Törmä, S. (2013). Semantic Linking of Building Information Models.
Seventh IEEE International Conference on Semantic Computing (ICSC 2013),
Irvine, CA

Semantic Linking of Building Information Models

Seppo Törmä
Department of Computer Science and Engineering
School of Science, Aalto University
Espoo, Finland
Email: seppo.torma@aalto.fi

***Abstract*—Information is produced and maintained in a building project in a concurrent and loosely coupled manner by a set of parties representing different disciplines. The multiple partial models produced in a project are interrelated since they represent different aspects of a same building. The standard conceptual schema of buildings – Industry Foundation Classes (IFC) – enables conceptual, type-level compatibility between models but solves interoperability problems only partially. We argue for the need for instance-level interoperability. The capability to link the individuals of different models together would support functionalities such as cross-model information access and aggregation, status monitoring, and change management. The nature of cross-model linking is investigated using a small case study and a set of principles for the implementation of linking are identified. The natural way to implement the principles using Linked Data technologies is outlined. In a qualitative evaluation against previous approaches, the linking approach turns out to be more flexible and expressive but more demanding of support from design tools and information management infrastructure.**

I. Introduction

Building information modeling (BIM) has evolved from object-based parametric 3D modeling [1]. By combining the geometric information with other properties – costs, materials, process, etc. – a range of new functionalities becomes possible, including cost estimations, quantity takeoffs, or insulation analyzes. There are many advanced, discipline-specific BIM design tools[1] that have had a significantly impact on the productivity and quality of individual design tasks.

Each design tool allows the creation of a *partial model* that represents the building from a particular perspective: requirements, architecture, structural design, mechanical systems, schedule, and so on (Figure 1). Tens or even hundreds of different partial models – also called *aspect models* [2], [3] or *discipline models* [4] – can be created in large projects.

Partial models are *different* perspectives to a *same* building. Each of them contains specific information required by some particular downstream activities. For instance, the architectural model represents the building from the perspective of usage and esthetics, while the structural model from the perspective of construction and stability. However, since the models are representations of the same physical end result, they are necessarily *interrelated*. Each model has areas of overlap with some other models, for instance, concerning the locations and dimensions of spaces and walls, alignment of entities, allocation of shared space (spatial overlaps), use of construction resources, and various groupings of entities.

The overlapping areas of designs must be in agreement with each other. Inconsistencies between models will result in potentially costly problems in a construction phase: spatial collisions, omissions, delays, and incompatible components. These often require redesign of aspects of a building, replanning of the work, or rework at the construction site. The problems can inflict the schedule of the project and have long-lasting effects on the eventual quality of the building.

While BIM has enabled advanced design tools and improved individual design tasks, so far it has had only minor impact on the processes and workflows in building projects. No large-scale, project-wide efficiency improvements have been actualized yet, despite improvements in individual tasks. What is needed is the *interoperation* between different design and construction tasks: the capability to *exchange* information between tasks and to *use* the information exchanged [5].

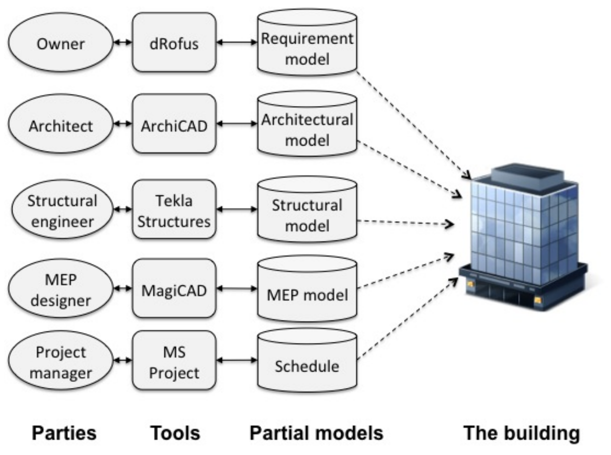

Fig. 1. Multiple partial models of a building

BIM interoperability technologies have been developed by BuildingSmart consortium, formerly known as the International Alliance for Interoperability (IAI). A central interoperability technology is IFC (Industry Foundation Classes[2]), a common data model for representing buildings. IFC has been developed since 1994 and specifies slightly over 600 entity types. It is being standardized by ISO as ISO16739.

IFC is specified in the interoperability domain of STEP (ISO10303 STEP – Standard for the Exchange of Product

[1]For a list of tools, see: `http://www.ifcwiki.org/`

[2]http://www.buildingsmart-tech.org/

Data) that uses EXPRESS language for schema definition, STEP files to instance data exchange, and GUIDs (Globally Unique Identifier[3]) to identify entities. Even though STEP is one of the largest standards of ISO, this representational approach is foreign to most application developers. The IFC schema, the most valuable result of the IFC effort, is largely independent of STEP technologies. It has been shown earlier [6]–[9] that the IFC schema and data files can be converted into representation used in the area of Semantic Web and Linked Data [10]–[12]. This approach is also adopted in this research.

