Formalizing a Language for Specifying Building Components from IFC

Ernest L. S. Abbott[1], **Kerwei Yeoh**[2] **and David K.H. Chua**[3]

Abstract

Identifying construction requirements is a complex task that is often overlooked. With the advent of Building Information Modelling (BIM), building components such as beams, columns and slabs are represented as objects using the IFC specification. This paper proposes a Building Component Specification Language which formalizes a method of specifying the elements which make up the spatial attribute of a construction requirement. The advantage of the proposed language is an easily understood mechanism which allows low level specifications to be easily represented using an iterative mechanism. This iterative mechanism allows the scaling of low level specifications to a higher level one, thus significantly reducing the complexity of user entry.

Keywords: BIM, Construction Requirements, IFC, Building Component Specification Language.

Introduction

Construction Requirements have been defined by Chua and Yeoh (2010) as being the requisite preconditions for construction to be carried out. In general, these construction requirements encompass the various regulatory, safety and engineering constraints applicable, and are important as an explicit representation of construction knowledge.

The lack of frameworks to represent these construction requirements within construction schedules has led to contractors being unable to adequately capture the impact of missing requirements on their schedules, leading to undesirable delays and potential cost overruns. Instead, contractors examine building plans and apply their construction experience to analyse the relationship between building components. They then determine what they consider to be an effective and efficient construction process. Consequently, the construction schedule is generated in an ad hoc manner. At best, these schedules are time consuming to plan, and become more tedious as the complexity of the project increases. At worst, the plans are erroneous and infeasible; amounting to unnecessary cost and rework (Yeoh and Chua 2014). To date, using construction requirements as a construction knowledge modelling tool has seen applications in automatic scheduling (Chua et al 2013), reasoning on building's structural functions (Yeoh 2012), as well as on extending reasoning capabilities for naval construction databases (Abbott et al 2010).

[1] Research Engineer, Dept. of Civil & Environmental Engineering, National University of Singapore, Block, E1A #07-03. No 1 Engineering Drive 2. Singapore 117576. (65) 65164643. Fax (65) 67791635. Email. ceeaels@nus.edu.sg

[2] Lecturer, Dept. of Civil & Environmental Engineering. National University of Singapore. Block E1A #07-03. No 1. Engineering Drive 2. Singapore 117576. (65) 65164643. Fax (65) 67791635. Email. ceeykw@nus.edu.sg

[3] Associate Professor. , Dept. of Civil & Environmental Engineering, National University of Singapore, Block, E1A #07-03. No 1 Engineering Drive 2. Singapore 117576. (65) 65162295. Fax (65) 67791635. Email. ceedavid@nus.edu.sg

To address the above problem, Yeoh and Chua (2014) provided a formal representation to describe the behaviour and function of a construction requirement. Their representation schema identified a requirement of requirements, namely: temporal, spatial and metric attributes. The representation schema provided a framework for describing the construction requirement using these three attributes in a machine readable format, as well as depicting the various interdependencies between them. Chua et al (2013) further elaborated on some of these ideas, implementing an automatic schedule generation system based on the functional construction requirements.

The major challenge of employing the above framework is the difficulty of specifying construction requirements at several levels of detail. Various spatial (topological) relationships could exist between individual building components. For example, a Beam is connected to Column. However, certain relationships must be defined on a set of components, such as a slab being supported by a beam truss system. This presents an added complexity as the relationship is not just between the slab and individual truss elements, but also between slab and the entire truss system.

With the advent of Building Information Modelling (BIM), building components such as beams, columns and slabs are represented as objects using the Industry Foundation Classes (IFC) specification. IFC is a platform neutral, object-based, open file format data exchange. More importantly, it is now an industry standard and is complied with by major BIM software platforms. Various building objects are parametrically described in an IFC file, thus making such a file suitable for incorporating into the representation of construction requirements. The current release of the IFC standard specification, IFC4 has 766 entities defined within (Liebich 2013).

This paper will introduce a Building Component Specification Language (BCSL) which will be used as a constructor to specify building systems based on the components and its associated parameters. This is a vital step in the description of construction requirements.

Background Review: Construction Requirements

The proposed Construction Requirements by Yeoh and Chua (2014) uses an upper level core ontology to establish the spatial, temporal and ordinal nature of construction requirements. As construction requirements may be expressed in many forms, from contractual conformance requirements, to site requirements expressed in natural language between contractors, a taxonomical schema is defined on the entities constituting the construction requirement.

