Multi-model Environment: Links between Objects in Different Building Models

A. Kiviniemi
VTT Building and Transport, Technical Research Centre of Finland, Espoo, Finland

M. Fischer
CIFE, Civil and Environmental Engineering, Stanford University, Palo Alto, USA

V. Bazjanac
Lawrence Berkeley National Laboratory, University of California, Berkeley, USA

ABSTRACT: The current IFC specifications include relations between objects and enable representation of complex structures in a building product model. However, several research projects have addressed the problem of one integrated model by pointing out the different content and structure of different design domains. The existing software products cannot support all features of the IFC specifications, and because of the structure of AEC industry there are no potential customers for applications which would cover all different information needs. We believe that there will be several instantiated models representing a building project, and these models share some parts of the information which must be linked between the models. However, IFC specifications do not enable links between objects in separate instantiated models. This paper will (1) discuss the reasons for the separation of instantiated models, (2) present the necessary extensions of the IFC specifications, (3) include examples of the links between the requirements model and architectural design model, and (4) discuss some possibilities how to implement this link in a model server environment.

1 INTRODUCTION

The problem of one integrated model in data exchange of a building project has been addressed in some earlier research projects. The "PM4D Final Report" (Kam & Fischer 2002) addressed it by pointing out the different content and structure of different design domains, although the report did not propose a solution for the problem. Also, John Haymaker recognized the need for several models in his Ph.D. research (Haymaker et al 2003). However, to our knowledge, a formal division of a project's data set into several models and solution for linkage between the objects in these sub-models has not been published prior the doctoral dissertation of the first author of this paper, which forms the basis of this paper (Kiviniemi 2005). The other two authors were advisers in the dissertation research.

2 SEPARATION OF THE INSTANTIATED MODELS

Our solution to address these problems was a concept that divides the instantiated model of a project, i.e., project's data set, into four separate main models: 1) Requirements Model, 2) Design Model(s), 3) Production Model(s), and 4) Maintenance Model. These linked sub-models form the integrated project information model, Figure 1.

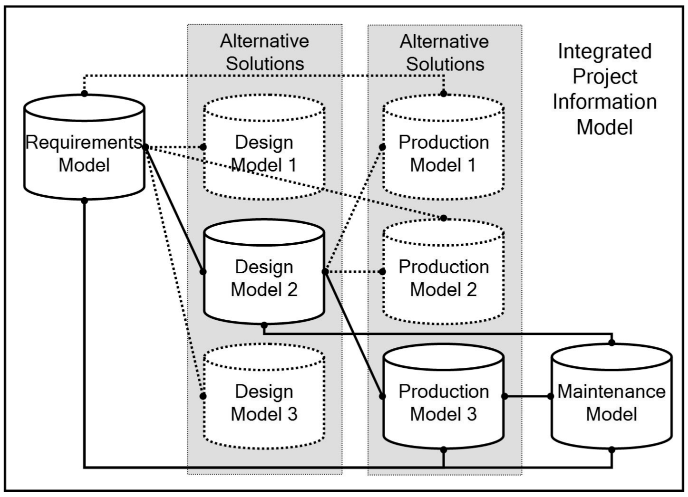

Figure 1: Integrated Project Information Model consisting of the four main models

This division of the project's data set does not mean that the information structure, model specification, would have to be separate models; it can be one specification. Our requirements model specification uses definitions from the current IFC specifications. Thus, the requirements model specification can be integrated with the IFC specifications. However, the instantiated model, i.e., project's data set, should be divided into several models. In fact, the information content in the different design and contractor domains is so different that there is a need for several design and production models, but this topic was not in the scope of our research.

