for testing the developed ontologies and knowledge operators, each case corresponding to different process stages. The results were then presented and later discussed in an attempt to answer the 8 objectives posed for the case study. Overall, the system functions correctly and the knowledge operators are able to construct and interpret model data correctly, with some limitations, which were outlined and discussed.

Chapter 8. Discussion and future work

Reflecting on the observations and findings from previous sections, this chapter provides a summary discussion of the research by revisiting the hypothesis and research questions. The limitations of the work are outlined, followed by planned future work to extend and improve the overall system and the developed ontologies.

8.1. Revisiting the hypothesis

The research hypothesis tested for this research was as follows:

A knowledge processing-based approach can allow a fast retrieval of information and automatic construction of evacuation models by leveraging existing BIM data and design knowledge to enhance the decision-making processes about building performance by considering different simulation scenarios on a large scale.

The hypothesis was then decomposed into 7 research questions, which are discussed below, based on findings from the previous chapters. Although it was initially envisaged that each chapter would focus on certain research questions, the findings from all chapters combined are used to evoke more comprehensive answers.

Q1) How are evacuation models and tools used for assessing design performance while considering their scope and limitations?

Evacuation models have been developed with the purpose of mimicking reality as close as possible in order to enable the prediction of human behaviour during fire evacuation events. Many models have been developed based on several methodologies, each with its own scope and limitations which need to be considered by safety engineers. CSTs are widely used to create very specific evacuation scenarios which are assessed by designers in attempts to prove a safe building layout for a building population. The scenario creation process is complex and inefficient requiring significant user input and configuration, as was concluded from the literature, but it also became evident in assessing the features of CSTs in Chapter 4. Additionally, one simulation scenario is insufficient to provide designers with enough insight into building performance, so dealing with a large scale of scenarios is required.

Q2) What is the current level of interoperability between CS for evacuation and BIM?

BIM is used across many disciplines to facilitate interoperability and collaboration between design disciplines. The CS field is at a relatively low level of integration compared to energy or cost analysis. There have been many attempts at CSTs and BIM tools integration, but these are limited to geometry alignment, and do not account for the complexity of information required to facilitate a complete and automatic way of simulating realistic evacuation scenarios. Most CSTs in retail have many import capabilities, including IFC, making them BIM compatible. However, this is still limited to geometry, and many tools lose all semantics attached to IFC objects on imports, as was outlined in Chapter 4.

Q3) What are the benefits of using ontologies for evacuation design, considering the BIM paradigm?

The IFC format has been used as a tool for providing information exchange, but it still presents many challenges. BIM is moving in the direction of knowledge processing, with the development of IfcOwl, thus being able to leverage web linked data as a tool to extend interoperability to other knowledge domains, which were not previously considered. Ontologies excel at integrating data and resources from different knowledge domains and design perspectives. Additionally, ontology reasoning capabilities offer new creative ways to interpret data, information and knowledge and allow a more realistic representation of human behaviour and design knowledge than conventional tools. However, the practical application of ontology-based systems requires extensive knowledge of the domains involved and their correct definition, often being an expensive process to carry out.

In addition to that, it was shown in Chapters 5 and 6 that an ontology representation of models allows a retrieval of contextual information for CS scenarios construction due to a semantic rich environment.

Considering the research questions above (Q1, Q2 and Q3), a novel way was sought in which an intelligent system would be able to account for the limitations mentioned above, and to benefit from the use of semantic linked data. This methodology was proven in the rest of the chapters, each viewing the problem from different lenses.

Q4) ***What are the requirements for an intelligent system capable of integrating resources relevant to the CS field within the context of automation and analysis feedback, whilst considering practical deployment and future extensibility?***

The primary requirement consists in an intelligent system being able to interface with more than one CST. Section 4.1 presented a detailed analysis of several CSTs with particular focus on their functionality and features from a software design perspective. The common features used by these tools were identified and can be categorised into 4 major functional categories: geometry, agent, event and analysis concepts. A more in-depth survey of the tools was conducted in order to identify a baseline of common concepts, which was used to define a generalised taxonomy for CSTs. This was required to allow the inclusion of multiple tools and models to account for the gaps identified when Q1 was posed. The basic underlying concepts and principles from this taxonomy were then used to define a fully functioning crowd simulation model in an ontology representation which enabled the generalisation and therefore the interfacing of CSTs with a knowledge base, as was demonstrated in Chapters 5 and 6.

The second requirement was to identify the information requirements that enable automation. Section 4.2 investigated ways in which a crowd simulation model can be made more automatic, by identifying what input is required for generating scenarios and what is required for providing meaningful knowledge about the performance of the design. These two requirements represented the two main stages through which an intelligent system is able to retrieve imbedded knowledge. The scenario generation stage (Section 4.2.1) outlines some basic concepts by which various sources of information available from existing knowledge or other design models can be considered for automatic scenario construction. A valid CSS requires not just geometric data, but also additional inputs which define its context. The potential sources of information which have an impact on scenario context definition were identified from various places including information models, design guidance, real building data, but most importantly user input, which is required to guide the process towards a realistic evacuation scenario. The second stage (Section 4.2.2) concerns feedback of output data and outlines key performance indicators used in practice to assess design performance, and what are the concerns around them in an automation context. While most factors can be quantified, the fact remains that CS analysis relies on user visualisation as well. These factors were further explored in Chapter 6 for practical deployment and proven to be reliable ways to retrieve information and knowledge about the design in Chapter 7.

In terms of extensibility for future needs, several sources of information were identified which can be used throughout the entire building lifecycle and not just the design phase, which was under the scope of this thesis. One of the primary sources which contribute the most was perceived to be from design guidance, as it is not expected to change, thus some were represented in ontology knowledge in Chapter 5. This is also coupled with the fact that ontologies are able to store data for future design use, as explored in Chapter 6.

Answers related to practical deployment and extensibility for the future could not be outlined solely from Chapter 4 but had to be further explored in the next ones to give more insight into the matter.

Q5) ***What are the challenges concerning information models and workflow processes being represented in a knowledge base considering the requirements for integration and knowledge retrieval?***

Chapter 5 outlined the required information models as part of the overall system design. One of the requirements for knowledge mining is to have a knowledge base in the first place. The information models developed and introduced represent the knowledge base for crowd simulation analysis and all its required resources, such as the digital building model, or design codes.

The CSS ontology sits at the core of the developed system, being able to generically conceptualise a crowd simulation model with its relevant results. The CSS ontology was developed based on common object and feature concepts present in several CST tools investigated in Chapter 4. To be able to retrieve knowledge in the first place, additional concepts were added to conceptualise user assumptions and simulation results. The FBA ontology was developed which is closely related to the CSS, as was seen in their alignment. The scope of the CSS ontology is to represent simulation events and record them, while the scope of the FBA ontology is to analyse the results across several CSS resources as potential answers to design performance objectives. Creating objective concepts in the FBA ontology is vital for supporting a performance design process, where the decision ultimately lies with the designer. It was also pointed out that the way in which an objective is defined is based on several factors such as the intent of the designers, the types of performance indicators and the capabilities of the CST used for running simulations. The important aspects of such a method was outlined, along with its limitation when relying on user definitions of objectives, which requires extensive knowledge of the process and ad-hoc re-definition of concepts. Several objectives and results concepts were defined in more detail, as they were used and tested for the developed prototype and fully deployed and tested in Chapters 6 and 7, however more objective types need to be identified from industry practice for completeness.

The inclusion of a CST ontology was required on a generic level to interface with the other models. However due to a complex structure of a program, focusing on the main objects using a simple ontology is recommended with the role to transfer the necessary information to the CSS ontology which sits above it. IfcOwl was used to conceptualise the digital built environment, which is more than capable to provide all the necessary geometric objects and other contextual information from its other semantics, such as object properties, as pointed out in Section 5.2.1. It was concluded that aligning the geometry of objects directly or using rules is impractical due to the format of the IFC itself, and the fundamental differences between geometric representations, making such a method highly inefficient for reasoning and querying, as was further confirmed through testing in Chapter 7 for the first use case.

Other semantic resources were introduced which can contribute to the automation and knowledge mining process, thus allowing an ontology-based system to 'understand' the circumstances of each digital building environment to a higher extent. The prime example given here was the inclusion of design occupancy factors present in UK approved documents for fire safety. Although these codes could have been retrieved differently, they were expressed in an ontology fashion in order to be able to distinguish between different space types with the assistance of the Uniclass 2015 classification system, which is used to classify model objects. The alignment of these resources had to rely on SWRL rules due to the large sizes of individuals in the Uniclass RDF dataset. This shows that the inclusion of future resources from the SW would require significant rework and complex alignment methods would have to be employed to ensure the retrieved data is correct. In the case of the UKSOC factors, this was proven to work in Chapter 7. Several limitations were identified in the initial alignment however, especially the inconsistencies in defining space types present in the design codes, as well as not accounting for every space type, which meant that many types of space factors still need to be identified. However, the best way is to use the most realistic data available, which makes design occupancy factors unreliable in most situations.

Q6) ***What needs to be considered for design knowledge storage and retrieval concerning building egress performance using evacuation models?***

The principal requirements to be able to process knowledge mining was to use operators, as was introduced in the methodology section. Chapter 6 began by introducing the two main types of operators used in this research: SWRL rules and SPARQL queries. These are most commonly used in conjunction with OWL models. Many rules and queries were developed to facilitate automation, intelligent system operation and relevant knowledge retrieval for the performance assessment stage of simulation scenarios.

Firstly, the main need is to consider a robust structure of the ontology knowledge base by separating the resources accordingly, as was shown in the initial alignment in Chapter

5 and in Section 6.2.1. This ensures model information is correctly linked, providing the necessary context for the knowledge operation stages.

Secondly, the relevant user input needs to be saved in the knowledge base, providing context to the analysis feedback stage and keeping track of design intent, as shown in Section 6.2.2. This further accounts for future extensibility needs, as posed by Q4, but also provides further context for knowledge retrieval processes.

The third need concerns the automation of scenario creation which needs to separate the information retrieval process into geometry and context. Operators for geometry can become very complex and should not rely on reasoning, but rather employ simpler approaches, also confirmed by initial alignment attempts between the CSS and IfcOwl in Chapter 5. They are required to provide the model with static elements, which only need to be retrieved once. The knowledge retrieval after this stage consists in making the computer system 'understand' the context correctly. A way to define contextual information was shown using SWRL rules, which can be used quite creatively, depending on available resources. The problem remains that the system is only as intelligent as the rules it is provided with. Unfortunately, there is no way to guarantee that these rules will always allow the system to 'understand' the model, unless the auxiliary resources are provided in the first place, such as pre-defined properties in the IFC model or otherwise specified by user input.

Finally, regarding the need concerning the analysis feedback stage of the process, several rules were constructed to conceptualise some of the PIs for performance assessment of CSM data. The knowledge embedded within these rules should be retrieved ad-hoc from separate SQL simulation data, on par with user objectives, due to the large datasets provided by each simulation scenario. The limitation here is that the construction of objectives, simulation results and rules to process them are interconnected. While only a few use cases were presented in Chapter 6, and deployed for testing in Chapter 7, there is still a need to identify how to implement more sophisticated analysis algorithms on various levels. The way to deal with knowledge retrieval and management on a large scale was conceptualised in Section 6.3.2, where the relationships between different knowledge models is shown considering change in design context over the building lifecycle. This is another step implemented in ensuring the future extensibility as posed by Q4. This paradigm can be applied for both a macroscopic scenario level and a microscopic object level. Due to time constraints, only the macroscopic level was implemented in practice and was proven to work efficiently in Chapter 7. However, the exploration in practice of the object level feedback would be beneficial in assessing more complex situations on a large scale, as was shown in Section 6.3.4 using several examples. Unfortunately, even with high expressivity provided by ontologies, the fact remains that such complex situations are hard to assess even by human designers in real situations due to a lack of formalised knowledge and agent behaviour un-predictability.

Q7) ***How reliable is a knowledge-based system in understanding the building model and other linked data resources in facilitating correct and efficient design support?***

The tests carried on the case study building in Chapter 7 reveal certain limitations concerning the methodology and others concerning the system itself. The ONTOCS system was able to successfully create correct scenarios for Stage I. Certain limitations in understanding the geometry were found, which make automatic models different from the manually constructed ones. However, this is subjective and depends on the situation. The context of each model was retrieved correctly, as they rely on the rules and queries to function correctly. The knowledge retrieval process at Stage I relies on the quality and correctness of the operators constructed. When comparing the scenarios from the first use case, it was observed that when there are more agents and fewer exits, the automatic scenarios are more reliable, regardless of the difference on the placement of agent entry points. For the Stage II, all objectives inputted were correctly interpreted except one case where data about a specific point was missing and thus the rule could not return a result. Making use of secondary resources to create the context of simulation models, specifically the use of design guide occupancy factors requires further investigation. The vague nature of design factors may not be the preferred choice, and it is highly dependent on building layout and its areas.

In terms of the efficiency of the queries for Stage I, it was observed that retrieving geometry from a graph database can be time consuming, due to the complex structure of the IfcOwl used as a source. Retrieving object properties is relatively more efficient by contrast. Reasoning queries on model context was also computed in a relatively speedy manner. These methods rely heavily on the quality of the operators and the expressed knowledge. Where knowledge is available explicitly reasoning is done much faster than in the case where reasoning uses other information resources such as the UKSOC ontology included. For Stage II, retrieving the answers for each objective is relatively fast, but it appears to grow significantly when multiple rules are intersected to find more specific results. This was further investigated by running scalability tests for each query in single and multi-objective cases. The results show that scalability would not be a problem when several dozen scenarios are applied, and that many queries remain constant, with some showing linearity. From the results it was apparent that the reasoning times scaled in steps, with the first encountered at 15 simulations, with slightly higher reasoning time than the simulation numbers before it. However, further tests would be required with sessions running a significantly higher number of simulations to identify the system and methodology limits. On the other hand, assuming an unrealistically high number of scenarios in parallel would not be beneficial in practice, defeating the purpose of making the process fast in the first place.

Finally, it is worth noting that other CSTs results will differ than those provided here by MassMotion. However, the general findings seem to indicate that although simulations created automatically are different from those constructed manually, the trends can be similar enough to be used with enough confidence in decision-making process for design evaluation.

It can be concluded that the hypothesis is true, semantic web data and model representations allow a fast retrieval of all the information requirements for CS construction and analysis. However, the implementations to do so require much investment in representing and linking all the relevant data, and correctly conceptualising the knowledge base with operators according to proven methods. This requires extensive knowledge of the domains involved and will require expertise for upkeeping a knowledge-based system up-to-date.

8.2. Research limitations

The limitations of this research and its developments are outlined here concerning the methodology, the system or the tools used in implementation and testing.

ONTOCS system limitations:

1) Dependency on code – the system itself is heavily reliant on its code packages to retrieve, transform and manipulate data from various resources including RDF graphs, SQL database and the user interface. This means that future developments need to integrate with its overall architecture. For example, the system converts IFC geometry from IfcOwl in memory directly into a MassMotion format. The inclusion of an additional CST would require a separate package solely for geometric conversion, in addition to some working with ontologies.
2) Dependency on the MassMotion tool – the system can only collaborate with the MassMotion CST at the moment. This was done for simplicity.
3) Interface restrictions – the system currently interfaces users over web pages through internet browsers in a very simplistic way, in which users cannot really interact directly with most of the model objects.

4) Code optimisation – the system was developed in an experimental context but would need more optimisation from a programming point of view, thus making the process faster overall. This however is not expected to improve reasoning times.

Knowledge base and operators limitations:

1) Level of knowledge formalisation - retrieval of knowledge depends on the level of the developed ontology and its operators. The developed ontologies do not encompass all available resources identified in Section 4.2, due to time constraints. This process requires extensive validation and testing.
2) The FBA ontology and its respective SWRL rules are currently limited to only a few types of PIs assessment. These do not represent feedback analysis on an object level but is set as an objective for future work.
3) SPARQL optimisation – over 32 queries were developed which work with the system retrieving data and knowledge. Geometry related queries are long due to the structure of the IfcOwl ontology. For the queries to become more efficient, a re-structuring and re-definition of the IfcOwl is recommended for future implementations.
4) Limits of imbedded knowledge – finding new knowledge about evacuation design performance is limited to the power of the knowledge operators used. CS analysis traditionally requires safety engineers to observe agent movement and thus relies on visualisation.

8.3. Future work

Considering the limitations above, the following will be addressed in future work:

- Integration of a second CST with the ONTOCS system and its workflow. This would allow a comparative analysis of simulation results.
- Further improvement of the ONTOCS interface to allow an object view of the schema and more data results in the form of graphs, aiding the users in assessing the evacuation progression if required.
- ONTOCS classes code optimisations will allow the system to work more efficiently and account for future extensions.
- Investigation into PIs for object level performance assessment. In light of the benefits provided by the object-oriented of the developed CSS schema, an object-level view of the feedback process would be better suited for comparing different results on a microscopic level. This would require an investigation from literature and expert consultations on what methods are employed in practice.

These would then have to be implemented in the CSS and FBA ontologies, and coded to work with the system.

- Investigation into methods to related scenarios and feedback models to changing IFC source models. Currently, the system makes use of a single IFC model which is the central ‘point of truth’ from which several simulations emerge, as was shown in Section 6.3.2. Dealing with changing versions of the IFC model would be a challenge to manage information and knowledge models correctly, whilst allowing a comparative analysis between different design stages in terms of model performance.
- Further testing is planned for identifying the apparent incremental steps for reasoning times from a scalability perspective. Additionally, a newer Stardog version has been released which promises better query and reasoning performance. This would also be a good choice for improving reasoning times.

Chapter 9. Conclusion and contribution

Taking into consideration the findings and developments presented in previous chapters, several conclusions can be drawn.

Section 2.1 revealed that the evaluation of evacuation time using CS models and tools is a complex and inefficient process which demands input of information from several fields of knowledge, including designer input across several stages during the process, as was also identified in Chapter 4.

Although multiple attempts to integrate BIM with CST have been carried out in the past with the purpose to speed up the design process, these are mostly limited to geometry and do not consider a more holistic view of the information requirements for creating and analysing evacuation models. This was concluded from the literature in Section 2.2.

The use of OWL ontologies is beneficial in providing a comprehensive layer of interoperability across multiple information models and knowledge domains available as web resources for the semi-automatic construction and analysis of CS models on a large scale, as was augmented in Section 2.3.

Based on the above conclusions from the literature, the proposed methodology and system under investigation aimed to prove that a knowledge-based system was able to provide automation and a greater degree of interoperability for the crowd simulation-based evacuation performance evaluation.

The developed OWL ontologies introduced in Chapter 5 conceptualise the crowd simulation analysis domain in detail, allowing intelligent computer agents to store and retrieve knowledge about a building design from multiple contexts. These ontologies were verified through testing (in Chapter 7) and validated by consulting field experts.

The developed knowledge operators and the workflow process provided by the ONTOCS system allows fast, correct and realistic scenario construction and analysis of simulation data on a large scale. This was evident from the results presented in Chapter 7. New knowledge can be retrieved about the design in accordance to design PIs on a scenario level, allowing a fast review of building performance in dozens of parallel contexts.

The developed OWL ontologies show great potential in storing and retrieving knowledge about the entire process, which can be extended to merge with other design fields (costs, energy, etc). Due to their holistic representation of information, they also show promise in exploring creative ways to evaluate design performance by considering an object-level view of model components (e.g. individual spaces, doors, assumptions, agents, etc.), as was discussed in Chapter 6.

The ONTOCS system has been validated on a real-life building showing its potential to operate of real data and to work efficiently with a large scale of scenarios. The findings from the case study in Chapter 7 strengthen the arguments brought forward from previous chapters, but also outlined several limitations and how to deal with them for future research developments.

The work carried out during this research project has contributed with several practical developments and with knowledge about the methodology adopted in delivering their implementation and testing.

<u>Practical research developments, in decreasing order of impact:</u>

1- ONTOCS (Ontology Crowd Simulation) software system, represents the core contribution of this research; an intelligent knowledge-based system capable of aggregating important data and information models which are relevant to evacuation design using CSTs. The intelligence imbedded within the system allows it to perform automatic scenario creation and to provide feedback to designers concerning building evacuation performance;

2- OWL ontologies in the field of crowd simulation:

   i. Crowd Simulation Scenario (CSS) – represents a generic view of a simulation model containing objects, assumptions and results, suitable to interface with a plethora of other resources such as IfcOwl, design codes and CSTs;

   ii. Feeback Analysis (FBA) – conceptualises the mechanism to store design objectives and retrieve knowledge about design performance across multiple scenario models from the CSS ontology data;

   iii. UK Spaces Occupancy Capacities (UKSOC) – conceptualises part of design codes from the UK approved documents regarding occupancy of spaces based on their functionality and area. It has been aligned with the IFC ontology and Uniclass classification system to provide meaningful data in a conceptual design scenario;

iv. MassMotion ontology – the simple representation of the MassMotion crowd simulation tool, according to its objects' relationships;

3- SWRL rules developed in conjunction with the ontologies above to provide logical operations on data and facilitate the correct retrieval of knowledge from existing resources; these represent imbedded knowledge about the construction of simulation models and about the correct analysis of the simulation output;

4- Taxonomy of common CST concepts – represents a baseline of objects which describe CS models and tools for them to be able to function. These concepts were identified from the analysis of several CSTs used in industry.

Knowledge contribution from theoretical design, analysis and testing, in decreasing order of impact:

1- By the sum of investigations carried to identify, represent and test the knowledge bases and operators within the scope of the research aims has contributed to knowledge about the limitations, benefits and challenges when employing such methods. This is concerned mostly with the field of CS, but the steps outlined in this thesis would also be easily replicated for other design disciplines, further benefiting from linked data concepts.

2- The investigations into CS and BIM interoperability has contributed to knowledge by looking at common concepts between the two fields while commenting on their behaviours and challenges when aligning them in Chapter 5. Some concepts within the BIM field behave differently because they meet different functions within a model when compared to a CS model. The alignment of geometry remains problematic within the context of full ontological alignment, while the BIM domain lacks the definition of certain concepts and resources to facilitate CS automation.

3- This study has relied heavily on using design regulations and guidance to conceptualise a performance assessment method in a machine-interpretable way. It has therefore contributed to knowledge by identifying the relevant sources of information from design via the literature, official documentations and also from field consultations with experts. Their reliability for automatic construction of models and knowledge retrieval was assessed in Chapter 7, along with the observations made during their testing.

4- Finally, this research has contributed to knowledge from the literature surveys in the fields of CS, building evacuation, BIM and ontologies, commenting on recent developments, limitations and potential benefits.

Bibliography

Abanda, F.H. Tah, J.H.M. and Keivani, R. 2013. Trends in built environment semantic Web applications: Where are we today? *Expert Systems with Applications* 40(14), pp. 5563–5577. Available at: http://dx.doi.org/10.1016/j.eswa.2013.04.027.

Abdul-Ghafour, S. Ghodous, P. Shariat, B. Perna, E. and Khosrowshahi, F. 2014. Semantic interoperability of knowledge in feature-based CAD models. *Computer-Aided Design* 56, pp. 45–57. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0010448514001195 [Accessed: 21 November 2014].

Alper, S.J. and Karsh, B.T. 2009. A systematic review of safety violations in industry. *Accident Analysis & Prevention* 41(4), pp. 739–754. Available at: http://linkinghub.elsevier.com/retrieve/pii/S000145750900061X.

Apel, S. Lengauer, C. Möller, B. and Kästner, C. 2008. An algebra for features and feature composition. In: *International Conference on Algebraic Methodology and Software Technology*. Springer., pp. 36–50.

Apel, S. Lengauer, C. Möller, B. and Kästner, C. 2009. An overview of feature-oriented software development. *Journal of Object Technology* 8(5), pp. 49–84.

Azhar, S. 2011. Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. *Leadership and management in engineering* 11(3), pp. 241–252.

Baiche, B. Walliman, N. and Ogden, R. 2006. Compliance with building regulations in England and Wales. *Structural Survey* 24(4), pp. 279–299. Available at: http://www.emeraldinsight.com/doi/10.1108/02630800610704427.

Bates, M.J. 2011. *Understanding information retrieval systems: management, types, and standards*. CRC Press.

Batory, D. 2005. Feature models, grammars, and propositional formulas. In: *International Conference on Software Product Lines*. Springer., pp. 7–20.

Beach, T.H. Rezgui, Y. Li, H. and Kasim, T. 2015. A rule-based semantic approach for automated regulatory compliance in the construction sector. *Expert Systems with Applications* 42(12), pp. 5219–5231. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0957417415001360.

Beetz, J. van Leeuwen, J. and de Vries, B. 2009. IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 23(1), p. 89. Available at: http://www.journals.cambridge.org/abstract_S0890060409000122.

Bejay Jayan 2016. *Real-time Multi-scale Smart Energy Management and Optimisation (REMO) for Buildingsand Their District*. PhD Thesis, Cardiff University

Bellinger, G. Castro, D. and Mills, A. 2004. Data, information, knowledge, and wisdom.

van Berlo, L. Derks, G. Pennavaire, C. and Bos, P. 2015. Collaborative Engineering with IFC: Common Practice in the Netherlands. *Proc. of the 32nd CIB W78 Conference 2015, 27th-29th October 2015, Eindhoven, The Netherlands* (Mvd), pp. 59–68.

Bos, P. 2012. Collaborative engineering with IFC : new insights and technology. *eWork and eBusiness in Architecture, Engineering and Construction, 9th ECPPM Conference Proceedings* , pp. 811–818.

Bradley, A. 2017. Uniclass2015 RDF dataset. Available at: http://www.icompe.engin.cf.ac.uk/ontologies/uniclass2015.

BuildingSMART 2017. BuildingSMART Linked Data Working Group. Available at: http://www.buildingsmart-tech.org/future/linked-data.

Bunnin, N. and Jiyuan, Y. 2004. *The Blackwell Dictionary of Western Philosophy*. Blackwell. Blackwell Publishing. doi: 10.1111/b.9781405106795.2004.x.

Butterfield, A. and Ngondi, G.E. eds. 2016. *A Dictionary of Computer Science*. Oxford University Press. Available at: http://www.oxfordreference.com/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975.

Cabinet Office 2011. *Government construction strategy, May 2011*. Available at: https://www.gov.uk/government/publications/government-construction-strategy.

Cassol, V. Oliveira, J. Musse, S.R. and Badler, N. 2016. Analyzing egress accuracy through the study of virtual and real crowds. In: *Virtual Humans and Crowds for Immersive Environments (VHCIE), IEEE*. IEEE., pp. 1–6.

Chen, A.Y. and Chu, J.C. 2016. TDVRP and BIM Integrated Approach for In-Building Emergency Rescue Routing. *Journal of Computing in Civil Engineering* 30(5), p. C4015003. Available at: http://ascelibrary.org/doi/10.1061/%28ASCE%29CP.1943-5487.0000522.

Chen, A.Y. and Huang, T. 2015. Toward BIM-Enabled Decision Making for In-Building Response Missions. *IEEE Transactions on Intelligent Transportation Systems* 16(5), pp. 2765–2773. Available at: http://ieeexplore.ieee.org/document/7101252/.

Chitty, R. and Fraser-Mitchell, J. 2003. *Fire safety engineering: a reference guide*. Available at: https://www.thenbs.com/PublicationIndex/documents/details?Pub=BRE&DocID=264317.

Choi, Junsik Choi, Junho and Kim, Inhan 2014. Development of BIM-based evacuation regulation checking system for high-rise and complex buildings. *Automation in Construction* 46, pp. 38–49. Available at: http://dx.doi.org/10.1016/j.autcon.2013.12.005.

Crotty, R. 2013. *The impact of building information modelling: transforming construction*. Routledge.

Damrongrat, C. Kanai, H. and Ikeda, M. 2013. Increasing situational awareness of indoor emergency simulation using multilayered ontology-based floor plan representation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8017 LNCS(PART 2), pp. 39–45. doi: 10.1007/978-3-642-39215-3_5.

Díaz, H. Alarcón, L.F. Mourgues, C. and García, S. 2017. Multidisciplinary Design Optimization through process integration in the AEC industry: Strategies and challenges. *Automation in Construction* 73, pp. 102–119. Available at: http://dx.doi.org/10.1016/j.autcon.2016.09.007.

Dibley, M.J. 2011. *An intelligent system for facility management*. PhD Thesis, Cardiff University

Dimyadi, J. Pauwels, P. Spearpoint, M. Clifton, C. and Amor, R. 2015. Querying a Regulatory Model for Compliant Building Design Audit. *Proc. of the 32nd CIB W78 Conference 2015, 27th-29th October 2015, Eindhoven, The Netherlands* , pp. 139–148. doi: 10.13140/RG.2.1.4022.6003.

Dimyadi, J. Clifton, C. Spearpoint, M. and Amor, R. 2016. Computerizing Regulatory Knowledge for Building Engineering Design. *Journal of Computing in Civil Engineering* 30(5), p. C4016001. Available at: http://ascelibrary.org/doi/10.1061/%28ASCE%29CP.1943-5487.0000572.

Druzdzel, M.J. and Flynn, R.R. 1999. Decision support systems. Encyclopedia of library and information science. A. Kent. *Marcel Dekker, Inc. Last Login* 10(3), p. 2010.

Duives, D.C. Daamen, W. and Hoogendoorn, S.P. 2013. State-of-the-art crowd motion simulation models. *Transportation Research Part C: Emerging Technologies* 37, pp. 193–209. Available at: http://dx.doi.org/10.1016/j.trc.2013.02.005.

Dürst, M. and Suignard, M. 2004. *Internationalized resource identifiers (IRIs).* W3C Submission, No. RFC 3987

Durupinar, F. 2010. From Audiences to Mobs: Crowd Simulation with Psychological Factors. PhD Thesis, Bilkent University

Eastman, C. Teicholz, P. Sacks, R. and Liston, K. 2011. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors, 2nd Edition* .Wiley. Available at: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470541377.html.

Euzenat, J. and Valtchev, P. 2004. Similarity-based ontology alignment in OWL-lite. In: Proceedings of the 16th *European Conference of Artificial Intelligence*, August 2004, Valencia, pp.333

Fang, J. El-Tawil, S. and Aguirre, B. 2016. New Agent-Based Egress Model Allowing for Social Relationships. *Journal of Computing in Civil Engineering* 30(4), p. 4015066. Available at: 10.1061/(ASCE)CP.1943-5487.0000532.

Farias, T.M. Roxin, A.-M. and Nicolle, C. 2015. IfcWoD , Semantically Adapting IFC Model Relations into OWL Properties. *Proc. of the 32nd CIB W78 Conference 2015, 27th-29th October 2015, Eindhoven, The Netherlands* , pp. 175–185.

Fernando, T.P. Wu, K.-C. and Bassanino, M.N. 2013. Designing a novel virtual collaborative environment to support collaboration in design review meetings. *Journal of Information Technology in Construction* 18(August), pp. 372–396.

Fidan, G. Dikmen, I. Tanyer, A.M. and Birgonul, M.T. 2011. Ontology for Relating Risk and Vulnerability to Cost Overrun in International Projects. *Journal of Computing in Civil Engineering* 25(4), pp. 302–315. Available at: http://ascelibrary.org/doi/10.1061/(ASCE)CP.1943-5487.0000090.

Fruin, J. 1992. Designing for pedestrians. *Public Transportation United States*

FSEG, Uiversity of Greenwich 2018. Exodus. Available at: https://fseg.gre.ac.uk/exodus/index.html [Accessed: 4 January 2018].

Gibbins, N. and Shadbolt, N. 2009. Resource Description Framework (RDF). In, Encyclopedia of Library and Information Sciences. University of Southampton Available at: https://eprints.soton.ac.uk/268264/1/gibbins-shadbolt-elis-rdf-v3.pdf [Accessed: 25 March 2018]

Government Digital Service 2018. Grenfell Tower. Available at: https://www.gov.uk/government/collections/grenfell-tower [Accessed: 28 February 2018].

Grover, R. and Froese, T.M. 2016. Knowledge Management in Construction Using a SocioBIM Platform: A Case Study of AYO Smart Home Project. *Procedia Engineering* 145, pp. 1283–1290. Available at: http://dx.doi.org/10.1016/j.proeng.2016.04.165.

Gwynne, S. Galea, E.R. Owen, M. Lawrence, P.J. and Filippidis, L. 1999. A review of the methodologies used in the computer simulation of evacuation from the built environment. *Building and Environment* 34(6), pp. 741–749. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0360132398000572.

Gwynne, S. Galea, E.R Lawrence, P.J and Filippidis, L. 2001. Modelling occupant interaction with fire conditions using the buildingEXODUS evacuation model. *Fire Safety Journal* 36(4), pp. 327–357. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0379711200000606.

Hardin, B. 2009. *BIM and Construction Management: Proven Tools, Methods, and Workflows*. Wiley. Available at: https://books.google.co.uk/books?id=pduhGycNlZMC.

Hardin, B. and McCool, D. 2015. *BIM and construction management: proven tools, methods, and workflows*. John Wiley & Sons.

Hay, D. 2006. Data Modeling, RDF, & OWL – Part One: An Introduction To Ontologies. Available at: http://tdan.com/data-modeling-rdf-owl-part-one-an-introduction-to-ontologies/5025.

Hebeler, J. Fisher, M. Blace, R. and Perez-Lopez, A. 2011. *Semantic web programming*. John Wiley & Sons.

Helbing, D. and Molnár, P. 1995. Social force model for pedestrian dynamics. *Physical Review E* 51(5), pp. 4282–4286. Available at: http://link.aps.org/doi/10.1103/PhysRevE.51.4282.

Hesham, O. and Wainer, G. 2016. Centroidal Particles for Interactive Crowd Simulation. In *Proceedings of the Summer Computer Simulation Conference*. Society for Computer Simulation International. pp.7

Hitzler, P. Krötzsch, M. Parsia, B. Patel-Schneider, P.F. and Rudolph, S. 2009. OWL 2 web ontology language primer. *W3C recommendation* 27(1), p. 123.

Home Office 2018. Fire and rescue incident statistics: England, year ending September 2017. Available at: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/679924/fire-and-rescue-incident-sep17-hosb0418.pdf [Accessed 23 March 2018]

Hopfe, C.J. and Hensen, J.L.M. 2011. Uncertainty analysis in building performance simulation for design support. *Energy and Buildings* 43(10), pp. 2798–2805. Available at: http://dx.doi.org/10.1016/j.enbuild.2011.06.034.

Horrocks, I. Patel-Schneider, P.F. and Van Harmelen, F. 2004. A semantic web rule language combining OWL and RuleML. *W3C submission*

Horrocks, I. 2005. OWL Rules, OK? *Rule Languages for Interoperability* April 27(34), pp. 1–5. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.215.4587.

