Developing the Crowd Simulation Scenario (CSS) ontology supporting building evacuation design

Calin Boje

Luxembourg Institute of Science and Technology,
Luxembourg, L-4362 Esch/Alzette,
calin.boje@list.lu

Abstract. The recurring interoperability problem within the construction industry has led to the exploration of linked data technologies. Evacuation design for buildings is a very complex domain which is evaluated using strict regulations and several models. Crowd Simulation Tools (CSTs) are increasingly used to assess building performance, but they require many iterations subjected to manual user input and assumptions from several sources. This article introduces the newly created Crowd Simulation Scenario (CSS) ontology, which is meant to represent the domain ontologically, but to also be used in practice for automation and feedback. The nature of simulation tools, the complex human behaviour which they describe, and the connection to the BIM were considered when developing the ontology.

Keywords: Evacuation, Crowd Simulation, Design, Ontology, IfcOwl

1 Introduction

Fire safety design is a complex multi-disciplinary process spanning across different knowledge fields, from structural fire resistance to human psychology. Fire design employs many regulations, which were improved over the years to enforce certain standards 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. 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.

The scope of this research is focused on assessing building performance with regards to evacuation of building occupants. In practice this is evaluated using Crowd Simulation Tools (CSTs), which are able to simulate in detail how people behave during an

evacuation event. This in turn allows designers to assess the performance of the building in multiple scenarios [1]. The entire process relies on expert designers using CSTs to create, run and analyse scenarios using several iterations, which is time consuming [2]. Additionally, each building layout is different and so is each scenario in terms of context [3]. 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 timely and holistic manner.

This research continues on the methodology and prototype system introduced in [4], which is able to create simulation scenarios according to design practice on the fly, and provide feedback on simulation results with minimum user input. This is achieved using ontology support, which aims to represent the CS knowledge domain due to several main benefits:

- Increased interoperability – a myriad of CSMs and CSTs are present in academia and industry, with no real consensus on data model schema or scope, thus limiting the application of CS on building design;
- Linking heterogeneous data – the creation of CS scenarios is subject to various data inputs (building, population, events), which often come from different sources. This allows the automation of scenario creation;
- Reasoning support – checking scenario data input and output is often a challenge for designers. Reasoning rules allow a means of validation and finding new knowledge about the building design.

This article in particular focuses on presenting the Crowd Simulation Scenario (CSS) ontology. The aims of the CSS ontology are twofold:

1) provide an ontological representation of the domain, regardless of the CSTs used in practice;
2) Facilitate practical implementation using software systems.

In terms of structure, this article outlines the methodology employed in the development of the CSS ontology in section 2. Following this, section 3 presents existing literature, industry tools and related ontologies. Section 4 outlines the main classes of the CSS ontology, with a mapping to IfcOwl provided in section 5. To demonstrate its use, section 6 provides several examples on querying the developed ontology. Finally, a summary and future work is provided in section 7.

2 Ontology modelling methodology

The development of the CSS ontology followed several steps (**Fig.1**):

1) A review of literature and design guidance on fire safety and evacuation using crowd simulation models;
2) A survey of current state of the art CSTs, specifically targeting their features and types of concepts they use to represent the model;
3) Development of main classes and properties with the aid of competency questions;

4) Testing the CSS in line with several other ontologies (such as IfcOwl) using a prototype system;
5) Expert consultation on the validity and completeness during open discussion sessions.

An iterative approach was adopted to the ontology design process, with the final steps (3, 4 and 5) having had significant influence one the structure and scope. The detailed results from steps 4) and 5) are out of scope for this article, but these are inherently present as the last version of the CSS ontology is shown. During the testing, other developed ontology models were connected to the CSS ontology. One of these represented several performance indicators which rely on CSS for retrieving simulation results data for feedback, while another graph represented the occupancy factors from UK fire safety guidance, to help automate scenario construction space by space. Most importantly however, is the IfcOwl ontology, which had the purpose to represent the digital building model. Parts of the CSS were therefore mapped to the IfcOwl during testing. The final linking is also presented in section 5 of this article.