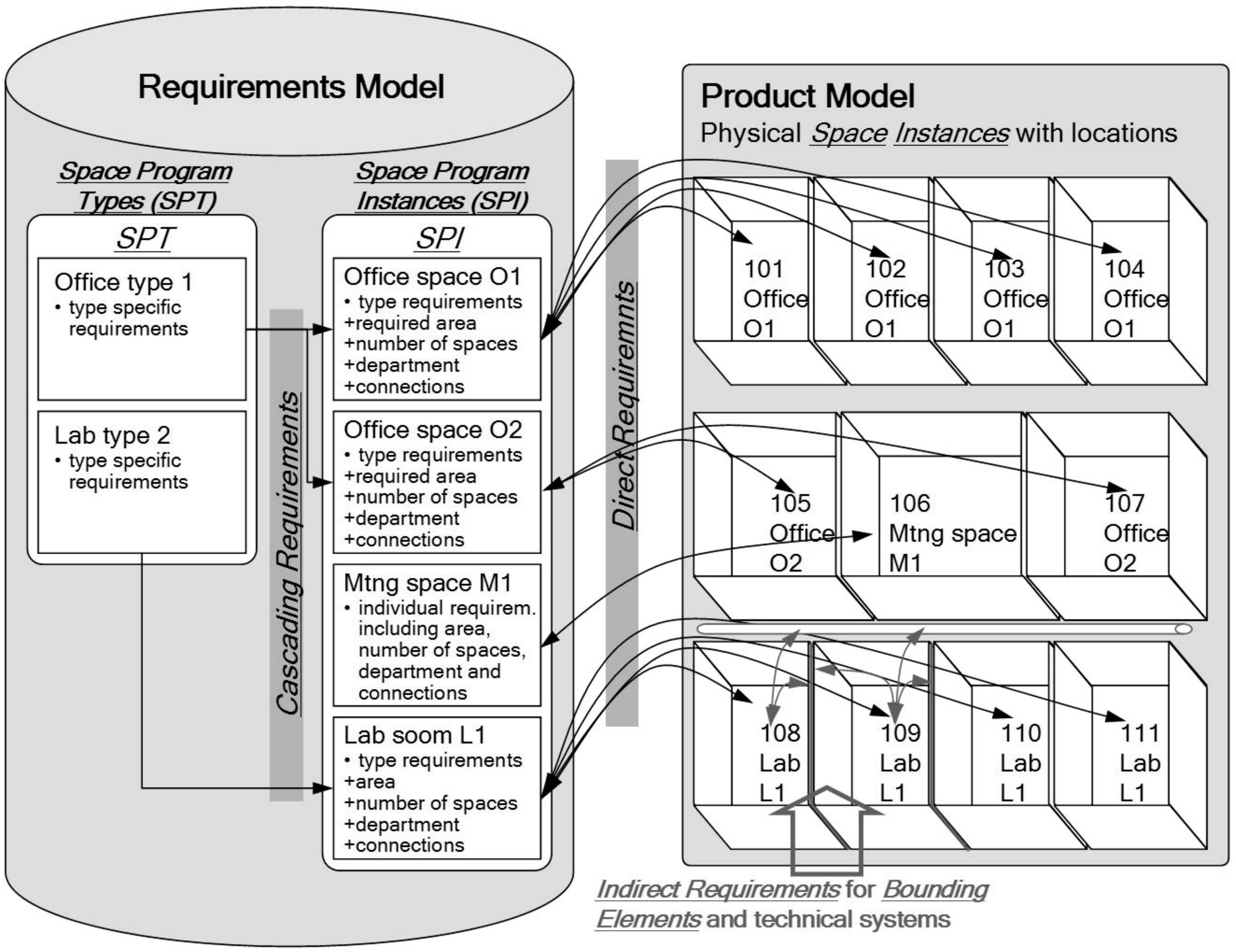

Figure 2: Concept used in the rapid prototyping to link requirements to a design model: Relations between Space Program Types (SPT), Space Program Instance (SPI), physical space instances and indirect requirements

There are several reasons for the separation of instantiated models. Our research focused on the requirements model and its connection to the architectural design model. Thus, most examples are from that area. However, most of the problems can be identified between different design models as well:

1. The data content and structure of the requirement and design models are different. For example, one requirements entity for spaces, a Space Program Instance (SPI, figure 2) can relate to a number of separate Instances with identical requirements in the design, production, and maintenance models. Similarly, for example, one slab or wall in the architectural design model can be several objects in the production model, or separate objects in the design and production models can be one object in the maintenance model.

2. Although the IFC specifications allow shared property sets, to our knowledge all IFC implementations are using instance-specific attribute sets, because the internal structures of design software do not support shared attributes. In practice it means that if the requirements are stored in the design model, the same requirements are multiplied in all instances, which can cause serious problems in the requirements management when the requirements evolve and must be updated (Kiviniemi 2005b).

3. Typically, the project team produces several alternative design proposals which all should meet the defined requirements. Thus, having one requirements model linked to the alternative design models is a logical structure instead of multiplying the same requirements to different design alternatives, which would easily lead to requirements management problems. Similarly, there can be several alternative production models and finally a separate maintenance model. All these models should be connected into one Integrated Project Information Model so that it is possible to access the content of the different models and compare the alternatives at any stage of the process (Figure 1).

4. The flexibility of the requirements model is greater if the models are separated and connected with a “thin” link, e.g., there is only one identifier in both models connecting the requirements and design objects (Kiviniemi 2005c). Adding or removing requirements in the requirements model specification does not change the design applications. In our prototype implementation, the only element needed for the link of space requirements was an ID in the

space object, which is supported by almost any design software. For indirect requirements, the functional demand is to recognize the connection between bounding elements and spaces, which is supported by some commercially available building-product-model-based software.

5. Another reason for the separation is to make the distinction between requirements and properties clear; for example sound insulation is a requirement for a space in the requirements model and a property of the bounding elements in the design model.

6. Separation of requirements and design models allows access control of requirements; it is possible to show the information to designers but not allow them to modify requirements if such control is wanted, for example, for project management or quality system purposes.