Howell, S.K. 2017. *Towards a semantic web of things for smart cities*. PhD Thesis, Cardiff University

INCONTROL Simulation Solutions 2018. Pedestrian Dynamics. Available at: http://www.pedestrian-dynamics.com/pedestrian-dynamics/pedestrian-dynamics.html.

Integrated Environmental Solutions Limited 2018. Simulex. Available at: https://www.iesve.com/software/ve-for-engineers/module/simulex/480.

Isikdag, U. Zlatanova, S. and Underwood, J. 2013. A BIM-Oriented Model for supporting indoor navigation requirements. *Computers, Environment and Urban Systems* 41, pp. 112–123. Available at: http://dx.doi.org/10.1016/j.compenvurbsys.2013.05.001.

Jalali, L. Mehrotra, S. and Venkatasubramanian, N. 2011. Interoperability of Multiple Autonomous Simulators in Integrated Simulation Environments. In: *2011 Spring Simulation Interoperability Workshop*. Available at: http://www.ics.uci.edu/~dsm/pubs/SIW-final.pdf.

Kalay, Y.E. 1998. P3: Computational environment to support design collaboration. *Automation in construction* 8(1), pp. 37–48.

Kang, K.C. Cohen, S.G. Hess, J.A. Novak, W.E. and Peterson, A.S. 1990. *Feature-oriented domain analysis (FODA) feasibility study*. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

Kannala, M. 2005. *Escape Route Analysis Based on Building Information Models : Design and Implementation*. Msc Dissertation, Helsinki University of Technology

Kaufman, K.A. and Michalski, R.S. 2005. From Data Mining to Knowledge Mining. In: *Handbook of Statistics*., pp. 47–75. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0169716104240020.

Khan, S.D. Crociani, L. Vizzari, G. 2014. Pedestrian and crowd studies: Towards the integration of automated analysis and synthesis. *SCS M&S Magazine* 3(3)

Kinateder, M. Ronchi, E. Nilsson, D. Kobes, M. Müller, M. Pauli, P. and Mühlberger, A. 2014. Virtual Reality for Fire Evacuation Research. *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on* 2, pp. 319–327. doi: 10.15439/2014F94.

Kinsey, M. Walker, G. Swailes, N. and Butterworth, N. 2015. *The Verification and Validation of MassMotion for Evacuation Modelling. Report, Ove Arup & Partners Ltd, 10th August 2015.*

Kobes, M. Helsloot, I. de Vries, B. and Post, J.G. 2010. Building safety and human behaviour in fire: A literature review. *Fire Safety Journal* 45(1), pp. 1–11. Available at: http://dx.doi.org/10.1016/j.firesaf.2009.08.005.

Kraus, L. Stanojević, M. Tomašević, N. and Mijović, V. 2011. A decision support system for building evacuation based on the EMILI SITE environment. *Proceedings of the 2011 20th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2011* , pp. 334–336. doi: 10.1109/WETICE.2011.59.

Krötzsch, M. 2010. *Description Logic Rules*. IOS Press.

Krötzsch, M. Maier, F. Krisnadhi, A. and Hitzler, P. 2011. A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In: *Proceedings of the 20th international conference on World wide web*. ACM., pp. 645–654.

Kuligowski, E. 2013. Predicting Human Behavior During Fires. *Fire Technology* 49(1), pp. 101–120. doi: 10.1007/s10694-011-0245-6.

Kuligowski, E.D. 2005. *A review of building evacuation models*. Gaithersburg, MD. Available at: http://nvlpubs.nist.gov/nistpubs/Legacy/TN/nbstechnicalnote1471.pdf.

Kuligowski, E.D. 2016a. Computer evacuation models for buildings. In: *SFPE Handbook of Fire Protection Engineering*. Springer., pp. 2152–2180.

Kuligowski, E.D. 2016b. Human behavior in fire. In: *SFPE Handbook of Fire Protection Engineering*. Springer., pp. 2070–2114.

Kvan, T. 2000. Collaborative design: What is it? *Automation in Construction* 9(4), pp. 409–415. doi: 10.1016/S0926-5805(99)00025-4.

Lee, D.-Y. Chi, H.-L. Wang, J. Wang, X. and Park, C.-S. 2016. A linked data system framework for sharing construction defect information using ontologies and BIM environments. *Automation in Construction* 68, pp. 102–113. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0926580516300838.

Lee, H. Shin, J. and Lee, J.-K. 2016. BIM-enabled Definition of a Path Object and its Properties to Evaluate Building Circulation using Numerical Data. *Journal of Asian Architecture and Building Engineering* 15(3), pp. 425–432. Available at: https://www.jstage.jst.go.jp/article/jaabe/15/3/15_425/_article.

Lee, J.M. 2010. Automated Checking of Building Requirements on Circulation Over a Range of Design Phases. *Building* , p. 172. Available at: http://smartech.gatech.edu/bitstream/handle/1853/34802/lee_jae_m_201008_phd.pdf;jsessionid=10FA65D2354A62739D238AB3B3A15577.smart2?sequence=1.

Lee, S. Kim, K. and Yu, J. 2014. BIM and ontology-based approach for building cost estimation. *Automation in Construction* 41, pp. 96–105. Available at: http://www.sciencedirect.com/science/article/pii/S092658051300188X [Accessed: 12 December 2014].

Li, T. Chang, H. Wang, M. Ni, B. Hong, R. and Yan, S. 2015. Crowded Scene Analysis: A Survey. *IEEE Transactions on Circuits and Systems for Video Technology* 25(3), pp. 367–386. Available at: http://ieeexplore.ieee.org/document/6898845/.

Luo, H. Peng, X. and Zhong, B. 2016. Ontology-based Emergency Plan Management of Metro Operation and its Application in Staff Training. In: *Creative Construction Conference 2016*., pp. 59–64.

Malsane, S. Matthews, J. Lockley, S. Love, P.E.D. and Greenwood, D. 2015. Development of an object model for automated compliance checking. *Automation in*

Construction 49(PA), pp. 51–58. Available at: http://dx.doi.org/10.1016/j.autcon.2014.10.004.

Marzouk, M. and Al Daoor, I. 2016. Simulation of labor evacuation: The case of housing construction projects. *HBRC Journal* , pp. 1–9. Available at: http://dx.doi.org/10.1016/j.hbrcj.2016.07.001.

Mashhadawi, M. 2016. *MassMotion Evacuation Model Validation*. Report, Department of Fire Safety Engineering, University of Lund.

Masinter, L. Berners-Lee, T. and Fielding, R.T. 2005. Uniform resource identifier (URI): Generic syntax. RFC 3986, The Internet Society.

McGuinness, D.L. and Van Harmelen, F. 2004. OWL web ontology language overview. *W3C recommendation* 10(10), p. 2004.

Moreno, A. Zlatanova, S. Bucher, B. Posada, J. Toro, C. and García-Alonso, A. 2011. Semantic Enhancement of a Virtual Reality Simulation System for Fire Fighting. In: *Proceedings of the Joint ISPRS Workshop on 3D City Modelling & Applications and the 6th 3D GeoInfo Conference*.

Motamedi, A. Wang, Z. Yabuki, N. Fukuda, T. and Michikawa, T. 2016. Signage Visibility Analysis and Optimization System Using BIM-enabled VR Environments. In: *16th International Conference on Computing in Civil and Building Engineering (ICCCBE2016)*., pp. 1–20.

Motik, B. and Rosati, R. 2010. Reconciling description logics and rules. *Journal of the ACM* 57(5), pp. 1–62. Available at: http://portal.acm.org/citation.cfm?doid=1754399.1754403.

Mott MacDonald 2018. STEPS. Available at: https://www.steps.mottmac.com/introducing-steps [Accessed: 4 January 2018].

Musse, S.R. and Thalmann, D. 2001. Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics* 7(2), pp. 152–164. Available at: http://ieeexplore.ieee.org/document/928167/.

Mustapha, K. and Frayret, J.-M. 2016. Agent-based Modeling and Simulation Software Architecture for Health Care. In: *Proceedings of the 6th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*. SCITEPRESS - Science and Technology Publications., pp. 89–100. Available at: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005972600890100.

NBS, R.E.L. 2017. Uniclass 2015. Available at: https://toolkit.thenbs.com/articles/classification/ [Accessed: 30 May 2017].

Nelson, H.E. 2002. Emergency movement. *The SFPE Handbook of Fire Protection Engineering.*

Niknam, M. and Karshenas, S. 2014. Integrating distributed sources of information for construction cost estimating using Semantic Web and Semantic Web Service technologies. *Automation in Construction* 57, pp. 222–238. Available at: http://dx.doi.org/10.1016/j.autcon.2015.04.003.

Nilsson, D. and Fahy, R. 2016. Selecting scenarios for deterministic fire safety engineering analysis: life safety for occupants. In: *SFPE Handbook of Fire Protection Engineering*. Springer., pp. 2047–2069.

Noy, N.F. Crubézy, M. Fergerson, R.W. Knublauch, H. Tu, S.W. Vendetti, J. and Musen, M.A. 2003. Protégé-2000: an open-source ontology-development and knowledge-acquisition environment. In: *AMIA... Annual Symposium proceedings. AMIA Symposium*. American Medical Informatics Association., p. 953.

Noy, N.F. and McGuinness, D.L. 2001. Ontology development 101: A guide to creating your first ontology. Available at: https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.

Oasys Limited 2018a. MassMotion. Available at: http://www.oasys-software.com/products/engineering/massmotion.html.

Oasys Limited 2018b. MassMotion case studies. Available at: https://www.oasys-software.com/case-studies/ [Accessed: 1 March 2018].

Oh, M. Lee, J. Hong, S.W. and Jeong, Y. 2015. Integrated system for BIM-based collaborative design. *Automation in Construction* 58, pp. 196–206. Available at: http://dx.doi.org/10.1016/j.autcon.2015.07.015.

Onorati, T. Malizia, A. Diaz, P. and Aedo, I. 2014. Modeling an ontology on accessible evacuation routes for emergencies. *Expert Systems with Applications* 41(16), pp. 7124–7134. Available at: http://dx.doi.org/10.1016/j.eswa.2014.05.039.

OpenBIMstandards 2017a. How is IfcOwl generated? Available at: http://openbimstandards.org/standards/ifcowl/how-is-ifcowl-generated/.

OpenBIMstandards 2017b. Web Ontology Language representation of the Industry Foundation Classes (IFC) schema. Available at: http://openbimstandards.org/standards/ifcowl/.

Pauwels, P. Van Deursen, D. Verstraeten, R. De Roo, J. De Meyer, R. Van de Walle, R. and Van Campenhout, J. 2011. A semantic rule checking environment for building performance checking. *Automation in Construction* 20(5), pp. 506–518. Available at: http://dx.doi.org/10.1016/j.autcon.2010.11.017.

Pauwels, P. de Farias, T.M. Zhang, C. Roxin, A. Beetz, J. and De Roo, J. 2016. Querying and reasoning over large scale building data sets. In: *Proceedings of the International Workshop on Semantic Big Data - SBD '16*. New York, New York, USA: ACM Press., pp. 1–6. Available at: http://dl.acm.org/citation.cfm?doid=2928294.2928303.

Pauwels, P. Krijnen, T. Terkaj, W. and Beetz, J. 2017. Enhancing the ifcOWL ontology with an alternative representation for geometric data. *Automation in Construction* 80, pp. 77–94. Available at: http://dx.doi.org/10.1016/j.autcon.2017.03.001.

Pauwels, P. and Roxin, A. 2016. SimpleBIM : From full ifcOWL graphs to simplified building graphs. In Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM) (pp. 11-18).

Pauwels, P. and Terkaj, W. 2016. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction* 63, pp. 100–133. Available at: http://dx.doi.org/10.1016/j.autcon.2015.12.003.

PD 7974 2004. *The application of fire safety engineering principles to fire safety design of buildings. Human factors. Life safety strategies. Occupant evacuation, behaviour and condition (Sub-system 6)*. British Standards Institution Group London UK

Perez, J. Arenas, M. and Gutierrez, C. 2006. Semantics and Complexity of SPARQL. 34(3), pp. 1–45. Available at: http://arxiv.org/abs/cs/0605124.

Pilone, D. and Pitman, N. 2005. *UML 2.0 in a Nutshell*. O'Reilly Media, Inc.

Plume, J. and Mitchell, J. 2007. Collaborative design using a shared IFC building model-Learning from experience. *Automation in Construction* 16(1), pp. 28–36. doi: 10.1016/j.autcon.2005.10.003.

Poveda, G. Serrano, E. and Garijo, M. 2014. Creating and Validating Emergency Management Services by Social Simulation and Semantic Web Technologies. In: *International Conference on Ubiquitous Computing and Ambient Intelligence*., pp. 460–467. Available at: http://link.springer.com/10.1007/978-3-319-13102-3_74.

Prud, E. and Seaborne, A. 2006. SPARQL query language for RDF.W3C Submission. Available at: https://www.w3.org/TR/rdf-sparql-query/ [Accessed 25 March 2018]

Purser, D. and Bensilum, M. 2001. Quantification of behaviour for engineering design standards and escape time calculations. *Safety Science* 38(2), pp. 157–182. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0925753500000667.

Rahm, E. and Bernstein, P.A. 2001. A survey of approaches to automatic schema matching. *VLDB Journal* 10(4), pp. 334–350. doi: 10.1007/s007780100057.

Raupp Musse, S. Ulicny, B. and Aubel, A. 2006. Groups and Crowd Simulation. *Handbook of Virtual Humans* (Benesch 1995), pp. 323–352. doi: 10.1002/0470023198.ch14.

Ronchi, E. Gwynne, S.M.V. Purser, D. A. and Colonna, P. 2013. Representation of the Impact of Smoke on Agent Walking Speeds in Evacuation Models. *Fire Technology* 49(2), pp. 411–431. doi: 10.1007/s10694-012-0280-y.

Ronchi, E. Kuligowski, E.D. Nilsson, D. Peacock, R.D. and Reneke, P.A. 2014. Assessing the Verification and Validation of Building Fire Evacuation Models. *Fire Technology* , pp. 197–219. Available at: http://dx.doi.org/10.1007/s10694-014-0432-3.

Ronchi, E. and Nilsson, D. 2013. Fire evacuation in high-rise buildings: a review of human behaviour and modelling research. *Fire Science Reviews* 2(1), p. 7. Available at: http://firesciencereviews.springeropen.com/articles/10.1186/2193-0414-2-7.

Rűppel, U. Lange, M and Wagenknecht, A 2006. Semantic integration of product model data in fire protection engineering. In: *eWork and eBusiness in Architecture, Engineering and Construction. ECPPM 2006: European Conference on Product and Process Modelling 2006 (ECPPM 2006), Valencia, Spain, 13-15 September 2006*., p. 115.

Rüppel, U. and Schatz, K. 2011. Designing a BIM-based serious game for fire safety evacuation simulations. *Advanced Engineering Informatics* 25(4), pp. 600–611. doi: 10.1016/j.aei.2011.08.001.

Scherer, R.J. and Schapke, S.-E. 2011. A distributed multi-model-based Management Information System for simulation and decision-making on construction projects. *Advanced Engineering Informatics* 25(4), pp. 582–599. Available at: http://linkinghub.elsevier.com/retrieve/pii/S1474034611000644 [Accessed: 19 October 2014].

Schevers, H. and Drogemuller, R. 2006. Converting the industry foundation classes to the web ontology language. *Proceedings - First International Conference on*

Semantics, Knowledge and Grid, SKG 2005 (Skg 2005), pp. 2005–2007. doi: 10.1109/SKG.2005.59.

Shafiq, M.T. Matthews, J. Lockley, and Stephen R. 2012. Requirements for Model Server Enabled Collaborating on Building Information Models. *International Journal of 3-D Information Modeling* 1(4), pp. 8–17. Available at: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/ij3dim.2012100102.

Shields, T. and Silcock, G.W. 1987. *Buildings and Fire*. Longman Group UK Limited 1987.

Sikos, L. 2015. *Mastering Structured Data on the Semantic Web:* . Apress. Available at: https://www.dawsonera.com:443/abstract/9781484210499.

Slamecka, V. and Hosch, W.L. 2008. Query language. *Britannica* . Available at: https://www.britannica.com/technology/query-language.

Stanford University 2018. Protégé. Available at: https://protege.stanford.edu/ [Accessed: 3 March 2018].

Stardog Union 2018. Stardog. Available at: http://stardog.com/.

Stenmark, D. 2002. Information vs. knowledge: The role of intranets in knowledge management. In: *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*. IEEE., pp. 928–937.

Stevenson, A. 2011. *Oxford dictionary of English*. OUP Oxford. Available at: https://books.google.co.uk/books?id=anecAQAAQBAJ.

Succar, B. 2009. Building information modelling framework: A research and delivery foundation for industry stakeholders. *Automation in Construction* 18(3), pp. 357–375. Available at: http://dx.doi.org/10.1016/j.autcon.2008.10.003.

Terkaj, W. and Šojić, A. 2015. Ontology-based representation of IFC EXPRESS rules: An enhancement of the ifcOWL ontology. *Automation in Construction* 57, pp. 188–201. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0926580515000886.

Thalmann, D. Musse, S.R. and Braun, A. 2007. Crowd simulation. Available at: http://link.springer.com/content/pdf/10.1007/978-3-319-08234-9_69-1.pdf.

The Building Regulations 2015. *Approved Document B (Fire Safety)*. Available at: http://gov.wales/topics/planning/buildingregs/approved-documents/part-b-fire/?lang=en.

The Eclipse Foundation 2018. Jetty Server. Available at: https://www.eclipse.org/jetty/ [Accessed: 12 February 2018].

Tomašević, N.M. Batić, M.Č. Blanes, L.M. Keane, M.M. and Vraneš, S. 2015. Ontology-based facility data model for energy management. *Advanced Engineering Informatics* 29(4), pp. 971–984. Available at: http://linkinghub.elsevier.com/retrieve/pii/S1474034615000981.

Trento, A. Fioravanti, A. and Simeone, D. 2012. Building-Use Knowledge Representation for Architectural Design. In: *Proceedings of eCAADe 2012*., pp. 683–690.

Trento, A. and Fioravanti, A. 2016. Human Behaviour Simulation to Enhance Workspace Wellbeing and Productivity A BIM and Ontologies implementation path. *Proceedings of eCAADe 2016* 2, pp. 315–325.

Ullrich, M. 2003. SemanticMiner – Ontology-Based Knowledge Retrieval 1. *Computer* 9(7), pp. 682–696. doi: 10.3217/jucs-009-07-0682 M4 - Citavi.

Venugopal, M. Eastman, C.M. and Teizer, J. 2015. An ontology-based analysis of the industry foundation class schema for building information model exchanges. *Advanced Engineering Informatics* 29(4), pp. 940–957. Available at: http://dx.doi.org/10.1016/j.aei.2015.09.006.

Vieira, R. de Paiva, D.C. and Musse, S.R. 2005. Ontology-based crowd simulation for normal life situations. In: *Computer Graphics International 2005*. IEEE., pp. 221–226.

Wang, B. Li, H. and Rezgui, Y. 2013. Intelligent building emergency management using building information modelling and game engine. *ICIC Express Letters* 7(3), pp. 1017–1023. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-84875355141&partnerID=40&md5=5225bffc88541f11317c888e9466ff77.

Wang, B. Li, H. Rezgui, Y. Bradley, A. and Ong, H.N. 2014. BIM Based Virtual Environment for Fire Emergency Evacuation. *The Scientific World Journal* 2014, pp. 1–22. Available at: http://www.hindawi.com/journals/tswj/2014/589016/.

Wang, S.-H. Wang, W.-C. Wang, K.-C. and Shih, S.-Y. 2015. Applying building information modeling to support fire safety management. *Automation in Construction* 59, pp. 158–167. Available at: http://www.sciencedirect.com/science/article/pii/S0926580515000205%5Cnhttp://linkinghub.elsevier.com/retrieve/pii/S0926580515000205.

Wang, S. and Wainer, G. 2015. A simulation as a service methodology with application for crowd modeling, simulation and visualization. *SIMULATION* 91(1), pp. 71–95. Available at: http://sim.sagepub.com/cgi/doi/10.1177/0037549714562994.

Yao, Y. Zeng, Y. Zhong, N. and Huang, X. 2007. Knowledge Retrieval (KR). *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, WI 2007* , pp. 729–735. doi: 10.1109/WI.2007.4427181.

Yoke, M. Low, H. Cai, W. and Zhou, S. 2007. A federated agent-based crowd simulation architecture. In *The 2007 European Conference on Modelling and Simulation, Prague, Czech Republic* (pp. 188-194).

Yu, L. 2014. *A Developer's Guide to the Semantic Web*. Berlin, Heidelberg: Springer Berlin Heidelberg. Available at: http://link.springer.com/10.1007/978-3-662-43796-4.

Zanni, M.A. Soetanto, R. Ruikar, K. 2017. Towards a BIM-enabled sustainable building design process: roles, responsibilities, and requirements. *Architectural Engineering and Design Management* 13(2), pp. 101–129. doi: 10.1080/17452007.2016.1213153.

Zhan, B. Monekosso, D.N. Remagnino, P. Velastin, S.A. and Xu, L.-Q. 2008. Crowd analysis: a survey. *Machine Vision and Applications* 19(5–6), pp. 345–357. Available at: http://link.springer.com/10.1007/s00138-008-0132-4.

Zhang, C. Beetz, J. and de Vries, B. 2013. Towards Model View Definition on Semantic Level : A State of the Art Review. In *Proceedings of the 20th International Workshop: Intelligent Computing in Engineering*. (July 2015), pp. 1–10.

Zhang, S. Boukamp, F. and Teizer, J. 2015. Ontology-based semantic modeling of construction safety knowledge: Towards automated safety planning for job hazard analysis (JHA). *Automation in Construction* 52, pp. 29–41. Available at: http://dx.doi.org/10.1016/j.autcon.2015.02.005.

Zheng, X. Zhong, T. and Liu, M. 2009. Modeling crowd evacuation of a building based on seven methodological approaches. *Building and Environment* 44(3), pp. 437–445. doi: 10.1016/j.buildenv.2008.04.002.

Zhou, P. and El-Gohary, N. 2017. Ontology-based automated information extraction from building energy conservation codes. *Automation in Construction* 74, pp. 103–117. Available at: http://dx.doi.org/10.1016/j.autcon.2016.09.004.

Zhou, S. Chen, D. Cai, W. Luo, L. Low, M.Y.H. Tian, F. Tay, V.-S.H Ong, D.W.S. and Hamilton, B.D. 2010. Crowd modeling and simulation technologies. *ACM Transactions on Modeling and Computer Simulation* 20(4), pp. 1–35. Available at: http://eprints.bournemouth.ac.uk/13285/1/licence.txt%5Cnhttp://portal.acm.org/citation.cfm?doid=1842722.1842725.

Appendix A – Simulation tools concepts

For each CST, certain concepts require specific input from the user side which define the behaviour of the object. E.g. An agent is programmed to evacuate a certain route, as instructed by the user input. The concepts which require 'Behaviour Input' are marked with an 'x' in the last column – BI.

Table A-1. MassMotion Concepts

No	Concept	Function	Notes	Feature category	BI
SCENE OBJECTS		Features which define the building environment.			
1	Bank	Contains connection objects within the scene.	Links, Ramps, Stairs are connection objects.	Visualisation Interface	
2	Collection	Contains any type of scene objects.	Cannot contain other Collections.	Visualisation Interface	
3	Perimeter	Reduces the number of available routes for agents in the scene.		Geometry	x
4	Zone	Define a conceptual area within the simulation.		Interface Geometry	
5	Transform	Contains multiple objects and allows their geometry to be altered collectively.		Interface Geometry	
6	Barrier	Represents regions which blocks agents' paths.		Geometry	
7	Cordon	Defines a surface which is used to track agent movement.	Does not impact simulation calculations.	Geometry Analysis	
8	Volume	Defines a volume which is used to track agents.	Does not impact simulation calculations.	Geometry Analysis	
9	Escalator	Model real-life escalators.		Geometry	
10	Floor	Models surfaces which allows agents to walk on.		Geometry	
11	Link	Connects two floors together.		Geometry	
12	Path	Guides agents on a well-defined curve.		Geometry	x
13	Portal	Allows agents to enter or exit a simulation.		Geometry	
14	Ramp	Represents inclined surfaces.	Add a vertical cost to agent movement.	Geometry	
15	Stair	Connects two floors of different elevations.		Geometry	

No	Concept	Function	Notes	Feature category	BI
16	Server	Model queuing behaviour of agents in more complex circumstances.		Geometry	x
17	Dispatch	Distributes agents across Server inputs.		Geometry	x
18	Reference Geometry	Includes geometry objects as generic models, with no specified purpose.	Does not impact simulation calculations.	Visualisation Geometry	
19	Visual	Enhances the appearance of the environment.	Does not impact simulation calculations.	Visualisation Geometry	
ACTIVITIES		Features which define events during a simulation.			
20	Action	Operation to be taken by agent.		Event Agent	x
21	Test	Operate a check on a single agent.		Event Interface Agent	
22	Avatar	Models the 3D representation of agents within the simulation.	Does not impact simulation calculations.	Visualisation Geometry Agent	
23	Circulate	Defines a specified route over time for agents to follow.	Agents move from one portal to another.	Event	x
24	Evacuate	Triggers when agents should begin evacuating the simulation.		Event	x
25	Broadcast	Applies a specified Action to agents within the simulation.		Event	x
26	Journey	Represent the act of people moving from A to B.	A journey may have multiple exits.	Event	
27	Gate Access	Opens of closes an access point objects dynamically.		Event	x
28	Profile	Defines the physical characteristics of Agent.		Agent	
29	Server Access	Controls ingress and egress to/from Server objects.		Event	x
30	Time	Reference to a specific point in time while the simulation is running.		Interface	
31	Timetable	Allows for more complex and coordinated definition of events.	Suitable for train, bus, plane schedules.	Event	x
32	Token	Allows access to agents for certain entry points.		Agent	
33	Trigger	Fires in response to certain conditions during the simulation.		Event	x

No	Concept	Function	Notes	Feature category	BI
34	Trip Matrix	Creates agents across multiple portals and distributes their exit through others.		Event	x
35	Vehicle	Simulates the controlled arrival and departure of agents at periodic intervals.		Event	
ANALYSIS		Features which allow users to analyse the outputs.			
36	Simulation Run	Represents a single iteration of a MassMotion simulation.	Start and end times must be specified.	Event Visualisation	
37	Agent Filter	Allows the selection of specific agent groups for analysis.		Analysis Visualisation	
38	Trip	Defines a particular route through the environment.	Consists of multiple areas, cordons, portals.	Analysis Event	
39	Graph	Queries and presents data for user analysis.		Analysis Visualisation	
40	Map	Adds colour coded contours on the environment to visualise events and behaviour.		Analysis Visualisation	
41	Tables	Queries and presents data in tabular form from the simulation ran.		Analysis	
42	Area	Conceptual surfaces which can be used for analysis.	Can include Volume, Collection, Zone, etc.	Analysis Visualisation	

Table A-2. Pedestrian Dynamics concepts

No	Entity	Function	Notes	Feature category	BI
DRAW ELEMENTS		Features which define the building environment.			
1	Height Layer	Defines a coplanar area where agents can walk.		Geometry	
2	Obstacle	Defines an area on which agents cannot walk.	It's usually placed within a Height Layer	Geometry	
3	Walkable area	Represents an area within a Height Layer where agents can walk.	Multiple areas can overlap.	Geometry	
4	Opening	Walking area created on top of an obstacle.	Allows agent to pass through otherwise un-walkable obstacles	Geometry	
5	Transfer	Connects two different Height Layers.		Geometry	
6	Stair	Models height between levels, and allows agent to cross them.	Automatically acts as a Transfer object.	Geometry	
7	Spiral Stair	Similar to Stair.		Geometry	
8	Escalator/Moving Walk	Models a slow moving conveyor belt which transports agents.	Agents can stand still on it. Acts as a Transfer object.	Geometry	
9	Passageway	Models a walkable area where agent flow direction I controlled.		Geometry	
ACTIVITY LOCATIONS		Features which define events at specific locations.			
10	Entry/Exit	Allows agents to enter or exit the model.	A single object can be both an exit and an entry.	Geometry	
11	Waiting	Forces agent to wait for specified/unspecified time.		Geometry	x
12	Waypoint	Acts as a destination point for agents when circulating.		Geometry	x
13	Commercial Facility	Models agents performing shopping activities.		Geometry	x
14	Service Facility	Models a location where agents receive a certain kind of service.		Geometry	x
15	Ticker Facility	Models a location where agent buy tickets for another activity.		Geometry	x
16	Access Control	Models an area which agents have restricted access to.		Geometry	x
ACTIONS		Features which define events during a simulation.			
17	Action Timer	Triggers time based actions during a simulation run.		Event	x
18	Action Area	Alters the behaviour or properties of certain agents.		Event	x

No	Entity	Function	Notes	Feature category	BI
19	Flow Shifter	Shifts the flow of an area.		Event	x
20	Flow Splitter	Splits certain flows of agents.		Event Geometry	x
21	Indicative Corridor	Forces agents to follow a specified route or queue for specific action.		Event	
22	Route	Models agent behaviour on moving from A to B		Event	x
23	Local Obstacle	Influences agents on which side to avoid an obstacle.		Geometry Mathematical	
SPECIAL ELEMENTS		Other features, specific to this tool.			
24	Stand	Models the physical infrastructure for a stand.	Stand Stairs and Stand Portals can be added to it.	Geometry	
25	Stand Stair	Make the rows on Stands reachable to agents.		Geometry	
26	Stand Portal	Connects a Stand to a Height Layer.		Geometry	
27	Stand Section	Groups Stand seats together.		Interface	
28	Stand Obstacle	Makes areas within a Stand which agents cannot walk on.		Geometry	
29	Transport Non Waiting Area	Defines an area where agents are forbidden from stopping.		Geometry	
30	Transport Network	Defines stop locations for transport elements.	Models rail tracks, bus stations, etc.	Event	
AGENT INPUT		Features which allow users to define agents.			
31	Agent Profile	Assigns a profile to a group of agents with similar characteristics.		Agent	
32	Route	Defines the path agents take through the model.		Agent	x
33	Agent 3D Model	Represents each agent with a 3D model.		Visualisation	
34	Agent Activities	Defines which Activity Locations an Agent can visit over time.		Agent	x
35	Agent Generator	Defines a number of agents of specific types when to appear.	Route can be specified.	Event	x
36	Arrival List	Defines a creation schedule of Agents.	Route can be specified.	Event	x
OUTPUT ELEMENTS		Features which allow users to analyse the outputs.			
37	Flow Counter	A line which counts agents passing over it.		Analysis	
38	Density Area	An area which overlays densities of agent traffic.		Analysis Visualisation	

No	Entity	Function	Notes	Feature category	BI
39	Output Layer	A layer which contains Output Elements.		Analysis	
40	Activity Route	A list of activities which were carried out by agents.		Analysis	
41	Density Map	Maps the density for the environment or agents over time.		Analysis Visualisation	
42	Frequency Map	Shows the number of agents passing through at certain times.		Analysis	
43	Travel Time Map	Shows the travel time of each agent using colour codes.		Analysis/ Visualisation	

Table A-3. STEPS concepts

No	Entity	Function	Notes	Feature category	BI
MAIN OBJECTS		Features which create the most important objects on the model.			
1	Distribution	Distributes values within a range.		Mathematical	
2	Curve	Specifies a relationship between two parameters.		Mathematical	
3	Colour Distribution	Distributes values within a range for colours.		Mathematical	
4	People Model	Defines the appearance of agents.		Agent	
5	People Type	Defines the properties of agents.		Agent	
6	People Group	Defines a set of agents at a given time or place.		Agent	
7	Family	Represents a group of agents which travel together.		Agent	
8	Shape	Represents a complex geometric construct from basic lines.		Geometry	
9	Mesh	Represents surfaces made from triangular shapes.	Can represent 2D and 3D objects.	Geometry	
10	Plane	Defines a surface which is walkable to the agents.		Geometry	
11	Path	Models stairs or unidirectional flows.	Superseded by Shaped Planes.	Geometry	
12	Plane Exit	Defines an exit point out of a plane for agents to use.	Links two planes.	Geometry	
13	Internal Door	Limits or alters the flow of people on one Plane.	Does not link Planes.	Geometry	
14	Checkpoint	Defines a location which agent can move towards.	Only functions in operation modes.	Geometry	
15	Route	Restricts movement of people via a defined path.	Only functions in operation modes.	Event Geometry	x
16	Matrix	Specifies agent destinations according to their origins.	Superseded by Route.	Event	x
17	Access	Specifies which agent can use which exit point within a plane.		Event	x
18	Blockage	Defines surfaces on Planes which agents cannot walk on.	Can be represented by points, lines, shapes, meshes.	Geometry	
19	Location	Specifies a smaller region on a Plane which is used to create agents.	Can be used to plot density maps.	Geometry	
20	Junction	Specifies the way in which agents can split between more Paths.		Mathematical	
21	Item	Geometry object used to enhance appearance.	Is not involved in model calculations.	Visualisation Geometry	

No	Entity	Function	Notes	Feature category	BI
22	Group	Groups different types of objects together.	Can be used for statistical analysis across objects.	Interface	
VEHICLE OBJECTS		Features which define moving vehicle objects during a simulation.			
23	Lift	Carries agents from one surface to another.	An object which moves across the environment.	Geometry	x
24	Train	Represents trains geometrically.	Is not involved in model calculations. Superseded by Vehicle	Geometry	
25	Vehicle	Dynamic surface which can transport agents across the environment.	Is made out of several components.	Geometry	x
26	Vehicle Model	Represents vehicles geometrically.		Geometry	
27	Vehicle Element Type	Represents a specific element as part of a Vehicle object.	Can be used to resemble doors, areas, etc.	Geometry	
28	Trajectory	Defines the path of a vehicle.		Event	x
EVENT OBJECTS		Features which define events during a simulation.			
29	Simulation Event	Specifies an interval of time for events to occur.		Event	x
30	Emergency Event	Triggers the act which simulates an evacuation at given times.		Event	x
31	People Event	Creates groups of agents within the model at given times.		Event	x
32	Population Event	Maintains a constant number of population flow on a given Plane.		Event	x
33	Exit Event	Dynamically alters the state of an Exit Plane as opened or closed to agents.		Event	x
34	Blockage Event	Dynamically alters the state of a surface to walkable or un-walkable by agents.		Event	x
35	Group Event	Changes the colour of grouped elements depending on their states.	Is purely for visualisation purposes	Visualisation	
36	Lift Event	Specifies the behaviour of Lift objects during a simulation.		Event	
37	Viewpoint Event	Moves the view of the model to a specific points at given times.		Visualisation	
38	Viewpath Event	Triggers a Viewpath object at a specified time during a simulation.		Visualisation	
39	Snapshot Event	Takes an image shot of the view at a given time.		Visualisation	