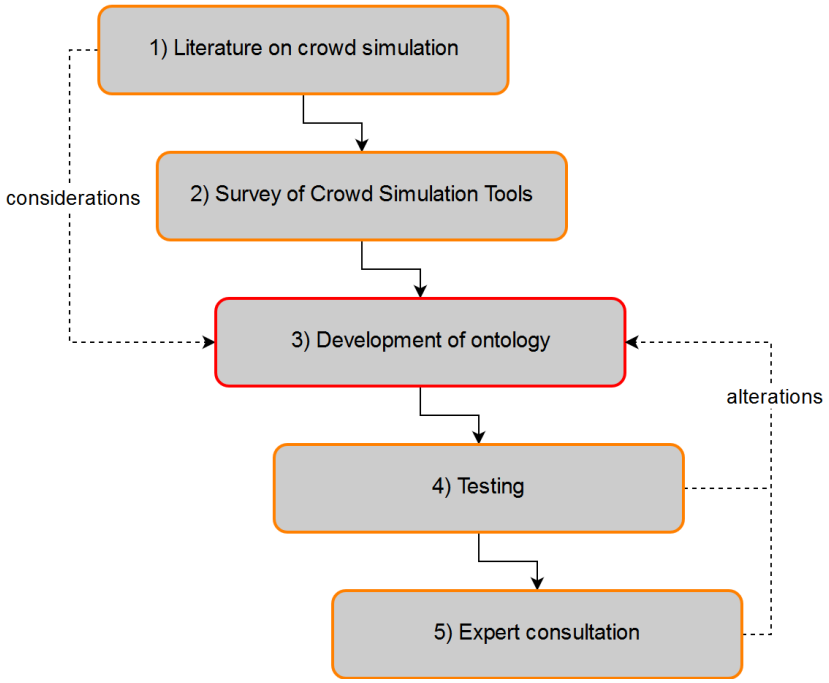

Fig 1. CSS ontology development method

3 Identification of crowd simulation concepts

3.1 Evacuation simulation models

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 [5] [2]. 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.” [6]

The entire process is heavily influenced by user input and follows three well-defined steps [7]:

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.

However, not all CSTs consider all types of scenarios. Some model only pedestrian movement, whilst others consider also the propagation of fire and smoke. Very often, the fire and crowd are simulated separately and later overlapped. Fire design assumes that people can evacuate the building safely, un-impeded by fire and smoke, as is done traditionally during live drills [8]. As such, this research does not take into account the fire element and instead focuses in more detail on the interaction between the agents (representing the people) and the digital model (representing the building).

3.2 Survey of Crowd simulation tools

A number of CSTs are available in industry and research, with various features that they provide to users and various concepts which make out the model. Several CSTs which are widely used in industry were investigated through testing and surveying of their documentation. This was used to establish a baseline of common functionalities

and concepts used in the field of CS which would serve as a basis for construction a common ontology. 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, and have been used on real-life projects on many occasions. Experts consider the CST validation process ongoing across the software tool's lifecycle.

The various concepts were categorized in an initial taxonomy of "things", which was later used to develop the CSS ontology. Several main categories stand out in **Fig. 2**:

- Geometry – concepts which represent the environment using geometry;
- Agent – concepts about people, with their behaviours and characteristics;
- Event – concepts about what and when things happen during a simulation;
- Analysis – concepts that report simulation results using various indicators.

The categories for Visualisation, Interface and Mathematical appear to vary greatly from one tool to another and are dependent on software design. As such, these were omitted for the construction of the CSS ontology.