7. Requirements are not attributes of design objects but independent entities, i.e., if the design changes so that a design object, such as a space, is removed, its requirements should remain unless the need for the space has changed too. Otherwise reliable comparison of the design solutions against the requirements is impossible.

8. A further important observation is that a Space Program Instance (SPI) in the requirements model has no geometrical locations, i.e., the requirements for bounding elements can relate to one space only. In the design, production, and maintenance models the bounding elements are always between two spaces; either between two rooms or as a part of the building envelope. This means that the requirements for the bounding elements must be aggregated from the requirements of the related spaces. They cannot be defined directly for the building elements in the same manner as the space requirements relate to the spaces (Figure 2).

3 NECESSARY IFC EXTENSIONS

3.1 *Link between objects in different models*

The information needed for the link between objects in different models is simple: The location and type of the model, and the IDs of the linked objects.

•The location of the model is the address where the linked model is stored. This can be a URL, address in the model server database, or some other address depending on the technical solution.

The type of the model is necessary because one object can be linked to several models, and the use of the link requires information of the purpose of the referenced model, for example, is it a requirements, design, or maintenance model. In our solution the list of models consists of six enumerations (requirements, design, production, maintenance, userdefined and notdefined), but the principle does not change if the list will be expanded with different domain models, such as structural, HVAC, electrical models.

The links between requirements and design models are from the design objects to the requirements objects. However, the links between the different design models must be two-directional, for example, a column instance in the architectural and structural model must be linked in both directions (Figure 3).

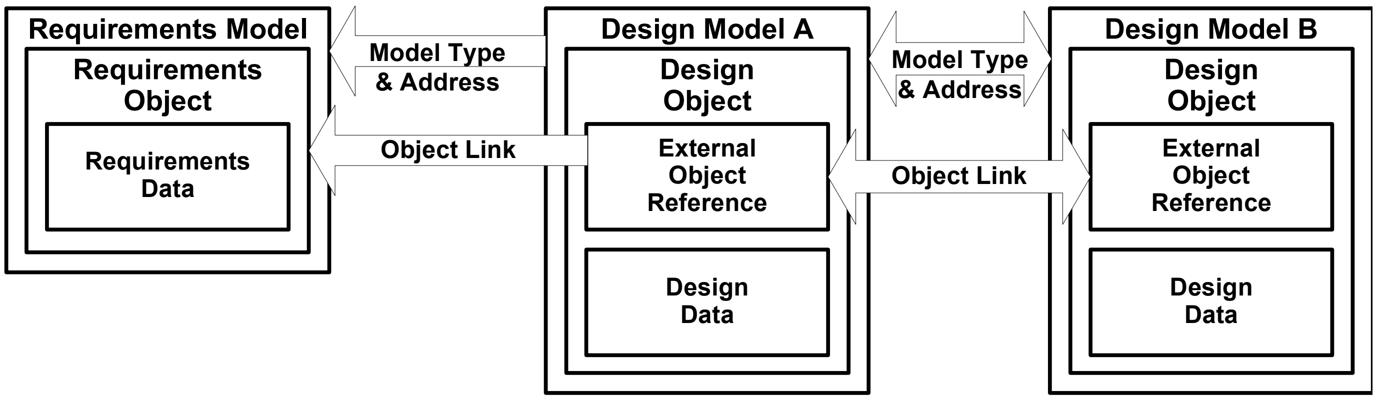

Figure 3: Links between objects in different models

3.2 *Existing IfcExternalReference*

The current IFC specifications include an element for external linkage, IfcExternalReference, but it does not include links between objects in different instantiated models; it only covers the links to library, classification and document, references. Thus, it was necessary to expand the current definition of the IfcExternalReference, which is now described by IAI in the following way (IFC 2005):

"An IfcExternalReference is the identification of information that is not explicitly represented in the current model or in the project database (as an implementation of the current model). Such information may be contained in classifications, documents or libraries.

Only the Location (e.g. as an URL) is given to describe the place where the information can be found. Also an optional ItemReference as a key to allow more specific references (as to sections or tables) is provided. The ItemReference defines a system interpretable method to identify the relevant part of information at the data source (given by Location).

In addition a human interpretable Name can be assigned to identify the information subject (e.g. classification code).

IfcExternalReference is an abstract supertype of all external reference classes."