No	Entity	Function	Notes	Feature category	BI
40	Movie Event	Triggers the recording of a simulation between time intervals.		Visualisation	
41	Clipping Plane Event	Triggers Clipping Plane objects at given times during a simulation.		Visualisation	
42	Tracking Event	Records the position of agents over specified time interval during simulation.		Visualisation	
SMOKE OBJECTS		Features which model smoke from fire.			
43	Surface	Represents smoke propagation visually.	Does not affect model calculations.	Visualisation	
44	Sample Plane	Modifies walking speeds of agents according to smoke concentration data.	Uses imported data from other software.	Mathematical Geometry	
45	Dose	Keeps track of the level of gas absorption by agents.		Analysis	
OUTPUT OBJECTS		Features which allow users to analyse the outputs.			
46	Basic Variable	Expresses certain statistical or raw data values about the simulation.		Analysis	
47	Expressions	Uses several Basic Variables to output specific data.		Analysis	
48	Variable	Monitors (saves or outputs) specified Basic Variables.		Analysis	
49	Condition	Reports specified condition status during a running simulation.		Analysis	
50	Display	Outputs data on screen during a simulation.		Analysis Interface	
51	Alert	Displays on screen a message when Conditon is met.		Analysis Interface	
52	Label	Labels model objects on view.		Interface	
53	Output	Specifies which data to be saved as output into results files.		Interface	
54	Output Map	Calculates and displays a contour of the events for analysis.	Can only output once simulation has finished.	Analysis Visualisation	
55	Scale	Sets up the colour codes to display over a range of result values.	Used for colouring Output Map.	Analysis Interface	
NAVIGATION OBJECTS		Features used for visualising the simulation.			
56	Viewpoint	Saves a specific view of the model.		Visualisation	
57	Viewpath	Connects Viewpoint objects in a sequence over time.		Visualisation	

No	Entity	Function	Notes	Feature category	BI
RENDERING OBJECTS		Features used for improving the visual aspect of the simulation.			
58	Clipping Plane	Hides or shows parts of the model.		Visualisation Interface	
59	Light	Adds lighting effects across the model for more realistic views.		Visualisation	
60	Material	Adds a material effect to model geometry.		Visualisation	
61	Texture Map	Adds images to 3D objects for more realistic representations.		Visualisation	

Table A-4. BuildingEXODUS concepts

No	Entity	Function	Notes	Feature category	BI
GEOMETRY		Features which define the building environment.			
1	Node	Basic unit used for constructing geometry in Exodus.		Geometry	
2	Free Space Node	Models free space.		Geometry	
3	Boundary Node	Models free space but alters on agent walking speed.	I created for nodes near boundary lines.	Geometry	
4	Seat Node	Models seats/chairs.		Geometry	
5	Stair Node	Models one lane on one riser of a staircase.		Geometry	
6	Discharge Node	Manipulates agent's movement near Internal Exits.	Used in conjunction with Attractor Node.	Geometry	
7	Attractor Node	Manipulates agent's movement near Internal Exits.	Used in conjunction with Discharge Node.	Geometry	
8	Landing	Models free space on staircase landings.		Geometry	
9	Census Region	Enables extraction of individual flow data.	Does not impact agent behaviour.	Geometry Analysis	
10	Internal Exit	Models an exit location within the environment.	Represents doors within a department	Geometry	
11	Source Node	Generates agents throughout the simulation.		Geometry	
12	Redirection Node	Models decision nodes for circulating agents.	Is used for agent itineraries.	Geometry	
13	Direction Node	Controls the direction of an agent's movement.		Geometry	x
14	External Exit Node	Ultimate exit point out of the simulation environment.		Geometry	
15	Transit Node	Represents more comprehensively lifts, staircases, corridors, etc.		Geometry	
16	Stair Transit Node	Models stairs.		Geometry	
17	Escalator Transit Node	Models escalators,		Geometry	
18	Lift Shaft Opening Transit Node	Models opening areas around lift shafts.		Geometry	
19	Corridor Transit Node	Models connection between horizontal spaces.	Has no vertical component.	Geometry	
20	Travelator Transit Node	Models a travelator with a specified direction.		Geometry	
21	Metered Gate Transit Node	Models metered barriers or ticked machines.		Geometry	

No	Entity	Function	Notes	Feature category	BI
22	Arc	Links node together, enabling agent movement.	Restricts flow of agents.	Geometry	
23	Line	Represent linear geometry objects, usually boundaries around free space.		Geometry	
24	Census Line	Monitors flow of agents over a given point.		Geometry Analysis	
25	Polygon	Represents more complex shapes formed from Lines.		Geometry	
26	Text Label	Label objects over the view.		Interface	
27	Sign	Represents real-life signage within the environment, which guide the agents.	Alters agent behaviour.	Geometry	x
28	Primary Link	Used to link floors.	Each Primary is assigned a Secondary Link.	Geometry	
29	Secondary Link	Used to link floors.	Each Secondary is assigned a Primary Link.	Geometry	
30	Ruler	Used to measure model geometry.		Interface	
31	Floor	Represents a floor/level within the environment.	Not clear if is set up in specific object.	Geometry Interface	
POPULATION		Features which define the agents within the simulation.			
32	Person/Occupant	Collection of attribute used to describe a person.		Agent	
33	Group	Collection of Persons/Occupants.		Agent	
34	Sub-Population	Collection of Groups		Agent	
35	Population	Collection of Sub-Populations, making up all the agents.		Agent	
36	Occupant Itinerary List	Defines a pre-defined list of tasks for Populations.		Agent Event	x
37	Occupant Exit Knowledge	Defines how many exits are known to agents.		Agent	x
38	Local Familiarity	Defines an agent's level of familiarity to the environment.		Agent	
39	Attribute	Defines specific values which model agent attributes.	Age, sex, speed, etc.	Agent	
40	Range	Defines a range of values.		Mathematical	
41	Distribution Curves	Distributes values within a range.		Mathematical	
SCENARIO		Features which describe a fire scenario.			
42	Hazzard	Models smoke, heat, irritation and toxic gases.		Event	x

No	Entity	Function	Notes	Feature category	BI
				Geometry	
43	Hazzard Evolution	Models the evolution of hazzards over time.		Event	x
44	Zone	Represents specific areas within the model made from multiple nodes.		Geometry	
45	Fire Scenario	Link Hazzards to Zones over time.		Event	x
46	Response Zone	A Zone which alters the behaviour of agents.	Can be used to delay or trigger agent evacuation.	Event Geometry	x
47	Response Time	Specifies the time when agents evacuate for a Response Zone.		Event	
48	Compartment Zone	A collection of nodes which acts as a separate compartment.		Geometry Analysis	
49	Obstacle Zone	Dynamically alters the walkable environment for agents.	Is defined by specific times.	Event Geometry	x
50	Exit Attractiveness	Models how likely an exit is chosen by agents.		Mathematical	
SIMULATION		Features which enable the animation and visualisation of a simulation.			
51	Simulation	Models a specifically set scenarios to run for calculation.		Event Visualisation	
52	Graph	Plots simulation data for user interpretation.		Analysis Visualisation	
53	Contour	Visualises in colour codes simulation data, over simulation geometry		Analysis Visualisation	
54	Zone Contour	Visualises in colour codes smoke data.		Analysis Visualisation	
55	Interrogation Objects	Allows the users to retrieve data in relation to geometry, agents or other events over simulation time.	Not clear if is set up in specific objects.	Analysis	

Table A-5. Simulex concepts

No	Entity	Function	Notes	Feature category	Behaviour input
BUILDING		Features which define the building environment.			
1	Floor	Represents the walkable area over which agents can travel on.	It is defined by the walls surrounding it, where the wall lines represent the limits.	Geometry	
2	Staircase	Allows agents to travel across from floor to floor.		Geometry	
3	Link	Connects a Floor to a Staircase		Geometry	
4	Exit	Final exit point for the building		Geometry	
5	DistMap	Overlays a color-coded mesh showing the distance from any point to nearest exit.		Analysis	
ROUTES		Features which specify how agents should evacuate the building.			
6	Most Remote	Agents take the highest distance route out of the model.		Event	
7	Test Position	Places an agent on the specified location and simulates it walk out to nearest exit.		Event	
8	Stop Testing	Stops all routes testing procedures.		Event	
PEOPLE		Features which define the building population.			
9	Person	Represents an individual agent within the model.		Agent	
10	Group	Defines a group of agents, with a number and a concentration over an area.		Agent	
11	Characteristics	Defines the physical properties of agents.		Agent	
SIMULATE		Features for running and visualising the simulation			
12	Being	Start the simulation process.		Visualisation	
13	Playback	Playback a simulation process.		Visualisation	

No	Entity	Function	Notes	Feature category	Behaviour input
14	Playback 3D	Playback a simulation process in 3D.		Visualisation	
15	Pause/Stop	Stops a simulation run.		Visualisation	

Table A-6. Summary of common CST concepts from previous tables

Tool	Geometry	Agent	Event	Analysis	Visualisation	Interface	Mathematical
MassMotion	13	2	13	6	5	3	0
Pedestrian Dynamics	22	3	9	7	1	1	0
STEPS	14	4	11	9	15	3	5
Exodus	31	8	7	4	0	2	3
Simulex	4	3	3	1	4	0	0

Appendix B – Developed ontologies

This section outlines the developed ontologies using diagrams and shows the ranges of object and data properties as they appear in the Protégé software.

Additionally, is shows the alignment of concepts between certain ontologies, and other resources which were used to define the ontologies as a reference.

Crowd Simulation Scenario (CSS) ontology

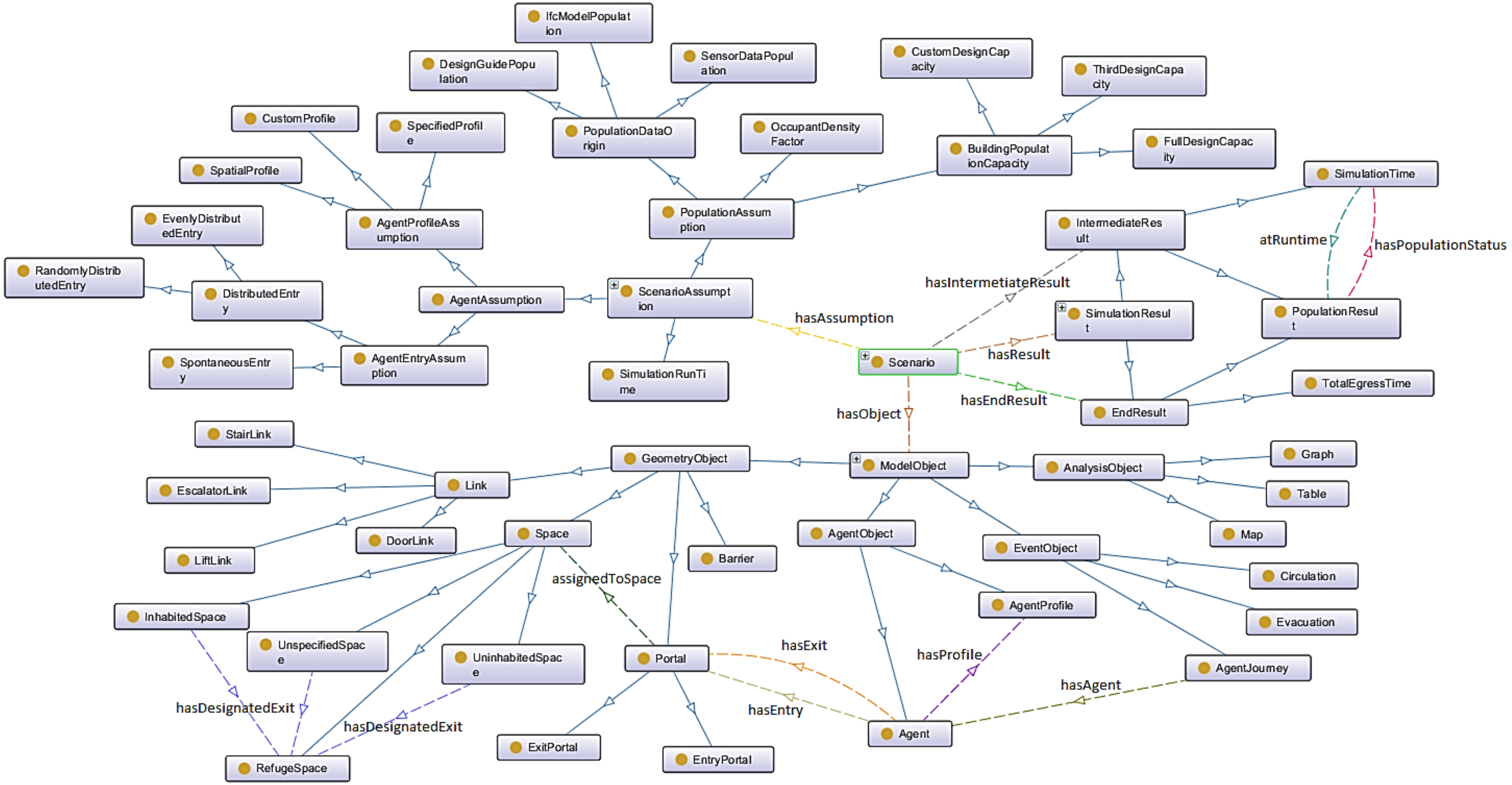
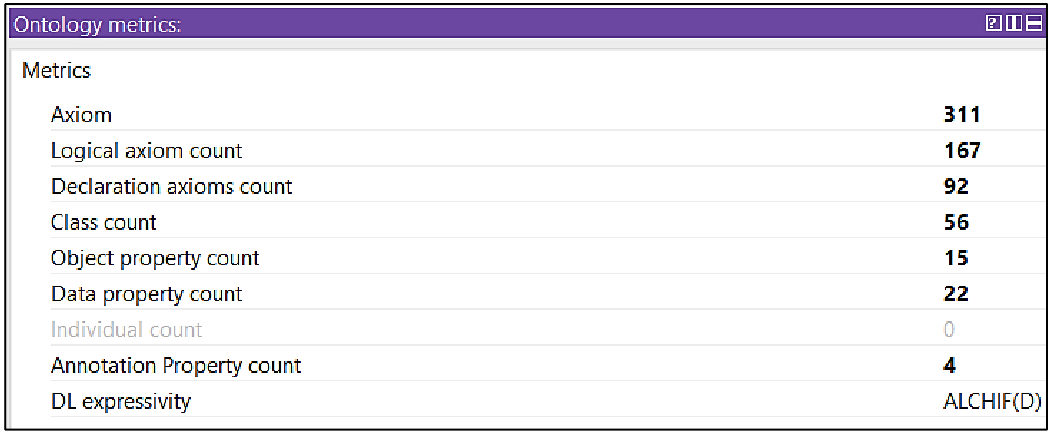


Figure B-1. CSS ontology classes and object properties connecting them

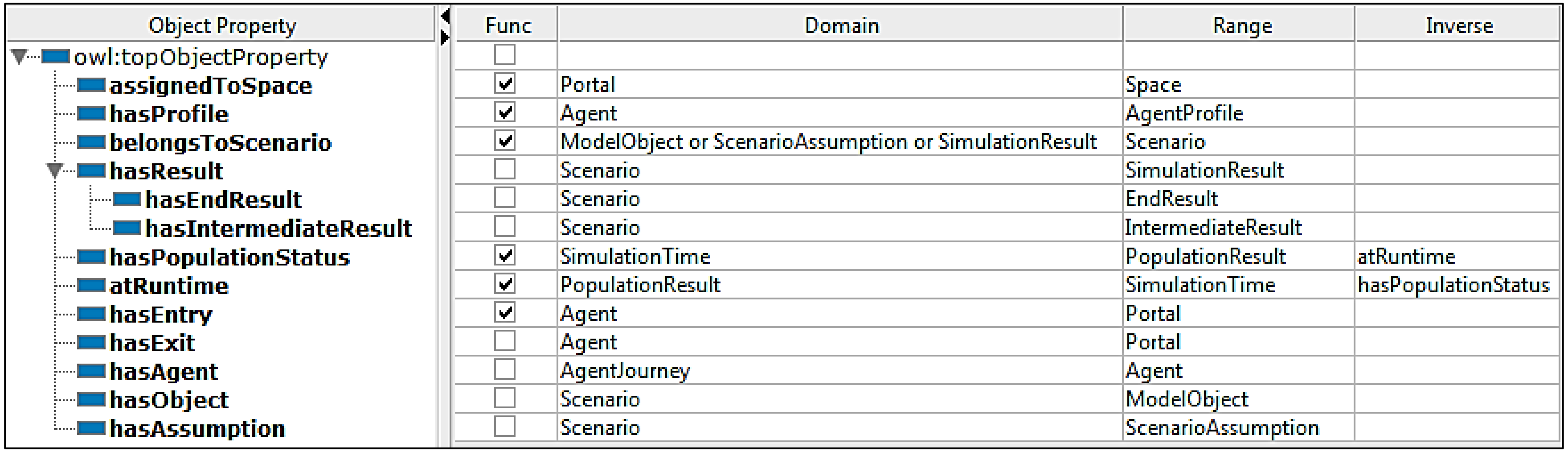


Ontology metrics:

Metrics

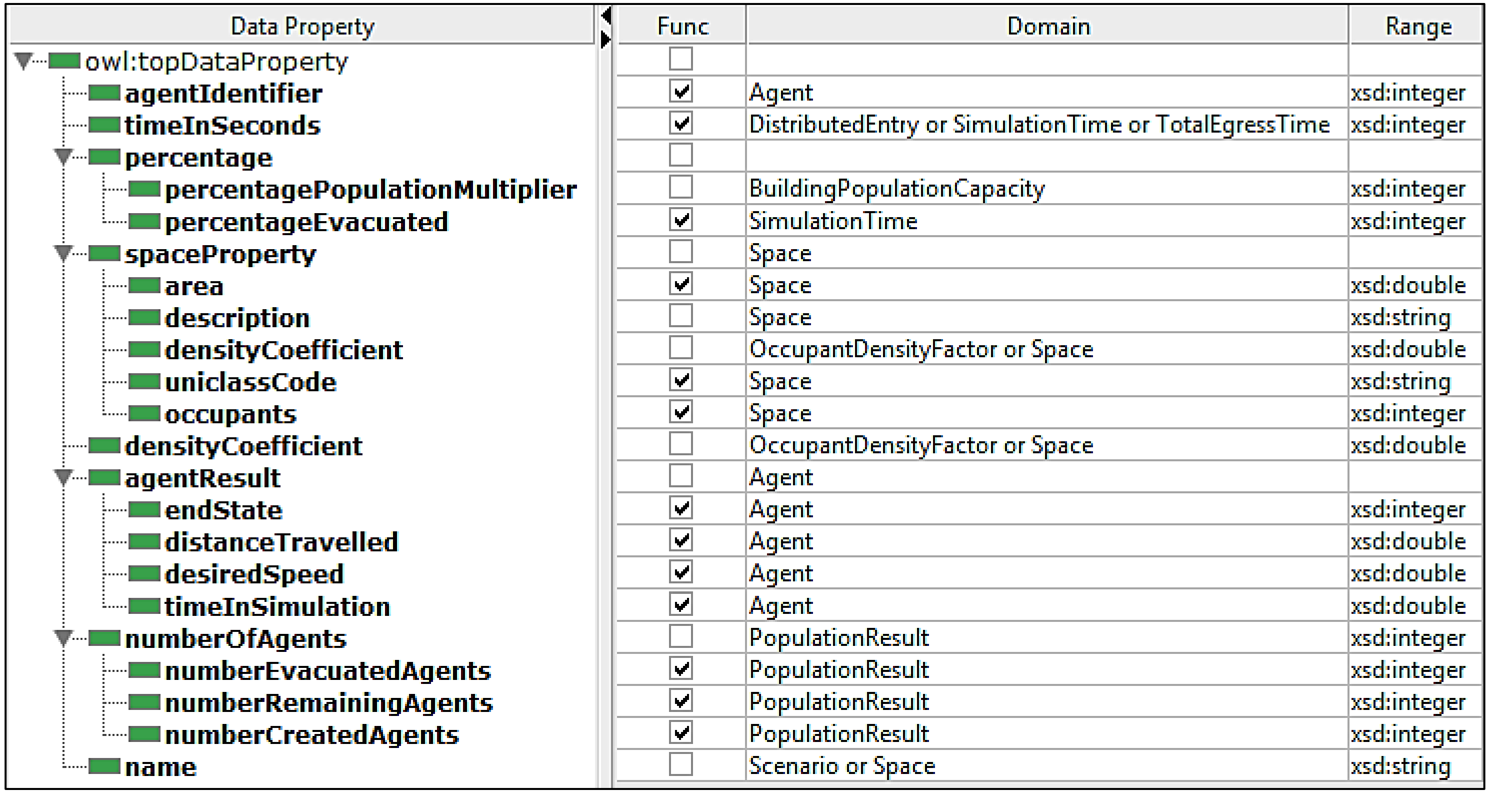
Axiom	**311**
Logical axiom count	**167**
Declaration axioms count	**92**
Class count	**56**
Object property count	**15**
Data property count	**22**
Individual count	0
Annotation Property count	**4**
DL expressivity	ALCHIF(D)

Figure B-2. CSS ontology metrics



Object Property	Func	Domain	Range	Inverse
owl:topObjectProperty	□			
assignedToSpace	☑	Portal	Space	
hasProfile	☑	Agent	AgentProfile	
belongsToScenario	☑	ModelObject or ScenarioAssumption or SimulationResult	Scenario	
hasResult	□	Scenario	SimulationResult	
hasEndResult	□	Scenario	EndResult	
hasIntermediateResult	□	Scenario	IntermediateResult	
hasPopulationStatus	☑	SimulationTime	PopulationResult	atRuntime
atRuntime	☑	PopulationResult	SimulationTime	hasPopulationStatus
hasEntry	☑	Agent	Portal	
hasExit	□	Agent	Portal	
hasAgent	□	AgentJourney	Agent	
hasObject	□	Scenario	ModelObject	
hasAssumption	□	Scenario	ScenarioAssumption	

Figure B-3. CSS ontology object properties



Data Property	Func	Domain	Range
owl:topDataProperty	□		
agentIdentifier	☑	Agent	xsd:integer
timeInSeconds	☑	DistributedEntry or SimulationTime or TotalEgressTime	xsd:integer
percentage	□		
percentagePopulationMultiplier	□	BuildingPopulationCapacity	xsd:integer
percentageEvacuated	☑	SimulationTime	xsd:integer
spaceProperty	□	Space	
area	☑	Space	xsd:double
description	□	Space	xsd:string
densityCoefficient	□	OccupantDensityFactor or Space	xsd:double
uniclassCode	☑	Space	xsd:string
occupants	☑	Space	xsd:integer
densityCoefficient	□	OccupantDensityFactor or Space	xsd:double
agentResult	□	Agent	
endState	☑	Agent	xsd:integer
distanceTravelled	☑	Agent	xsd:double
desiredSpeed	☑	Agent	xsd:double
timeInSimulation	☑	Agent	xsd:double
numberOfAgents	□	PopulationResult	xsd:integer
numberEvacuatedAgents	☑	PopulationResult	xsd:integer
numberRemainingAgents	☑	PopulationResult	xsd:integer
numberCreatedAgents	☑	PopulationResult	xsd:integer
name	□	Scenario or Space	xsd:string

Figure B-4. CSS ontology data properties

Feedback Analysis (FBA) ontology

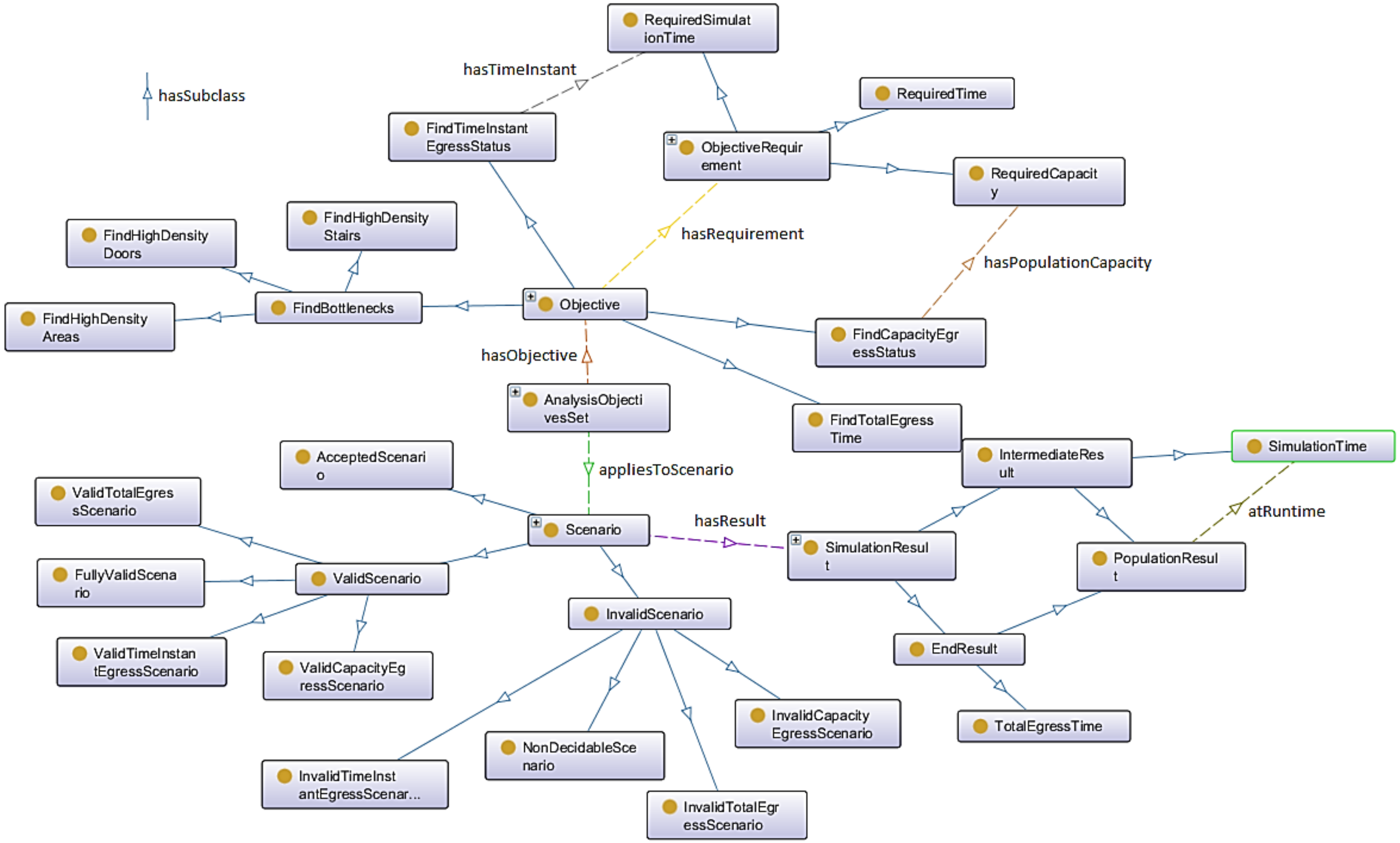
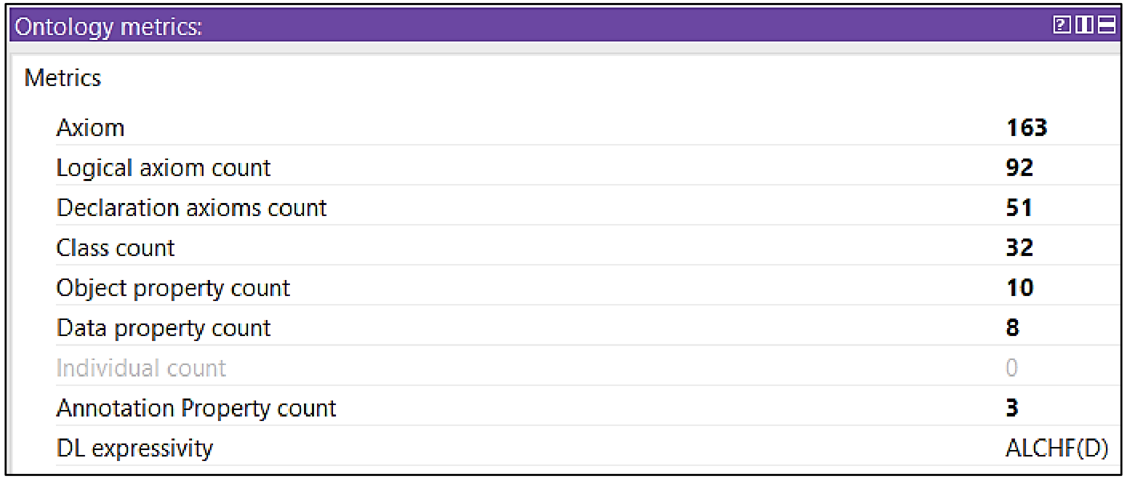


Figure B-5. FBA ontology classes and object properties connecting them

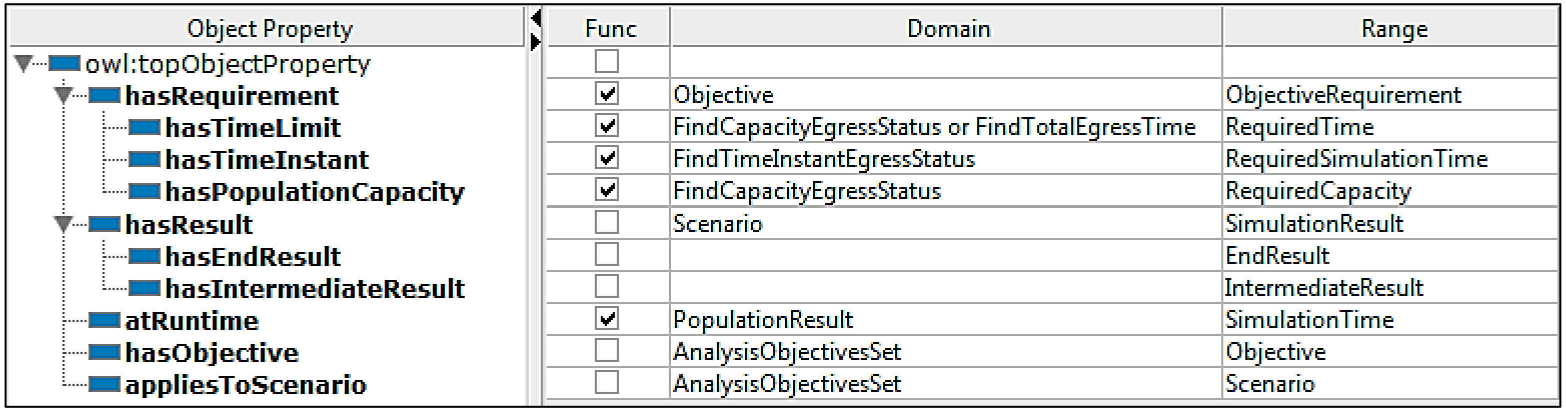


Ontology metrics:

Metrics

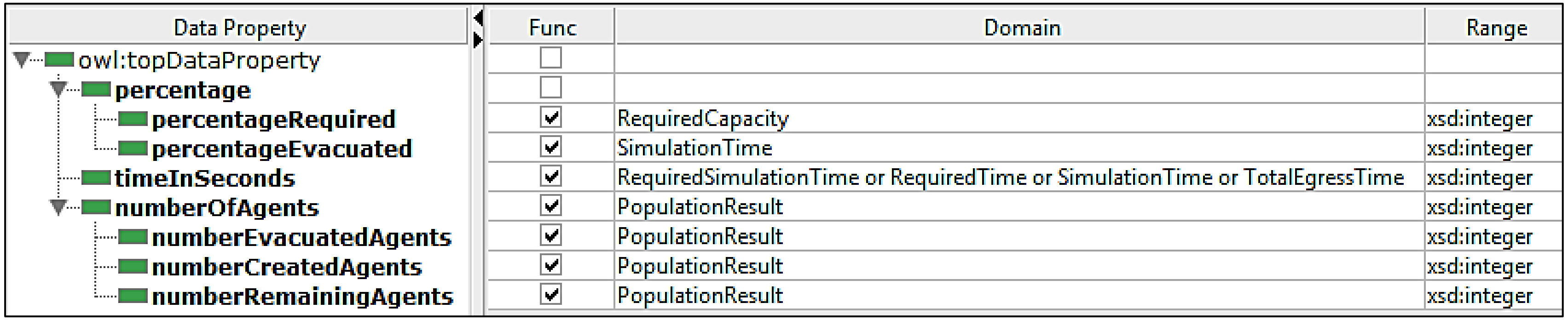
Axiom	**163**
Logical axiom count	**92**
Declaration axioms count	**51**
Class count	**32**
Object property count	**10**
Data property count	**8**
Individual count	0
Annotation Property count	**3**
DL expressivity	ALCHF(D)

Figure B-6. FBA ontology metrics



Object Property	Func	Domain	Range
owl:topObjectProperty	☐		
hasRequirement	☑	Objective	ObjectiveRequirement
hasTimeLimit	☑	FindCapacityEgressStatus or FindTotalEgressTime	RequiredTime
hasTimeInstant	☑	FindTimeInstantEgressStatus	RequiredSimulationTime
hasPopulationCapacity	☑	FindCapacityEgressStatus	RequiredCapacity
hasResult	☐	Scenario	SimulationResult
hasEndResult	☐		EndResult
hasIntermediateResult	☐		IntermediateResult
atRuntime	☑	PopulationResult	SimulationTime
hasObjective	☐	AnalysisObjectivesSet	Objective
appliesToScenario	☐	AnalysisObjectivesSet	Scenario

Figure B-7. FBA ontology object properties



Data Property	Func	Domain	Range
owl:topDataProperty	□		
percentage	□		
percentageRequired	☑	RequiredCapacity	xsd:integer
percentageEvacuated	☑	SimulationTime	xsd:integer
timeInSeconds	☑	RequiredSimulationTime or RequiredTime or SimulationTime or TotalEgressTime	xsd:integer
numberOfAgents	☑	PopulationResult	xsd:integer
numberEvacuatedAgents	☑	PopulationResult	xsd:integer
numberCreatedAgents	☑	PopulationResult	xsd:integer
numberRemainingAgents	☑	PopulationResult	xsd:integer

Figure B-8. FBA ontology data properties

UK Spaces Occupant Capacity (UKSOC) ontology

Table B-1. Spaces occupant capacities (adapted from The Building Regulations 2015 Appendix C3 – Methods of measurement)

Table C1 Floor space factors (1)		
Type of accommodation (2)(3)		Factor (m^2/pers)
1	Standing spectator areas, bar areas (within 2m of serving point) similar refreshment areas	0.3
2	Amusement arcade, assembly hall (including a general purpose place of assembly), bingo hall, club, crush hall, dance floor or hall, venue for pop concert and similar events and bar areas without fixed seating	0.5
3	Concourse, queuing area or shopping mall (4)(5)	0.7
4	Committee room, common room, conference room, dining room, licensed betting office (public area), lounge or bar (other than in 1 above), meeting room, reading room, restaurant, staff room or waiting room (6)	1
5	Exhibition hall or studio (film, radio, television, recording)	1.5
6	Skating rink	2
7	Shop sales area (7)	2
8	Art gallery, dormitory, factory production area, museum or workshop	5
9	Office	6
10	Shop sales area (8)	7
11	Kitchen or library	7
12	Bedroom or study-bedroom	8
13	Bed-sitting room, billiards or snooker room or hall	10
14	Storage and warehousing	30
15	Car park	2/pers
Notes	1. As an alternative to using the values in the table, the floor space factor may be determined by reference to actual data taken from similar premises. Where appropriate, the data should reflect the average occupant density at a peak trading time of year.	
	2. Where accommodation is not directly covered by the descriptions given, a reasonable value based on a similar use may be selected.	
	3. Where any part of the building is to be used for more than one type of accommodation, the most onerous factor(s) should be applied. Where the building contains different types of accommodation, the occupancy of each different area should be calculated using the relevant space factor.	
	4. Refer to section 5 of BS 5588-10:1991 Code of practice for shopping complexes for detailed guidance on the calculation of occupancy in common public areas in shopping complexes.	
	5. For detailed guidance on appropriate floor space factors for concourses in sports grounds refer to “Concourses” published by the Football. Licensing Authority (ISBN: 0 95462 932 9).	
	6. Alternatively the occupant capacity may be taken as the number of fixed seats provided, if the occupants will normally be seated.	
	7. Shops excluding those under item 10, but including - supermarkets and department stores (main sales areas), shops for personal services such as hairdressing and shops for the delivery or collection of goods for cleaning, repair or other treatment or for members of the public themselves carrying out such cleaning, repair or other treatment.	
	8. Shops (excluding those in covered shopping complexes but including department stores) trading predominantly in furniture, floor coverings, cycles, prams, large domestic appliances or other bulky goods, or trading on a wholesale self-selection basis (cash and carry).	