The upper level core ontology (shown in Figure 1) starts with the characterization of a core immutable set of attributes. These core attributes are then instrumental in defining the entities making up the construction requirement, by providing a fundamental basis for describing their key characteristics. These in turn form the knowledge constructs of the requirement.

Figure 1. Approach Adopted for defining Construction Requirements

Core Attributes of Construction Requirement Entities

A construction requirement is defined by its constituent entities. An entity of a construction requirement may be defined by establishing one or more of the following characteristics:

- Tangibility (Spatial): Tangible entities are entities which can be perceived, and inherently have spatial attributes. These tangible entities are extracted from the IFC model.
- Temporal behaviour: Temporal behaviour is derived from the schedule, and is typically associated with the individual IFC element to obtain a fourth dimension.
- Measurability (Ordinal): Measurability refers to the perceptible measure. These may be inferred directly or indirectly from the parameters of the IFC elements.

The above establishes a "requirement" of requirements, covering the spatial, temporal and measurable aspects. This paper focuses on the rules that describe the set of entities making up the spatial aspects of the construction requirement.

Illustrating Construction Requirements via an Example

A simple example of a simple construction requirement is now established using concepts from prior sections. A construction requirement is the confluence of purpose and operation. Hence, the taxonomy of a construction requirement can be defined as: Purposive, Operational and the necessary conditions defining the interaction between the two.

Purposive and Operational

The purpose for a construction requirement may be defined as fulfilling the desired intention. This intention takes the form of a function. A function is the action of performing an intention, and is physical, thus directly involving both spatial and temporal dimensions (entities). For example, requirement R1 could be "R1: *IfcColumn* C1 Supports *IfcBeam* B1". The "Support" indicates a physical function by C1, to be used by B1.

The purpose of the requirement is intimately tied to the IFC entities within the system. For these functions, further categorization into function users and function providers is necessary. Users are the requesters of the function, providing the purpose for the requirement; Providers provide the function, exhibiting the operational behaviour needed to fulfil the requirement. Using the above example "R1: C1 supports B1", C1 is the function provider (operational), while B1 is the function user (purpose).

Necessary Conditions

The necessary conditions are the conditions which must be fulfilled before the requirement is available for proceeding.

Challenges and Research Motivation

Firstly, the representation of a simple functional construction requirement will be illustrated. The functional requirement R1 is modelled using the above schema as shown in the following figure:

Figure 2. Functional Requirement Example

In Figure 2, the purposive and operational aspects of the requirement are identified with the *function user* and *function provider* respectively. *IfcBeam* B1 requires a support, and this support is to be provided by *IfcColumn* C1. Additional details regarding the definition of *function user* and *function provider* will not be discussed within the context of this paper.

The above example illustrates a single construction requirement. This construction requirement is at an extremely low level, meaning that it applies to a small subset of building components. However, a single construction project is likely to encompass numerous construction requirements, and some of these requirements may be at a higher level. For example, a requirement could be that all beams have to be supported by columns.

The implication of the above is that a method of specifying building components covering both higher and lower levels must be devised. Furthermore, the specification method must allow lower level sets of elements to be automatically generated from the higher level specification. To achieve this goal, this paper proposes the Building Component Specification Language (BCSL) to derive a partial model based on specified IFC attributes and/or parameters.

Review of Related Literature

The current literature on related technologies consists of database query languages. Structured Query Language (SQL) is the standard, with many various dialects available. SQL is based on relational algebra, and inspects structured data represented by this relational algebra. While it is a de facto standard, SQL is generic and not domain specific, making it difficult to implement on IFC data models.

Filtering of IFC data models is accomplished by several specialized approaches, which implicitly handles the internal data representation. One example is the Partial Model Query Language (PMQL) by Adachi (2003). This is based on an XML representation, and provides Select, Update and Delete operators. It is implemented by integration with the IFC Model Server. Users use the Select statement to define their own filtering, where this may comprise several nested Select statements.

Another example is the Generalized Model Subset Definition Schema developed by Weise et al (2003). Again, this achieves a filtering of the entire IFC model to a partial model using a declarative approach. During the filtering, objects are filtered first at the instance level, and then processed on the defined model view definition. Due to this declarative approach, ad-hoc queries which can be performed which filters according to attribute and type values.

Object Interaction Query (oIQ) by Ogueta and Carlsson (2014) introduces the idea of context awareness to evaluate the interactions between building components in BIM. While it is not written in IFC, it queries a proprietary BIM database to extract the necessary quantities and parameters. oIQ makes a clear distinction between the direct and indirect parameter queries needed for context aware evaluation, and demonstrate the use of this in a design example.