<u>*EXPRESS specification (IFC 2005):*</u>

```
ENTITY IfcExternalReference
   ABSTRACT SUPERTYPE OF  (ONE OF
   (IfcLibraryReference, IfcClassificationReference,
   IfcDocumentReference));
      Location : OPTIONAL IfcLabel;
      ItemReference : OPTIONAL IfcIdentifier;
      Name : OPTIONAL IfcLabel;
   WHERE
      WR1 : EXISTS(ItemReference) OR
      EXISTS(Location) OR EXISTS(Name);
END_ENTITY;
```

3.3 *Modified IfcExternalReference*

To address the limitations we modified the existing IfcExternalReference by adding one new subtype, NewExternalObjectReference:

<u>*EXPRESS specification:*</u>

```
ENTITY IfcExternalReference
   ABSTRACT SUPERTYPE OF (ONE OF
   (IfcLibraryReference, IfcClassificationReference,
   IfcDocumentReference,
   NewExternalObjectReference));
      Location   :  OPTIONAL IfcLabel;
      ItemReference   :  OPTIONAL IfcIdentifier;
      Name   :  OPTIONAL IfcLabel;
   WHERE
      WR1 :  EXISTS(ItemReference) OR
      EXISTS(Location) OR EXISTS(Name);
END_ENTITY;
```

This modification required some small modifications in the existing IfcRelAssociates entity, and addition of four new entities in the specification: NewExternalObjectReference, NewRelAssociatesExternalObject, NewModelInformation, and NewModelTypeEnum shown in Figure 4. The detailed EXPRESS specification is documented in the published dissertation (Kiviniemi 2005c).

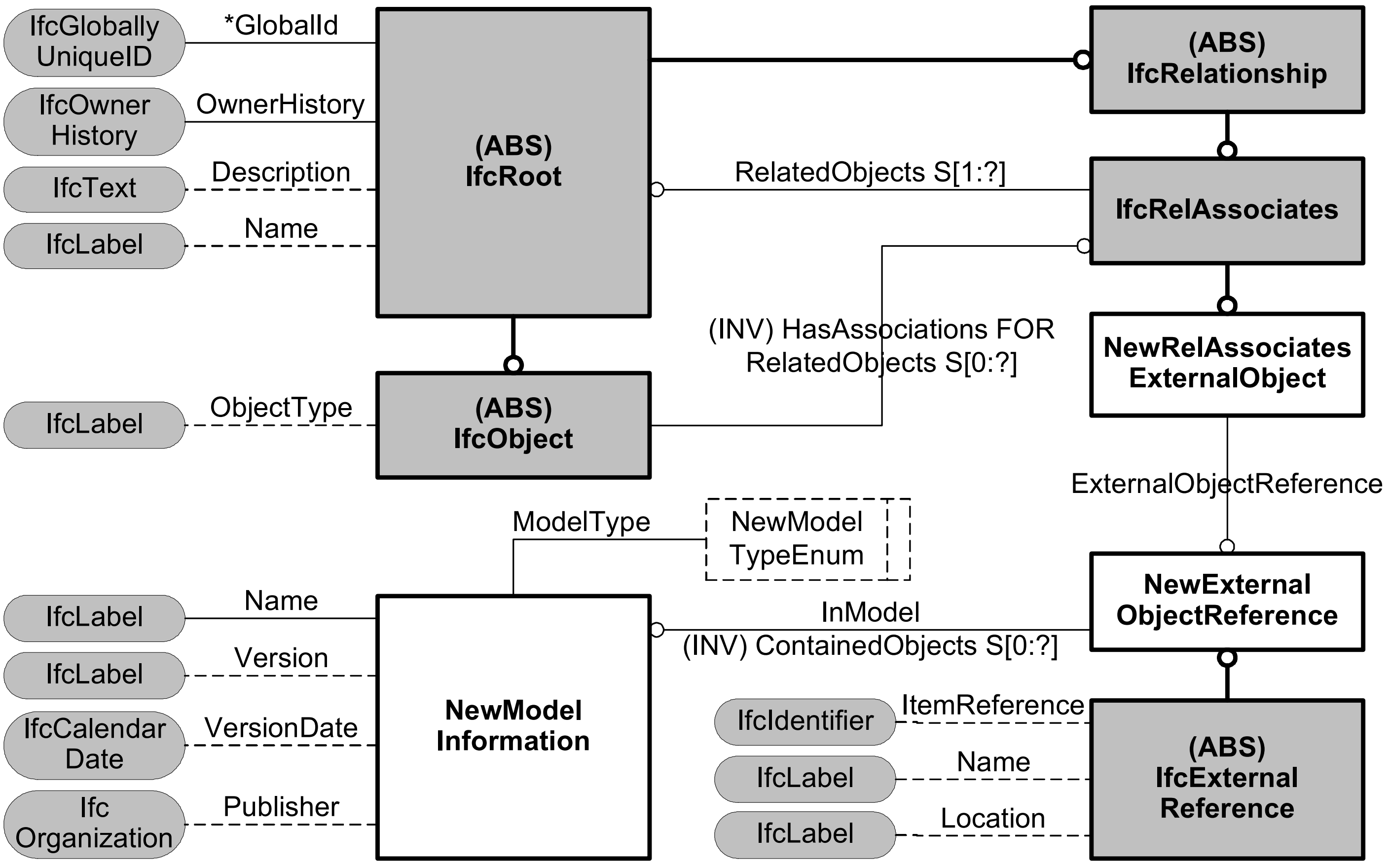

Figure4: Location and structure of the object link between models in the IFC specification. Grey background indicates the existing entities