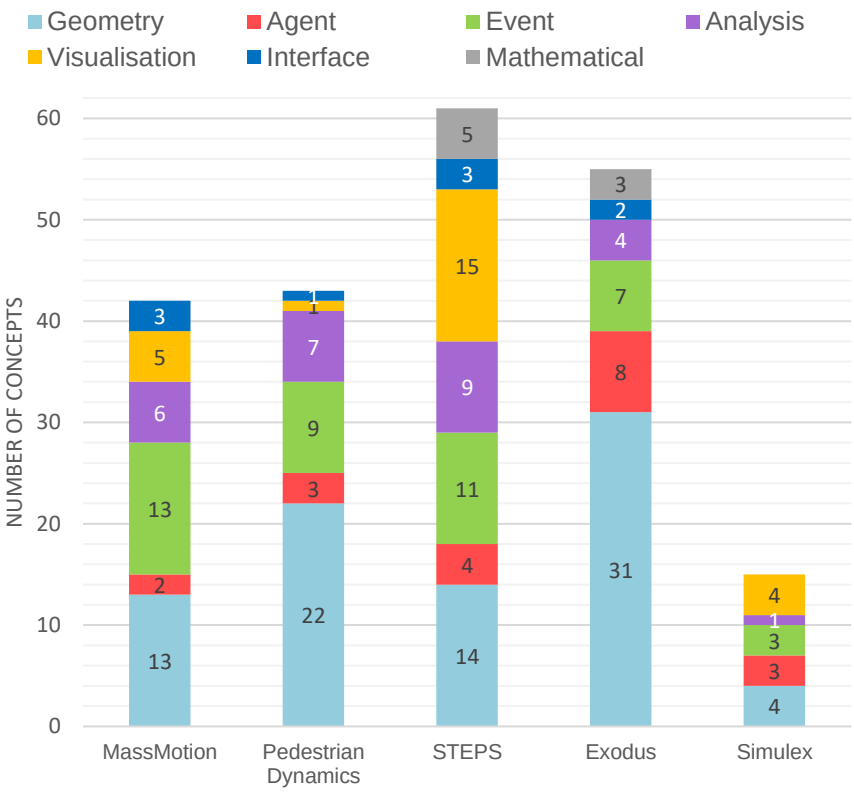

Fig. 2. Concept types identified in several CSTs

3.3 Related ontologies

[9] 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. [10] 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.

[11] 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. [12] proposes an ontological representation of the building plans, according to different functionalities so that evacuation events can be represented more comprehensively. [13] uses ontologies and ambient intelligence to gather knowledge about how evacuations progress in a building. [14] present a framework using ontology support for disaster response, with a wider scope, not focused on the details of building evacuation.

[15] represents smoke propagation using an OWL model with reasoning support, with the purpose of aiding rescuers identify a smoke-free route within a building. However, its focus is not on crowd simulation, nor does it represent the building inhabitants as a CSM would.

While some of the related works above attempt to represent human behaviour in buildings or fire safety events to various extents, none of the investigated ontologies above have a scope on crowd simulation evacuation, nor do they consider interoperability with BIM or other CSTs used in practice.

The CSS ontology on the other hand, was envisaged to represent a BIM-based design domain, and allow to interface via a professional CST in practice, offering the entire process increased interoperability and ontology support.

4 CSS main concepts

Following these aims, several competency questions were outlined, which were used to construct the initial ontology, whilst also considering the common terminology previously investigated around existing commercial tools and models from literature. The primary classes of the CSS ontology are shown in **Fig. 3**.
The core competency question relevant to the CSS ontology is the following:

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

Model objects make up the previously mentioned categories which are represented virtually (and programmatically) within a simulation model, which will find equivalents in most existing CSTs.
Assumptions conceptualise design choices via the main *ScenarioAssumption* class, which are part of the required input, thus a subclass of *UserInput*;
Results conceptualise results about each simulation run, via the *SimulationResult* class. This is required for the performance analysis of the scenario in practice.