BIMQL (Mazairac and Beetz 2013) lies at the confluence of PMQL and SPARQL, and is able to work on complex IFC models which reside on BIMServer. Created with a graph based mechanism, it is able to traverse complex referential models. It potentially features a Create, Read, Update and Delete functionality, even though only the Create and Update features are currently implemented.

QL4BIM extends the basic SQL query mechanism by providing a spatial semantic based on topological predicates (Daum and Borrman 2014). The topological predicates are developed based on octree based approaches with Boundary Representation (Brep) methods.

This enhances the query semantic, allowing more expressive partial model extraction to be carried out.

The above technologies represent the current state of the art in the extraction of partial models from IFC models. However, they are not able to provide a means of quickly specifying a high level specification, required for a construction requirement.

Building Component Specification Language Framework

Development of the Specification Language

The development of any user driven language has to be easy to use, since users are not required to have any prior programming knowledge. To this end the language has to have a simple and intuitive syntax and structure, with references to natural language. This paper is concerned with the theoretical framework for such a language and how it can be used in the automated generation of construction requirements from an IFC data format.

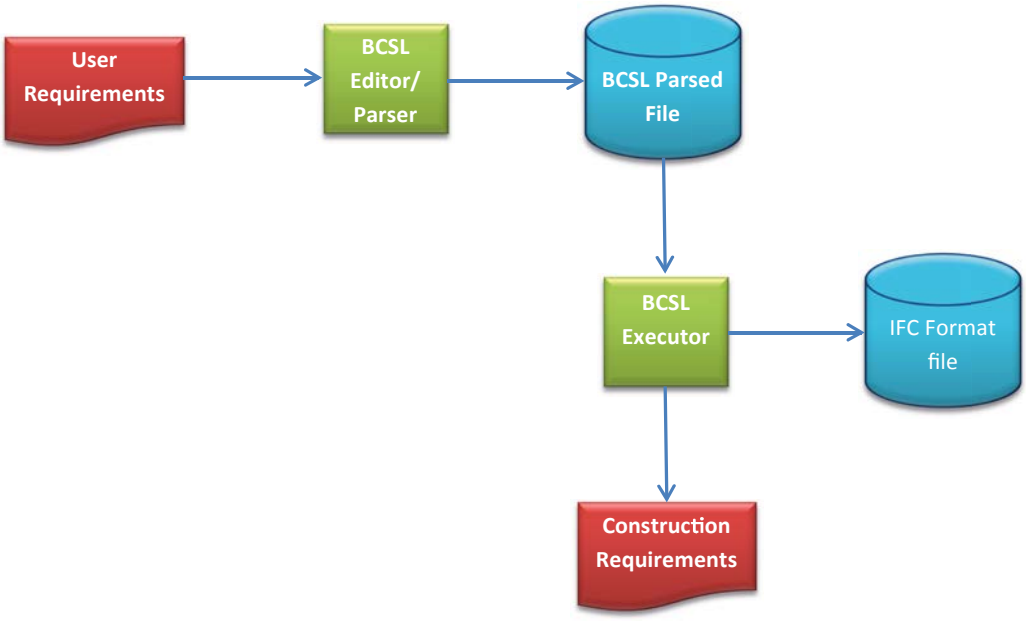

Figure 3: BCSL Workflow

Figure 3 shows the usage flow of the BCSL. BCSL distinguishes IFC data from the BCSL language. The IFC file contains the project specific IFC data, while the BCSL processor contains the generic specification language syntax. The user enters simple English like commands via the BCSL editor/parser. The result of this is an output that can be understood by the subsequent module, the *Language Executor*. The *Language Executor* reads an IFC file and produces construction requirements.

IFC File Data

IFC comprises 8 domains: - Building control, Plumbing and Fire protection, Structural Elements, Structural Analysis, HAVC, Electrical, Architecture and Construction Management. Not all domains are present in every IFC file. The domains exported to an IFC file depends on the contents of the original files. This paper focuses on the major structural elements necessary for construction (*Beam, Column, Slab* and *Wall*), but may be extended as required.