4 EXAMPLES OF LINKS BETWEEN REQUIREMENTS AND DESIGN MODEL

The requirements model linked to the design model was developed to enable the links between client requirements and design model, and its main purpose was to improve requirements management in the AEC process. The extension of IFC specifications documented in the previous chapter was necessary for this requirements model. The current IFC specifications include some requirements, but the IfcSpaceProgram is the only independent requirement object, and it includes only very few requirements, which are only related to spaces. All other requirements are property sets attached to the design objects, which causes problems described in chapter 2. In addition, the requirements in the current IFC specification are inconsistent and include many mistakes, such as duplication of the same requirements in several property sets in the specification (Kiviniemi 2005d). Thus it was necessary to develop a systematic way to define requirements and their relations to the design model.

The direct requirements in the system are requirements defined and maintained by some member of the project team. Direct requirements can cause indirect requirements to the building elements on lower level in the hierarchy or to the systems, including the technical systems, such as structural, electrical, HVAC and security systems, Figure 5.

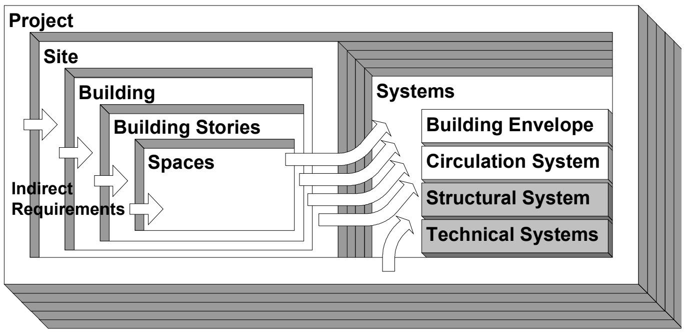

Figure 5: Levels of detail in the requirements model

A practical example of this is a space with requirements for area, sound insulation and temperature. Area is the only requirement directly affecting the physical space object. The sound insulation requirement causes indirect requirements for the bounding elements (walls, doors, windows), and the temperature requirement causes indirect requirements for the HVAC system and bounding elements.

The content of the requirements model developed in our research was limited to the architectural design of office and laboratory buildings, but most of the principles can be applied also to other building types and design domains. The basic idea to link the requirements and design models is based on the hierarchical composition of the IFC specification: the requirements are linked directly to five levels of detail in the model (project, site, building, story and spaces) or to the systems (building envelope and circulation system), Figure 6.