The current way to share building information models is predominantly based either on *point-to-point exchange* of IFC files, or sharing of IFC files through *centralized model repositories* [13]. Models can also be shared through federated databases, distributed event-based systems, service-oriented architecture, and cloud [1], [14], [15]. All of these methods solve the easy part of the interoperability problem, the information *exchange* between tasks while the difficult part, the *use* of the exchanged information in the receiving tasks remains largely unsolved: it requires human interpretation and manual work.

The utilization of information across models can take many forms: the *access* to properties of linked entities across models, *aggregation* of information from several models enabling consistency checks and advanced analyzes, *monitoring the status* of entities across models (designed/fabricated/installed), and *management of changes* that affect entities across models.

To support these tasks, we need to know *how individuals in different models are related*. There are at least three possible approaches to that. (1) In *comparison* the relations of objects are determined by geometrically or visually comparing two models, each time from the scratch. (2) *Fusion* means that objects in different models are merged together in a central server. The original objects loose their independence and the centralized objects will have all their properties and relations. (3) In *linking* approach a variety of meaningful relations are maintained between the objects describing same real-world entities. The objects remain independent in different models.

The objectives of this study are (1) to analyze the needs and types of links across BIM models, (2) to outline a Linked Data approach for link management, and (3) to evaluate linking method against the comparison and fusion approaches.

The method is a case study of a minimal "building" consisting just of a wall with a door and a few penetrating pipes. In several workshops of the DRUM project[4] an expert group developed three partial models of the case building, and identified the central interactions between the models.

Six principles for the implementation of linking are identified: two-level linking (model-level and instance-level), model-level defined link generation actions, two-directional linking, external linksets, public linksets, and independent utilization of linksets. The representation and architecture based on Linked Data are presented. When compared with other approaches, linking turns out to be more flexible and expressive, but it requires tool and infrastructure support to work.

Below the characteristics of BIM are first reviewed to justify the need for cross-model linking. A case study is processed to identify the types of cross-model links. An implementation based on Linked Data is outlined. A qualitative evaluation of cross-model linking against previous methods is provided. The paper is concluded with a discussion.

II. Managing BIM information

Building projects have specific characteristics that reduce the applicability of traditional centralized, tightly-coupled, high-investment, and closed information management solutions. Below we give an overview of these characteristics and present possible interoperability approaches.

A. Characteristics

Distributed, building-focused information. Numerous models are created and maintained during a building project in a distributed manner, ranging from requirements, architecture, and structural design to financial calculations, fabrication drawings, procurement specifications, and schedules. For the clarity of legal responsibilities, the maintenance of each model is typically done by the one party that has the proper tools and expertise for it. Despite the uniqueness of ownership, however, all relevant information should be readily available to all parties whose work depend on it.

Organizational fragmentation. As a building is a large, unique, and complex physical entity, often the combined effort of hundreds of companies is needed to define, design, fabricate, and construct it. A set of companies with heterogeneous information management practices and capabilities comes together to form a unique and short-living organizational configuration. Due to short partnerships, business processes or information systems wont be harmonized across companies. The fragmentation is further increased by contractual relationships. Competitive bidding, that is typically used to award contracts, increases the variability from one project to another, and contractual boundaries add friction to information exchange. Building projects have a network-like organization established by pairwise contractual relationships, often without a unique central party. There can be long chains of subcontractors and even relatively independent subprojects. Such a peculiar organizational environment calls for a loosely-coupled, decentralized approach to information management.

Ubiquitous changes. The common assumption that earlier models would be freezed when subsequent models are created fails time after time in building projects. Changes are ubiquitous during a project and cause a lot of extra work. A change can propagate from earlier models to subsequent models, or backwards, or through resource constraints to parallel models. An innocent change in requirements, such as increasing the floor height, can lead to complete revision of all other models. It is wise to accept that any model can change at any point of a project potentially causing extensive changes in other models.

[3]GUIDs: `http://www.ietf.org/rfc/rfc4122.txt`

[4]DRUM (Distributed Transactional Building Information Management, 2010-2013) is a collaborative research effort funded by RYM/Tekes (Finland).

This implies that the master data should to be kept close to the designers having the required expertise and tools.