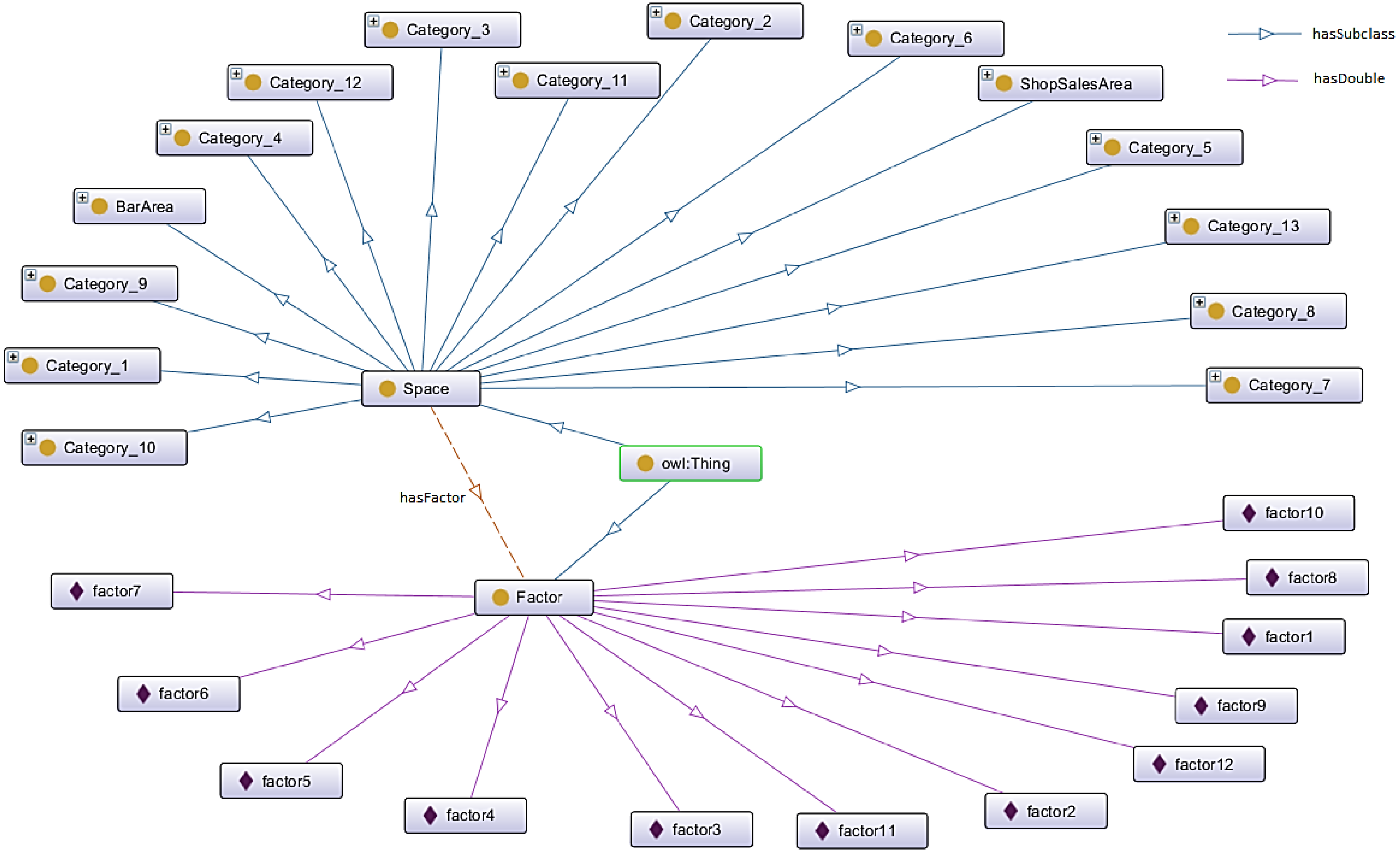


Figure B-9. UKSOC ontology with main classes and individuals

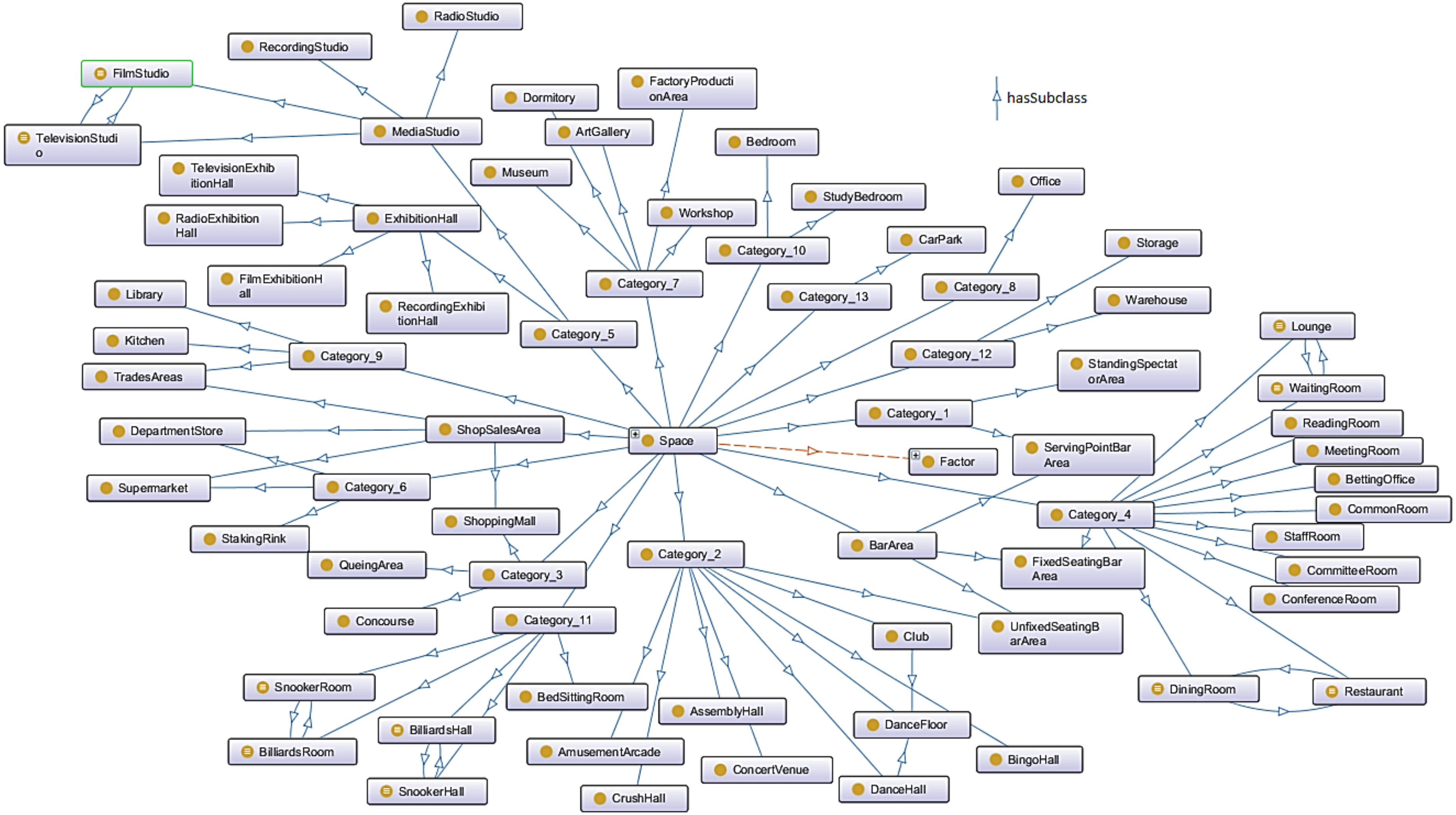
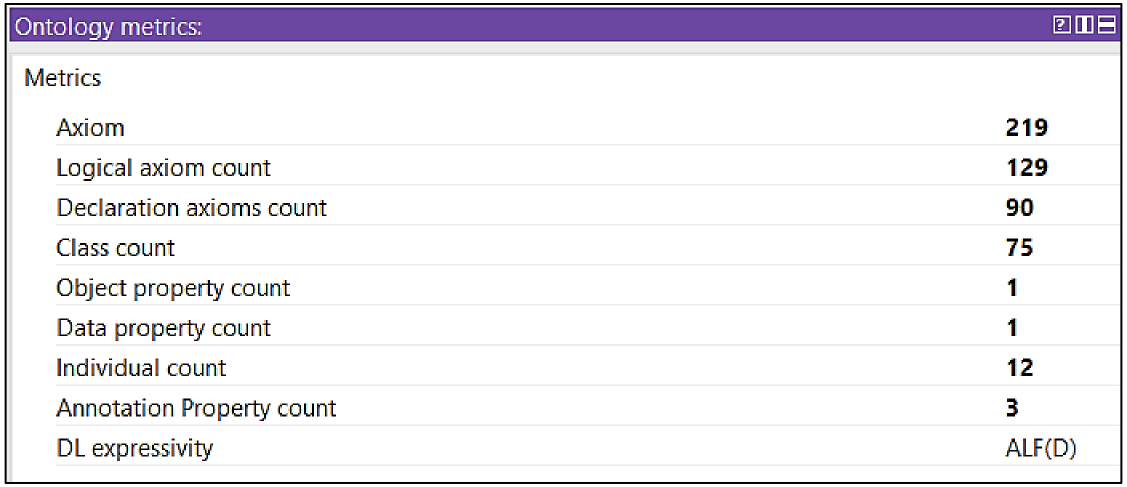


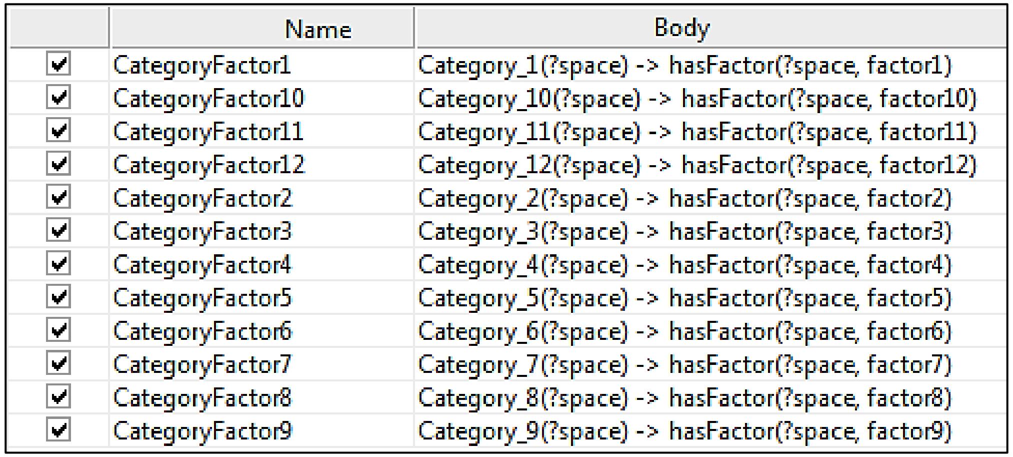
Figure B-10. UKSOC ontology classes and object properties connecting them



Metrics

Axiom	**219**
Logical axiom count	**129**
Declaration axioms count	**90**
Class count	**75**
Object property count	**1**
Data property count	**1**
Individual count	**12**
Annotation Property count	**3**
DL expressivity	ALF(D)

Figure B-11. UKSOC ontology metrics



	Name	Body
☑	CategoryFactor1	Category_1(?space) -> hasFactor(?space, factor1)
☑	CategoryFactor10	Category_10(?space) -> hasFactor(?space, factor10)
☑	CategoryFactor11	Category_11(?space) -> hasFactor(?space, factor11)
☑	CategoryFactor12	Category_12(?space) -> hasFactor(?space, factor12)
☑	CategoryFactor2	Category_2(?space) -> hasFactor(?space, factor2)
☑	CategoryFactor3	Category_3(?space) -> hasFactor(?space, factor3)
☑	CategoryFactor4	Category_4(?space) -> hasFactor(?space, factor4)
☑	CategoryFactor5	Category_5(?space) -> hasFactor(?space, factor5)
☑	CategoryFactor6	Category_6(?space) -> hasFactor(?space, factor6)
☑	CategoryFactor7	Category_7(?space) -> hasFactor(?space, factor7)
☑	CategoryFactor8	Category_8(?space) -> hasFactor(?space, factor8)
☑	CategoryFactor9	Category_9(?space) -> hasFactor(?space, factor9)

Figure B-12. UKSOC ontology SWRL rules matching individuals of specific classes to factors

MassMotion (MM) ontology

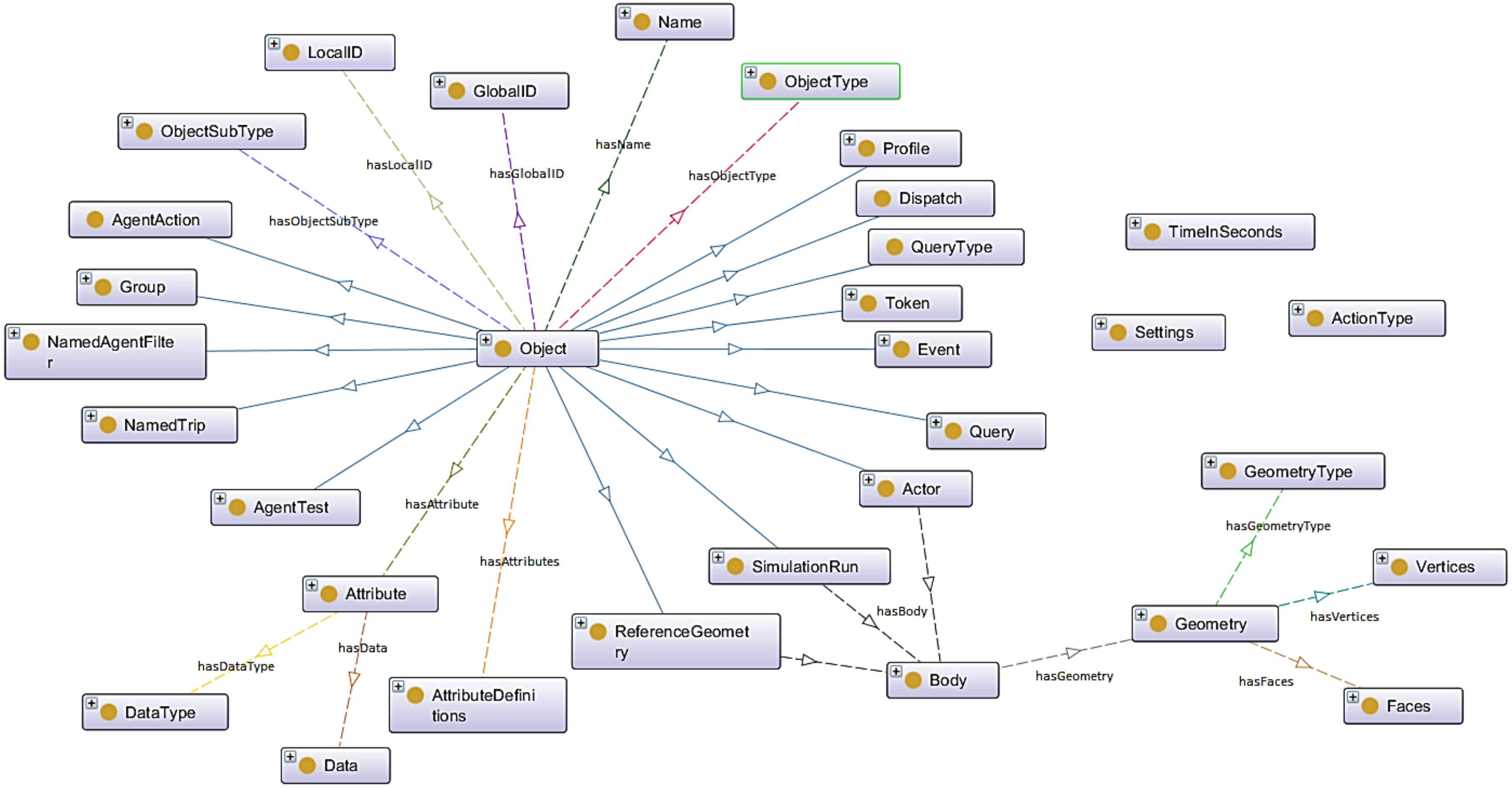
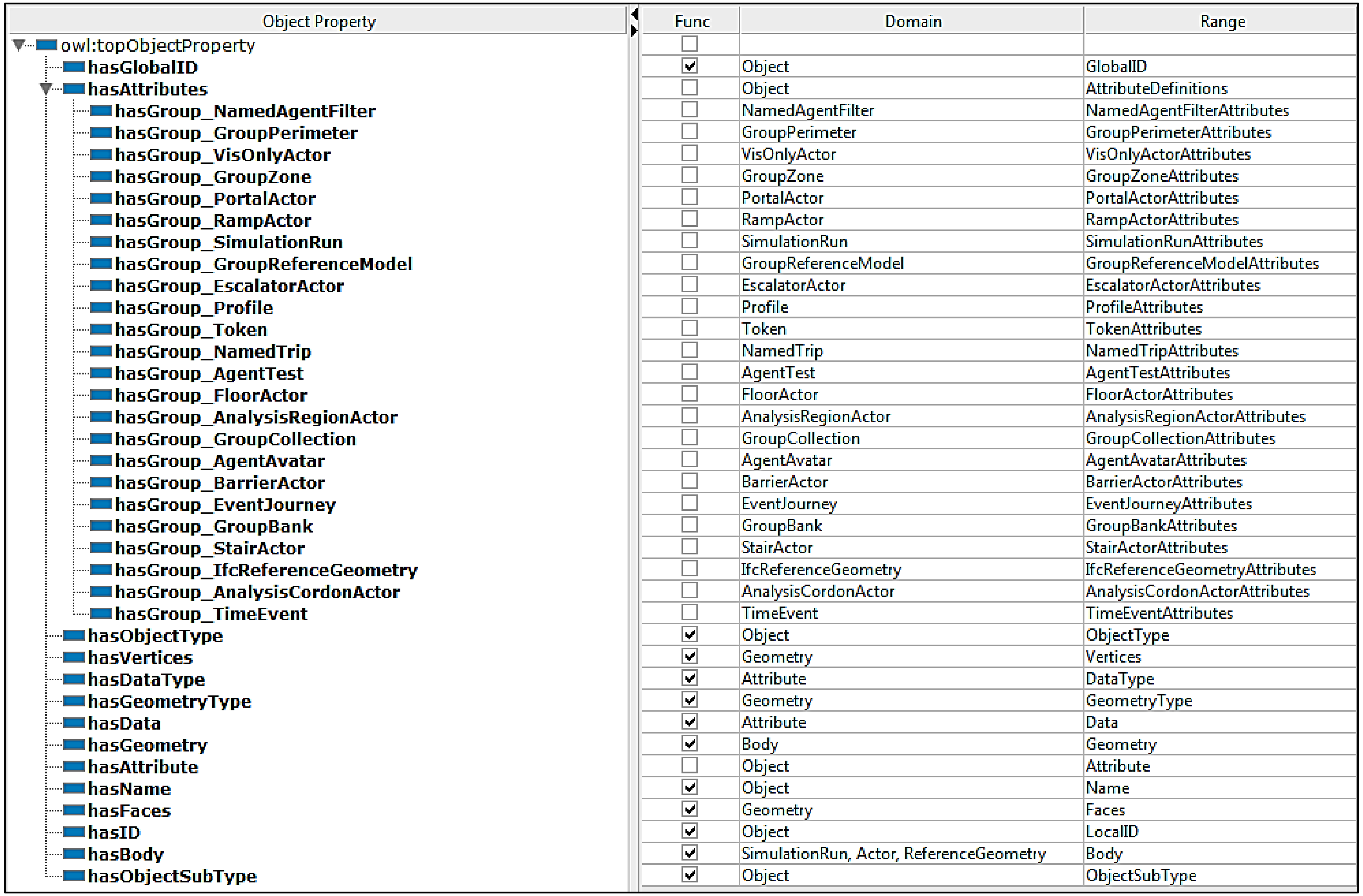
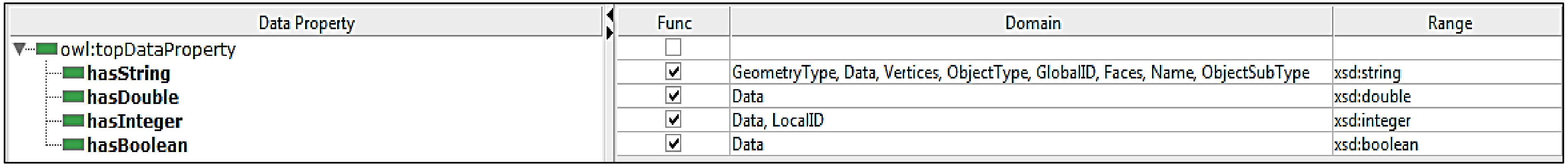


Figure B-13. MM ontology with main upper classes and object properties connecting them



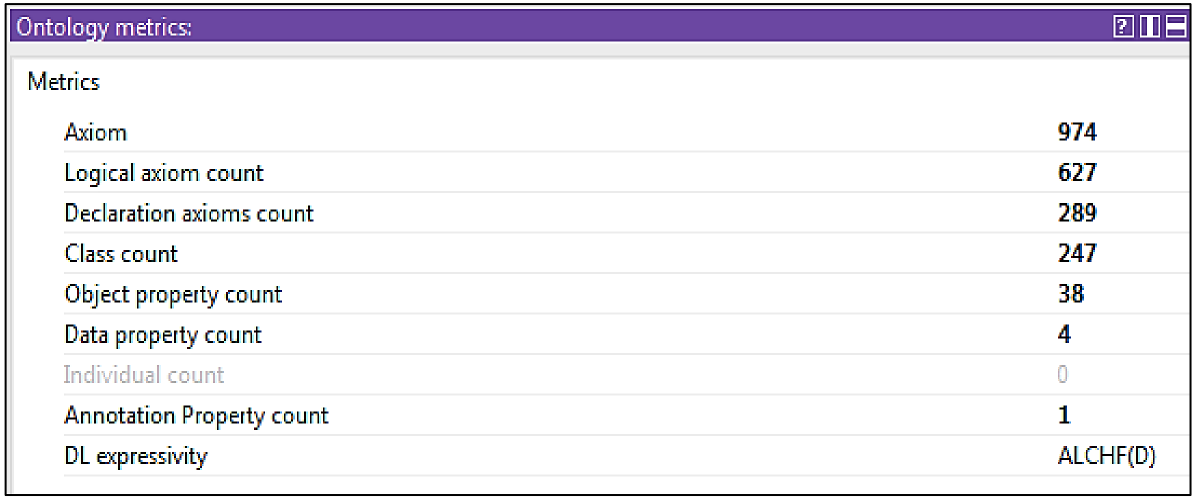
Object Property	Func	Domain	Range
owl:topObjectProperty	□		
hasGlobalID	☑	Object	GlobalID
hasAttributes	□	Object	AttributeDefinitions
hasGroup_NamedAgentFilter	□	NamedAgentFilter	NamedAgentFilterAttributes
hasGroup_GroupPerimeter	□	GroupPerimeter	GroupPerimeterAttributes
hasGroup_VisOnlyActor	□	VisOnlyActor	VisOnlyActorAttributes
hasGroup_GroupZone	□	GroupZone	GroupZoneAttributes
hasGroup_PortalActor	□	PortalActor	PortalActorAttributes
hasGroup_RampActor	□	RampActor	RampActorAttributes
hasGroup_SimulationRun	□	SimulationRun	SimulationRunAttributes
hasGroup_GroupReferenceModel	□	GroupReferenceModel	GroupReferenceModelAttributes
hasGroup_EscalatorActor	□	EscalatorActor	EscalatorActorAttributes
hasGroup_Profile	□	Profile	ProfileAttributes
hasGroup_Token	□	Token	TokenAttributes
hasGroup_NamedTrip	□	NamedTrip	NamedTripAttributes
hasGroup_AgentTest	□	AgentTest	AgentTestAttributes
hasGroup_FloorActor	□	FloorActor	FloorActorAttributes
hasGroup_AnalysisRegionActor	□	AnalysisRegionActor	AnalysisRegionActorAttributes
hasGroup_GroupCollection	□	GroupCollection	GroupCollectionAttributes
hasGroup_AgentAvatar	□	AgentAvatar	AgentAvatarAttributes
hasGroup_BarrierActor	□	BarrierActor	BarrierActorAttributes
hasGroup_EventJourney	□	EventJourney	EventJourneyAttributes
hasGroup_GroupBank	□	GroupBank	GroupBankAttributes
hasGroup_StairActor	□	StairActor	StairActorAttributes
hasGroup_IfcReferenceGeometry	□	IfcReferenceGeometry	IfcReferenceGeometryAttributes
hasGroup_AnalysisCordonActor	□	AnalysisCordonActor	AnalysisCordonActorAttributes
hasGroup_TimeEvent	□	TimeEvent	TimeEventAttributes
hasObjectType	☑	Object	ObjectType
hasVertices	☑	Geometry	Vertices
hasDataType	☑	Attribute	DataType
hasGeometryType	☑	Geometry	GeometryType
hasData	☑	Attribute	Data
hasGeometry	☑	Body	Geometry
hasAttribute	□	Object	Attribute
hasName	☑	Object	Name
hasFaces	☑	Geometry	Faces
hasID	☑	Object	LocalID
hasBody	☑	SimulationRun, Actor, ReferenceGeometry	Body
hasObjectSubType	☑	Object	ObjectSubType

Figure B-14. MM ontology object properties



Data Property	Func	Domain	Range
owl:topDataProperty	□		
hasString	☑	GeometryType, Data, Vertices, ObjectType, GlobalID, Faces, Name, ObjectSubType	xsd:string
hasDouble	☑	Data	xsd:double
hasInteger	☑	Data, LocalID	xsd:integer
hasBoolean	☑	Data	xsd:boolean

Figure B-15. MM ontology data properties



Ontology metrics:

Metrics

Axiom	**974**
Logical axiom count	**627**
Declaration axioms count	**289**
Class count	**247**
Object property count	**38**
Data property count	**4**
Individual count	0
Annotation Property count	**1**
DL expressivity	ALCHF(D)

Figure B-16. MM ontology metrics

Alignment between UKSOC and Uniclass2015 ontologies

The alignment between UKSOC and the Uniclass2015 classification system is based on matching spaces with identical or similar names, as is shown in Table B-2 below. Some comments were made outlining conflicts and/or relationship better suited for integration (subclass or equivalency).

Table B-2. Aligned common spaces between UKSOC categories and Uniclass categories with comments

Category	No	UKSOC	UNICLASS 2015			Note
		Type of space	Uniclass equivalent		Uniclass categories	
		Description	Code	Title	Sub-group Title	
1	1	Standing spectator areas	SL_90_20_83	Spectator standing areas	Common spaces	equivalency
	2	Bar areas (within 2m of serving point)	SL_40_20_06	Bars	Dining spaces	subclasses, but ambiguous with 12 and 22
2	4	Amusement arcade	SL_40_05_03	Amusement arcades	Amusement spaces	equivalency
	5	Assembly hall	SL_25_10_05	Assembly halls	Educational spaces	equivalency
	6	Bingo hall	SL_40_05_43	Indoor play spaces	Amusement spaces	subclass
	7	Club	SL_40_60_21	Dance floors	Performing arts spaces	subclass
	8	Crush hall	SL_90_10_27	Entrance halls	Circulation spaces	subclass
	9	Dance floor	SL_40_60_21	Dance floors	Performing arts spaces	equivalency
	10	Dance hall	SL_40_60_21	Dance floors	Performing arts spaces	subclass
	11	Venue for pop concert and similar events	SL_90_20_05	Audience standing areas	Common spaces	equivalency
	12	Bar areas without fixed sitting	SL_40_20_06	Bars	Dining spaces	subclasses, but ambiguous with 2 and 22
3	13	Concourse	SL_80_10_16	Concourses	Loading and embarkation spaces	equivalency
	14	Queuing area	SL_90_20_69	Queuing areas	Common spaces	equivalency
	15	Shopping mall (4) (5)				no direct equivalent
			SL_20_50_12	Checkout points	Commercial spaces	category about queuing areas
4	16	Committee room	SL_20_70_15	Court rooms	Judicial spaces	equivalency
	17	Common room	SL_25_10_15	Common rooms	Educational spaces	equivalency
	18	Conference room	SL_25_70_13	Conference rooms	Information spaces	equivalency
	19	Dining room	SL_40_20_27	Enclosed dining areas	Dining spaces	subclass, equivalent to Restaurant
			SL_40_20_28	Food courts	Dining spaces	subclass

Category	No	UKSOC	UNICLASS 2015			Note
		Type of space	Uniclass equivalent		Uniclass categories	
		Description	Code	Title	Sub-group Title	
	20	Licensed betting office (public area)	SL_90_20_89	Ticket offices	Common spaces	subclass
	21	Lounge	SL_90_20_96	Waiting rooms	Common spaces	equivalency
	22	Bar (other than in 1 above)	SL_40_20_06	Bars	Dining spaces	subclasses, but ambiguous with 2 and 12
	23	Meeting room	SL_20_15_50	Meeting rooms	Administrative spaces	equivalency
	24	Reading room	SL_25_70_72	Reading rooms	Information spaces	equivalency
	25	Restaurant	SL_40_20_28	Food courts	Dining spaces	subclass, equivalent to Dining room
			SL_40_20_27	Enclosed dining areas	Dining spaces	subclass
			SL_40_20_59	Outdoor dining areas	Dining spaces	subclass
	26	Staff room	SL_90_20_08	Breakout spaces	Common spaces	equivalency
	27	Waiting room	SL_90_20_96	Waiting rooms	Common spaces	equivalency, equivalent of Lounge
5	28	Exhibition hall	SL_25_50	Exhibition spaces	Exhibition spaces	subclass, but ambiguity with Museums
	29	Studio (film, radio, television, recording)	SL_75_10	Communications spaces	Communications spaces	subclass
			SL_40_60_78	Sound recording studios	Performing arts spaces	equivalency
			SL_75_10_73	Radio studios	Communications spaces	equivalency
			SL_75_10_93	Television studios	Communications spaces	equivalency
6	30	Skating rink	SL_42_95_40	Ice skating rinks	Winter sports spaces	equivalency
7	31	Shop sales area (7)	SL_20_50_22	Department store shop floors	Commercial spaces	subclass
			SL_20_50_85	Supermarket shop floors	Commercial spaces	subclass
			SL_20_50_51	Market stalls	Commercial spaces	subclass
8	32	Art gallery	SL_25_50_42	Internal galleries	Exhibition spaces	subclass
	33	Dormitory	SL_45_10_24	Dormitories	Living spaces	equivalency
	34	Factory production area	SL_30_50	Manufacturing spaces	Manufacturing spaces	equivalency
	35	Museum	SL_25_50	Exhibition spaces	Exhibition spaces	subclass
	36	Workshop	SL_30_60_50	Maintenance workshops	Cleaning and maintenance spaces	subclass
9	37	Office	SL_20_15_59	Offices	Administrative spaces	subclass
			SL_20_55_45	Letter sorting offices	Postal communications spaces	subclass
			SL_20_55_60	Parcel sorting offices	Postal communications spaces	subclass

Category	No	UKSOC	UNICLASS 2015			Note
		Type of space	Uniclass equivalent		Uniclass categories	
		Description	Code	Title	Sub-group Title	
			SL_20_85_80	Security offices	Security spaces	subclass
			SL_45_10_16	Concierge offices	Living spaces	subclass
			SL_80_10_60	Passport control offices	Loading and embarkation spaces	subclass
10	38	Shop sales area (8)	SL_20_50_72	Retail kiosks	Commercial spaces	subclass
			SL_20_50_36	Hair and beauty salons	Commercial spaces	subclass
			SL_20_50_29	Financial and professional services outlets	Commercial spaces	subclass
			SL_20_50_32	Food and drink outlets	Commercial spaces	subclass
			SL_20_50_87	Tattoo and piercing parlours	Commercial spaces	subclass
11	39	Kitchen	SL_35_60_56	Non-domestic kitchens	Food management spaces	
			SL_45_10_23	Domestic kitchens	Food management spaces	
			SL_45_10_44	Kitchen-dining rooms	Food management spaces	
	40	Library	SL_25_70_47	Library rooms	Information spaces	equivalency
12	41	Bedroom	SL_45_10_09	Bedrooms	Living spaces	subclass
			SL_45_10_57	Nursing home bedrooms	Living spaces	subclass
	42	Study-bedroom	SL_45_10_08	Bedroom-studies	Living spaces	subclass
13	43	Bed-sitting room	SL_45_10_08	Bedroom-studies	Living spaces	subclass
	44	Billiards room	SL_42_40_79	Snooker, billiards and pool halls	Indoor activity spaces	subclass
	45	Billiards hall	SL_42_40_79	Snooker, billiards and pool halls	Indoor activity spaces	subclass
	46	Snooker room	SL_42_40_79	Snooker, billiards and pool halls	Indoor activity spaces	subclass
	47	Snooker hall	SL_42_40_79	Snooker, billiards and pool halls	Indoor activity spaces	subclass
14	48	Storage	SL_90_50	Storage spaces	Storage spaces	Equivalency, entire sub-group
	49	Warehousing	SL_30_90	Warehousing and distribution spaces	Warehousing and distribution spaces	Equivalency, entire sub-group
15	50	Car park	SL_80_45_40	Indoor vehicle parking spaces	Highway storage and maintenance spaces	subclass
			SL_80_45_59	Outdoor vehicle parking spaces	Highway storage and maintenance spaces	subclass