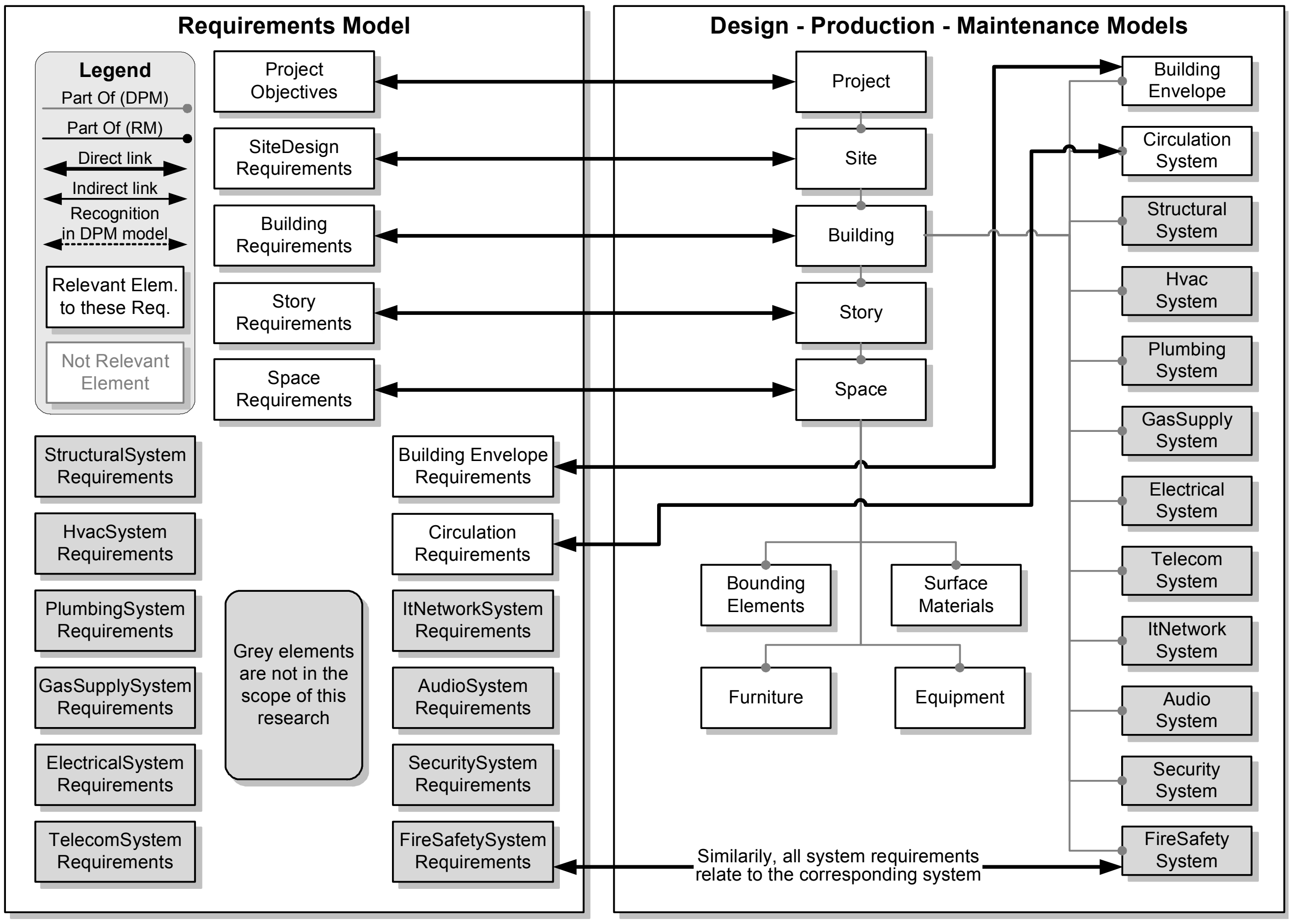

Figure 6: Basic relations between requirements and design-production-maintenance models

In total our requirements model contains 300 requirements in 14 main and 35 sub-categories. These requirements are organized in the specification into 7 main-level and 30 sub-level requirements objects which have direct links to 5 levels of detail and 2 systems in the building product model plus indirect links to 4 levels of detail and 12 systems.

The lighting requirements for a space are an example to illustrate the detailed content of the requirements model. They are linked directly to the spaces in the design model. In addition they have indirect link to the electrical system and they need the recognition of some space-related elements in the design-production-maintenance models, e.g. the link to windows and colors of furniture, equipment and surface materials is a needed functionality, Figure 7.