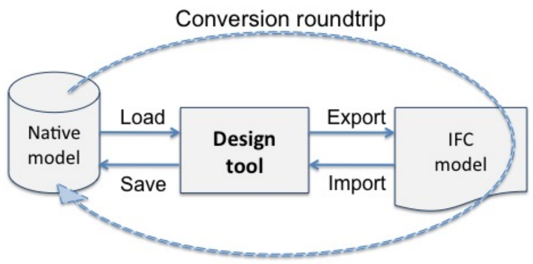

Fig. 2. Conversion roundtrip: native-IFC-native

No conversion roundtrip. The design tools all store their models in their proprietary native format, but they are able to export the models in IFC. However, the IFC version of the model can, in practice, only be used in a read-only mode, since the conversion roundtrip — exporting native model to IFC and importing that back to a native format, the load-export-import-save cycle in Figure 2 — is generally unsuccessful. Information is lost in a roundtrip and the result is a restructured native model that is difficult to develop further without extensive manual repairs. In particular, all parametric information is lost, since it cannot be represented in IFC. Consequently, changes to building information should be done first to a native model, and then re-exported to IFC.

External data sources and services. There is a need to link building information models to external information, and conversely, to refer from external information to entities in building information models. Many existing data sources are very relevant in building projects: messages, annotations, reports, bids, contracts, schedules, spreadsheets, photos, video, maps, social networks, and RFID systems. A 3D BIM model could provide users a conceptually easy index to integrate such building related information both during the project and in the operational phase. Moreover, a linking capability would reduce the future needs to extend IFC schema to cover new domains.

B. Interoperability approaches

IFC provides *type-level interoperability* across tools that are able to export and import IFC files. Using IFC, visualization tools can provide a combined visualization of multiple partial models and analyze collisions of entities across models based on their geometries. However, many important ways to use the information across models and tasks require *instance-level interoperability* solutions that specify how individuals in different models are related to each other. Examples are:

1) *Access* to information relevant to an entity from related models. For instance, a MEP model has a duct whose dimensioning requires information about the properties of the spaces that it serves; the space entities and their properties, however, are located in the architectural model. Cross-model links between the duct and space entities could provides an access path to the properties.
2) *Aggregation* of information from several models to carry out consistency *checks* or various *analyzes* (e.g. fire resistance, or acoustic or thermal insulation). Model checking tools like Solibri Model Checker[5] nowadays can utilize information from several models; this process could be improved with proper cross-model linking.
3) *Status monitoring*: Has an entity been designed, detailed, ordered, fabricated, transported, or installed. Sharing of up-to-date status information is indispensable to integrate the design (digital) and construction (physical) sides of a building project, and facilitates the evaluation of options for late-phase changes [18].
4) *Change management* across models could be supported in different phases. Before making a change, (1) different options could be evaluated based on impact to other models. Before committing a change, (2) change requests could be sent, allowing other parties to comment. After committing, (3) focused notifications to the affected parties could be automatically generated.

There are at least three possible approaches for relating individuals in different models: comparison, fusion, and linking.

1) *Comparison* is the current practice. The relations of objects are determined by comparing two models, each time from the scratch and mostly based on geometric information. This is typically a manual task: designers visually inspect and interpret models to find out corresponding or clashing objects. BIM viewers (e.g. Tekla BIMSight[6]) that can combine several models into one geometric model can help the user in visual inspection. Parts of the task can be automated with cross-model checking tools such as Solibri Model Checker.
2) *Fusion* means that objects in different models are merged together. This kind of solution can be implemented in centralized model servers [16], [17]. The fused objects will have the union of all the properties and relations of the corresponding objects in different models. Since the models loose their independence regarding the fused objects, centralized information management and careful planning of design workflows is needed.
3) *Linking* means that a variety of semantically meaningful relations are maintained between the objects denoting the same real-world entities [9]. The objects maintain their identities and independence in different models, and a rich set of relations can be represented. This requires infrastructure solutions for the creation, storage, and maintenance of links.

The fusion approach in practice requires centralized management of related partial models. It leads into a *central as master* architecture that requires round tripping between central and native models when changes need to be done.

However, several characteristics mentioned above — no roundtrip, frequent changes, need for necessary expertise and tools, and need for unambiguous legal responsibilities —

[5]http://solibri.com
[6]http://www.teklabimsight.com/

strongly point to the *native as master* architecture in which the native models of different BIM tools contain the master data. The native models remain *independent and modifiable*. They are exported to IFC and exchanged with other project parties in *read-only* mode. No project party should directly modify the data created by others. Any needs to revise other models should happen through change requests to owners of those models. Both the comparison and linking approaches are compatible with the *native as master* architecture.

III. Case study

Below the semantics of the linking between models is analyzed using a minimal case study. The goal is to identify different kinds of relations between objects as well as the processes of their emergence and the structures that mediate them. Real-life BIM models are large, containing tens of thousands of entities, but the use case below contains as little structure as possible to enable a detailed description of links. It was developed in the DRUM project in several workshops among domain experts in the period of 11/2012—5/2013.