Table B-3. Alignment SWRL rules between the UKSOC and Uniclass2015 ontology. Implements Table B-2

No	Rule name	SWRL code
1	CF-Category_1-01-SpectatorStandingAreas	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_90_20_83") -> uksoc:hasFactor(?spaceClass, uksoc:factor1)
2	CF-Category_1-02-BarsServingAreas	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_40_20_06") -> uksoc:hasFactor(?spaceClass, uksoc:factor1)
3	CF-Category_2-01-AmusementArcades	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_40_05_03") -> uksoc:hasFactor(?spaceClass, uksoc:factor2)
4	CF-Category_2-02-AssemblyHalls	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_25_10_05") -> uksoc:hasFactor(?spaceClass, uksoc:factor2)
5	CF-Category_2-03-IndoorPlaySpaces	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_40_05_43") -> uksoc:hasFactor(?spaceClass, uksoc:factor2)
6	CF-Category_2-04-DanceFloor	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_40_60_21") -> uksoc:hasFactor(?spaceClass, uksoc:factor2)
7	CF-Category_2-05-EntranceHalls	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_90_10_27") -> uksoc:hasFactor(?spaceClass, uksoc:factor2)
8	CF-Category_2-06-AudienceStandingAreas	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_90_20_05") -> uksoc:hasFactor(?spaceClass, uksoc:factor2)
9	CF-Category_3-01-Concourses	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^

No	Rule name	SWRL code
		swrlb:matches(?id, "SL_80_10_16") -> uksoc:hasFactor(?spaceClass, uksoc:factor3)
10	CF-Category_3-02-QueuingAreas	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_90_20_69") -> uksoc:hasFactor(?spaceClass, uksoc:factor3)
11	CF-Category_3-03-CheckoutPoints	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_50_12") -> uksoc:hasFactor(?spaceClass, uksoc:factor3)
12	CF-Category_4-01-CourtRooms	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_70_15") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
13	CF-Category_4-02-CommonRooms	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_25_10_15") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
14	CF-Category_4-03-ConferenceRoom	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_25_70_13") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
15	CF-Category_4-04-TicketOffices	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_90_20_89") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
16	CF-Category_4-05-WaitingRooms	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_90_20_96") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
17	CF-Category_4-06-UnfixedSeatingBarAreas	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_40_20_06") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
18	CF-Category_4-07-MeetingRooms	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_15_50") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)

No	Rule name	SWRL code
19	CF-Category_4-08-ReadingRooms	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_25_70_72") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
20	CF-Category_4-09-EnclosedDiningAreas	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_40_20_27") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
21	CF-Category_4-10-FoodCourts	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_40_20_28") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
22	CF-Category_4-11-OutdoorDiningAreas	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_40_20_59") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
23	CF-Category_4-12-BreakoutSpaces	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_90_20_08") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
24	CF-Category_4-13-BreakoutSpaces	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_90_20_08") -> uksoc:hasFactor(?spaceClass, uksoc:factor4)
25	CF-Category_5-01-ExhibitionHall	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:containsIgnoreCase(?id, "SL_25_50") -> uksoc:hasFactor(?spaceClass, uksoc:factor5)
26	CF-Category_5-02-RadioStudios	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_75_10_73") -> uksoc:hasFactor(?spaceClass, uksoc:factor5)
27	CF-Category_5-03-RecordingStudios	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_40_60_78") -> uksoc:hasFactor(?spaceClass, uksoc:factor5)
28	CF-Category_5-04-TelevisionStudios	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^

No	Rule name	SWRL code
		swrlb:matches(?id, "SL_75_10_93") -> uksoc:hasFactor(?spaceClass, uksoc:factor5)
29	CF-Category_6-01-IceSkatingRinks	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_42_95_40") -> uksoc:hasFactor(?spaceClass, uksoc:factor6)
30	CF-Category_6-02-DepartmentStoreShopFloors	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_50_22") -> uksoc:hasFactor(?spaceClass, uksoc:factor6)
31	CF-Category_6-03-SupermarketShopFloors	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_50_85") -> uksoc:hasFactor(?spaceClass, uksoc:factor6)
32	CF-Category_6-04-MarketStalls	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_50_51") -> uksoc:hasFactor(?spaceClass, uksoc:factor6)
33	CF-Category_7-01-ExhibitionHalls	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:containsIgnoreCase(?id, "SL_25_50") -> uksoc:hasFactor(?spaceClass, uksoc:factor7)
34	CF-Category_7-02-InternalGalleries	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_25_50_42") -> uksoc:hasFactor(?spaceClass, uksoc:factor7)
35	CF-Category_7-03-Dormitories	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_42_10_24") -> uksoc:hasFactor(?spaceClass, uksoc:factor7)
36	CF-Category_7-04-ManufacturingSpaces	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:containsIgnoreCase(?id, "SL_30_50") -> uksoc:hasFactor(?spaceClass, uksoc:factor7)
37	CF-Category_7-05-MaintenanceWorkshops	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_30_60_50") -> uksoc:hasFactor(?spaceClass, uksoc:factor7)
38	CF-Category_8-01-Offices	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^

No	Rule name	SWRL code
		swrlb:matches(?id, "SL_20_15_59") -> uksoc:hasFactor(?spaceClass, uksoc:factor8)
39	CF-Category_8-02-LetterSortingOffices	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_55_45") -> uksoc:hasFactor(?spaceClass, uksoc:factor8)
40	CF-Category_8-03-ParcelSortingOffices	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_55_60") -> uksoc:hasFactor(?spaceClass, uksoc:factor8)
41	CF-Category_8-04-SecurityOffices	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_85_80") -> uksoc:hasFactor(?spaceClass, uksoc:factor8)
42	CF-Category_8-05-ConciergeOffices	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_45_10_16") -> uksoc:hasFactor(?spaceClass, uksoc:factor8)
43	CF-Category_8-06-PassportControlOffices	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_80_10_60") -> uksoc:hasFactor(?spaceClass, uksoc:factor8)
44	CF-Category_9-01-FoodManagementSpaces	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:containsIgnoreCase(?id, "SL_35_60") -> uksoc:hasFactor(?spaceClass, uksoc:factor9)
45	CF-Category_9-02-LibraryRooms	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_25_70_47") -> uksoc:hasFactor(?spaceClass, uksoc:factor9)
46	CF-Category_9-03-FinancialAndProfessionalServicesOutlets	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_50_29") -> uksoc:hasFactor(?spaceClass, uksoc:factor9)
47	CF-Category_9-04-FoodAndDrinksOutlets	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_50_32") -> uksoc:hasFactor(?spaceClass, uksoc:factor9)
48	CF-Category_9-05-HairAndBeautySalons	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_50_36") -> uksoc:hasFactor(?spaceClass, uksoc:factor9)
49	CF-Category_9-06-RetailKiosks	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_50_72") -> uksoc:hasFactor(?spaceClass, uksoc:factor9)

No	Rule name	SWRL code
50	CF-Category_9-07-TatooAndPiercingParlours	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_20_50_87") -> uksoc:hasFactor(?spaceClass, uksoc:factor9)
51	CF-Category_10-01-Bedrooms	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_45_10_09") -> uksoc:hasFactor(?spaceClass, uksoc:factor10)
52	CF-Category_10-02-NursingHomBedrooms	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_45_10_57") -> uksoc:hasFactor(?spaceClass, uksoc:factor10)
53	CF-Category_10-03-BedroomStudies	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_45_10_08") -> uksoc:hasFactor(?spaceClass, uksoc:factor10)
54	CF-Category_11-01-SnookerBilliardAndPoolHalls	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:matches(?id, "SL_42_40_79") -> uksoc:hasFactor(?spaceClass, uksoc:factor11)
55	CF-Category_12-01-StorageSpaces	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:containsIgnoreCase(?id, "SL_90_50") -> uksoc:hasFactor(?spaceClass, uksoc:factor12)
56	CF-Category_12-02-WarehousingAndDistributionSpaces	uniclass2015:Space(?spaceClass) ^ smpo:identifier(?spaceClass, ?id) ^ swrlb:containsIgnoreCase(?id, "SL_30_90") -> uksoc:hasFactor(?spaceClass, uksoc:factor12)

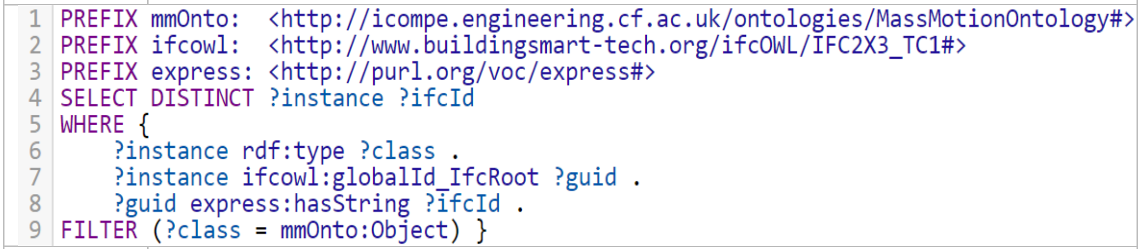
Appendix C – SPARQL queries

Table C-1. SPARQL queries operating to IfcOwl

SPARQL QUERIES
Link for IFC schema concepts: http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ Documentation reference for geometry construction: BS ISO 10303-42:1994

Q-IFC-1, Find objects

Question	Which are the instances with an IFC identifier which are also MassMotion instances?

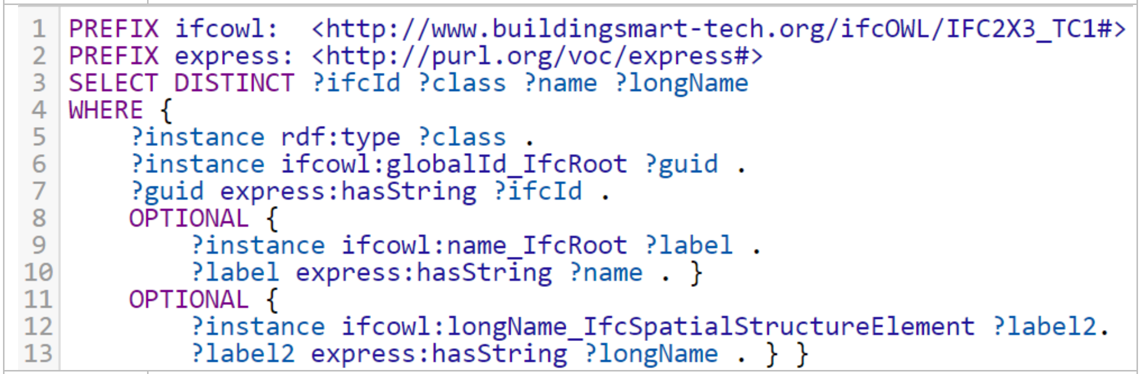


```
PREFIX mmOnto:  <http://icompe.engineering.cf.ac.uk/ontologies/MassMotionOntology#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
PREFIX express: <http://purl.org/voc/express#>
SELECT DISTINCT ?instance ?ifcId
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
FILTER (?class = mmOnto:Object) }
```

Function	Finds all IfcOwl individuals from the IFC model, which have a unique identifier. Narrows the answers down using a filter, which limits those found to also be a MassMotion class.	
	IFC concepts of interest: IfcIdentifier	Requires reasoning? YES

Q-IFC-2, Get IFC Types

Question	What are the IFC types of individuals with unique identifiers and any IFC name labels?



```
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
PREFIX express: <http://purl.org/voc/express#>
SELECT DISTINCT ?ifcId ?class ?name ?longName
WHERE {
     ?instance rdf:type ?class .
     ?instance ifcowl:globalId_IfcRoot ?guid .
     ?guid express:hasString ?ifcId .
     OPTIONAL {
         ?instance ifcowl:name_IfcRoot ?label .
         ?label express:hasString ?name . }
     OPTIONAL {
         ?instance ifcowl:longName_IfcSpatialStructureElement ?label2.
         ?label2 express:hasString ?longName . } }
```

Function	Find the specific class of each object at the lowest level in the hierarchy tree which is part of the IfcOwl graph and optionally finds their labels (names or long names)	
	IFC concepts of interest: IfcIdentifier, IfcLabel	Requires reasoning? NO

Q-IFC-3, Get IFC Storeys

Question What is the elevation and identifier of each IfcBuildingStorey class instance?

```
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
PREFIX express: <http://purl.org/voc/express#>
SELECT DISTINCT ?ifcId ?elevation
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:elevation_IfcBuildingStorey ?lengthMeasure .
    ?lengthMeasure express:hasDouble ?elevation .
FILTER (?class = ifcowl:IfcBuildingStorey) }
```

Function Finds all storeys within the model and retrieves their elevations and its IFC identifier to match it in memory.

<u>IFC concepts of interest:</u> IfcIdentifier, IfcBuildingStorey	Requires reasoning? NO

Q-IFC-4, Find inhabited spaces

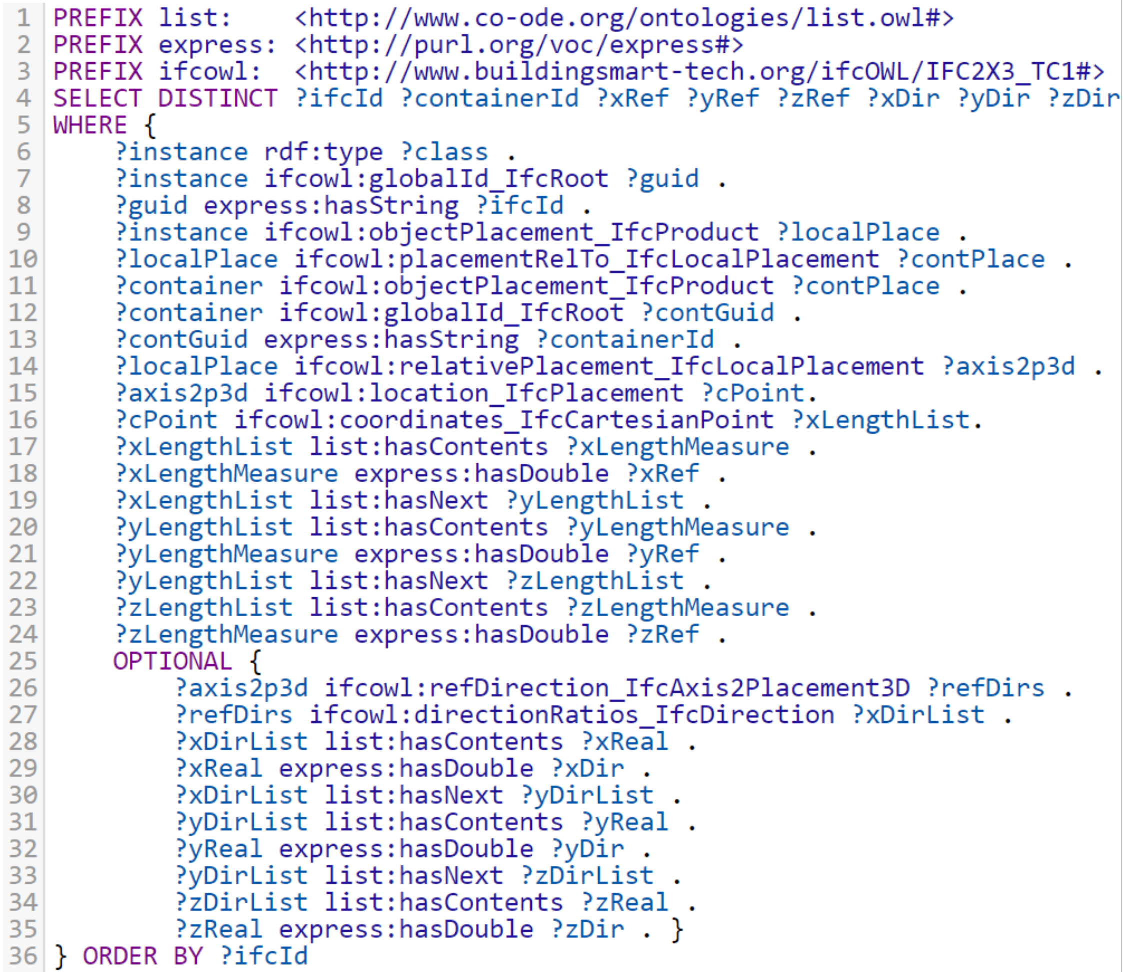
Question What are the shape definition types of each IFC object?

```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?shapeType ?actualShapeType ?profileType
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape.
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList .
    { ?repList list:hasContents ?shapeRep .}
    UNION { ?repList list:hasNext ?secondList .
            ?secondList list:hasContents ?shapeRep } .
    ?shapeRep ifcowl:items_IfcRepresentation ?shape .
    ?shape rdf:type ?shapeType .
    OPTIONAL {
        {?shape ifcowl:sweptArea_IfcSweptAreaSolid ?profDef .}
        UNION { ?shapeRep ifcowl:items_IfcRepresentation ?boolItem .
                ?boolItem ifcowl:firstOperand_IfcBooleanResult ?actualShape .
                ?actualShape rdf:type ?actualShapeType .
                ?actualShape ifcowl:sweptArea_IfcSweptAreaSolid ?profDef . }
                ?profDef rdf:type ?profileType . }
} ORDER BY ?ifcId
```

Function Finds those IFC objects which have a shape, or a geometric representation within the model. Does not extract the shape, only the type of basic shape definition it is constructed from, according to the IFC schema specification.

<u>IFC concepts of interest:</u> IfcIdentifier, IfcProduct, IfcProductDefinitionShape, IfcRepresentation, IfcSweptAreaSolid	Requires reasoning? NO

<table>
<tr><th colspan="3">Q-IFC-5, Get IFC Placements</th></tr>
<tr><td>Question</td><td colspan="2">What are the relative coordinates of each geometric IFC object relative to its parent?</td></tr>
<tr><td colspan="3">



<pre>
PREFIX list: <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl: <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?containerId ?xRef ?yRef ?zRef ?xDir ?yDir ?zDir
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:objectPlacement_IfcProduct ?localPlace .
    ?localPlace ifcowl:placementRelTo_IfcLocalPlacement ?contPlace .
    ?container ifcowl:objectPlacement_IfcProduct ?contPlace .
    ?container ifcowl:globalId_IfcRoot ?contGuid .
    ?contGuid express:hasString ?containerId .
    ?localPlace ifcowl:relativePlacement_IfcLocalPlacement ?axis2p3d .
    ?axis2p3d ifcowl:location_IfcPlacement ?cPoint.
    ?cPoint ifcowl:coordinates_IfcCartesianPoint ?xLengthList.
    ?xLengthList list:hasContents ?xLengthMeasure .
    ?xLengthMeasure express:hasDouble ?xRef .
    ?xLengthList list:hasNext ?yLengthList .
    ?yLengthList list:hasContents ?yLengthMeasure .
    ?yLengthMeasure express:hasDouble ?yRef .
    ?yLengthList list:hasNext ?zLengthList .
    ?zLengthList list:hasContents ?zLengthMeasure .
    ?zLengthMeasure express:hasDouble ?zRef .
    OPTIONAL {
        ?axis2p3d ifcowl:refDirection_IfcAxis2Placement3D ?refDirs .
        ?refDirs ifcowl:directionRatios_IfcDirection ?xDirList .
        ?xDirList list:hasContents ?xReal .
        ?xReal express:hasDouble ?xDir .
        ?xDirList list:hasNext ?yDirList .
        ?yDirList list:hasContents ?yReal .
        ?yReal express:hasDouble ?yDir .
        ?yDirList list:hasNext ?zDirList .
        ?zDirList list:hasContents ?zReal .
        ?zReal express:hasDouble ?zDir . }
} ORDER BY ?ifcId
</pre>

</td></tr>
<tr><td>Function</td><td colspan="2">Retrieves the coordinates in (x, y, z) of each IFC object with a geometric representation within the model. These coordinates are relative to its parent object. The parent of each object is also found in this query and then matched in memory to find the absolute position.</td></tr>
<tr><td></td><td>IFC concepts of interest:
IfcIdentifier, IfcProduct,
IfcLocalPlacement, IfcObjectPlacement,
IfcAxis2Placement3D,
IfcCartesianPoint,
IfcDirection</td><td>Requires reasoning?
NO</td></tr>
<tr><td colspan="3"></td></tr>
</table>

Q-IFC-6, Get IFC Placements (spaces)

Question	What are the relative coordinates of IfcSpace instances relative to its parent?

```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?xRef ?yRef ?zRef ?xDir ?zDir ?yDir
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape .
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList
    ?repList list:hasContents ?shapeRep .
    ?shapeRep ifcowl:items_IfcRepresentation ?exAreaSolid .
    ?exAreaSolid ifcowl:position_IfcSweptAreaSolid ?axis2p3d .
    ?axis2p3d ifcowl:location_IfcPlacement ?cpoint.
    ?cpoint ifcowl:coordinates_IfcCartesianPoint ?xlengthList.
    ?xlengthList list:hasContents ?xlengthMeasure .
    ?xlengthMeasure express:hasDouble ?xRef .
    ?xlengthList list:hasNext ?ylengthList .
    ?ylengthList list:hasContents ?ylengthMeasure .
    ?ylengthMeasure express:hasDouble ?yRef .
    ?ylengthList list:hasNext ?zlengthList .
    ?zlengthList list:hasContents ?zLengthMeasure .
    ?zLengthMeasure express:hasDouble ?zRef .
    OPTIONAL{
        ?axis2p3d ifcowl:refDirection_IfcAxis2Placement3D ?refDirs .
        ?refDirs ifcowl:directionRatios_IfcDirection ?xDirList .
        ?xDirList list:hasContents ?xReal .
        ?xReal express:hasDouble ?xDir .
        ?xDirList list:hasNext ?yDirList .
        ?yDirList list:hasContents ?yReal .
        ?yReal express:hasDouble ?yDir .
        ?yDirList list:hasNext ?zDirList .
        ?zDirList list:hasContents ?zReal .
        ?zReal express:hasDouble ?zDir . }
FILTER (?class = ifcowl:IfcSpace)
} ORDER BY ?ifcId
```

Function	Retrieves the coordinates in (x, y, z) of each IFC Space object with a geometric representation within the model. These coordinates are relative to its parent object. The query is similar to 5, and it accounts for some spaces having a different definition of relative position coordinates, stored in the IfcExtrudedAreaSolid concept.	
	IFC concepts of interest: IfcIdentifier, IfcProduct, IfcLocalPlacement, IfcObjectPlacement, IfcAxis2Placement3D, IfcCartesianPoint, IfcSpace, IfcDirection	Requires reasoning? NO

Q-IFC-7, Get IFC Placements (mapped)

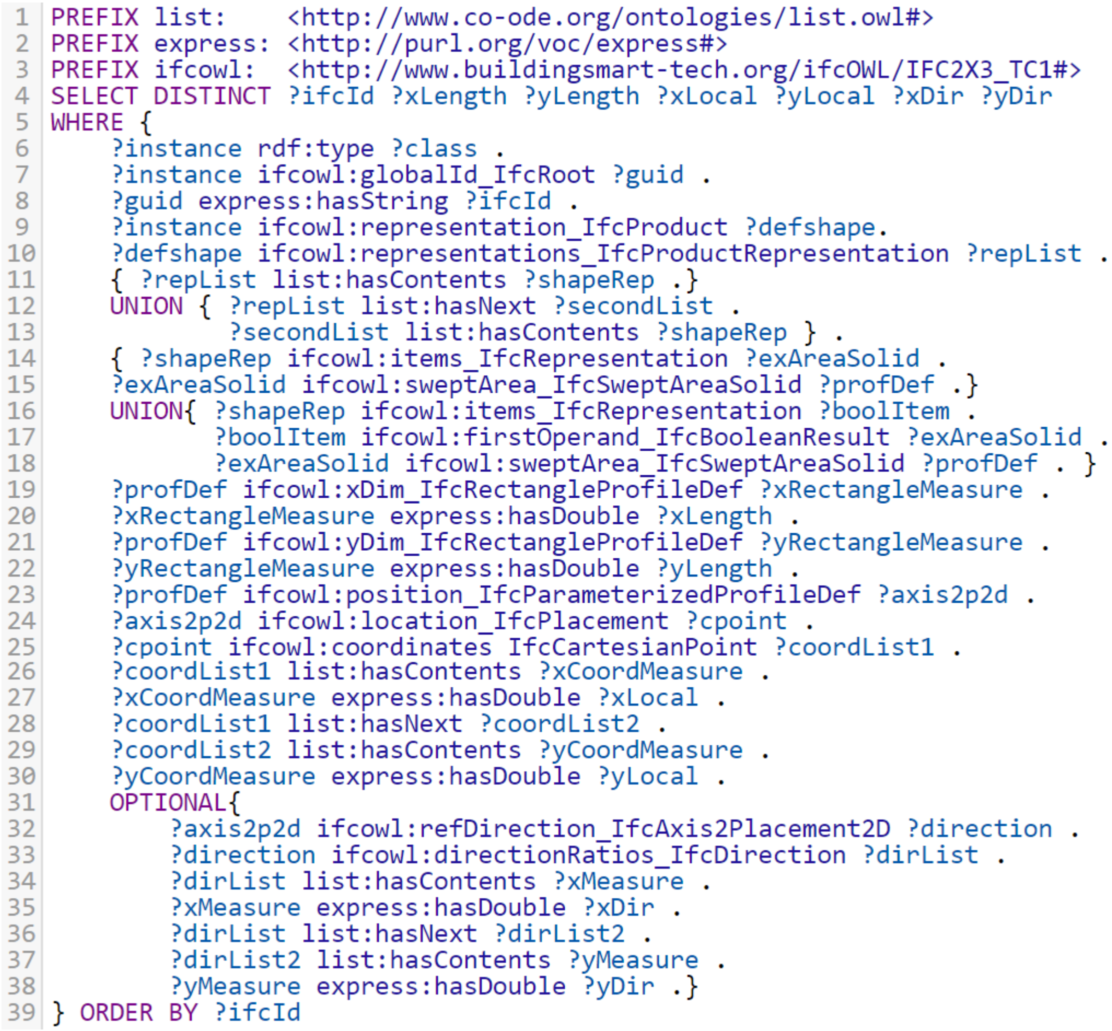
Question	What are the relative coordinates of IFC Columns relative to its parent?

```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?xRef ?yRef ?zRef ?xDir ?zDir ?yDir ?xAxis ?yAxis ?zAxis
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape .
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList .
    { ?repList list:hasContents ?shapeRep .}
    UNION { ?repList list:hasNext ?secondList .
        ?secondList list:hasContents ?shapeRep } .
        ?shapeRep ifcowl:items_IfcRepresentation ?mappedItem .
    ?mappedItem ifcowl:mappingSource_IfcMappedItem ?repMap .
    ?repMap ifcowl:mappedRepresentation_IfcRepresentationMap ?shapeRep2 .
    ?shapeRep2 ifcowl:items_IfcRepresentation ?exAreaSolid .
    ?exAreaSolid ifcowl:position_IfcSweptAreaSolid ?axis2p3d .
    ?axis2p3d ifcowl:location_IfcPlacement ?cpoint.
    ?cpoint ifcowl:coordinates_IfcCartesianPoint ?xlengthList.
    ?xlengthList list:hasContents ?xlengthMeasure .
    ?xlengthMeasure express:hasDouble ?xRef .
    ?xlengthList list:hasNext ?ylengthList .
    ?ylengthList list:hasContents ?ylengthMeasure .
    ?ylengthMeasure express:hasDouble ?yRef .
    ?ylengthList list:hasNext ?zlengthList .
    ?zlengthList list:hasContents ?zLengthMeasure .
    ?zLengthMeasure express:hasDouble ?zRef .
    OPTIONAL{
        ?axis2p3d ifcowl:refDirection_IfcAxis2Placement3D ?refDirs .
        ?refDirs ifcowl:directionRatios_IfcDirection ?xDirList .
        ?xDirList list:hasContents ?xReal .
        ?xReal express:hasDouble ?xDir .
        ?xDirList list:hasNext ?yDirList .
        ?yDirList list:hasContents ?yReal .
        ?yReal express:hasDouble ?yDir .
        ?yDirList list:hasNext ?zDirList .
        ?zDirList list:hasContents ?zReal .
        ?zReal express:hasDouble ?zDir . }
    OPTIONAL{
        ?axis2p3d ifcowl:axis_IfcAxis2Placement3D ?axisDir .
        ?axisDir ifcowl:directionRatios_IfcDirection ?xAxisList .
        ?xAxisList list:hasContents ?xAxisReal .
        ?xAxisReal express:hasDouble ?xAxis .
        ?xAxisList list:hasNext ?yAxisList .
        ?yAxisList list:hasContents ?yAxisReal .
        ?yAxisReal express:hasDouble ?yAxis .
        ?yAxisList list:hasNext ?zAxisList .
        ?zAxisList list:hasContents ?zAxisReal .
        ?zAxisReal express:hasDouble ?zAxis .   }
FILTER (?class = ifcowl:IfcColumn)
} ORDER BY ?ifcId
```

Function	Retrieves the coordinates in (x, y, z) of each IFC Column object with a geometric representation within the model. These coordinates are relative to its parent object. The query is similar to 5 and 6, and it accounts for some columns having a different definition of relative position coordinates, stored in the IfcRepresentationMap concept.	
	<u>IFC concepts of interest:</u> IfcIdentifier, IfcProduct, IfcLocalPlacement, IfcObjectPlacement, IfcAxis2Placement3D, IfcRepresentationMap, IfcCartesianPoint, IfcColumn, IfcDirection	Requires reasoning? NO

Q-IFC-8, Get IFC Rectangle shapes

Question	Which are the instances with unique identifiers which have a rectangular shape definition?



```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?xLength ?yLength ?xLocal ?yLocal ?xDir ?yDir
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape.
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList .
    { ?repList list:hasContents ?shapeRep .}
    UNION { ?repList list:hasNext ?secondList .
            ?secondList list:hasContents ?shapeRep } .
    { ?shapeRep ifcowl:items_IfcRepresentation ?exAreaSolid .
    ?exAreaSolid ifcowl:sweptArea_IfcSweptAreaSolid ?profDef .}
    UNION{ ?shapeRep ifcowl:items_IfcRepresentation ?boolItem .
           ?boolItem ifcowl:firstOperand_IfcBooleanResult ?exAreaSolid .
           ?exAreaSolid ifcowl:sweptArea_IfcSweptAreaSolid ?profDef . }
    ?profDef ifcowl:xDim_IfcRectangleProfileDef ?xRectangleMeasure .
    ?xRectangleMeasure express:hasDouble ?xLength .
    ?profDef ifcowl:yDim_IfcRectangleProfileDef ?yRectangleMeasure .
    ?yRectangleMeasure express:hasDouble ?yLength .
    ?profDef ifcowl:position_IfcParameterizedProfileDef ?axis2p2d .
    ?axis2p2d ifcowl:location_IfcPlacement ?cpoint .
    ?cpoint ifcowl:coordinates IfcCartesianPoint ?coordList1 .
    ?coordList1 list:hasContents ?xCoordMeasure .
    ?xCoordMeasure express:hasDouble ?xLocal .
    ?coordList1 list:hasNext ?coordList2 .
    ?coordList2 list:hasContents ?yCoordMeasure .
    ?yCoordMeasure express:hasDouble ?yLocal .
    OPTIONAL{
        ?axis2p2d ifcowl:refDirection_IfcAxis2Placement2D ?direction .
        ?direction ifcowl:directionRatios_IfcDirection ?dirList .
        ?dirList list:hasContents ?xMeasure .
        ?xMeasure express:hasDouble ?xDir .
        ?dirList list:hasNext ?dirList2 .
        ?dirList2 list:hasContents ?yMeasure .
        ?yMeasure express:hasDouble ?yDir .}
} ORDER BY ?ifcId
```

Function	Finds the elements which are defined as rectangular in nature and retrieve the basic values to construct its shape.	
	IFC concepts of interest: IfcIdentifier, IfcProduct, IfcRepresentation, IfcProductRepresentation, IfcSweptAreaSolid, IfcRectangleProfileDefinition IfcAxis2Placement2D, IfcCartesianPoint, IfcDirection	Requires reasoning? NO

Q-IFC-9, Get IFC Rectangle shapes (mapped)

Question	Which are the instances with unique identifiers which are defined as rectangular based on a mapped shape?

```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?profileType ?xLength ?yLength ?xLocal ?yLocal ?xDir ?yDir
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape.
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList .
    { ?repList list:hasContents ?shapeRep .}
    UNION { ?repList list:hasNext ?secondList .
            ?secondList list:hasContents ?shapeRep } .
    ?shapeRep ifcowl:items_IfcRepresentation ?mappedItem .
    ?mappedItem ifcowl:mappingSource_IfcMappedItem ?repMap .
    ?repMap ifcowl:mappedRepresentation_IfcRepresentationMap ?shapeRep2 .
    ?shapeRep2 ifcowl:items_IfcRepresentation ?extAreaSolid .
    ?extAreaSolid ifcowl:sweptArea_IfcSweptAreaSolid ?profDef .
    ?profDef rdf:type  ?profileType .
    ?profDef ifcowl:xDim_IfcRectangleProfileDef ?xRectangleMeasure .
    ?xRectangleMeasure express:hasDouble ?xLength .
    ?profDef ifcowl:yDim_IfcRectangleProfileDef ?yRectangleMeasure .
    ?yRectangleMeasure express:hasDouble ?yLength .
    ?profDef ifcowl:position_IfcParameterizedProfileDef ?axis2p2d .
    ?axis2p2d ifcowl:location_IfcPlacement ?cpoint .
    ?cpoint ifcowl:coordinates_IfcCartesianPoint ?coordList1 .
    ?coordList1 list:hasContents ?xCoordMeasure .
    ?xCoordMeasure express:hasDouble ?xLocal .
    ?coordList1 list:hasNext ?coordList2 .
    ?coordList2 list:hasContents ?yCoordMeasure .
    ?yCoordMeasure express:hasDouble ?yLocal .
    OPTIONAL{
        ?axis2p2d ifcowl:refDirection_IfcAxis2Placement2D ?direction .
        ?direction ifcowl:directionRatios_IfcDirection ?dirList .
        ?dirList list:hasContents ?xMeasure .
        ?xMeasure express:hasDouble ?xDir .
        ?dirList list:hasNext ?dirList2 .
        ?dirList2 list:hasContents ?yMeasure .
        ?yMeasure express:hasDouble ?yDir .}
FILTER (?class = ifcowl:IfcColumn)
} ORDER BY ?ifcId
```