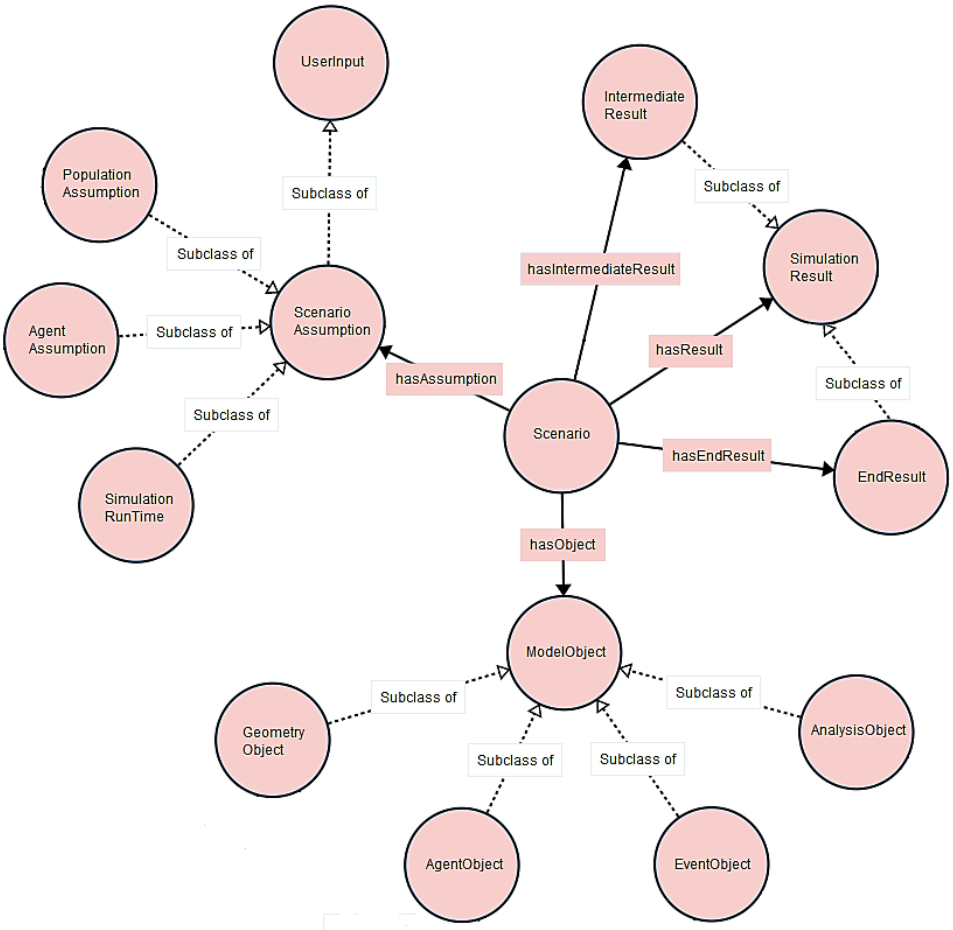

Fig. 3. Main classes of the CSS ontology

4.1 Model objects

The *ModelObject* class specifically includes concepts which are present within the model simulated. In addition to the major categories mentioned above, the *GeometryObject* class includes several important sub-classes which describe the essential building environment:

- *Barrier* – the objects whose geometry impedes agent movement; agents will by default avoid barriers in their path.
- *Space* - the most characteristic object type for all CSTs is the one defining the walkable surfaces, which allows agents to effectively exist and act within the model. They are represented virtually within a model as surfaces without a 3D component. This 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 can easily identify regions within a model. The functionality of a space was required in order to refer to spaces in other specific circumstances.

- *Link* – the connection between Spaces within the model, vertically and horizontally;
- *Portal* - the place where populations of agents enter or leave a model; this can coincide with *Link* and *Space* for some *CSTs*, but is also a more precise object, which can reside within a *Link* or *Space* object.

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

For example, an *InhabitedSpace* refers to a space within the virtual model which has agents assigned to it at the start of a simulation; this is also considered inhabited in reality (e.g. an office is inhabited by X people). A *RefugeSpace* acts 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, and correctly assign which spaces are inhabited, and which are egress destinations.