A. Models

The case study consists essentially of *a wall with an opening filled by a door, in which the wall is penetrated by a duct and two pipes*. Three different partial models representing it are shown in figure 3:

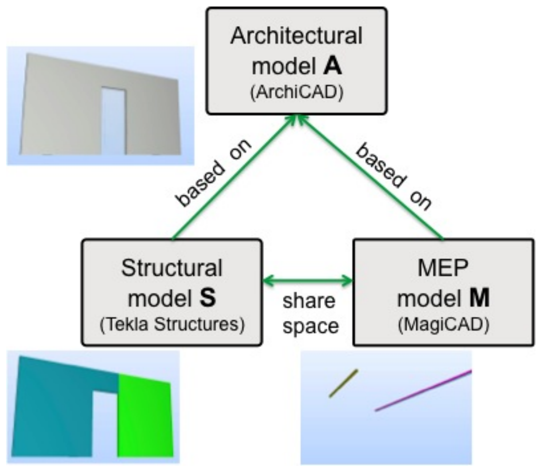
Architectural model A (ArchiCAD)
based on
based on
Structural model S (Tekla Structures)
share space
MEP model M (MagiCAD)

Fig. 3. Three models with model level relations

- *Architectural model A*: a wall, an opening, and a door.
- *Structural model S*: two structural walls (precast concrete elements) and an opening.
- *MEP (Mechanical, Electrical, and Piping) model M*: a duct and two pipes.

B. Model-level links

Figure 3 shows two kinds of relations between the models:

1) The structural model S and MEP model M are *based on* the architectural model A. It means that the structural engineer and MEP engineer take A as a reference model when starting to design their models.
2) The structural model S and MEP model M *share* the same physical *space*. It means they need to be checked against each other for spatial overlaps (clashes/collisions). Overlaps must be solved by equipping the models, for instance, with voids enveloping the overlaps.

Model-level relations are part of the model metadata; they are based on the contractual obligations of different project partners and they are known well before any of these models have any content yet.

More generally, the model-level relations between models m_1 and m_2 can be divided into the following categories:

- *Sequential*: Model m_2 is *based on* model m_1. Model m_2 receives requirements and initial data from m_1, and is generally an elaboration or implementation of m_1. The start time and end time of the design of m_2 cannot be earlier that those of m_1.
- *Parallel*: Models m_1 and m_2 *compete with* or *complement* each other. Models m_1 and m_2 have indirect dependencies through shared resources or interfaces, which means that conflicting decisions can be made on them.
  - *Model m_2 compete of a resource with model m_1*. The typical case is physical space: only one entity can reside in a same spatial point. Other capacity limited resources are cranes and unloading areas.
  - *Model m_2 complements model m_1*. Models of adjacent building zones have an indirect dependency through a shared interface. The interface is narrow if zones are different building masses and wide if they are different floors. The structures and openings at both sides of an interface must be aligned.

As noted, model-level links are *known in advance* and they *determine the specific tasks that generate instance-level links*.

C. Instance-level links

When the development of the models begins, the architectural model A is created first. It ends up containing three interesting entities: a wall, an opening, and a door. These are shown at the left side in Figure 4. The entities are related inside the model through *fills* and *voids* links. This information is created with an architectural design tool (e.g., ArchiCAD) and exported to IFC.

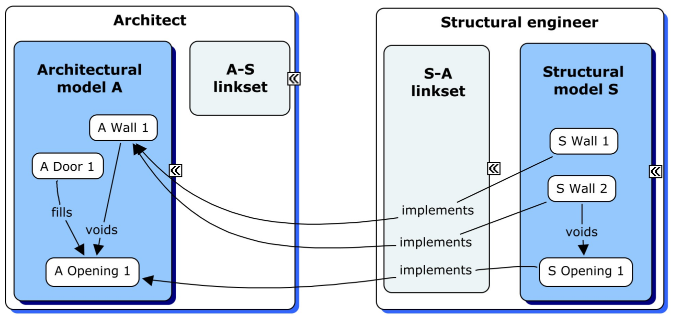

Fig. 4. Two interlinked models.

Next the architect exchanges the architectural model A with a structural engineer SE who loads it as a reference model to a design tool (e.g., Tekla Structures). The designer creates the structural entities – two walls and an opening – and links these entities to the corresponding architectural entities. The structural model S is shown in the right side of Figure 4.

Figure 4 the linkset *S-A* is shown next to S. It contains the links between structural and architectural entities created by SE. The type of the links shown in the figure is *implements*, indicating the fact that architectural entities pose requirements for the structural design and structural entities are their implementations. It should be noted that with sufficient support from the design tool the links can be generated automatically. Tekla Structures, for instance, supports the conversion of architectural entities to structural entities, and during this process could also produce the links between the entities.

The figure also shows the linkset *A-S* that should contain those links between the two models created by the architect. At this stage of the process this linkset is empty but later on some linking information will also accumulate in it.

It should be stressed that links must be maintained in external linksets. They cannot be directly included in the model datasets since that would require hard-to-achieve changes to the conceptual schemas of the native models and the IFC and to the related implementations.

D. Change management utilizing the links

We now consider two kinds of changes. In the first case, the SE realizes that the concrete elements (*SWall1* and *SWall2*) need to be made thicker. This would eventually require a compatible change from the architect.