The basic data exported from these components is their geometrical description together with their location. Additional parameters, if entered in the exporting software, will also be in the IFC file. These parameters are not mandatory and so may be absent. Two types of additional parametric data are identified: Direct and Indirect parameters. Direct parameters are directly extracted from the IFC element. Indirect parameters are derived from the geometric description of the building object or between a set of objects, and subject to further manipulation. For example, the weight of an element is an indirect parameter, derived by assuming a notional density for the material of the component, multiplied by its volume.

Table 1: Building Elements and Their Parameters (Indirect Parameters in Italics)

Building Element	Parameters Exported
Beam	Nominal Length, Cross Section Area, Outer Surface Area, Total Surface Area, Gross Volume, Net Volume, *Gross Weight*, *Net Weight*
Column	Nominal Length, Cross Section Area, Outer Surface Area, Total Surface Area, Gross Volume, Net Volume, *Gross Weight*, *Net Weight*
Slab	Nominal Width, Perimeter, Gross Footprint Area, Net Footprint Area, Gross Volume, Net Volume, *Gross Weight*, *Net Weight*
Wall	Nominal Length, Nominal Width, Nominal Height, Gross Footprint Area, Gross Side Area Left, Net Side Area Left, Gross Side Area Right, Net Side Area Right, Gross Volume, Net Volume

IFC components do not only have the basic parameters as shown in Table 1, but have inverses attributes. The formal definition of an inverse attribute is, "Inverse attributes are excluded from files but define queries for obtaining related data and enforcing referential integrity. Inverse attributes are similar to the term 'navigation property' in entity-relational programming frameworks." (Liebiech 2013).

The inverse attributes that are of interest in this paper are, *ConnectedTo*, *ConnectedFrom* which are self-explanatory. The inverse attribute *IsConnectionRealization* is not considered as this is a connection that requires additional information to make the connection.

Structure of BCSL

The BCSL consists of basic constructs that are derived from natural language, thus making it easily comprehensible and intuitive. The underlying concept of the BCSL is that of Sets. Each of the four main structural components of interest, *Beam, Column, Slab* and *Wall* are thought of as sets. There is a BEAMSET, COLUMNSET, SLABSET and WALLSET. Each set retrieved is given a name, and the conditions of retrieval are specified. To retrieve a set of building components, the language statement has the following format:

SET *name*: ***ELEMENT_1.LEVEL****(n) Relationship* ***ELEMENT_2.LEVEL****(n+1)* [**WHERE** *n=i* **TO** *n = j*]

The capitalized embolden words are language keywords, while the italicized words are variables or attributes.

- SET is a placeholder for a set of reserved language keywords. In this paper, the allowable SET comprises COLUMNSET, BEAMSET, WALLSET, SLABSET and ELEMENTSET. The SET identifies the type of element to be retrieved.
- *name* is the name of the extracted set. This can be any name the user chooses. This is the retrieved set name.

- ELEMENT_1.LEVEL refers to a specific building component at a specific building level. An example is COLUMN.LEVEL. The COLUMN component is a building element from the IFC Model.
- *n, n+1* are variables of the levels within the structure. Within the IFC file the elevations have arbitrarily named, without a globally standard convention. To make the system usable, levels are derived from information in the IFC file. Level 1 is considered the level with the lowest elevation in the construction.
- [WHERE *n=i* TO *n = j*] is optional. The value *n* may be a defined value, e.g. 10 or may be left as *n*, in which case the WHERE key word could be used. However, if the level is defined as *n* and there is no WHERE keyword, the language defaults to *i=1* to *j=maximum level -1*. We further restrict *i* to be always less than *j*.
- *Relationship* is the topological relationship between elements, for example, *ConnectedTo* and *ConnectedFrom*.

To retrieve a set of columns which supports beams in a 10 storey building, the following specification can be used. The language iterates through the ten levels in the building to retrieve all columns that fulfil the condition that they are connected to beams.

COLUMNSET *SET0*: COLUMN.LEVEL(*n*) *ConnectedTo* BEAM.LEVEL(*n+1*) [WHERE *n=1* TO *n=10*]

Not all columns support beams in the LEVEL immediately above. To retrieve the set of columns that fulfil this condition the following statement would be used.

COLUMNSET *SET1*: COLUMN.LEVEL(*n*) *ConnectedTo* BEAM.LEVEL(*m*) [WHERE *n=1* TO *n = j-1*] [WHERE *m=n+1* TO *m = j*]

Columns associated with beams have been identified in the COLUMNSET statement above. Beams associated with slabs by means of support are retrieved using the following BEAMSET statement.