Function	Finds the elements which are defined as rectangular in nature and retrieve the basic values to construct its shape. Is nearly identical to 8, however a mapped shapes in IFC belongs to one object which can be copied by other identical objects, to save space.	
	<u>IFC concepts of interest:</u> IfcIdentifier, IfcProduct, IfcRepresentation, IfcProductRepresentation, IfcSweptAreaSolid, IfcRectangleProfileDefinition IfcAxis2Placement2D, IfcCartesianPoint, IfcDirection, IfcMappedItem	Requires reasoning? NO

Q-IFC-10, Get IFC Arbitrary shapes

Question	Which are the instances with unique identifiers that have an arbitrary shaped definition?	

```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?firstPointList ?xAxisDir ?yAxisDir ?zAxisDir
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape.
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList .
    { ?repList list:hasContents ?shapeRep .}
    UNION { ?repList list:hasNext ?secondList .
            ?secondList list:hasContents ?shapeRep } .
    {?shapeRep ifcowl:items_IfcRepresentation ?exAreaSolid .
     ?exAreaSolid ifcowl:sweptArea_IfcSweptAreaSolid ?profDef .}
    UNION { ?shapeRep ifcowl:items_IfcRepresentation ?boolItem .
            ?boolItem ifcowl:firstOperand_IfcBooleanResult ?exAreaSolid .}
    ?exAreaSolid ifcowl:sweptArea_IfcSweptAreaSolid ?profDef .
    ?profDef ifcowl:outerCurve_IfcArbitraryClosedProfileDef ?polyLine .
    ?polyLine ifcowl:points_IfcPolyline ?firstPointList .
    OPTIONAL{
        ?exAreaSolid ifcowl:position_IfcSweptAreaSolid ?axis2p3d .
        ?axis2p3d ifcowl:axis_IfcAxis2Placement3D ?axisDir .
        ?axisDir ifcowl:directionRatios_IfcDirection ?dirList .
        ?dirList list:hasContents ?xMeasure .
        ?xMeasure express:hasDouble ?xAxisDir .
        ?dirList list:hasNext ?dirList2 .
        ?dirList2 list:hasContents ?yMeasure .
        ?yMeasure express:hasDouble ?yAxisDir .
        ?dirList2 list:hasNext ?dirList3 .
        ?dirList3 list:hasContents ?zMeasure .
        ?zMeasure express:hasDouble ?zAxisDir .}
} ORDER BY ?ifcId
```

Function	Identifies objects with an arbitrary shape definition and retrieves the first point and its coordinates, which is part of a finite list of points used to define an arbitrary perimeter.	
	IFC concepts of interest: IfcIdentifier, IfcProduct, IfcRepresentation, IfcProductRepresentation, IfcSweptAreaSolid, IfcArbitraryClosedProfileDefinition, IfcPolyline, IfcAxis2Placement3D, IfcCartesianPoint, IfcDirection,	Requires reasoning? NO

Q-IFC-1, Get Arbitrary shapes (mapped)

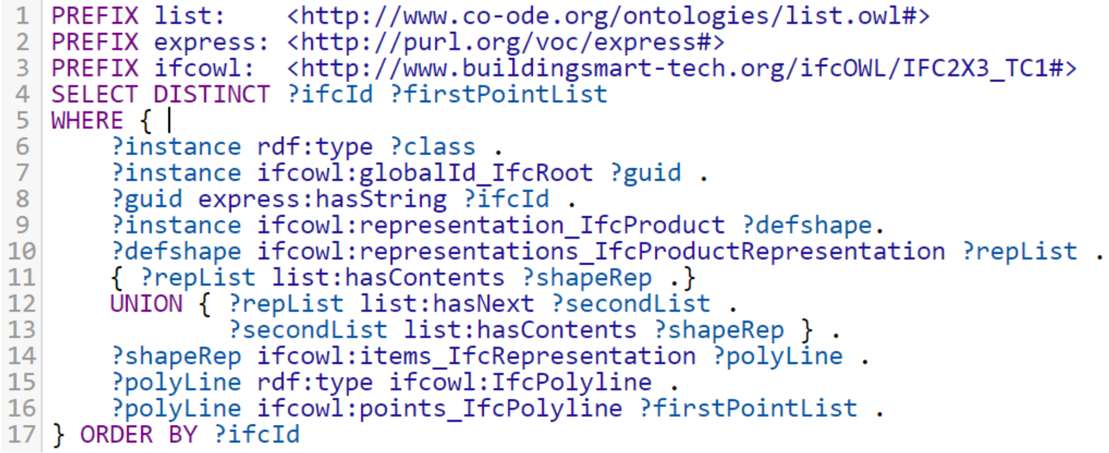
Question	Which are the instances with unique identifiers which are defined by an arbitrary shape, mapped to a source object?

```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?profileType ?firstPointList ?xAxisDir ?yAxisDir ?zAxisDir
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape.
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList .
    { ?repList list:hasContents ?shapeRep .}
    UNION { ?repList list:hasNext ?secondList .
            ?secondList list:hasContents ?shapeRep } .
    ?shapeRep ifcowl:items_IfcRepresentation ?mappedItem .
    ?mappedItem ifcowl:mappingSource_IfcMappedItem ?repMap .
    ?repMap ifcowl:mappedRepresentation_IfcRepresentationMap ?shapeRep2 .
    ?shapeRep2 ifcowl:items_IfcRepresentation ?extAreaSolid .
    ?extAreaSolid ifcowl:sweptArea_IfcSweptAreaSolid ?profDef .
    ?profDef rdf:type  ?profileType .
    ?profDef ifcowl:outerCurve_IfcArbitraryClosedProfileDef ?polyLine .
    ?polyLine ifcowl:points_IfcPolyline ?firstPointList .
    OPTIONAL{
        ?exAreaSolid ifcowl:position_IfcSweptAreaSolid ?axis2p3d .
        ?axis2p3d ifcowl:axis_IfcAxis2Placement3D ?axisDir .
        ?axisDir ifcowl:directionRatios_IfcDirection ?dirList .
        ?dirList list:hasContents ?xMeasure .
        ?xMeasure express:hasDouble ?xAxisDir .
        ?dirList list:hasNext ?dirList2 .
        ?dirList2 list:hasContents ?yMeasure .
        ?yMeasure express:hasDouble ?yAxisDir .
        ?dirList2 list:hasNext ?dirList3 .
        ?dirList3 list:hasContents ?zMeasure .
        ?zMeasure express:hasDouble ?zAxisDir .}
FILTER (?class = ifcowl:IfcColumn)
} ORDER BY ?ifcId
```

Function	Identifies objects with an arbitrary shape definition and retrieves the first point and its coordinates, which is part of a finite list of points used to define an arbitrary perimeter. This is nearly identical to 10, however the geometry is mapped to another source object, for storage reasons.	
	<u>IFC concepts of interest:</u> IfcIdentifier, IfcProduct, IfcRepresentation, IfcProductRepresentation, IfcSweptAreaSolid, IfcArbitraryClosedProfileDefinition, IfcPolyline, IfcAxis2Placement3D, IfcCartesianPoint, IfcDirection, IfcMappedItem	Requires reasoning? NO

Q-IFC-12, Get Polyline first point

Question: What are the instances with unique identifiers which are defined with a polyline shape?



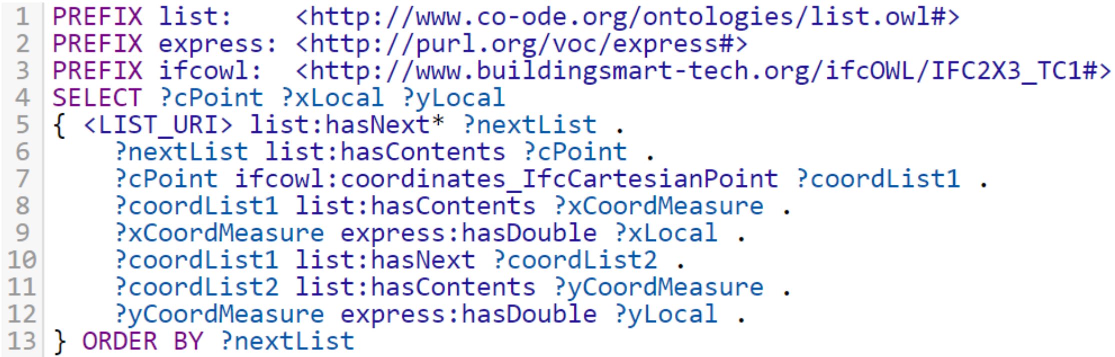
```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?firstPointList
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape.
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList .
    { ?repList list:hasContents ?shapeRep .}
    UNION { ?repList list:hasNext ?secondList .
            ?secondList list:hasContents ?shapeRep } .
    ?shapeRep ifcowl:items_IfcRepresentation ?polyLine .
    ?polyLine rdf:type ifcowl:IfcPolyline .
    ?polyLine ifcowl:points_IfcPolyline ?firstPointList .
} ORDER BY ?ifcId
```

Function: Identifies objects represented by polylines and retrieves the first point from a list of segments which define a curve made from n-1 linear segments.

IFC concepts of interest:	Requires reasoning?
IfcIdentifier, IfcProduct, IfcRepresentation, IfcProductRepresentation, IfcPolyline	NO

Q-IFC-13, Get Polyline next points

Question: What are the coordinate points of the next point on a polyline?



```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT ?cPoint ?xLocal ?yLocal
{ <LIST_URI> list:hasNext* ?nextList .
    ?nextList list:hasContents ?cPoint .
    ?cPoint ifcowl:coordinates_IfcCartesianPoint ?coordList1 .
    ?coordList1 list:hasContents ?xCoordMeasure .
    ?xCoordMeasure express:hasDouble ?xLocal .
    ?coordList1 list:hasNext ?coordList2 .
    ?coordList2 list:hasContents ?yCoordMeasure .
    ?yCoordMeasure express:hasDouble ?yLocal .
} ORDER BY ?nextList
```

Function: It is used in conjuction with 12 to retrieve polyline points from a nested list. It iteratively goes through all the points and retrieves their coordinates in (x,y,z). A variable “LIST_URI” has to be provided for each object defined by a polyline.

IFC concepts of interest:	Requires reasoning?
IfcCartesianPoint	NO

Q-IFC-14, Get IFC BREP shapes

Question	Which are the instances with unique identifiers that are represented by faceted breps and polyloops?

```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?shapeRep ?face ?polyloop ?orientation ?firstPointList
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape.
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList .
    { ?repList list:hasContents ?shapeRep .}
    UNION { ?repList list:hasNext ?secondList .
            ?secondList list:hasContents ?shapeRep } .
    {?shapeRep ifcowl:items_IfcRepresentation ?facetedBrep . }
    UNION{ ?shapeRep ifcowl:items_IfcRepresentation ?boolItem .
           ?boolItem ifcowl:firstOperand_IfcBooleanResult ?facetedBrep .}
    ?facetedBrep ifcowl:outer_IfcManifoldSolidBrep ?closedShell .
    ?closedShell ifcowl:cfsFaces_IfcConnectedFaceSet ?face .
    ?face ifcowl:bounds_IfcFace ?faceOuterBound .
    ?faceOuterBound ifcowl:bound_IfcFaceBound ?polyloop .
    ?faceOuterBound ifcowl:orientation_IfcFaceBound ?boolean .
    ?boolean express:hasBoolean ?orientation .
    ?polyloop ifcowl:polygon_IfcPolyLoop ?firstPointList .
} ORDER BY ?ifcId
```

Function	Finds objects which have a geometry represented by breps faces and polylines. Each face is composed of polyline, and the entire object is composed by several faces. Retrieves the URI for the polyloops, which are queried at 16.	
	IFC concepts of interest: IfcIdentifier, IfcProduct, IfcRepresentation, IfcProductRepresentation, IfcManifoldSolidBrep, IfcFace, IfcPolyLoop	Requires reasoning? NO

Q-IFC-15, Get IFC BREP shapes (mapped)

Question	Which are the instances with unique identifiers that have brep shapes mapped to other source objects?

```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?profileType ?face ?polyloop ?orientation ?firstPointList
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape.
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList .
    { ?repList list:hasContents ?shapeRep .}
    UNION { ?repList list:hasNext ?secondList .
            ?secondList list:hasContents ?shapeRep } .
            ?shapeRep ifcowl:items_IfcRepresentation ?mappedItem .
    ?mappedItem ifcowl:mappingSource_IfcMappedItem ?repMap .
    ?repMap ifcowl:mappedRepresentation_IfcRepresentationMap ?shapeRep2 .
    ?shapeRep2 ifcowl:items_IfcRepresentation ?facetedBrep .
    ?facetedBrep rdf:type ?profileType .
    ?facetedBrep ifcowl:outer_IfcManifoldSolidBrep ?closedShell .
    ?closedShell ifcowl:cfsFaces_IfcConnectedFaceSet ?face .
    ?face ifcowl:bounds_IfcFace ?faceOuterBound .
    ?faceOuterBound ifcowl:bound_IfcFaceBound ?polyloop .
    ?faceOuterBound ifcowl:orientation_IfcFaceBound ?boolean .
    ?boolean express:hasBoolean ?orientation .
    ?polyloop ifcowl:polygon_IfcPolyLoop ?firstPointList .
} ORDER BY ?ifcId
```

Function	Finds objects which have a geometry represented by breps faces and polylines. Similar to 14, however the brep shape belongs to a source object to which other objects map to. Each face is composed of polyline, and the entire object is

composed by several faces. Retrieves the URI for the polyloops, which are queried at 16.

IFC concepts of interest:	Requires reasoning?
IfcIdentifier, IfcProduct, IfcRepresentation, IfcProductRepresentation, IfcManifoldSolidBrep, IfcFace, IfcPolyLoop, IfcMappedItem	NO

Q-IFC-16, Get Polyloop next points

Question: What are the coordinates of the next point on the specified polyloop?

```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT ?cPoint ?xCoord ?yCoord ?zCoord
{ <LIST_URI> list:hasNext* ?nextList .
    ?nextList list:hasContents ?cPoint .
    ?cPoint ifcowl:coordinates_IfcCartesianPoint ?coordList1 .
    ?coordList1 list:hasContents ?xCoordMeasure .
    ?xCoordMeasure express:hasDouble ?xCoord .
    ?coordList1 list:hasNext ?coordList2 .
    ?coordList2 list:hasContents ?yCoordMeasure .
    ?yCoordMeasure express:hasDouble ?yCoord .
    ?coordList2 list:hasNext ?coordList3 .
    ?coordList3 list:hasContents ?zCoordMeasure .
    ?zCoordMeasure express:hasDouble ?zCoord .
} ORDER BY ?nextList
```

Function: It is used in conjunction with 14 or 15 to retrieve polyloop points from a nested list. It iteratively goes through all the points and retrieves their coordinates in (x,y,z). A variable “LIST_URI” has to be provided for each object defined by a polyloop.

IFC concepts of interest:	Requires reasoning?
IfcPolyLoop, IfcCartesianPoint	NO

Q-IFC-17, Get IFC Extrusions

Question: Which are the instances with unique identifiers that have extrusions for their defined geometry shape?

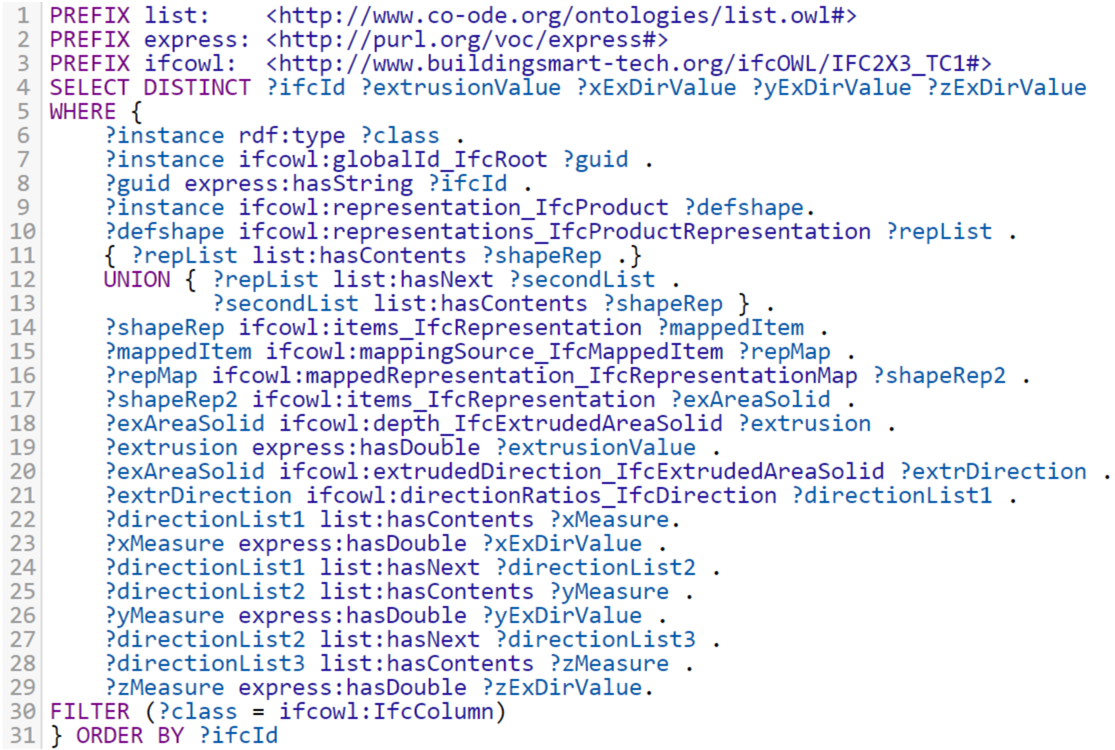
```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?extrusionValue ?xExDirValue ?yExDirValue ?zExDirValue
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape.
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList .
    { ?repList list:hasContents ?shapeRep .}
    UNION { ?repList list:hasNext ?secondList .
            ?secondList list:hasContents ?shapeRep } .
    {?shapeRep ifcowl:items_IfcRepresentation ?exAreaSolid .
     ?exAreaSolid ifcowl:sweptArea_IfcSweptAreaSolid ?profDef .}
    UNION{ ?shapeRep ifcowl:items_IfcRepresentation ?boolItem .
           ?boolItem ifcowl:firstOperand_IfcBooleanResult ?exAreaSolid .}
    ?exAreaSolid ifcowl:depth_IfcExtrudedAreaSolid ?extrusion .
    ?extrusion express:hasDouble ?extrusionValue .
    ?exAreaSolid ifcowl:extrudedDirection_IfcExtrudedAreaSolid ?extrDirection .
    ?extrDirection ifcowl:directionRatios_IfcDirection ?directionList1 .
    ?directionList1 list:hasContents ?xMeasure.
    ?xMeasure express:hasDouble ?xExDirValue .
    ?directionList1 list:hasNext ?directionList2 .
    ?directionList2 list:hasContents ?yMeasure .
    ?yMeasure express:hasDouble ?yExDirValue .
    ?directionList2 list:hasNext ?directionList3 .
    ?directionList3 list:hasContents ?zMeasure .
    ?zMeasure express:hasDouble ?zExDirValue.
} ORDER BY ?ifcId
```

Function: Finds objects which have a representation in 3D, usually described by an extrusion length and direction from their original 2D shape.

	IFC concepts of interest: IfcIdentifier, IfcProduct, IfcRepresentation, IfcProductRepresentation, IfcSweptAreaSolid, IfcExtrudedAreaSolid, IfcDirection, IfcCartesianPoint	Requires reasoning? NO

Q-IFC-18, Get IFC Extrusions (mapped)

Question	Which are the instances with unique identifiers that have extrusions for their mapped geometry shape?

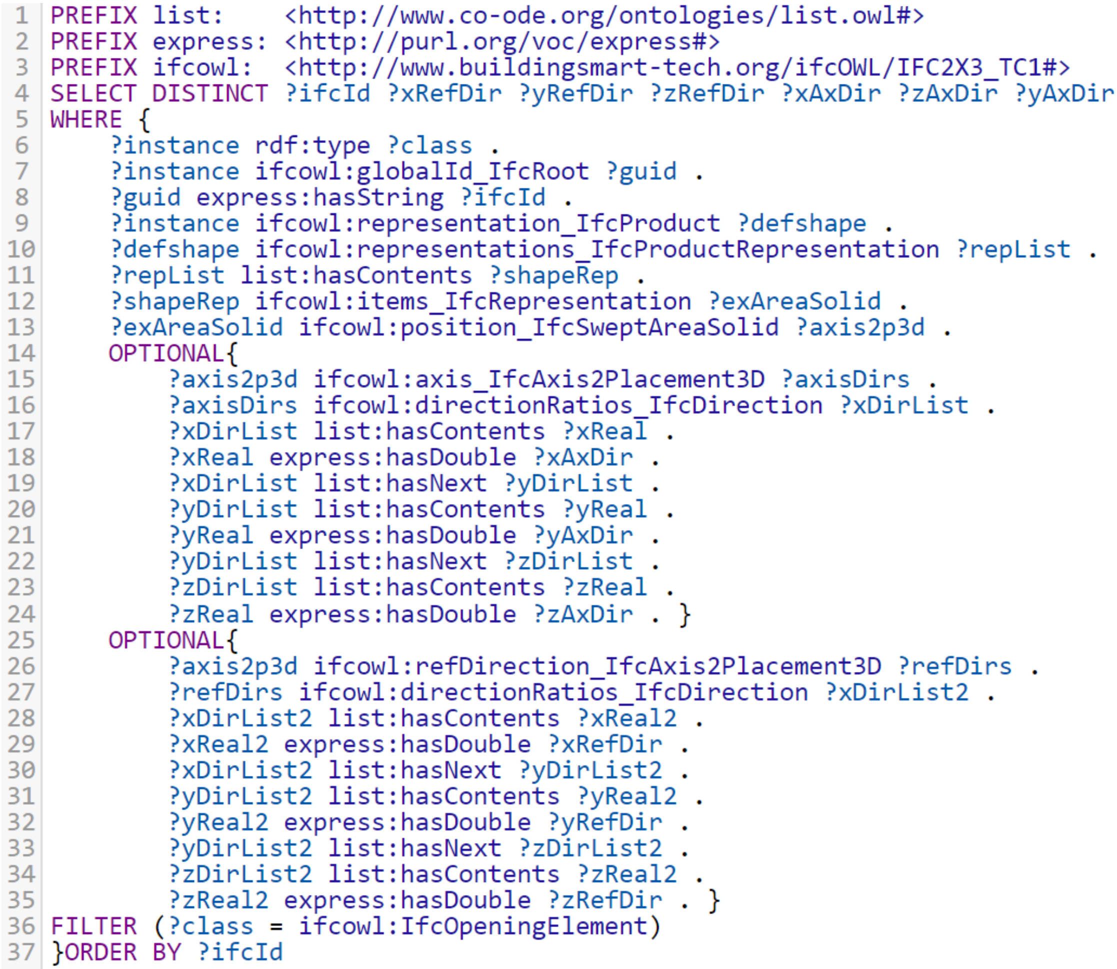


```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?extrusionValue ?xExDirValue ?yExDirValue ?zExDirValue
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape.
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList .
    { ?repList list:hasContents ?shapeRep .}
    UNION { ?repList list:hasNext ?secondList .
            ?secondList list:hasContents ?shapeRep } .
    ?shapeRep ifcowl:items_IfcRepresentation ?mappedItem .
    ?mappedItem ifcowl:mappingSource_IfcMappedItem ?repMap .
    ?repMap ifcowl:mappedRepresentation_IfcRepresentationMap ?shapeRep2 .
    ?shapeRep2 ifcowl:items_IfcRepresentation ?exAreaSolid .
    ?exAreaSolid ifcowl:depth_IfcExtrudedAreaSolid ?extrusion .
    ?extrusion express:hasDouble ?extrusionValue .
    ?exAreaSolid ifcowl:extrudedDirection_IfcExtrudedAreaSolid ?extrDirection .
    ?extrDirection ifcowl:directionRatios_IfcDirection ?directionList1 .
    ?directionList1 list:hasContents ?xMeasure.
    ?xMeasure express:hasDouble ?xExDirValue .
    ?directionList1 list:hasNext ?directionList2 .
    ?directionList2 list:hasContents ?yMeasure .
    ?yMeasure express:hasDouble ?yExDirValue .
    ?directionList2 list:hasNext ?directionList3 .
    ?directionList3 list:hasContents ?zMeasure .
    ?zMeasure express:hasDouble ?zExDirValue.
FILTER (?class = ifcowl:IfcColumn)
} ORDER BY ?ifcId
```

Function	Finds objects which have a representation in 3D, usually described by an extrusion length and direction from their original 2D shape. It is nearly identical to 16, however these objects use a different source object to copy its shape.	
	IFC concepts of interest: IfcIdentifier, IfcProduct, IfcRepresentation, IfcProductRepresentation, IfcSweptAreaSolid, IfcExtrudedAreaSolid, IfcDirection, IfcCartesianPoint, IfcMappedItem	Requires reasoning? NO

Q-IFC-19, Get IFC Orientations

Question	Which are the IfcOpeningElements with unique identifiers with a rectangular shaped definition?

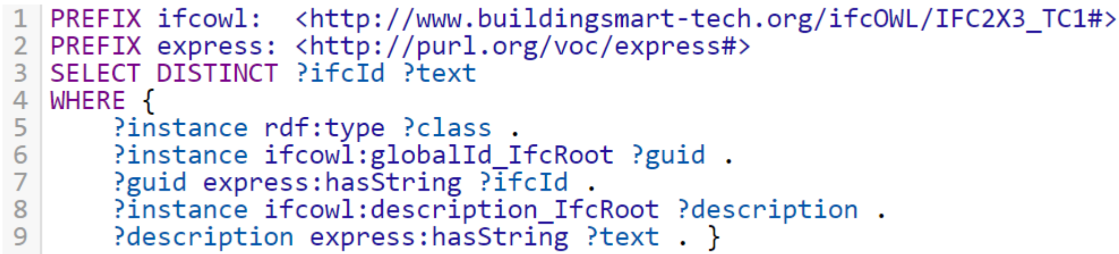


```
PREFIX list:    <http://www.co-ode.org/ontologies/list.owl#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
SELECT DISTINCT ?ifcId ?xRefDir ?yRefDir ?zRefDir ?xAxDir ?zAxDir ?yAxDir
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:representation_IfcProduct ?defshape .
    ?defshape ifcowl:representations_IfcProductRepresentation ?repList .
    ?repList list:hasContents ?shapeRep .
    ?shapeRep ifcowl:items_IfcRepresentation ?exAreaSolid .
    ?exAreaSolid ifcowl:position_IfcSweptAreaSolid ?axis2p3d .
    OPTIONAL{
        ?axis2p3d ifcowl:axis_IfcAxis2Placement3D ?axisDirs .
        ?axisDirs ifcowl:directionRatios_IfcDirection ?xDirList .
        ?xDirList list:hasContents ?xReal .
        ?xReal express:hasDouble ?xAxDir .
        ?xDirList list:hasNext ?yDirList .
        ?yDirList list:hasContents ?yReal .
        ?yReal express:hasDouble ?yAxDir .
        ?yDirList list:hasNext ?zDirList .
        ?zDirList list:hasContents ?zReal .
        ?zReal express:hasDouble ?zAxDir . }
    OPTIONAL{
        ?axis2p3d ifcowl:refDirection_IfcAxis2Placement3D ?refDirs .
        ?refDirs ifcowl:directionRatios_IfcDirection ?xDirList2 .
        ?xDirList2 list:hasContents ?xReal2 .
        ?xReal2 express:hasDouble ?xRefDir .
        ?xDirList2 list:hasNext ?yDirList2 .
        ?yDirList2 list:hasContents ?yReal2 .
        ?yReal2 express:hasDouble ?yRefDir .
        ?yDirList2 list:hasNext ?zDirList2 .
        ?zDirList2 list:hasContents ?zReal2 .
        ?zReal2 express:hasDouble ?zRefDir . }
FILTER (?class = ifcowl:IfcOpeningElement)
}ORDER BY ?ifcId
```

Function	Finds openings in walls and retrieves their shapes and basic points for geometry construction. Openings are usually rectangular in nature, thus most of the door and window openings are found.	
	IFC concepts of interest: IfcIdentifier, IfcProduct, IfcRepresentation, IfcProductRepresentation, IfcSweptAreaSolid, IfcAxis2Placement3D, IfcDirection, IfcCartesianPoint	Requires reasoning? NO

Q-IFC-20, Get descriptions

Question	Which instances with unique identifiers have textual descriptions attached to them? And what is this description?

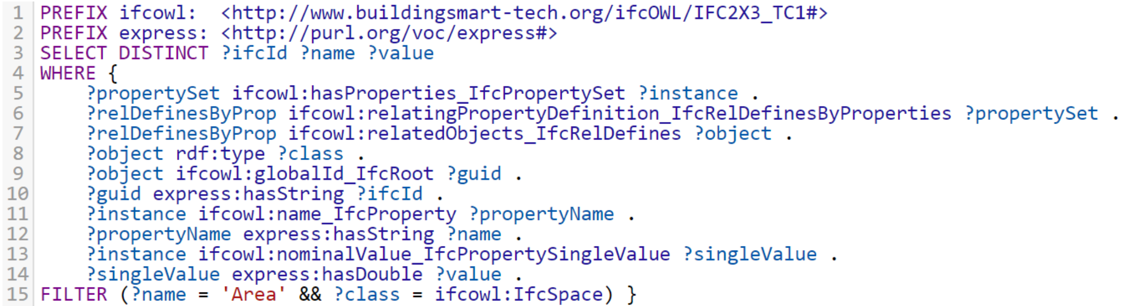


```
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
PREFIX express: <http://purl.org/voc/express#>
SELECT DISTINCT ?ifcId ?text
WHERE {
    ?instance rdf:type ?class .
    ?instance ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:description_IfcRoot ?description .
    ?description express:hasString ?text . }
```

Function	Looks for objects which have a specific description property, which is can be either inputted by the users in a BIM tool or be available from the product factory specifications.	
	<u>IFC concepts of interest:</u> IfcIdentifier, IfcText	Requires reasoning? NO

Q-IFC-21, Get areas

Question	Which are the IfcSpace type instances with unique identifiers that have a property defined “Area” and what is its value?