SE can first use the linkset *S-A* to evaluate if the change is possible and sees that both structural walls participate in the implementation of the architectural wall *AWall1*. It is therefore likely to be affected by the change. SE checks if there are specific reasons why *AWall1* cannot be changed, and if none exist, SE goes on to implement the change in S.

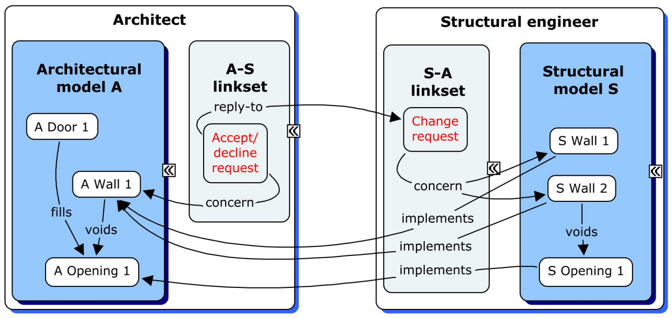

Fig. 5. Coordination in change management

There are now two ways to procede. A simple way is to just notify the architect about the change, assuming that she will try to comply with it. However, she may not be able to do that, or she may need to make a change in a manner requiring further changes from SE.

A much more flexible alternative is a "reification" of this interaction process resembling the way it is done in real projects. Once SE has completed the change, a proposal for a change, more commonly known as a *change request*, is created concerning *SWall1* and *SWall2*. Since the object representing the request is not part of either of the models, a natural place to record it is in the linkset *S-A* (Figure 5).

Based on linking of *SWall1* and *SWall2* it can be automatically determined who all need to be informed about the newly appeared change request. Here the only links are to the architectural model A and hence the architect is notified.

The architect needs to independently evaluate – based on her expertise – whether to change the architectural model to comply with the request. If she decides to do so, she can implement the change and produce a reply to the request. The reply is recorded in the linkset *A-S* with reference to changed architectural wall *AWall1* and to the change request object created by SE (Figure 5).

Change requests and replies record the intermediate states of the possibly emerging or disappearing links. Since the actions are carried out by humans, these intermediate states can be long lasting and cannot be regarded as ephemeral transitions. They cannot be managed in the execution state of an application but require a persistent external representation. After the change is completed, the change requests and replies can either be removed to save space, or can be retained to record the history and rationale of the design decisions.

In the second use case, the architect wants to change the door into a much heavier fire door. To support this use case the architect must be able to access the linkset *S-A* created by SE. This indicates the need for *public linksets*.

The use case also shows that although there are no direct links between the door and any entities in the structural model, there can be indirect effects. In particular, the support for the door in the precast concrete elements may require reinforcements that SE must evaluate. The effect of the change can mediated by a link in a component upwards in the containment hierarchy. If the architect can use the linking published from the structural model, it appears that the *AOpening1* has a link to *SOpening1* in the structural model. From the opening entity it is possible to access information about the neighbor elements, the two structural walls.

The architect thus evaluates whether the change is possible. If the status information of the related structural walls reveal that they have already been fabricated, it is worth considering alternative solutions to the fire resistance problem. But assume the architect proceeds to make the change to *ADoor1*. A change request is created and notification is sent to the SE due to the indirect relation of *ADoor1* to *SOpening1*.

E. Cross-model information access

When creating the MEP model (for instance, with MagiCAD[7]), the mechanical engineer uses the architectural model as a reference. The MEP model consists of one duct and two

[7]http://www.magicad.com/

pipes. The *MDuct1* is linked during the design to the space elements *ASpace1* and *ASpace2* surrounding the *AWall1* with the relations *serve* (Figure 6). The duct provides air flow to the spaces and consequently the properties of the spaces (such as dimension and the nature of use) provide information to the dimensioning of the duct. The linking thus provides an access path the information in the architectural model that is relevant to the mechanical engineering.

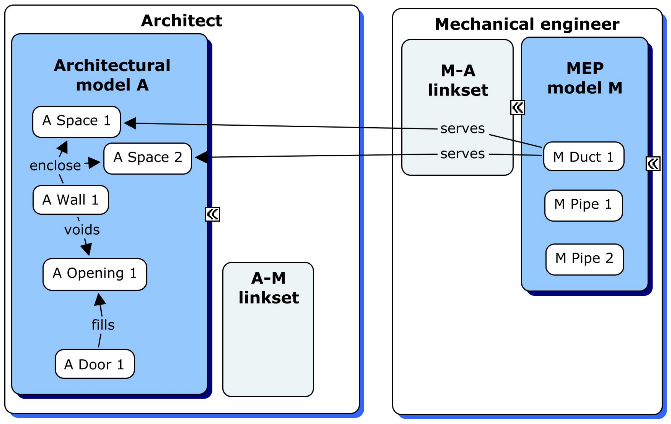

Fig. 6. A MEP model *M* linked to *A*

F. Overlaps and provisions for voids

Once the MEP model has been created, it needs to be checked for spatial overlaps with the structural model, as there is a *share space* link between the models. Three overlaps are identified: each entity in M has one with the one the structural walls in S. They are recorded in the linkset *M-S* (Figure 7).