4.2 Scenario assumptions

These refer to concepts which are supposed to keep track of the assumed scenario context and are usually in relationship with the *EventObject* and *AgentObject* classes and their subclasses. The *ScenarioAssumption* subclasses therefore answer to questions such as:

- *'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 influence the behaviour of agents and therefore the performance of the design. Within CSTs these are usually user input assumptions. Each CST has several pre-set values for these inputs, such as different types of agent profiles. Knowing the differences in assumptions between several parallel scenarios is vital in identifying design problems.

'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 that assumed building occupancy factors changing with region. However, there are several viable resources where population data can be retrieved, such as the BIM, occupancy data tables or design guides – depending on the building lifecycle. The most reliable source of information is preferred. For example, in a building design stage, it would be preferred to know the intended occupancy of each space. If not, design guidance can provide initial estimates based on space area. For existing buildings, real-time occupancy would be the most representative of the truth.

4.3 Simulation results

Simulation outputs are usually presented to users via several analysis features such as tables or overlay maps. These have already been defined as *AnalysisObject* concepts (subclasses of *ModelObject*) in their own right. However, the data which these objects use are usually stored separately, often recorded in memory or databases and later retrieved on user demand.

The *SimulationResult* class therefore accounts for this and conceptualises the different types of results, based on how they are stored. Thus, the class *EndResult* encompasses definitive outputs which are retrieved at the end of the simulation through its general reporting stage, where a summary of simulation results is presented. 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*, however, is meant to store data dynamically, according to user objectives, and to provide data at specific calculated time steps during a simulation. This is a special requirement for crowd simulation data as events and agent movement relate to *SimulationTime*. Additionally, the performance of the design is monitored over time, thus being important for the analysis stage.

4.4 Agent relationships

Within a simulation, the most dynamic objects are those describing the building inhabitants. Therefore, the *Agent* class is one of the most complex concepts, due to its role to represent a person with not just its physical properties, but also its behaviour.

Each *Agent* created within a simulation tries to represent a distinct individual, mimicking reality with higher fidelity. However, in practice this is usually done by grouping different agents into several well-defined typologies, with common traits, and very often with common events and paths to follow. Each agent is able to interact with the environment and other agents, increasing their dynamism and impacting their decisions and behaviours. **Fig. 4** shows an overview of the *Agent* class and its relationships to its nearest sibling classes within the CSS ontology.

'How are an agent's attributes defined?'