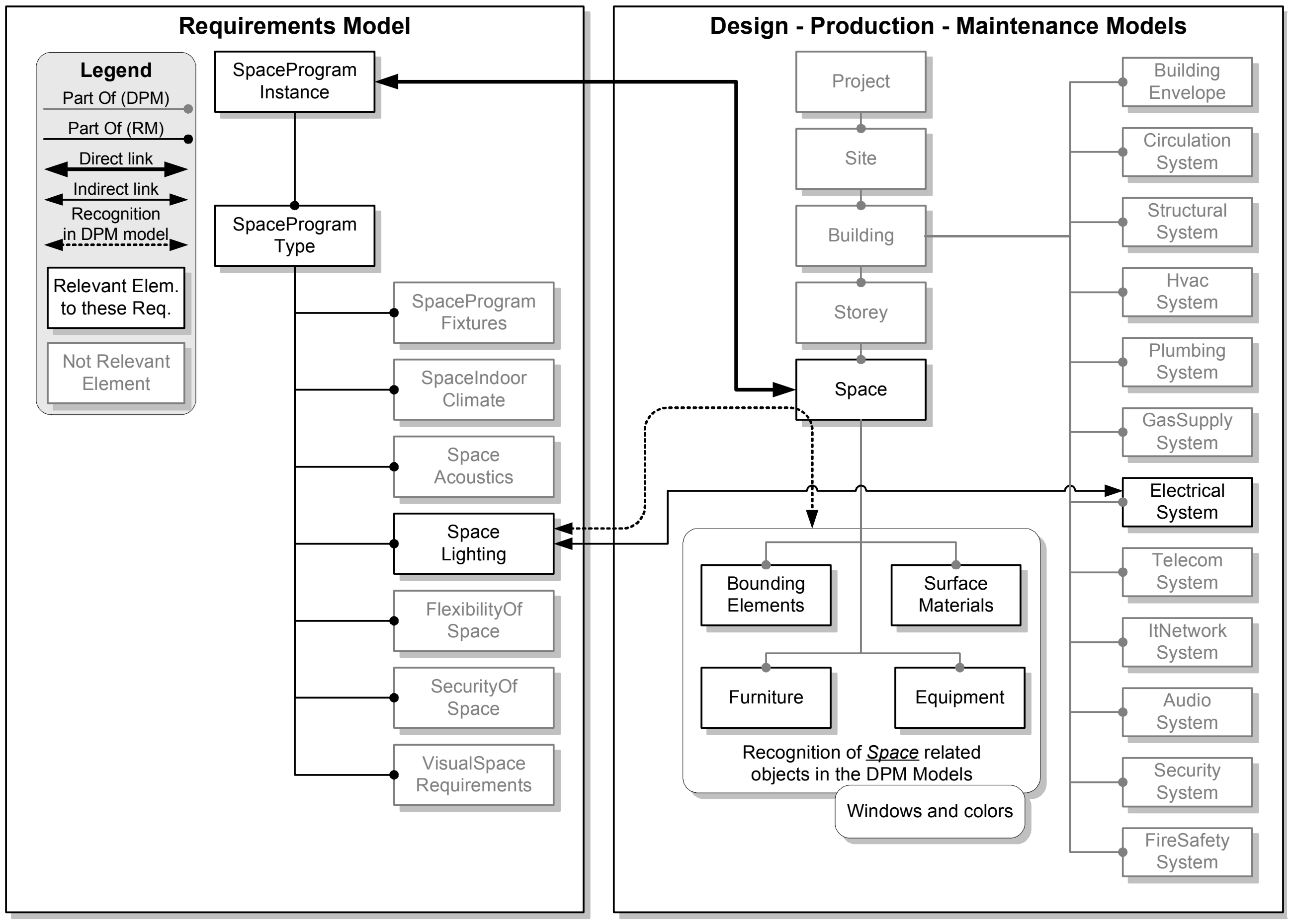

Figure 7: Space lighting requirements

5 THE USE OF MODEL SERVERS IN MULTI-MODEL ENVIRONMENT

The problem of file-based data exchange is the different internal data structures of the existing tools. A data set exported and imported between several software applications will almost certainly loose some data, because there is no place for all the data elements in all applications. It is hardly possible to believe that the software developers would solve this problem, because of the complexity of the data. In addition, the development of other software products would inevitably lead the constant need to change the data structures of all software products used in the AEC processes if the data should be maintained fully in the file-based exchange.

The only feasible solution seems to be an independent data platform; IFC compliant model server, which can store all information shared between the applications. In this case the interface between the application and model server handles only the data which is meaningful for the application and all other data in the model server will stay untouched.

However, this means quite complex data structures on the server, and control of the right to access and change information on the server. The AEC processes are iterative, and the design solutions affect each other. The changes must be approved by the owner of the data, which means that a proposed change by one player must not affect directly to the data of other players. A practical example of this is that if the HVAC engineer wants to move a duct which is going through a beam, the hole can not move before the structural engineer has approved its new position. This can be done only if the domain specific data sets are maintained independently; when the need of the hole for a duct in HVAC database is linked to the actual hole in the structural database, and the locations and sizes of these two objects can be checked and possible conflicts solved.