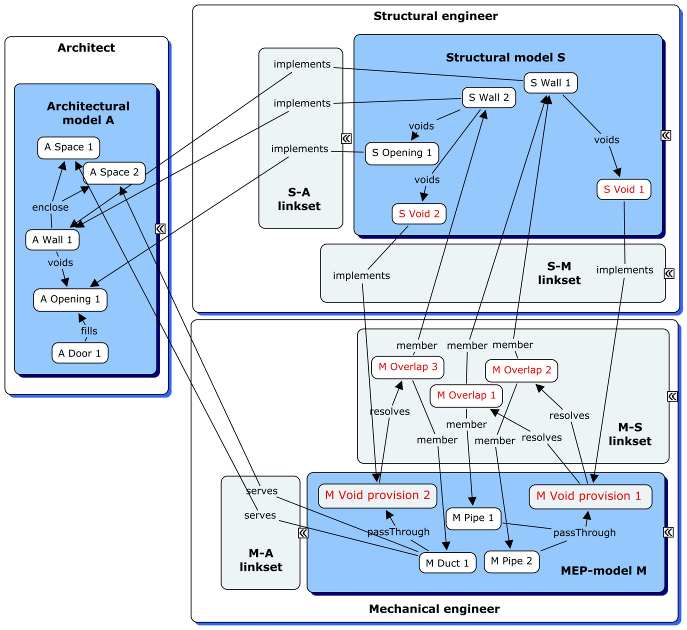

Fig. 7. Spatial overlaps and provisions for voids

The mechanical engineer creates two *provision for void* entities (specified in IFC), one of which handles the overlap of the duct and another that handles the overlaps of the two pipes. These act as requirements for the structural model. They are specified in a manner to envelope the overlaps in a way that once they are implemented in the structural model, the conflicts caused by overlaps are solved.

SE is notified about the provisions for voids since they are linked through an overlap entity with entities in the S. He or she designs the compatible voids, which can in some cases be done almost automatically.

G. Conclusions from the case study

Apparently, some aspects of cross-model linking could be actualized in a number of different ways, even with just local and private linkset files. However, to support the different actions in the case study, we have identified the following principles for the implementation of linking:

- *Two level linking*: Links exist at two different levels: model level and instance level. The identities of models and the model level links are known before any content for the models is created. In the case study two types of model level links were used: *based on* and *share space*.
- *Model level links determine instance level link generation activities*: For instance, the link 'm_2 *based on* m_1' means that m_1 will be used as a reference model when M_2 is created, and the links between the instances of m_1 and m_2 are established during the design work.
- *Links generated at both ends*: Instance level links can be independently generated at both ends of a model level link, especially in change situations. There must be either separate linksets owned by each parties or a common linkset modifiable by both.
- *Linksets contain all stuff between the models*: To avoid the need to extend the conceptual schemas of the native models and the IFC, all relations between two models should be managed in external linksets. The linksets are also a natural place to include various cross-model coordination objects (overlaps, change requests, replies).
- *Public linksets*: If linksets are published so that both parties of model level relation can utilize them, there are more opportunities for interoperation between the tasks, especially in change management.
- *Independent utilization at both ends*: In change management situation both parties should utilize the linksets to independently evaluate the impact of a change based on their discipline-specific expertise.

IV. Linked Building Data

The principles above indicate the need to support separate and public datasets and linksets that could be easily utilized by heterogeneous parties. While there might be a number of different implementation options available, the Linked Data technologies provide a natural fit with the principles, as outlined below.

Linked Data is based on the Semantic Web technologies for representation, publication, and browsing of structural data on the Web. It consists of the following technologies:

- URIs (Uniform Resource Identifiers) for identifiers.
- RDF (Resource Description Framework) for representation of data in a graph form.
- OWL (Web Ontology Language) for representing the conceptual schema.
- SPARQL, an SQL-type language for graph queries.

In ISO 10303 STEP (Standard for the Exchange of Product Data) the following technologies are used:

- GUIDs (Globally Unique Identifiers) for identifiers.
- STEP physical files (ISO 10303-21) as an exchange format. The file structure is compact; it relies heavily on internal definitions accessed via line numbers.
- EXPRESS, an object oriented data definition language, for representing the conceptual schema.

These map to each other according to I. An obvious advantage of the Linked Data approach is the existence of a query language that the IFC approach is lacking.