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

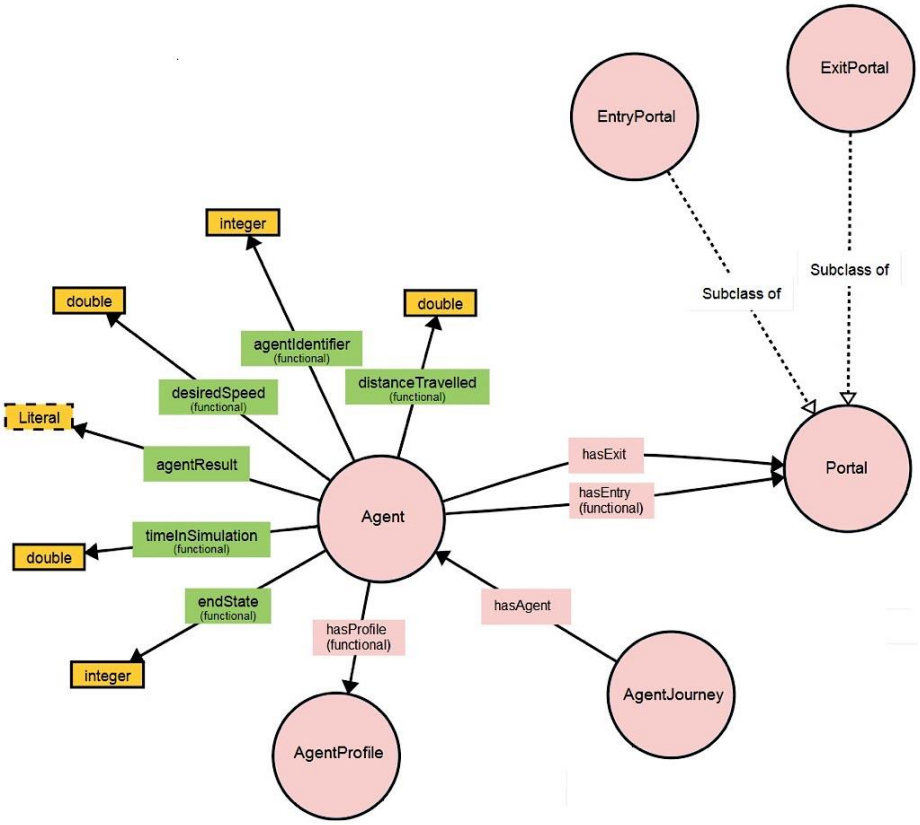

Fig. 4. CSS ontology Agent class and its relationships

The data properties shown in yellow in **Fig. 4** store agent specific data about its identity and behaviour within a model. Example competency questions on Agent data properties include:

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

5 Mapping CSS to IfcOwl

As mentioned in section 2, the CSS ontology was tested in practice using a prototype system [4], where IfcOwl version 2x3_TC1 was considered the BIM source. The initial alignment was done by looking at the similarities in terms of terminology and structure between the ontologies. Any additional common concepts which were relevant to the testing were also added later, with a list of the aligned concepts shown in **Table 1**.

All of the identified common concepts deal exclusively with the geometric parts of the model. No evident concepts which are relevant to the agents, or events are present within the IFC schema, mainly due to the different scopes of the two models.

Table 1. Alignment between CSS and IfcOwl

Subject	Predicate	Object
ifc: IfcWall	**rdfs: subClassOf**	css: Barrier
ifc: IfcWallStandardCase		
ifc: IfcCurtainWall		
ifc: IfcColumn		
ifc: IfcRailling		
ifc: IfcFurnishingElement		
ifc: IfcSpace	**owl: equivalentClass**	css: Space
ifc: IfcDoor		css: DoorLink
ifc: IfcStair		css: StairLink
css: LiftLink	**rdfs: subClassOf**	ifc: IfcTransportElement
css: EscalatorLink		

No object or data properties were identified, mainly due to the object-oriented structure of the IFC schema. The mapping of the geometry itself was not envisaged as part of the CSS ontology in relation to IfcOwl, mainly due to the nature in which IFC stores geometry, which can be cumbersome to deal with on the ontology level. Instead, the CSS ontology attempts to identify the objects within the BIM and attempt to understand their roles within a simulation. The importing of geometry should be taken care of separately, depending on the CST which is chosen for the evacuation simulation.

The majority of aligned concepts which represent and *IfcProduct* have been classified as a sub-class of *Barrier*. For the scope of CSS, the specific nature of the static environment is not vital. The important aspect is to identify the building elements which create a 'barrier' for agents to avoid when walking on surfaces. This includes elements such as walls, columns, furniture, railings, etc. Other IFC products which are not directly in the path of inhabitants were not included in the alignment, so as to simplify the geometry to its essential components.

The specific role of the CSS is to describe a scenario's context, and therefore by using reasoning via rules it was possible to extract extra information from an IfcOwl model. However, this did not result in a clear alignment. The most used object for identifying the context relied on the *IfcProperty* class from IfcOwl, which was used to differentiate between different spaces. However, this is highly dependent on the data being present explicitly within the IFC instance model in the first place.

6 Use case – querying the CSS ontology

To demonstrate the use of the CSS ontology for creating simulation scenarios, two example queries representing two workflow steps are outlined below.