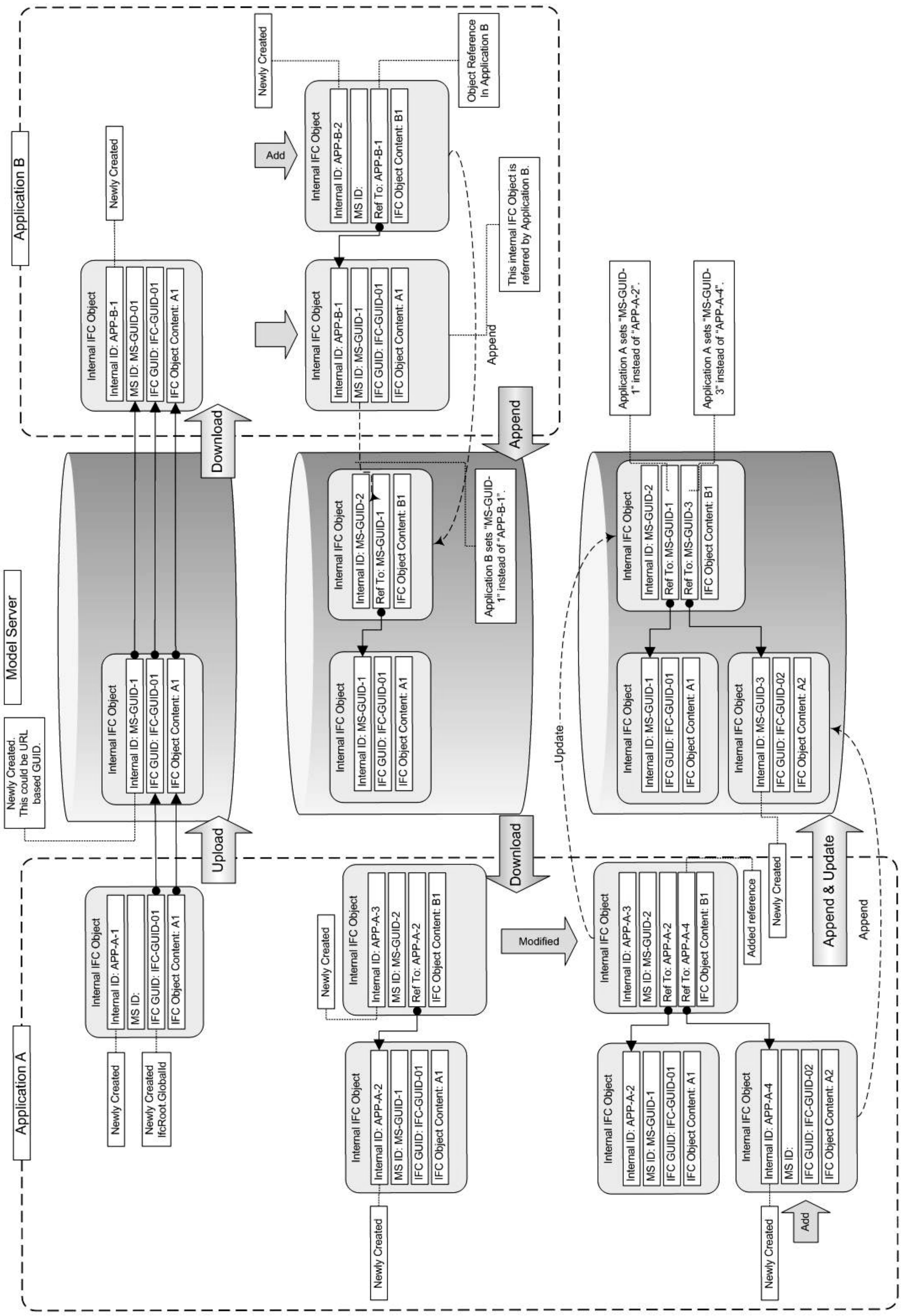

Figure 8: Model server object reference linkage (Adachi 2005)

Model server technology provides an excellent platform for this purpose. The IFC specifications have created a wide framework and defined objects, which can be stored and linked in the model server database as domain specific sub-models linked to each other, Figure 8.

6 CONCLUSIONS

Because of the numerous independent players in the AEC processes and tools supporting the individual tasks in these processes, the building information models must be able to handle different information content and compositions. Thus, an integrated instantiated project model is extremely difficult to implement into practical software tools and maintain in the design and construction process.

If the information sharing is based on file exchange, the integrity of the data is practically impossible to maintain because of the different data structures in different software products. The model server environment improves the situation significantly, because it can include all necessary data for different tools. However, because of access and control purposes it is necessary to divide the data into different subsets. This division can support also different compositions of data sets, which is often needed for different AEC tasks; what one domain sees as one object might be several objects for another domain.

Integrated product models can be utilized in the industry efficiently only if the different domain specific views are supported, but without links between the different domain-specific sub-models the situation will be almost the same as in the drafting-based processes; the data will be fragmented and changes in one model will be easily ignored in the other models, which will cause conflicts between different documents. Thus, links between the sub-models are necessary for information management.

Our research has pointed out a possible technical solution for the links between objects in different models, and also created a detailed documentation of the relation between requirements and design models in the architectural design. The next phase in our research will be the implementation of a set of software tools for integrated multi-model environment on the level which can be tested and validated in practical projects.

ACKNOWLEDGMENTS

The research has been funded by Tekes (Technology Agency of Finland), VTT (Technical Research Center of Finland), Confederation of Finnish Construction Industries RT, NCC Oy, Pöyry Oy, Skanska Oy, and YIT Rakennus Oy.

Jiri Hietanen, Patrick Houbaux, and Kari Karstila, who gave us invaluable insights in the development of the link between the requirements and design models.

REFERENCES

Adachi, Yoshinobu: Solution for the model server object reference linkage, sent as a part of the GUID problem discussions in the BLIS group, January 2005. Published in Kiviniemi, Arto 2005, page 121

Haymaker, John; Suter, Ben; Kunz, John; & Fischer, Martin: PERSPECTORS: Automating the Construction and Coordination of Multidisciplinary 3D Design Representations. CIFE Technical Report TR145, Stanford University 2003: http://cife.stanford.edu/online.publications/TR145.pdf

IFC 2x2 Addendum 1 web site 2005: http://www.iai-international.org/Model/R2x2_add1/

Kam, Calvin & Fischer, Martin: PM4D Final Report. CIFE Technical Report TR143, Stanford University 2002: http://www.stanford.edu/group/4D/download/c1.html

Kiviniemi, Arto: Requirements Management Interface to Building Product Models. Ph.D. Dissertation and CIFE Technical Report TR161, Stanford University 2005: http://cife.stanford.edu/online.publications/TR161.pdf

Kiviniemi, Arto 2005b: “Property Sets: Requirements in the Building Element Attributes” pages 97-98

Kiviniemi, Arto 2005c: “Links between the Requirements and DPM Models”, pages 132-136

Kiviniemi, Arto 2005d: “Space-Related Requirements in the IFC Specifications”, pages 104-115.