	IFC	**Linked Data**
Schema	EXPRESS	OWL
Data	STEP Physical File	RDF
Identifiers	GUID	URI
Queries		SPARQL

TABLE I
MAPPING BETWEEN REPRESENTATIONS

Since OWL is mostly more expressive and flexible language than EXPRESS, the conversion of IFC schema into an OWL ontology is straightforward. There are some minor complications dealing with procedural constraints in EXPRESS which cannot be directly translated into OWL, global vs. local names of relations, and mapping of datatypes. General guidelines for the conversion are given in [7].

The properties and relations of IFC objects are represented as RDF statements. The GUID-identified entities in IFC models would be identified with URIs using, for instance, the following URI scheme:

```
http://<domain>/<project>/<guid>
```

The conversion of IFC into RDF datasets uses an IFC ontology, produced by converting the IFC schema into OWL [7]. A publicly accessible converter for IFC data is available at the Ghent University [8], and an efficient IFC-to-RDF converter has been developed in the DRUM project [9]. For linksets RDF provides a natural representation due to its graph-based model consisting of links between URIs.

Based on the principles identified above, a natural way to apply Linked Data in building projects is for each party to convert its native model to IFC and from IFC to RDF and store it in an RDF store. The linksets should be directly generated in RDF and also stored in an RDF store (as a separate graph). All relevant parties should then be provided with a read-only access to the models and linksets. When instance-level interoperability needs arise, the SPARQL endpoints of relevant RDF stores can then be queried.

The existing dataset description schemes such as VoID (Vocabulary for Interlinked Datasets) [19] could be used for the representation of model metadata – including the model identities and access endpoints – but to capture the information about model-level links, extensions would be required.

V. Evaluation

How does the linking approach compare with the comparison and fusions approaches? Table II provides the summary of a qualitative evaluation between the three proposed approaches to determining cross-model correspondences between objects.

	Comparison	**Fusion**	**Linking**
1. Nature of relationships			
Possible types		One	Multiple
Cardinality		1-to-1	n-to-m
Reason for a link	Geometry	Defined	Defined
Time of linking	Comparison	Modeling	Modeling
2. Supported operations			
Cross-model access	No	Yes	Yes
Change propagation	No	Forced	Negotiated
External links	Separately	Separately	Integrated
Provides history	No	If in IFC	If needed
3. Decentralization			
Master model	Native	Central	Native
Needs roundtrips	No	Yes	No
Model coupling	Loose	Tight	Loose
4. Infrastructure requirements			
Link bookkeeping	No	No	Linksets
Requires metadata	No	No	Yes

TABLE II
QUALITATIVE EVALUATION OF THE APPROACHES

There are four categories of properties evaluated:

1) *Nature of relationships*. Linking is clearly more expressive approach than fusion, as it can support different relationship types and n-to-m links, while fusion is limited just to coreference relation and 1-to-1 mappings. It is difficult to say what kinds of relations the comparison approach supports since the relations have not been explicated and can reside just in the head of a user. It should be noted that if the only practical option is to use the coreference relation, it is likely to lead into abuse of the semantics of that relation in the ways described in [20]: in practice coreference would be used for a whole range of relations from loose similarity to exact equality.
2) *Supported operations*. The main weakness of the comparison approach is the poor support for the cross-model operations: it does not provide any solution to access the relevant information from other models, to manage changes across models, to support external linking, or maintain history. Linking approach is strong in all of these operations, and fusion in some of them.
3) *Decentralization*. The fusion approach is centralized and tightly coupled, leading to the need of round tripping

when changes happen. The other approaches fit better to the fragmented and volatile nature of building projects.

4) *Infrastructure requirements*. Linking approach requires that there are separate mechanisms for linkset management and the metadata from models is provided. There are not similar requirements in the other approaches.

VI. Discussion

In summary, comparison is a simple approach with poor support for instance-level interoperability. Fusion, on the other hand, is the wrong solution to the problem, although some of its properties may look promising. It creates actual connections between models but the connections are limited to coreference and hence are too strong and constrained. Moreover, it leads into a centralized architecture that complies poorly with the fragmented and dynamic nature of building projects.

Linking is the most flexible and expressive of the approaches. It allows the representation of a variety of semantic relations and at the desired level of detail. It poses some requirements to the infrastructure, since the linksets need to be maintained. However, the sizes of linksets are likely to be manageable: they are estimated to be roughly of the same order of magnitude as the models themselves.

When building information management is implemented with Linked Data technologies, the BIM world is opened into the Web. The use of Web-oriented representation framework makes it possible to connect entities in building information models to all kinds of other data on the Internet. The building information model would thus create an user understandable index to access and browse that information. In addition to creating a bridge to the world of Web, Linked Data and Semantic Web technologies provide opportunities to use reasoning tools with building information models [21].

VII. Conclusions and future work

Although it is generally recognized that building information consists of multiple partial models, the question of how the models are related has largely been side-stepped.