The first step involves identifying the necessary model objects which are provided for the simulation tool. A large part of these objects, are those representing the building environment, which are rooted in the BIM model, in this case within the IfcOwl model, which was translated in order to be processed in a graph database environment. **Fig. 5** shows an example query on selecting all the ifcOwl ontology individuals, which are

now implicitly *css:ModelObject* individuals due to the mapping between the two ontologies, as presented in section 5. This is the first step in filtering the necessary objects to be exchanged from the BIM to the CST, thus enabling knowledge already expressed in the ontologies to be applied in a similar way to a Model View Definition (MVD) protocol. The secondary effect of this is that it implicitly leaves out all the objects which are out of scope for a crowd simulation scenario (roofs, curtain walls, foundations, etc).

```
PREFIX css:     <http://icompe.engineering.cf.ac.uk/ontologies/CrowdSimulationScenario#>
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 = css:ModelObject) }
```

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

Fig. 5. Example SPARQL query selecting IfcOwl individuals which are needed to construct crowd simulation models

Once these objects are identified, they are stored, and further information is retrieved using various other SELECT queries, which are always matched in memory using the *IfcIdentifier*. The structure the IFC schema and the long nature of SPARQL queries prevents the efficient retrieval of all the data in one go.

The second step involves understanding the context of the model. The example given below involves identifying the spaces within the building which are known to be inhabited by people, referred to in the CSS ontology as *InhabitedSpace* individuals, as mentioned in section 4.1. This is done using the SWRL rule presented in **Fig. 6**, which is triggered when the SPARQL query in **Fig. 7** is made to the ontology server.

```
Reasoning: If a Space has occupants assigned to it,
           it is considered an InhabitedSpace

css:Space(?space) ^ css:occupants(?space, ?number)

-> css:InhabitedSpace(?space)
```

Fig. 6. A SWRL rule within the CSS ontology which identified spaces inhabited by agents

Based on the available information from the BIM, external design data or user input, a *Space* is considered inhabited if it has occupants. In a building design context, occupancy information is not always explicit within the BIM for each space. To account for this limitation, external design guidance data linked to the CSS ontology multiplies the

Space area by occupancy factors, thus suggesting initial populations for each *Space*. In the case of existing buildings, this capability could be replaced by the use of IoT sensors which are able to capture the locations of the inhabitants, thus creating scenarios which are highly representative of a real case evacuation.

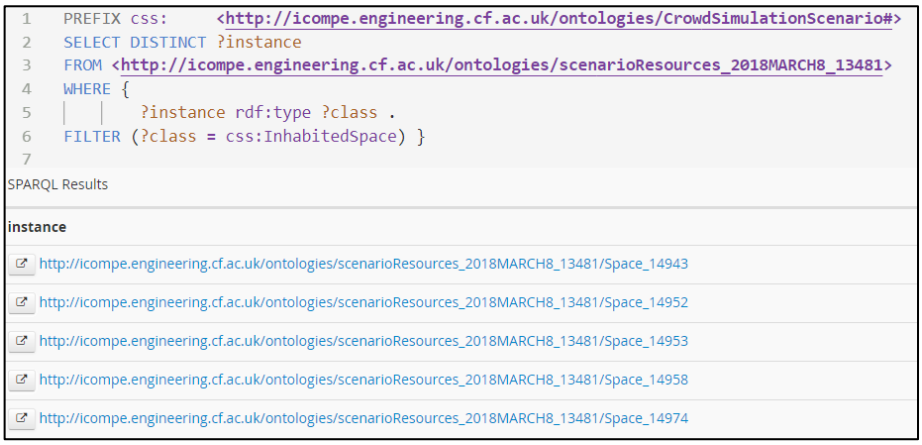

Fig. 7. Example SPARQL query selecting the InhabitedSpace individuals

7 Summary and future work

The ontology for representing a crowd simulation domain within the context of building evacuation was outlined based on the described methodology. Its structure was designed to generically represent the domain, as well as to be used in conjunction with several CSTs, and includes concepts for referring to the build environment, the inhabitants, the events which describe the evacuation as well as the results for analysis. The connection to the BIM was described based on the IFC schema, using the IfcOwl ontology, shown in section 5. Although this proved to be a reliable source of information, the common objects identified are exclusively related to geometry. Additional context can be inferred from the ontology using rules, as shown in section 6.