BEAMSET *SET2*: BEAM.LEVEL(*n*) *ConnectedTo* SLAB.LEVEL(*n*) [WHERE *n=1* TO *n = 10*]

Slabs may be extracted as a separate set, using the same techniques above. Omitting the attribute simply retrieves all elements in a particular level.

SLABSET *SET3*: SLAB.LEVEL(*1*)
SLABSET *SET4:* SLAB.LEVEL(*n*) [WHERE *n=1* TO *n = 10*]

The above language allows low level specifications of building components to be generalized to a higher level, by providing an iterative mechanism. This addresses the identified research challenge of being able to automatically scale low level specifications from a higher level one.

Boolean Manipulation of Sets

Having identified the sets of components, Boolean algebra can be used to extend the information. The various Boolean operators have also been incorporated as language keywords for implementation. These include: UNION, INTERSECT and NOT. For example,

to obtain a full set of connectivity among columns, beams and slabs, the following construction would be used.

ELEMENTSET *elemset*: COLUMNSET *SET0* UNION BEAMSET *SET2* UNION SLABSET *SET3*

Not all beams support slabs. To retrieve those that do not support any slab the following BEAMSET statement is used. This returns the complement of the BEAMSET *SET2*, where an intermediate BEAMSET *SET5a* is called, and a set difference operation carried out to remove all elements in BEAMSET *SET2* from *SET5a*.

BEAMSET *SET5*: BEAM.LEVEL(*n*) NOT *ConnectedTo* SLAB.LEVEL(*n*) [WHERE *n=i* TO *n = j*]

BEAMSET *SET5a*: BEAM.LEVEL(*n*)

The Boolean manipulation allows complex specifications of the IFC model to be created. This allows partial models comprising specific building components to be generated, which will be used to define the spatial elements within the construction requirement.

Conclusion

Construction Requirements represent the key preconditions for construction to take place, and omission of these requirements may cause delays in the construction schedule. However, properly defining each and every construction requirement is a tedious process due to the numerous requirements available. A method of generalizing the elements to facilitate the representation of these requirements is thus needed.

This paper proposes a Building Component Specification Language (BCSL) to overcome this problem. The key innovation of the BCSL lies in the iterative method used which allows the scaling of high level to low level specifications.

Future work involves developing a more general representation of the topological relationships between the elements. This will enable a spatial semantic to facilitate the development of construction requirements from a full IFC model. The authors will also demonstrate the approach on a full-fledged construction method. The demonstration of this is too long to include in this paper.

References

Abbott, E. L. S., Liu, Z., Chua, D. K. H., and Lim, C. L., 2010. Extraction of Ship Product Design Data. *In Proceedings of the 2010 International Conference on Engineering, Project, and Production Management*.

Adachi, Y., 2003. Overview of partial model query language. *In*: *Proceedings of the 10th International Conference on Concurrent Engineering*.

Chua, D. K. H., Nguyen, T. Q., and Yeoh, K. W., 2013. Automated construction sequencing and scheduling from functional requirements. *Automation in Construction*, 35, 79-88.

Chua, D. K. H., and Yeoh, K. W., 2010. PDM++: Planning framework from a construction requirements perspective. *Journal of Construction Engineering and Management*, 137(4), 266-274.

Daum, S., and Borrmann, A., 2014. Processing of Topological BIM Queries using Boundary Representation Based Methods. *Advanced Engineering Informatics*, 28(4), 272-286.

Liebich, T., 2013. [online]. Available from: http://www.buildingsmart-tech.org/ifc/IFC4/final/html/index.htm [Accessed 17 April 2015].

Mazairac, W., and Beetz, J., 2013. BIMQL–An open query language for building information models. *Advanced Engineering Informatics*, 27(4), 444-456.

Ogueta, C. S., and Carlsson, M., 2014. Object Interaction Query: A Context Awareness Tool for Evaluating BIM Components' Interactions. *Blucher Design Proceedings*, *1*, 269-273.

Weise, M., Katranuschkov, P., and Scherer R.J., 2003. Generalized model subset definition schema. *In: Proceedings of the 20th CIB-W78 Conference on Information Technology in Construction.*

Yeoh, K.W., 2012. *Construction Requirements-Driven Planning and Scheduling with Spatial-Temporal Constraints using an Artificial Intelligence approach*. Thesis (PhD), National University of Singapore, Singapore.

Yeoh, K.W., and Chua, D. K. H., 2014. Representing Requirements of Construction from an IFC Model. *Computing in Civil and Building Engineering* (2014): pp. 331-338.