With a case study we identify the needs for linking between different partial models produced in a building project. When compared with the other approaches to specify cross-model relations, the linking approach turns out to be more flexible and expressive, although it requires support from design tools and the information management infrastructure. The advantages are, however, clear and in our view worth the costs. Linked Data technologies provide a natural starting point for the implementation for the principles identified.

The future research in the DRUM project will be focused on the change management in the Linked Building Data framework – change discovery, change analysis, and linkset maintenance – as well as a larger case study.

Acknowledgments

This research has been done in the DRUM/PRE project (Distributed Transactional Building Information Management) funded by RYM/Tekes (Finland). Special thanks to Leif Granholm, Jukka Suomi, Heikki Haikonen and Ragnar Wessman (Tekla), Heikki Kulusjärvi (Solibri), Pauli Keinonen (Progman), Ville Pietilä (M.A.D.), and Jyrki Oraskari and Nam Vu Hoang (Aalto) for their valuable insights and contribution in the development of the case study.

References

[1] C. Eastman, P. Teicholz, R. Sacks, and K. Liston, *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. Wiley, 2011.

[2] G. Van Nederveen and F. Tolman, "Modelling multiple views on buildings," *Automation in Construction*, vol. 1, no. 3, pp. 215–224, 1992.

[3] C. M. Eastman, *Building product models: computer environments supporting design and construction*. CRC PressI Llc, 1999.

[4] M. A. Rosenman and J. S. Gero, "Modelling multiple views of design objects in a collaborative cad environment," *Computer-aided design*, vol. 28, no. 3, pp. 193–205, 1996.

[5] *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*, IEEE, New York, NY, 1990.

[6] J. Beetz, J. Van Leeuwen, and B. De Vries, "Ifcowl: A case of transforming express schemas into ontologies," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 23, no. 01, pp. 89–101, 2009.

[7] J. Beetz, "Facilitating distributed collaboration in the aec/fm sector using semantic web technologies," Ph.D. dissertation, PhD Thesis, Eindhoven University of Technology. ISBN 978-90-6814-618-9, 2009.

[8] P. Pauwels, R. De Meyer, and J. Van Campenhout, "Interoperability for the design and construction industry through semantic web technology," *Semantic Multimedia*, pp. 143–158, 2011.

[9] S. Törmä, J. Oraskari, and N. Vu Hoang, "Distributed transactional building information management," in *Proceedings of the First Workshop in Linked Data in Architecture and Construction*, P. Pauwels, Ed. Ghent University, March 2012.

[10] T. Berners-Lee, "Linked data-design issues," *W3C*, vol. 2009, no. 09/20, 2006.

[11] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 5, no. 3, pp. 1–22, 2009.

[12] T. Heath and C. Bizer, "Linked data: Evolving the web into a global data space," *Synthesis Lectures on the Semantic Web: Theory and Technology*, vol. 1, no. 1, 2011.

[13] A. Cleveland Jr, "Interoperability platform," *Bentley White Paper*, vol. 29, 2008.

[14] U. Isikdag, G. Aouad, J. Underwood, and S. Wu, "Building information models: a review on storage and exchange mechanisms," *Bringing ITC knowledge to work, 24th W*, vol. 78, 2007.

[15] U. Isikdag and J. Underwood, "Two design patterns for facilitating building information model-based synchronous collaboration," *Automation in Construction*, vol. 19, no. 5, pp. 544–553, 2010.

[16] L. van Berlo, J. Beetz, P. Bos, H. Hendriks, and R. van Tongeren, "Collaborative engineering with IFC: new insights and technology," in *eWork and eBusiness in Architecture, Engineering and Construction*, G. Gudnason and R. Scherer, Eds. CRC Press, 2012, iSBN: 978-0415621281.

[17] V. Singh, N. Gu, and X. Wang, "A theoretical framework of a bim-based multi-disciplinary collaboration platform," *Automation in Construction*, vol. 20, no. 2, pp. 134–144, 2011.

[18] J. Aro, "Automated exchange of distributed status information of building elements," Master's thesis, School of Science, Aalto University, August 2013.

[19] K. Alexander and M. Hausenblas, "Describing linked datasets-on the design and usage of void, the vocabulary of interlinked datasets," in *In Linked Data on the Web Workshop (LDOW 09), in conjunction with WWW 09*, 2009.

[20] H. Halpin, P. J. Hayes, J. P. McCusker, D. L. McGuinness, and H. S. Thompson, "When owl:sameas isn't the same: An analysis of identity in linked data," in *The Semantic Web–ISWC 2010*. Springer, 2010, pp. 305–320.

[21] P. Pauwels, D. Van Deursen, R. Verstraeten, J. De Roo, R. De Meyer, R. Van de Walle, and J. Van Campenhout, "A semantic rule checking environment for building performance checking," *Automation in Construction*, vol. 20, no. 5, pp. 506–518, 2011.