Future work will look at new ways to connect to the building environment, possibly through the use of newer more efficient ontologies, such as BOT, but most importantly to consider ways to connect to other potential sources of information available on the web, such as sensors or other IoT devices which can better describe the inhabitants of existing buildings.

References

[1] E. Ronchi, E. D. Kuligowski, D. Nilsson, R. D. Peacock, and P. A. Reneke, “Assessing the Verification and Validation of Building Fire Evacuation Models,” *Fire Technol.*, pp. 197–219, 2014.

[2] S. D. Khan, L. Crociani, and G. Vizzari, “Pedestrian and crowd studies:

Towards the integration of automated analysis and synthesis," *SCS M&S Mag.*, vol. 3, no. 3, 2014.

[3] D. Nilsson and R. Fahy, "Selecting scenarios for deterministic fire safety engineering analysis: life safety for occupants," in *SFPE Handbook of Fire Protection Engineering*, Springer, 2016, pp. 2047–2069.

[4] C. Boje and H. Li, "Crowd simulation-based knowledge mining supporting building evacuation design," *Adv. Eng. Informatics*, vol. 37, no. April, pp. 103–118, Aug. 2018.

[5] B. Zhan, D. N. Monekosso, P. Remagnino, S. A. Velastin, and L.-Q. Xu, "Crowd analysis: a survey," *Mach. Vis. Appl.*, vol. 19, no. 5–6, pp. 345–357, Oct. 2008.

[6] S. Gwynne, E. R. Galea, M. Owen, P. J. Lawrence, and L. Filippidis, "A review of the methodologies used in the computer simulation of evacuation from the built environment," *Build. Environ.*, vol. 34, no. 6, pp. 741–749, Nov. 1999.

[7] E. D. Kuligowski, "Computer evacuation models for buildings," in *SFPE Handbook of Fire Protection Engineering*, Springer, 2016, pp. 2152–2180.

[8] PD 7974, "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, 2004.

[9] E. D. Kuligowski, "Human behavior in fire," in *SFPE Handbook of Fire Protection Engineering*, Springer, 2016, pp. 2070–2114.

[10] A. Trento, A. Fioravanti, and D. Simeone, "Building-Use Knowledge Representation for Architectural Design," in *Proceedings of eCAADe 2012*, 2012, vol. 1, pp. 683–690.

[11] T. Onorati, A. Malizia, P. Diaz, and I. Aedo, "Modeling an ontology on accessible evacuation routes for emergencies," *Expert Syst. Appl.*, vol. 41, no. 16, pp. 7124–7134, Nov. 2014.

[12] C. Damrongrat, H. Kanai, and M. Ikeda, "Increasing situational awareness of indoor emergency simulation using multilayered ontology-based floor plan representation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8017 LNCS, no. PART 2, pp. 39–45, 2013.

[13] G. Poveda, E. Serrano, and M. Garijo, "Creating and Validating Emergency Management Services by Social Simulation and Semantic Web Technologies," in *International Conference on Ubiquitous Computing and Ambient Intelligence*, 2014, pp. 460–467.

[14] C. Prudhomme, C. Cruz, A. Roxin, and F. Boochs, "A Framework to Improve the Disaster Response Through a Knowledge-Based Multi-Agent System," *Int. J. Inf. Syst. Cris. Response Manag.*, vol. 9, no. 3, pp. 96–109, 2018.

[15] N. Nuo, W. Tay, N. K. Ubota, and J. Botzheim, "Building Ontology for Fire Emergency Planning and Support," vol. 8, pp. 13–22, 2010.