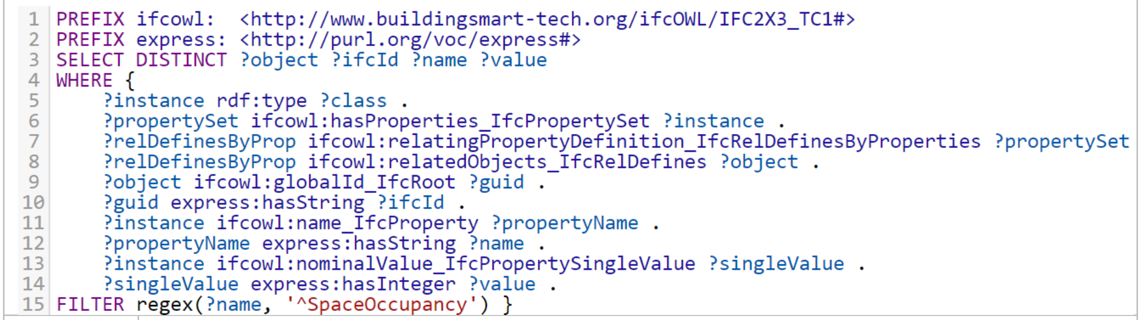


```
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
PREFIX express: <http://purl.org/voc/express#>
SELECT DISTINCT ?ifcId ?name ?value
WHERE {
    ?propertySet ifcowl:hasProperties_IfcPropertySet ?instance .
    ?relDefinesByProp ifcowl:relatingPropertyDefinition_IfcRelDefinesByProperties ?propertySet .
    ?relDefinesByProp ifcowl:relatedObjects_IfcRelDefines ?object .
    ?object rdf:type ?class .
    ?object ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:name_IfcProperty ?propertyName .
    ?propertyName express:hasString ?name .
    ?instance ifcowl:nominalValue_IfcPropertySingleValue ?singleValue .
    ?singleValue express:hasDouble ?value .
FILTER (?name = 'Area' && ?class = ifcowl:IfcSpace) }
```

Function	Finds the spaces which have a specifically named property defined, which in this case is labelled “Area”. This is an IfcPropertySingleValue which usually has a name and a value defined. These types of properties differ from one BIM tool to another and they are also frequently defined by users to attach specific information about objects.	
	<u>IFC concepts of interest:</u> IfcIdentifier, IfcPropertySet, IfcRelDefinesByProperties, IfcPropertySingleValue.	Requires reasoning? NO

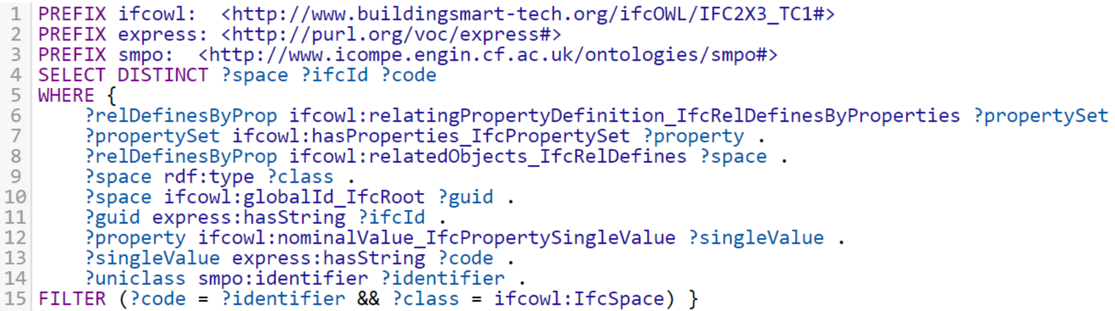
Table C-2. SPARQL queries operating on CSS and other resources

<table>
<tr><td colspan="2">SPARQL QUERIES</td></tr>
<tr><td colspan="2">Q-RES-1, Get occupancy</td></tr>
<tr><td>Question</td><td>Which are the instances with unique identifiers in the IFC model that have a property named “SpaceOccupancy” and what is its value?</td></tr>
<tr><td colspan="2">



```
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
PREFIX express: <http://purl.org/voc/express#>
SELECT DISTINCT ?object ?ifcId ?name ?value
WHERE {
    ?instance rdf:type ?class .
    ?propertySet ifcowl:hasProperties_IfcPropertySet ?instance .
    ?relDefinesByProp ifcowl:relatingPropertyDefinition_IfcRelDefinesByProperties ?propertySet
    ?relDefinesByProp ifcowl:relatedObjects_IfcRelDefines ?object .
    ?object ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?instance ifcowl:name_IfcProperty ?propertyName .
    ?propertyName express:hasString ?name .
    ?instance ifcowl:nominalValue_IfcPropertySingleValue ?singleValue .
    ?singleValue express:hasInteger ?value .
FILTER regex(?name, '^SpaceOccupancy') }
```

</td></tr>
<tr><td rowspan="2">Function</td><td>Finds the IfcSinglePropertyValue labelled “SpaceOccupancy” and the objects to which it is attached. Retrieves this value which is used to save it in memory. This value is an integer which represents the number of people that occupy a space, which can be attached by designers to building spaces.</td></tr>
<tr><td>Requires reasoning?
NO</td></tr>
<tr><td colspan="2">Q-RES-2, Get classifications</td></tr>
<tr><td>Question</td><td>Which are the IfcSpace type instances with unique identifiers in the IFC model that have a property which matches existing Uniclass2015 codes in available resources?</td></tr>
<tr><td colspan="2">

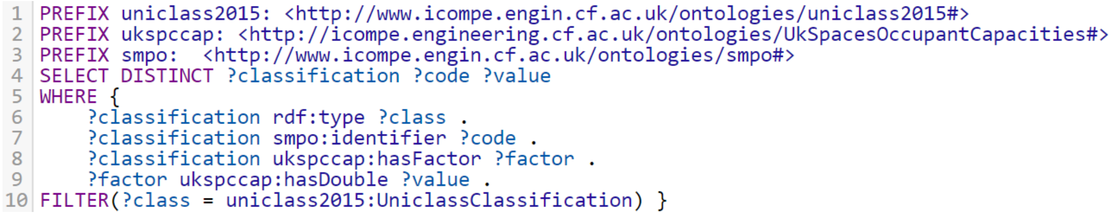


```
PREFIX ifcowl:  <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
PREFIX express: <http://purl.org/voc/express#>
PREFIX smpo:  <http://www.icompe.engin.cf.ac.uk/ontologies/smpo#>
SELECT DISTINCT ?space ?ifcId ?code
WHERE {
    ?relDefinesByProp ifcowl:relatingPropertyDefinition_IfcRelDefinesByProperties ?propertySet
    ?propertySet ifcowl:hasProperties_IfcPropertySet ?property .
    ?relDefinesByProp ifcowl:relatedObjects_IfcRelDefines ?space .
    ?space rdf:type ?class .
    ?space ifcowl:globalId_IfcRoot ?guid .
    ?guid express:hasString ?ifcId .
    ?property ifcowl:nominalValue_IfcPropertySingleValue ?singleValue .
    ?singleValue express:hasString ?code .
    ?uniclass smpo:identifier ?identifier .
FILTER (?code = ?identifier && ?class = ifcowl:IfcSpace) }
```

</td></tr>
<tr><td rowspan="2">Function</td><td>Finds the IfcSinglePropertyValue which has identity data related to the Uniclass2015 classification, saved in the variable “?identifier” and the IfcSpace objects to which it is attached. Reasons if this value is present in the Uniclass2015 dataset ontology graphs and returns only those instances to which the classification can be confirmed.</td></tr>
<tr><td>Requires reasoning?
YES</td></tr>
</table>

Q-RES-3, Match occupancy factors

Question	What are the UK regulations spaces occupancy factors for Uniclass2015 classification codes?

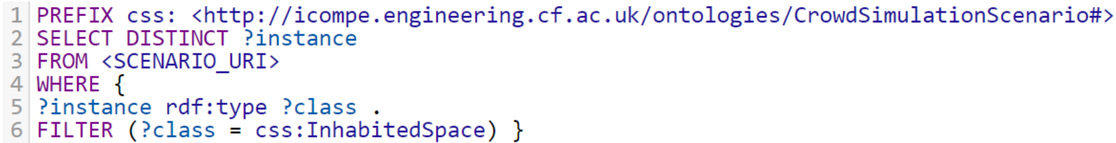


```
PREFIX uniclass2015: <http://www.icompe.engin.cf.ac.uk/ontologies/uniclass2015#>
PREFIX ukspccap: <http://icompe.engineering.cf.ac.uk/ontologies/UkSpacesOccupantCapacities#>
PREFIX smpo:  <http://www.icompe.engin.cf.ac.uk/ontologies/smpo#>
SELECT DISTINCT ?classification ?code ?value
WHERE {
    ?classification rdf:type ?class .
    ?classification smpo:identifier ?code .
    ?classification ukspccap:hasFactor ?factor .
    ?factor ukspccap:hasDouble ?value .
FILTER(?class = uniclass2015:UniclassClassification) }
```

Function	Finds the values of the occupancy factors for each Uniclass2015 classification code in the building model. Reasons in conjunction with the alignment rules from Appendix B6.
	Requires reasoning? YES

Q-RES-4, Find inhabited spaces

Question	Which are the instances from a specific scenario that are classified as a InhabitedSpace class in the CSS ontology?

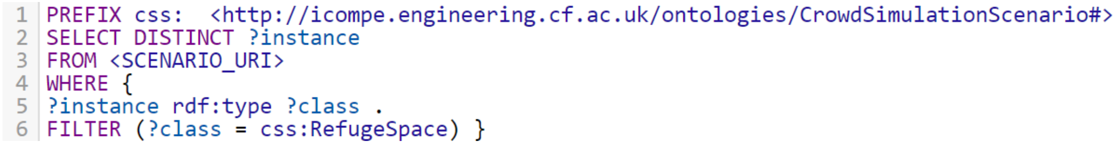


```
PREFIX css: <http://icompe.engineering.cf.ac.uk/ontologies/CrowdSimulationScenario#>
SELECT DISTINCT ?instance
FROM <SCENARIO_URI>
WHERE {
?instance rdf:type ?class .
FILTER (?class = css:InhabitedSpace) }
```

Function	Finds the instance that are reasoned by rules in Appendix C, Table C3 as “InhabitedSpace” class.
	Requires reasoning? YES

Q-RES-5, Find exit spaces

Question	Which are the instances from a specific scenario that are classified as a RefugeSpace class in the CSS ontology?



```
PREFIX css:  <http://icompe.engineering.cf.ac.uk/ontologies/CrowdSimulationScenario#>
SELECT DISTINCT ?instance
FROM <SCENARIO_URI>
WHERE {
?instance rdf:type ?class .
FILTER (?class = css:RefugeSpace) }
```

Function	Finds the instance that are reasoned by rules in Appendix C, Table C3 as “RefugeSpace” class.
	Requires reasoning? YES

Appendix D – ONTOCS system interface

This section showcases the workflow process which guides the designers through the developed web interface. The system provides the entire experience as a service, running from an internet browser. The workflow process reflects the steps described in Section 7.1.1. Additionally, it includes some steps where users can validate the models to ensure the system correctly reconstructs the IFC models, and that the scenarios are correctly created.

Figures 7-3 to 7-10 show the separate navigation web pages which guide the user through the process of generating and analysing scenarios on a large scale. The interface has an important role in the capabilities in which users can contribute to the context of each simulation, as can be seen in Figure 7-6, or Figure 7-9. The input of the user assumptions and objectives are important in involving the designers as much as possible in the process, to help the system configure more realistic scenarios, and more relevant to each situation under analysis.

The main limitation of the system lies in its ability to ensure the validity of the generated models, which was addressed by allowing users to download each model file and manually check it. During testing, it was observed that if certain sources of data is missing, some scenarios can be incomplete, thus these cannot be executed by simulation tools. This is a cause of the OWA of ontologies and rules implemented. In such situations, the interface can be used demand correct user input. For example, a scenario cannot be created without specifying where to look for the population data (IFC model or Design codes, etc); however, if the data is missing from the start, the scenario can still be generated, but with no population. This results in an empty model, without any event objects, which will eventually provide no results for analysis.

The interface shows only some of the capability of an ontology-based system. The main benefit, which can be seen in Figures 7-9 and 7-10, is the ability to aggregate the data across multiple scenarios on large scales, as was presented conceptually in Section 4.2.2. Because the data is connected, the extent to which information and knowledge about the design can be generated is also dependent on the level of interface implementation.

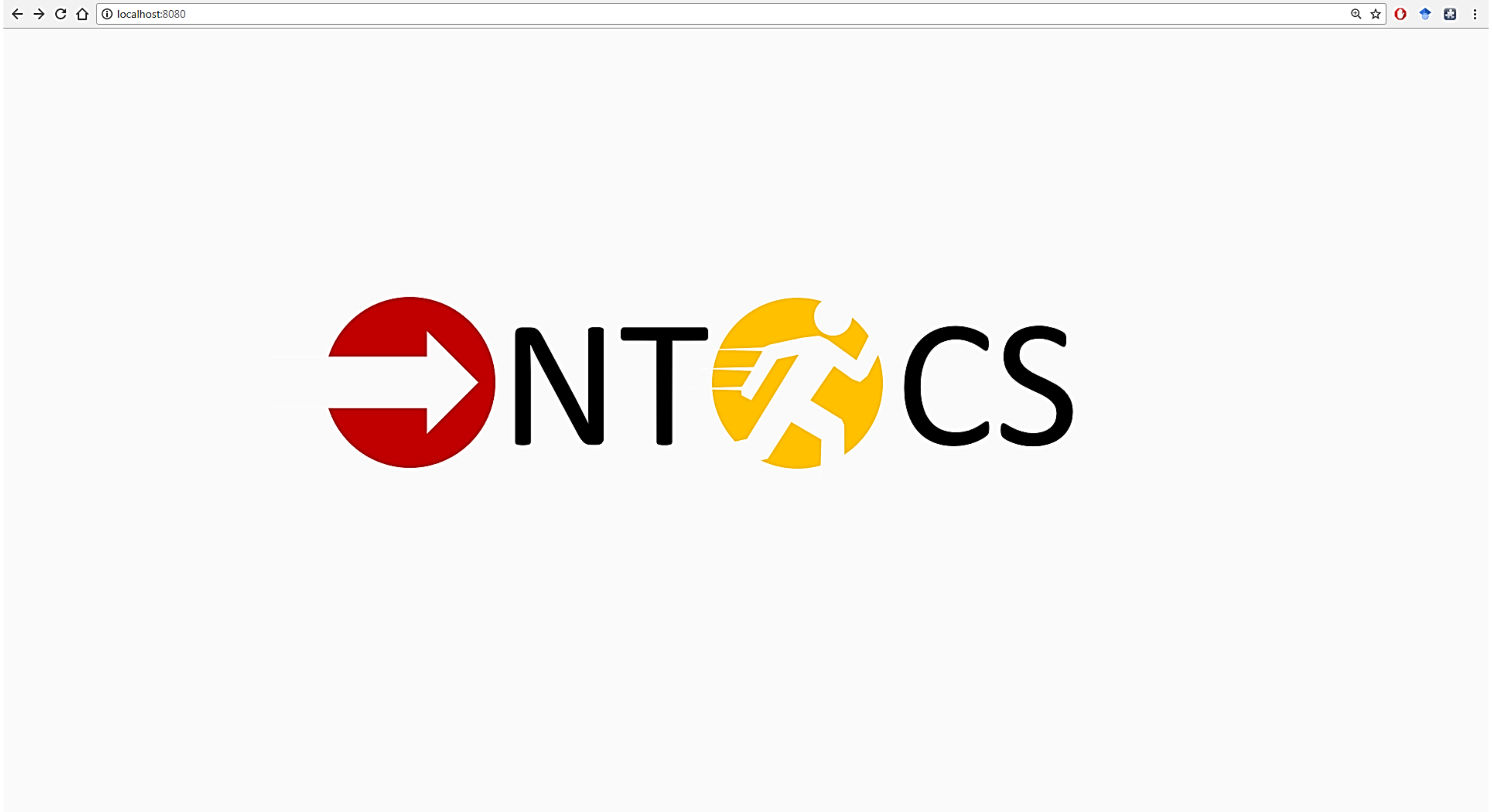
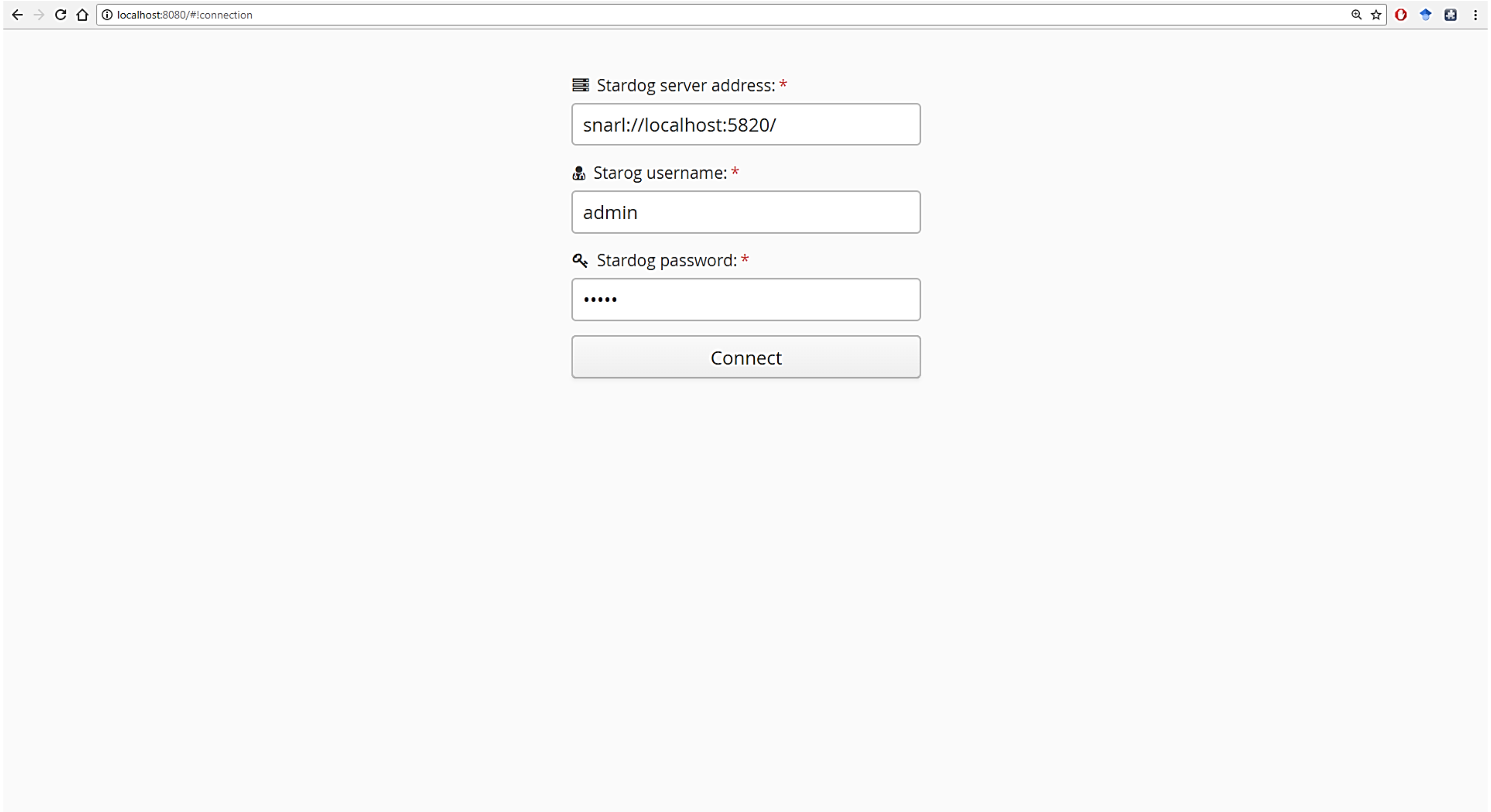


Figure D-1. ONTOCS welcome page



Stardog server address: *

snarl://localhost:5820/

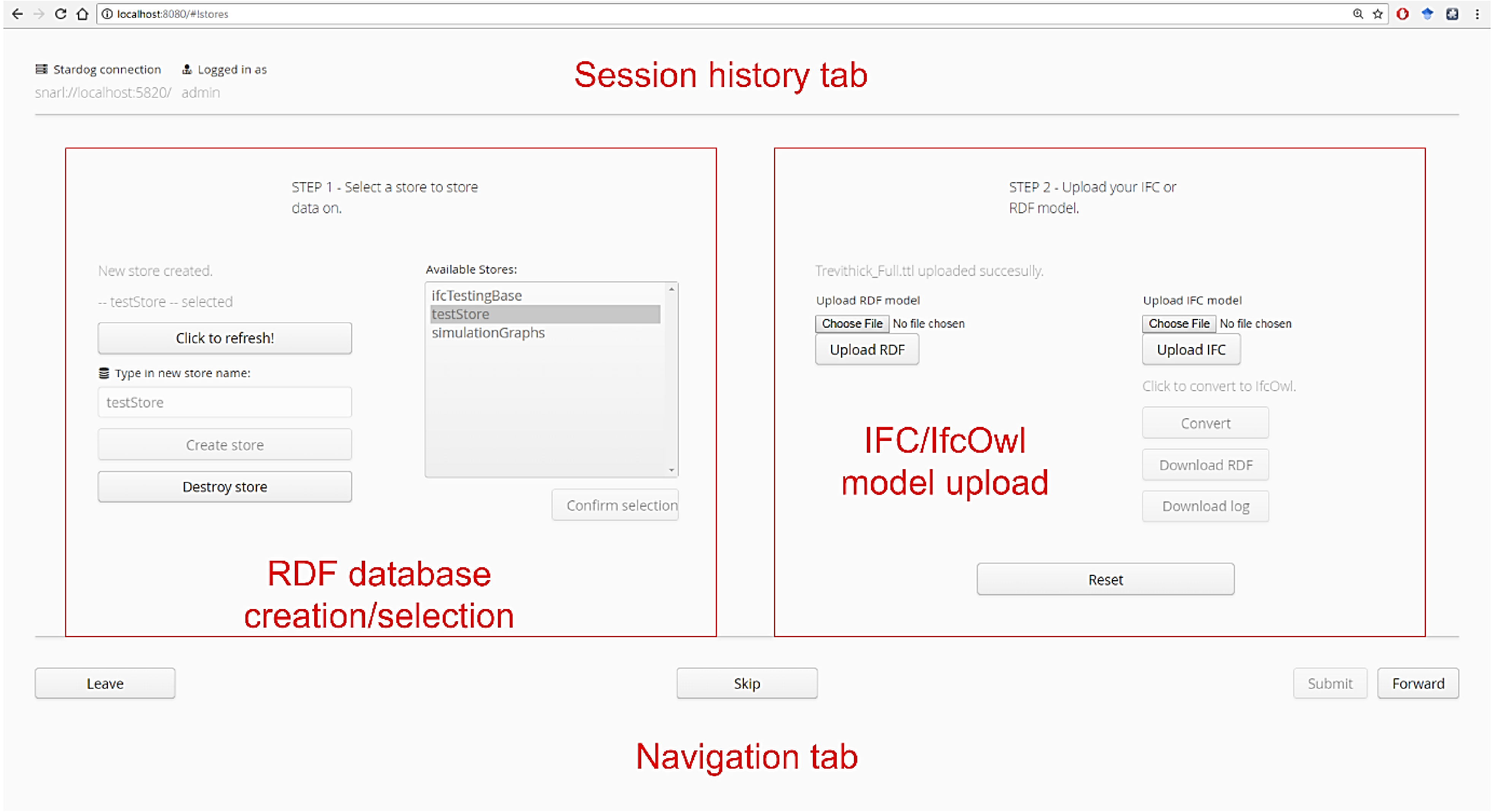
Starog username: *

admin

Stardog password: *

•••••

Figure D-2. ONTOCS user log in page



STEP 1 - Select a store to store data on.

New store created.

-- testStore -- selected

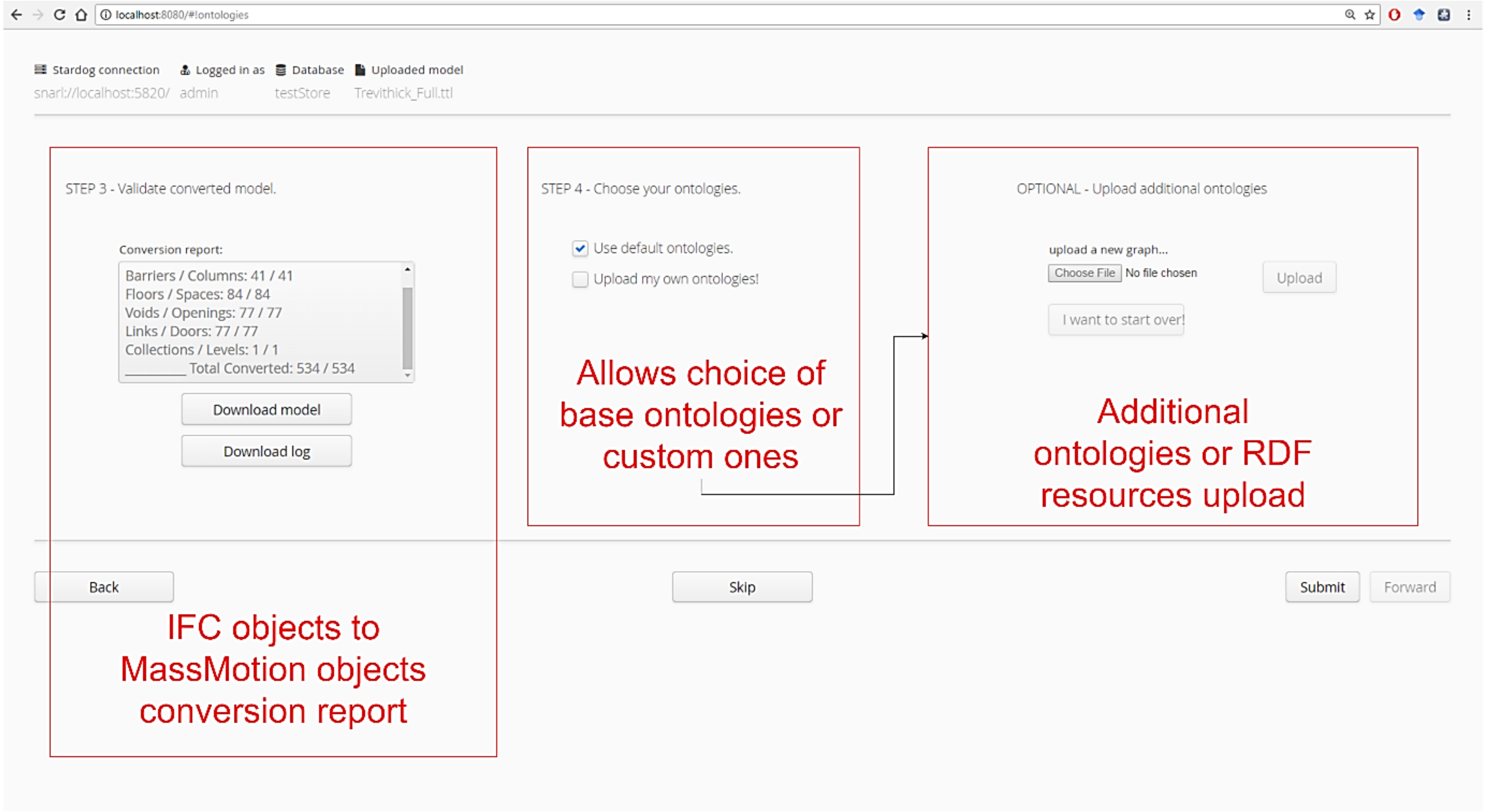
Type in new store name: testStore

STEP 2 - Upload your IFC or RDF model.

Trevithick_Full.ttl uploaded succesully.

Click to convert to IfcOwl.

Figure D-3. ONTOCS database selection and IFC model upload page



localhost:8080/#!ontologies

Stardog connection: snarl://localhost:5820/
Logged in as: admin
Database: testStore
Uploaded model: Trevithick_Full.ttl

STEP 3 - Validate converted model.

Conversion report:

```
Barriers / Columns: 41 / 41
Floors / Spaces: 84 / 84
Voids / Openings: 77 / 77
Links / Doors: 77 / 77
Collections / Levels: 1 / 1
________ Total Converted: 534 / 534
```

STEP 4 - Choose your ontologies.

☑ Use default ontologies.

□ Upload my own ontologies!

OPTIONAL - Upload additional ontologies

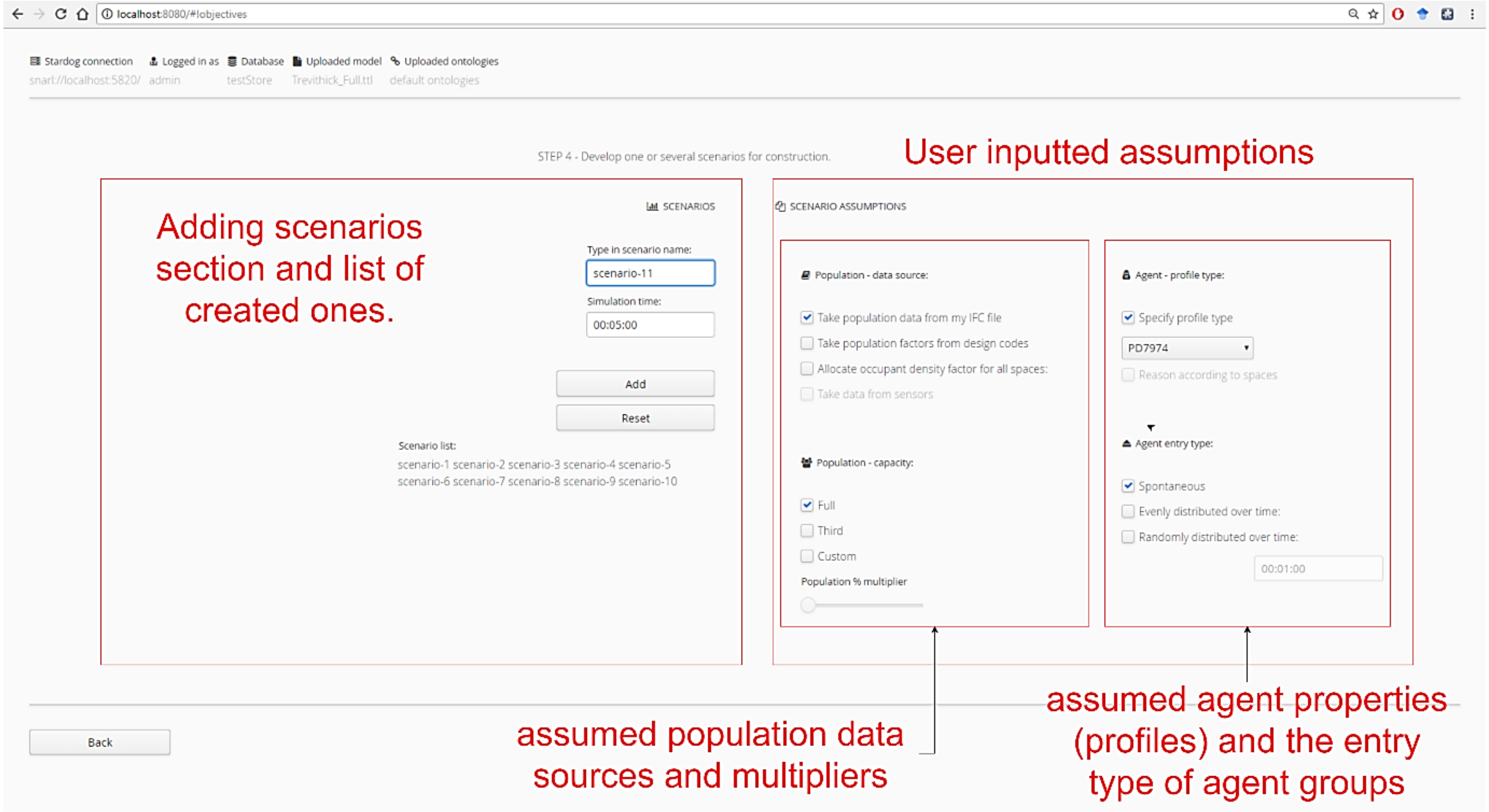
upload a new graph... No file chosen

IFC objects to MassMotion objects conversion report

Allows choice of base ontologies or custom ones

Additional ontologies or RDF resources upload

Figure D-4. ONTOCS geometry conversion report from IFC to MassMotion and ontology upload page



localhost:8080/#!objectives

Stardog connection: snarl://localhost:5820/
Logged in as: admin
Database: testStore
Uploaded model: Trevithick_Full.ttl
Uploaded ontologies: default ontologies

STEP 4 - Develop one or several scenarios for construction.

User inputted assumptions

Adding scenarios section and list of created ones.

SCENARIOS

Type in scenario name: scenario-11

Simulation time: 00:05:00

Scenario list:
scenario-1 scenario-2 scenario-3 scenario-4 scenario-5
scenario-6 scenario-7 scenario-8 scenario-9 scenario-10

SCENARIO ASSUMPTIONS

Population - data source:

☑ Take population data from my IFC file
□ Take population factors from design codes
□ Allocate occupant density factor for all spaces:
□ Take data from sensors

Population - capacity:

☑ Full
□ Third
□ Custom

Population % multiplier

Agent - profile type:

☑ Specify profile type
PD7974
□ Reason according to spaces

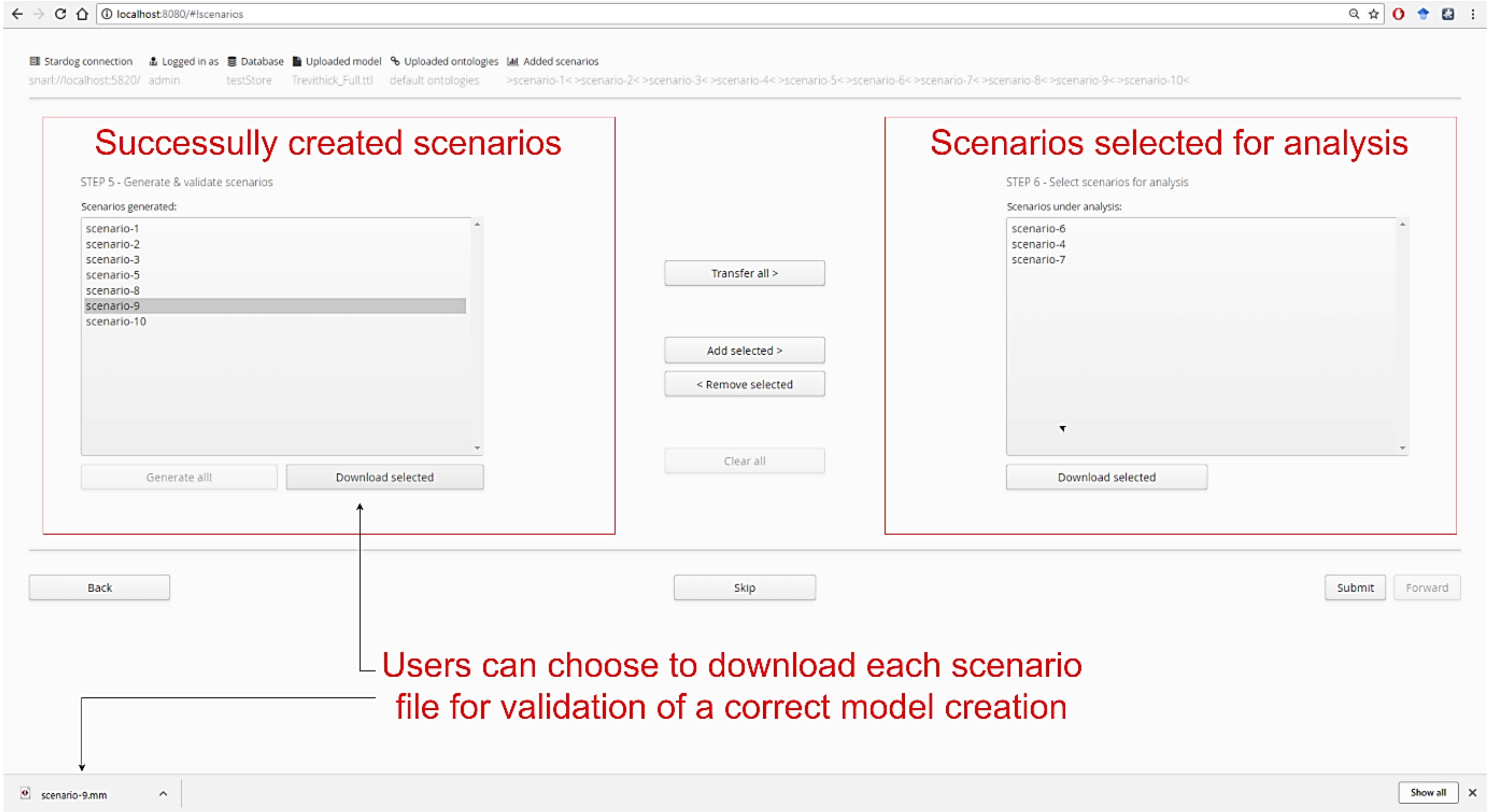
Agent entry type:

☑ Spontaneous
□ Evenly distributed over time:
□ Randomly distributed over time:
00:01:00

assumed population data sources and multipliers

assumed agent properties (profiles) and the entry type of agent groups

Figure D-5. ONTOCS scenario creation page



localhost:8080/#!scenarios

Stardog connection: snarl://localhost:5820/
Logged in as: admin
Database: testStore
Uploaded model: Trevithick_Full.ttl
Uploaded ontologies: default ontologies
Added scenarios: >scenario-1< >scenario-2< >scenario-3< >scenario-4< >scenario-5< >scenario-6< >scenario-7< >scenario-8< >scenario-9< >scenario-10<

Successully created scenarios

STEP 5 - Generate & validate scenarios

Scenarios generated:

scenario-1
scenario-2
scenario-3
scenario-5
scenario-8
scenario-9
scenario-10

Scenarios selected for analysis

STEP 6 - Select scenarios for analysis

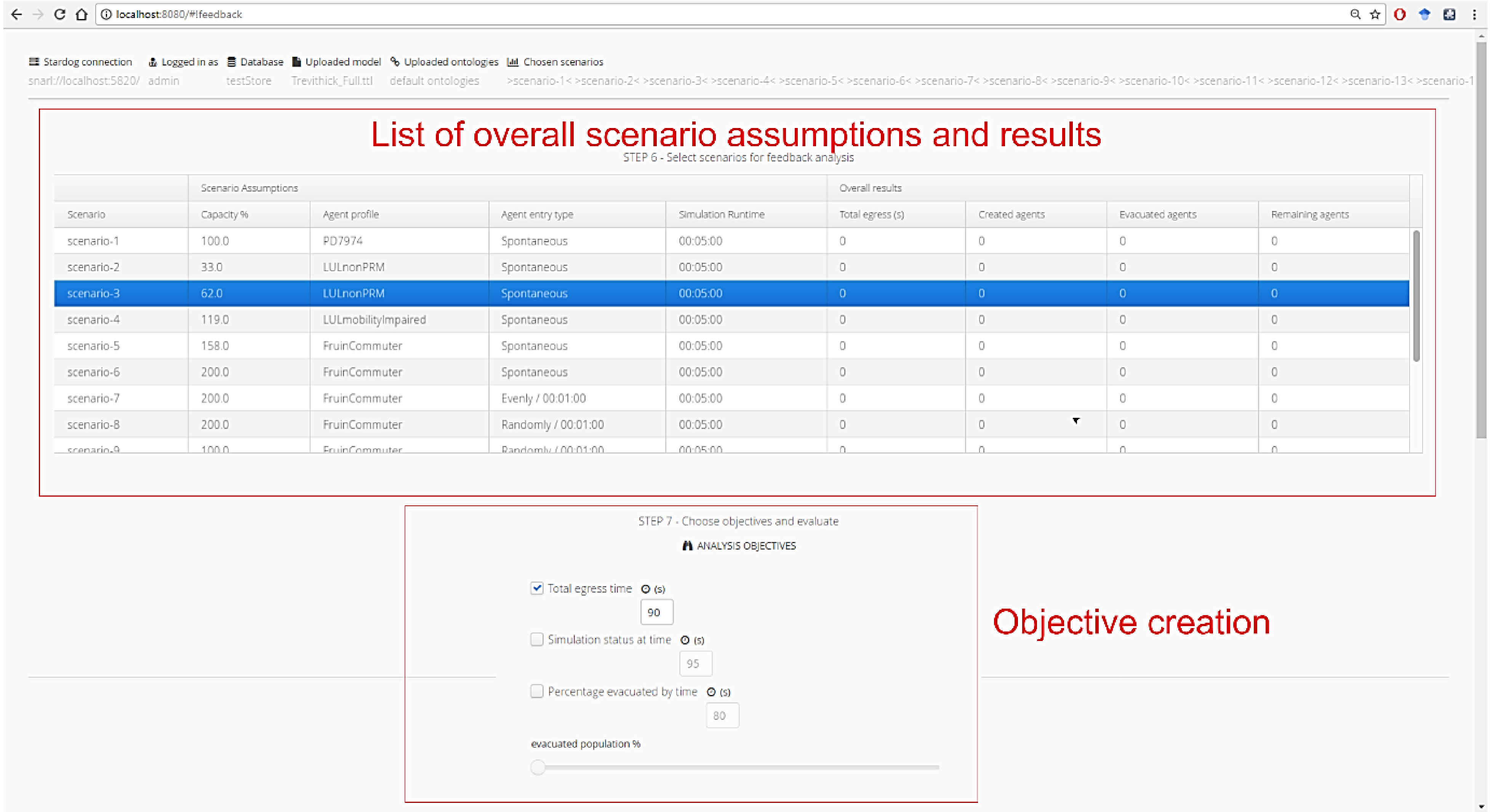
Scenarios under analysis:

scenario-6
scenario-4
scenario-7

Users can choose to download each scenario file for validation of a correct model creation

scenario-9.mm

Figure D-6. ONTOCS scenario generation, validation and selection for analysis page



Stardog connection	Logged in as	Database	Uploaded model	Uploaded ontologies	Chosen scenarios
snarl://localhost:5820/	admin	testStore	Trevithick_Full.ttl	default ontologies	>scenario-1< >scenario-2< >scenario-3< >scenario-4< >scenario-5< >scenario-6< >scenario-7< >scenario-8< >scenario-9< >scenario-10< >scenario-11< >scenario-12< >scenario-13< >scenario-1

List of overall scenario assumptions and results

STEP 6 - Select scenarios for feedback analysis

	Scenario Assumptions				Overall results			
Scenario	Capacity %	Agent profile	Agent entry type	Simulation Runtime	Total egress (s)	Created agents	Evacuated agents	Remaining agents
scenario-1	100.0	PD7974	Spontaneous	00:05:00	0	0	0	0
scenario-2	33.0	LULnonPRM	Spontaneous	00:05:00	0	0	0	0
scenario-3	62.0	LULnonPRM	Spontaneous	00:05:00	0	0	0	0
scenario-4	119.0	LULmobilityImpaired	Spontaneous	00:05:00	0	0	0	0
scenario-5	158.0	FruinCommuter	Spontaneous	00:05:00	0	0	0	0
scenario-6	200.0	FruinCommuter	Spontaneous	00:05:00	0	0	0	0
scenario-7	200.0	FruinCommuter	Evenly / 00:01:00	00:05:00	0	0	0	0
scenario-8	200.0	FruinCommuter	Randomly / 00:01:00	00:05:00	0	0	0	0
scenario-9	100.0	FruinCommuter	Randomly / 00:01:00	00:05:00	0	0	0	0

STEP 7 - Choose objectives and evaluate

ANALYSIS OBJECTIVES

☑ Total egress time (s)

90

□ Simulation status at time (s)

95

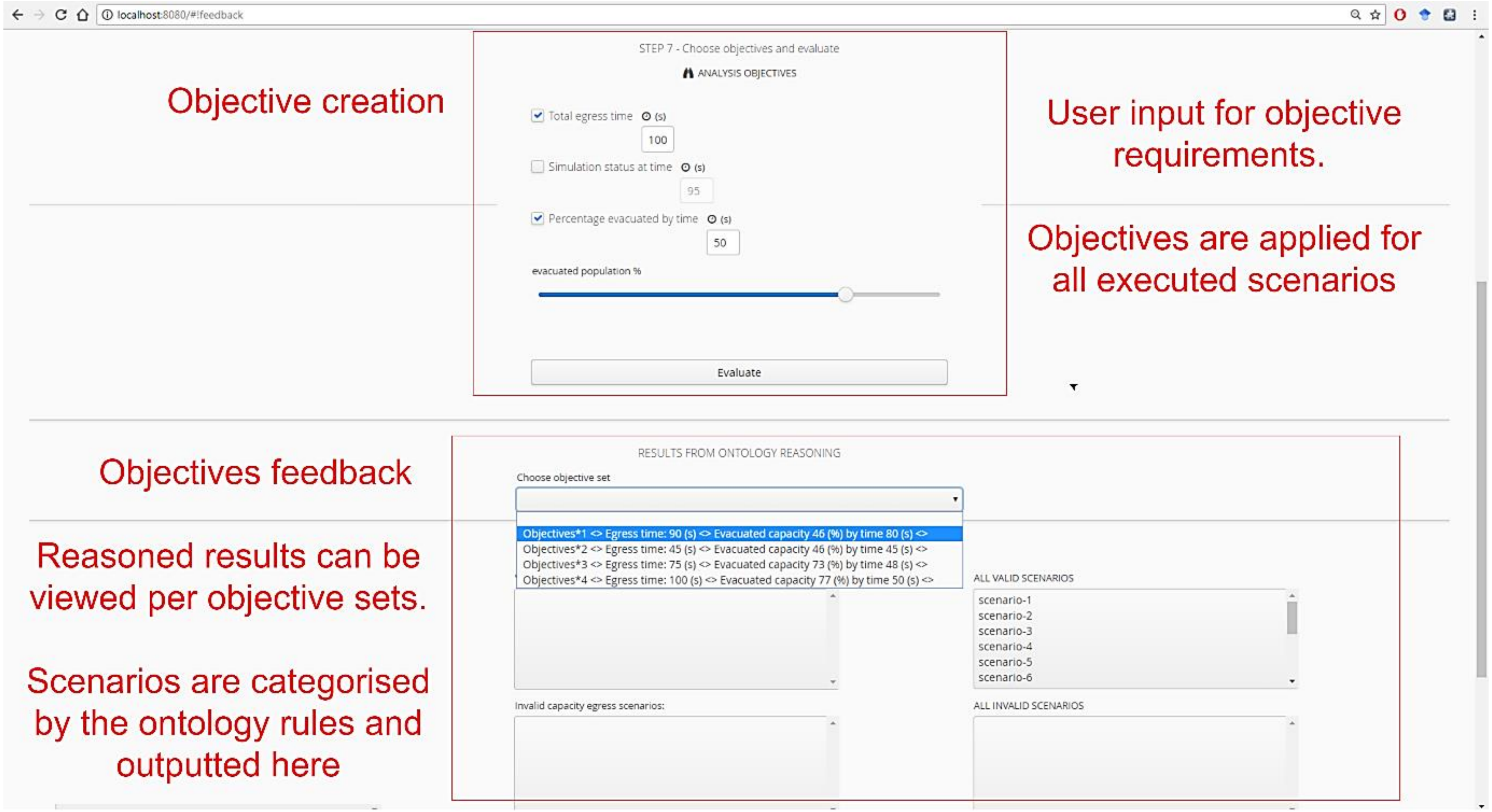
□ Percentage evacuated by time (s)

80

evacuated population %

Objective creation

Figure D-7. ONTOCS scenario level results and feedback page



localhost:8080/#!feedback

Objective creation

STEP 7 - Choose objectives and evaluate

ANALYSIS OBJECTIVES

☑ Total egress time (s)

100

□ Simulation status at time (s)

95

☑ Percentage evacuated by time (s)

50

evacuated population %

User input for objective requirements.

Objectives are applied for all executed scenarios

Objectives feedback

RESULTS FROM ONTOLOGY REASONING

Choose objective set

Objectives*1 <> Egress time: 90 (s) <> Evacuated capacity 46 (%) by time 80 (s) <>
Objectives*2 <> Egress time: 45 (s) <> Evacuated capacity 46 (%) by time 45 (s) <>
Objectives*3 <> Egress time: 75 (s) <> Evacuated capacity 73 (%) by time 48 (s) <>
Objectives*4 <> Egress time: 100 (s) <> Evacuated capacity 77 (%) by time 50 (s) <>

Reasoned results can be viewed per objective sets.

Scenarios are categorised by the ontology rules and outputted here

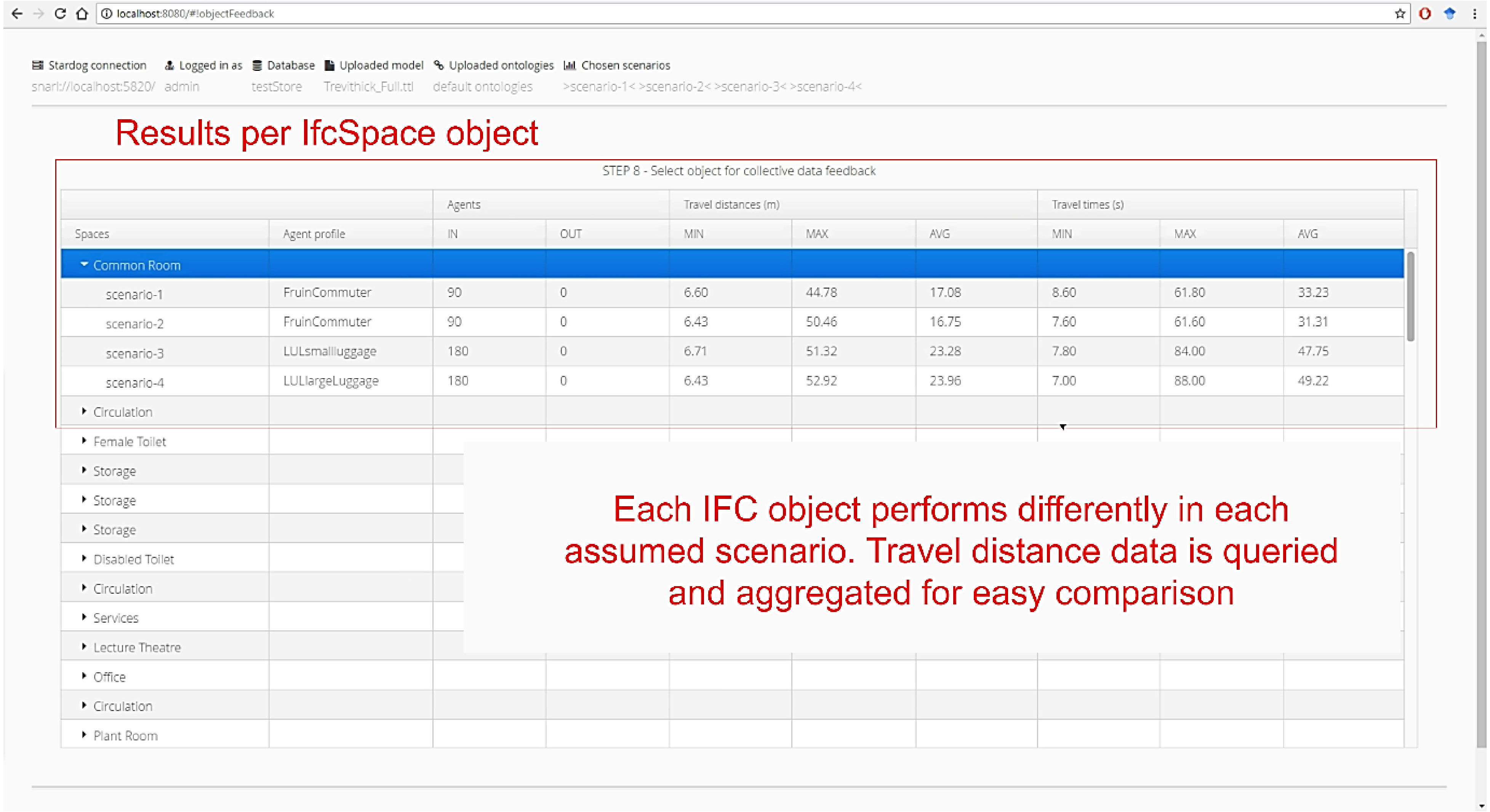
ALL VALID SCENARIOS

scenario-1
scenario-2
scenario-3
scenario-4
scenario-5
scenario-6

Invalid capacity egress scenarios:

ALL INVALID SCENARIOS

Figure D-8. ONTOCS objects creation and reasoning results viewing page



localhost:8080/#!objectFeedback

Stardog connection: snarl://localhost:5820/ | Logged in as: admin | Database: testStore | Uploaded model: Trevithick_Full.ttl | Uploaded ontologies: default ontologies | Chosen scenarios: >scenario-1< >scenario-2< >scenario-3< >scenario-4<

Results per IfcSpace object

STEP 8 - Select object for collective data feedback

		Agents		Travel distances (m)			Travel times (s)		
Spaces	Agent profile	IN	OUT	MIN	MAX	AVG	MIN	MAX	AVG
Common Room									
scenario-1	FruinCommuter	90	0	6.60	44.78	17.08	8.60	61.80	33.23
scenario-2	FruinCommuter	90	0	6.43	50.46	16.75	7.60	61.60	31.31
scenario-3	LULsmallluggage	180	0	6.71	51.32	23.28	7.80	84.00	47.75
scenario-4	LULlargeLuggage	180	0	6.43	52.92	23.96	7.00	88.00	49.22
Circulation									
Female Toilet									
Storage									
Storage									
Storage									
Disabled Toilet									
Circulation									
Services									
Lecture Theatre									
Office									
Circulation									
Plant Room									

Each IFC object performs differently in each assumed scenario. Travel distance data is queried and aggregated for easy comparison

Figure D-9. ONTOCS object level results page

Appendix E – Case study data

This section presents a summary of the data gathered about the real-life building which was used for the case study and additional results from testing.

Each numbered room from the Table E1 is shown on a detailed floor plan from Figure E-1. This data was inputted into the Revit model during the modelling stage and is also present in the IFC and IfcOwl models throughout the testing.

Additional results gathered from case study use cases are provided in Tables E2 and E3 below. Their data was plotted in charts in Chapter 7 for visualisation. Table E3 shows average times of query taken from 5 measurements across each objective case. The full data was not made available in print due to its size.

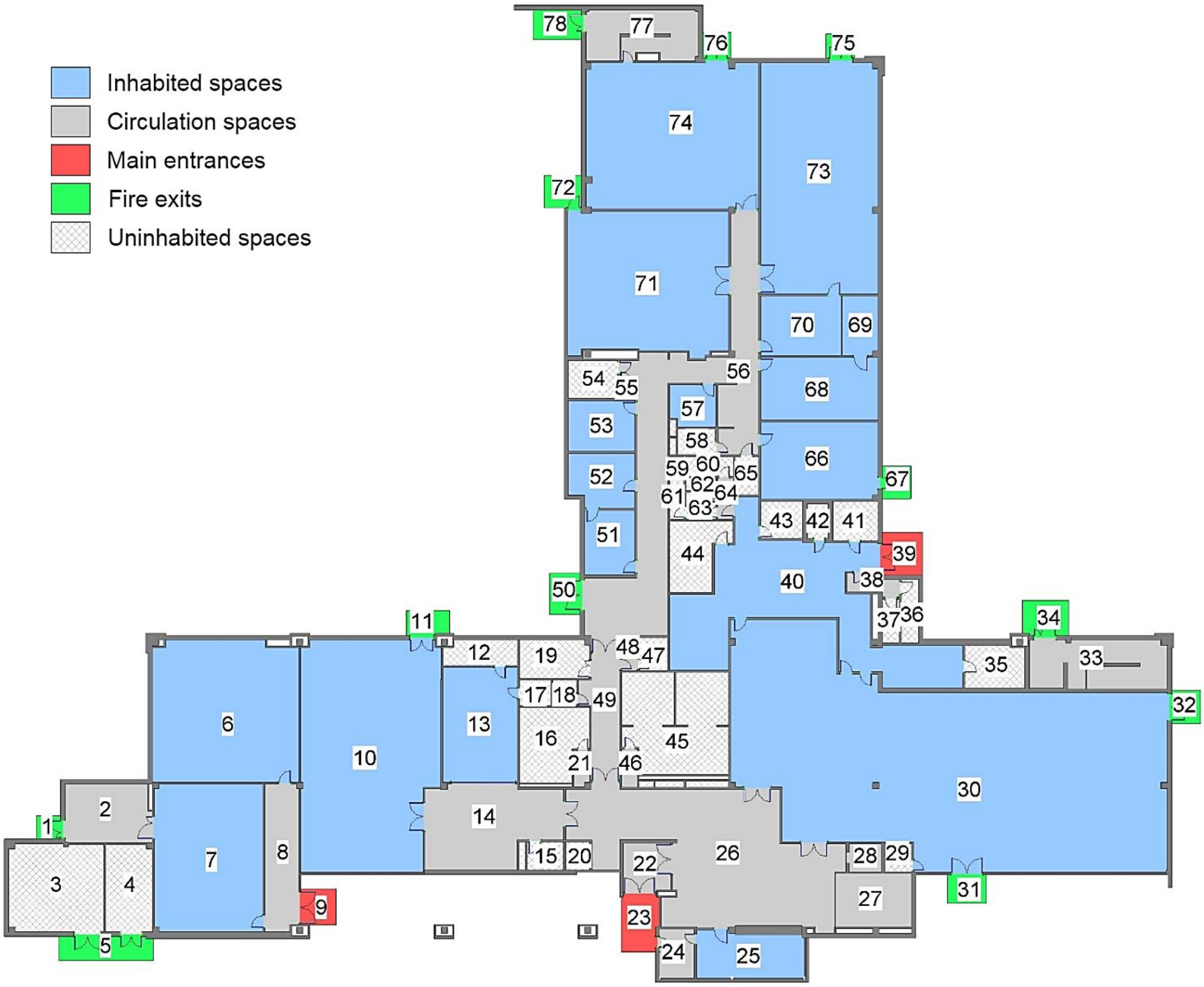


Figure E-1. Case study building layout with numbered spaces

Table E-1. Building spaces data exported from Revit

No	Name	Uniclass description	Uniclass code	Area	Space Occupancy	Comments
1	Exit	Fire refuge spaces	SL_20_90_30	2.18 m²		
2	Staircase	Stairways	SL_90_10_87	19.34 m²		
3	Plant Room	Plant rooms	SL_90_90_64	32.60 m²		
4	Storage	Storage rooms	SL_90_50_84	15.92 m²		
5	Exit	Fire refuge spaces	SL_20_90_30	8.83 m²		
6	Lecture Theatre	Lecture theatres	SL_25_10_47	80.25 m²	50	
7	Office	Offices	SL_20_15_59	62.52 m²	11	
8	Circulation	Hallways	SL_90_10_36	19.51 m²		
9	Entrance	Fire refuge spaces	SL_20_90_30	4.93 m²		Fire Exit
10	Common Room	Common rooms	SL_25_10_15	120.26 m²	90	
11	Exit	Fire refuge spaces	SL_20_90_30	4.15 m²		
12	Storage	Storage rooms	SL_90_50_84	7.29 m²		
13	Shop	Food and drink outlets	SL_20_50_32	33.58 m²	5	
14	Circulation	Hallways	SL_90_10_36	39.82 m²		
15	Services	Wall services voids	SL_90_90_96	4.02 m²		
16	Female Toilet	Toilets	SL_35_80_89	18.45 m²		
17	Storage	Storage rooms	SL_90_50_84	3.20 m²		
18	Disabled Toilet	Toilets	SL_35_80_89	2.40 m²		
19	Storage	Storage rooms	SL_90_50_84	10.65 m²		
20	Services	Wall services voids	SL_90_90_96	2.66 m²		
21	Circulation	Hallways	SL_90_10_36	2.20 m²		
22	Circulation	Hallways	SL_90_10_36	9.15 m²		
23	Entrance	Fire refuge spaces	SL_20_90_30	8.07 m²		is fire refuge
24	Circulation	Hallways	SL_90_10_36	6.75 m²		
25	Security	Security offices	SL_20_85_80	18.84 m²	3	
26	Circulation	Hallways	SL_90_10_36	103.12 m²		
27	Staircase	Stairways	SL_90_10_87	16.04 m²		
28	Lift	Lift shafts	SL_90_60_50	3.67 m²		

No	Name	Uniclass description	Uniclass code	Area	Space Occupancy	Comments
29	Services	Wall services voids	SL_90_90_96	2.99 m²		
30	Dining Area	Enclosed dining areas	SL_40_20_27	305.93 m²	130	
31	Exit	Fire refuge spaces	SL_20_90_30	4.35 m²		
32	Exit	Fire refuge spaces	SL_20_90_30	3.78 m²		
33	Staircase	Stairways	SL_90_10_87	25.06 m²		
34	Exit	Fire refuge spaces	SL_20_90_30	6.22 m²		
35	Storage	Storage rooms	SL_90_50_84	9.96 m²		
36	Storage	Storage rooms	SL_90_50_84	4.29 m²		
37	Storage	Storage rooms	SL_90_50_84	2.94 m²		
38	Circulation	Hallways	SL_90_10_36	3.76 m²		
39	Entrance	Fire refuge spaces	SL_20_90_30	6.75 m²		Can act as a fire exit
40	Kitchen	Cooking spaces	SL_35_60_16	86.19 m²	5	
41	Storage	Storage rooms	SL_90_50_84	6.93 m²		
42	Storage	Storage rooms	SL_90_50_84	2.82 m²		
43	Storage	Storage rooms	SL_90_50_84	6.02 m²		
44	Storage	Storage rooms	SL_90_50_84	13.65 m²		
45	Male Toilet	Toilets	SL_35_80_89	40.43 m²		
46	Circulation	Hallways	SL_90_10_36	2.41 m²		
47	Storage	Storage rooms	SL_90_50_84	3.76 m²		
48	Circulation	Hallways	SL_90_10_36	2.10 m²		
49	Circulation	Hallways	SL_90_10_36	15.60 m²		
50	Exit	Fire refuge spaces	SL_20_90_30	4.94 m²		
51	Meeting Room	Meeting rooms	SL_20_15_50	12.66 m²	9	
52	Office	Offices	SL_20_15_59	13.34 m²	5	
53	Office	Offices	SL_20_15_59	13.24 m²	5	
54	Changing Room	Changing rooms	SL_90_20_13	7.33 m²		
55	Storage	Storage rooms	SL_90_50_84	0.79 m²		
56	Circulation	Hallways	SL_90_10_36	85.70 m²		
57	Office	Offices	SL_20_15_59	7.30 m²	2	
58	Disabled Toilet	Toilets	SL_35_80_89	3.78 m²		
59	Female Toilet	Toilets	SL_35_80_89	1.58 m²		
60	Disabled Toilet	Toilets	SL_35_80_89	3.65 m²		

No	Name	Uniclass description	Uniclass code	Area	Space Occupancy	Comments
61	Male Toilet	Toilets	SL_35_80_89	1.78 m²		
62	Shower Room	Showers	SL_35_80_80	1.97 m²		
63	Male Toilet	Toilets	SL_35_80_89	2.64 m²		
64	Circulation	Hallways	SL_90_10_36	2.41 m²		
65	Storage	Storage rooms	SL_90_50_84	3.55 m²		
66	Office	Offices	SL_20_15_59	35.18 m²	7	
67	Exit	Fire refuge spaces	SL_20_90_30	3.69 m²		
68	Office	Offices	SL_20_15_59	28.82 m²	5	
69	Office	Offices	SL_20_15_59	8.22 m²		
70	Control Room	Experiment control rooms	SL_25_30_28	18.82 m²	5	
71	Laboratory	Engineering laboratories	SL_25_30_27	88.26 m²	5	
72	Exit	Fire refuge spaces	SL_20_90_30	4.68 m²		
73	Laboratory	Engineering laboratories	SL_25_30_27	105.58 m²	30	
74	Laboratory	Engineering laboratories	SL_25_30_27	98.55 m²	6	
75	Exit	Fire refuge spaces	SL_20_90_30	2.71 m²		
76	Exit	Fire refuge spaces	SL_20_90_30	2.82 m²		
77	Staircase	Stairways	SL_90_10_87	20.89 m²		
78	Exit	Fire refuge spaces	SL_20_90_30	5.59 m²		
79	Services	Wall services voids	SL_90_90_96	0.39 m²		
80	Services	Wall services voids	SL_90_90_96	0.75 m²		
81	Services	Wall services voids	SL_90_90_96	1.12 m²		
82	Services	Wall services voids	SL_90_90_96	0.75 m²		
83	Services	Wall services voids	SL_90_90_96	0.40 m²		
84	Services	Wall services voids	SL_90_90_96	0.37 m²		

Table E-2. STAGE I query times for the run of 36 scenarios in parallel

Query code	Role	Reasoning	Time measurements (milliseconds)										AVG
			1	2	3	4	5	6	7	8	9	10	
Q-IFC-2	geometry	no	733	264	284	177	354	182	145	234	237	201	**281**
Q-IFC-3	geometry		22	11	10	17	33	29	12	21	24	15	**19**
Q-IFC-4	geometry		651	652	736	693	683	677	631	572	632	564	**649**
Q-IFC-5	geometry		1680	1444	1466	1502	1480	1500	1567	1437	1482	1447	**1501**
Q-IFC-6	geometry		1479	1449	1456	1566	1448	1470	1401	1391	1389	1395	**1444**
Q-IFC-7	geometry		2007	1838	1868	2026	1999	2086	1866	1853	1887	1888	**1932**
Q-IFC-8	geometry		1173	1142	1136	1157	1146	1179	1134	1145	1138	1141	**1149**
Q-IFC-9	geometry		1154	1136	1131	1187	1139	1172	1124	1129	1126	1136	**1143**
Q-IFC-10	geometry		2264	2098	1885	2100	2181	2467	2166	2105	2251	2373	**2189**
Q-IFC-11	geometry		1764	1666	1686	1839	2122	2015	1965	1822	1802	1892	**1857**
Q-IFC-14	geometry		1029	1033	1030	1031	1040	1033	1039	1039	1036	1032	**1034**
Q-IFC-15	geometry		15753	12664	12784	16487	15979	18236	14322	18516	22781	25366	**17289**
Q-IFC-17	geometry		1058	1043	1055	1067	1054	1045	1052	1043	1051	1057	**1053**
Q-IFC-18	geometry		1040	1032	1036	1061	1045	1045	1051	1056	1038	1045	**1045**
Q-IFC-19	geometry		1236	822	826	1126	968	850	852	1003	819	911	**941**
Q-IFC-1	context	yes	1235	933	795	767	851	753	741	636	673	669	**805**
Q-IFC-20	context	no	28	19	15	25	43	23	25	28	29	40	**28**
Q-IFC-21	context		220	201	198	223	332	250	194	241	192	249	**230**
Q-RES-1	context		450	316	251	254	222	191	318	423	235	382	**304**
Q-RES-2	context		7729	7323	7390	6982	7299	7385	8521	7241	6927	7062	**7386**
Q-RES-3	context	yes	165678	148797	151504	146634	142949	143510	140474	139552	140850	142746	**146269**
Q-RES-4	context		1493	806	872	638	494	509	492	404	384	648	**674**
Q-RES-5	context		43	66	52	35	32	23	56	21	16	109	**45**

Table E-3. STAGE II query times for increasing numbers of scenarios (part1/3)

Query	Objectives applied	Scenarios											
		1	2	3	4	5	6	7	8	9	10	11	12
		Average time (milliseconds)											
Q-FBA-1	single (a)	731	1408	1211	1206	1077	1135	931	970	1016	1084	977	997
	multiple (a & b)	1055	1413	1742	1712	1619	1540	2030	2211	2187	2191	2112	2196
Q-FBA-2	single (b)	571	705	623	645	669	708	717	703	824	762	717	675
	multiple (a & b)	598	832	993	893	954	850	833	827	904	842	857	871
Q-FBA-3	single (a)	350	984	715	694	680	742	579	605	653	608	554	697
	single (b)	173	242	229	217	230	220	206	212	227	218	228	241
	multiple (a & b)	804	1495	1804	1657	1595	1636	1900	1957	2141	2100	1981	2049
Q-FBA-4	single (a)	6	6	10	10	10	8	10	11	9	9	10	9
	single (b)	7	5	9	8	9	7	9	8	8	7	7	7
	multiple (a & b)	33376	33317	33312	33321	33371	33309	33398	33341	33393	33290	33381	33274
Q-FBA-5	single (a)	62	106	100	103	92	112	92	105	87	108	100	112
	multiple (a & b)	68	128	172	154	140	156	144	143	141	132	136	134
Q-FBA-6	single (b)	216	255	241	255	259	276	246	227	237	218	237	221
	multiple (a & b)	217	463	530	450	381	451	419	394	399	386	377	401
Q-FBA-7	single (a)	60	94	97	90	97	94	83	88	82	90	86	93
	single (b)	201	231	219	210	223	230	210	207	228	221	225	235
	multiple (a & b)	277	589	663	593	501	582	530	516	551	530	551	533

Table E-4. STAGE II query times for increasing numbers of scenarios (part2/3)

Query	Objectives applied	Scenarios											
		13	14	15	16	17	18	19	20	21	22	23	24
		Average time (milliseconds)											
Q-FBA-1	single (a)	1017	1021	1331	1382	1302	1360	1618	1365	1292	1376	1415	1459
	multiple (a & b)	2182	598	3655	3766	3926	3905	3753	3710	3709	3979	3924	3866
Q-FBA-2	single (b)	797	1035	740	762	723	849	759	819	742	772	921	768
	multiple (a & b)	858	750	1125	1165	1096	1077	1192	1063	1187	1098	1106	1123
Q-FBA-3	single (a)	628	635	941	1016	904	921	1061	962	928	962	968	968
	single (b)	240	473	299	275	287	309	299	291	306	317	289	279
	multiple (a & b)	2082	597	3713	3947	3981	4108	3986	3923	3939	4216	4063	4056
Q-FBA-4	single (a)	8	10	9	10	12	9	8	14	9	9	9	8
	single (b)	7	7	7	7	7	7	12	7	6	7	11	7
	multiple (a & b)	33354	24808	33244	33290	33277	33306	33304	33297	33243	33283	33244	33323
Q-FBA-5	single (a)	96	104	110	130	124	123	138	111	127	118	108	114
	multiple (a & b)	144	65	174	198	177	185	178	166	165	189	180	168
Q-FBA-6	single (b)	229	509	293	292	316	331	298	283	308	303	301	285
	multiple (a & b)	395	329	617	639	625	654	626	653	622	680	637	641
Q-FBA-7	single (a)	100	102	122	120	153	110	124	103	125	104	116	102
	single (b)	229	548	290	291	282	310	300	288	288	313	286	273
	multiple (a & b)	521	421	784	807	804	853	792	777	797	808	786	804

Table E-5. STAGE II query times for increasing numbers of scenarios (part3/3)

Query	Objectives applied	Scenarios											
		25	26	27	28	29	30	31	32	33	34	35	36
		Average time (milliseconds)											
Q-FBA-1	single (a)	1438	1446	1308	1247	1213	1248	1288	1347	1204	1205	1270	1297
	multiple (a & b)	3896	3896	3925	3686	3692	3656	3817	3856	3660	3672	3808	3514
Q-FBA-2	single (b)	719	741	775	646	675	672	662	661	657	651	650	778
	multiple (a & b)	1116	1170	1037	961	994	981	1002	1018	974	1003	1012	969
Q-FBA-3	single (a)	999	962	950	816	919	933	904	928	863	905	876	951
	single (b)	290	283	346	273	300	287	295	296	285	286	276	325
	multiple (a & b)	4192	4148	4179	3891	3896	3852	4013	4076	3871	3833	3945	3641
Q-FBA-4	single (a)	10	8	10	11	12	11	8	11	8	10	9	9
	single (b)	6	8	9	8	7	8	6	8	7	6	6	8
	multiple (a & b)	33283	33299	33348	33451	33329	33394	33369	33360	33667	33403	33398	33344
Q-FBA-5	single (a)	122	113	102	117	142	106	133	119	122	117	111	123
	multiple (a & b)	182	199	185	182	174	179	172	173	174	174	176	176
Q-FBA-6	single (b)	299	289	321	285	292	286	310	305	295	289	293	340
	multiple (a & b)	651	641	638	569	611	618	627	639	624	606	631	593
Q-FBA-7	single (a)	115	110	107	108	109	115	110	106	120	261	103	111
	single (b)	284	316	323	277	302	300	279	290	285	273	275	325
	multiple (a & b)	826	789	821	751	759	818	814	791	781	747	795	749