CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

PREMISS - Requirements Management Interface to Building Product Models: Problem Definition and Research Issues

By

Arto Kiviniemi, Martin Fischer, Vladimir Bazjanac & Boyd Paulson

CIFE Working Paper #092
OCTOBER 2004

STANFORD UNIVERSITY

If you would like to contact the authors, please write to:

c/o CIFE, Civil and Environmental Engineering Dept.,
Stanford University
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

Executive Summary

The purpose of this report is to define the important problems for a requirements management interface to building product models and outline the research needed to address these problems. The goal is to create a Requirements Model Specification, to enable active connections between the requirements for a building project and the Building Product Model based design solution. The focus of the Requirements Model Specification will be in the clients' (owner's and end-users') requirements, but it will also include connections to some external requirements, like for example requirements set by the community or regulations, like building code. However, the detailed requirements analysis of the external requirements for building projects is not in the scope of this report.

In current AEC practice client requirements are typically recorded in a building program, which, depending on the building type, covers various aspects from the overall goals, activities and spatial needs to very detailed material and condition requirements. This documentation is used as the starting point of the design process, but as the design progresses, it is usually left aside and design changes are made incrementally based on the previous design solution. As a consequence of several small changes and without any conscious decisions to change the scope, this can lead to a solution that may no longer meet the original requirements. In addition, design is by nature an iterative process and the proposed solutions often also cause evolution in the client requirements. However, the requirements documentation is usually not updated accordingly. In the worst case the changes are recorded just in the memory of the participants, and in the best case in meeting or personal notes. Finding the latest updates and evolution of the requirements from the documentation is very difficult, if not impossible.

This process can lead to an end result, which is significantly different from the documented client requirements. Some important client requirements may not be satisfied, and even if the design process was based on agreed-upon changes in the scope and requirements, differences in the requirements documents and in the completed building can lead to well-justified doubts about the quality of the design and construction process.

Our observation is that even a simple active link between the client requirements and design tools can increase the usage of requirements documentation throughout the design and construction process and facilitate necessary updates of the client requirements. The key limitation is the lack of a theory to link the requirements to the design systems.

The research needed to formalize a Requirements Model Specification linked to Building Product Model based design tools to improve the quality of the process and the documentation includes the following points of departure:

- Design as an information process
- Existing client requirements documentation
- Lawrence Berkeley National Laboratory's Design Intent Tool for technical systems
- Existing IFC Specification and its implementation
- Building Lifecycle Interoperable Software (BLIS) views to the IFC Specification.

The research is part of CIFE's Virtual Design and Construction (VDC) framework; objects in the requirements model represent Desired Product Form in the Product-Organization-Process (POP) ontology.

The proposed plan consists of four phases; analysis of client requirements, development of a Requirements Model Specification and its links to the IFC Specification, extension of the BLIS view for IFC implementation, and validation of the Requirements Model Specification.

The anticipated scientific contributions of this research are: specification for the link between client requirements and product model objects, specification for the aggregation of indirect requirements from the direct requirements to the building elements, extended view "Room Program -> Architectural Design" for the IFC product model implementation. In addition, we believe that our research will also create a basis for several future research topics in this area.

The anticipated main contributions on a practical level are the proposed framework for requirements management during the design process and interaction between the design solutions and client requirements.

Table of Contents

Table of Figures

History of the “PREMISS” Name

The research project is named “PREMISS” based on a loose abbreviation of its original name (Product Model Extension for Requirements Management Interfaces) and this name is used in this document to identify some parts to the project, like “PREMISS Database” or “PREMISS Model”. PREMISS was selected because it is an old British form of the word “premise” and thus, fits well to describe the main topic of our research – client requirements for a building project.

Introduction

The purpose of this report is to define the starting point and plan for research to create a *Requirements Model Specification* [i] to enable active connections between *Requirements* for a building project and its *Building Product Model* based design solution. The main focus of the *Requirements Model Specification* will be on the *Clients' Requirements* but the specification will also cover some aspects of the *Requirements* set by the community and regulatory agencies, like the building code.

A building program specifying the project's goals and *Requirements* for all the spaces is the typical *Client Requirements* documentation in building projects, though there are also several other methods to capture *Client Requirements*. Regardless of the capturing method, the *Requirements*, depending on the project type, consist of more or less detailed information about the required properties; net area, activities, connections to other spaces, security, appropriate or desired materials, conditions like daylight, lighting, temperature, sound level, etc. Many *Requirements* also "cascade"; e.g., create additional *Requirements* to building elements bounding the space and systems serving the space. Moreover, an important part of the design process is that some *Requirements* can be in conflict; the *Project Team* must often prioritize and make trade-offs between different *Requirements*, which creates the need to update the *Requirements*, and thus, manage and document the changes to the *Requirements* and the design solution.

In practice several factors make it virtually impossible that all the participants know and remember all the relevant *Requirements* and especially their relationships to each other and to the design solutions. The main reasons for this argument are:

- The amount and complexity of project information,
- The duration of projects,
- The need for designers to work simultaneously on many projects,
- Changing stakeholders in different project phases, and
- Shifting design focus, e.g., moving from overall problem solving to detailed technical solutions.

1.1 AEC Process

The Stanford project guidelines "Heartbeat" (Figure 1) represent a typical description of the design and construction process. Though it is basically correct, it creates an image of a sequential process, where the *Requirements* are set in the programming phase and design is just solving the needs documented in the programming phase.

[i] Definitions of all terms formatted in *Underline Italic* are in Appendix 2

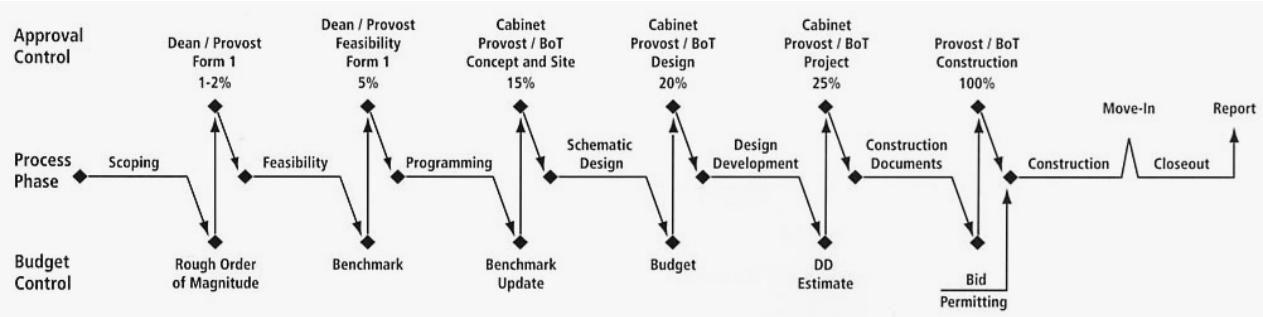

Figure 1: "Heartbeat", the Project Delivery Process at Stanford [Stanford 2001 [1]]

However, this is not the case; design is by nature an iterative process, and the provided design solutions affect also the *Client* expectations, thus causing evolution of the *Requirements*. Though the intensity of *Requirements* definition and design activities, and the character of the changes, are different in different stages of the process, we argue that the process should be described as partly parallel activities, including *Requirements Management* through the whole process and several stages where local authorities check if the design and construction meet the regulations [Figure 2].

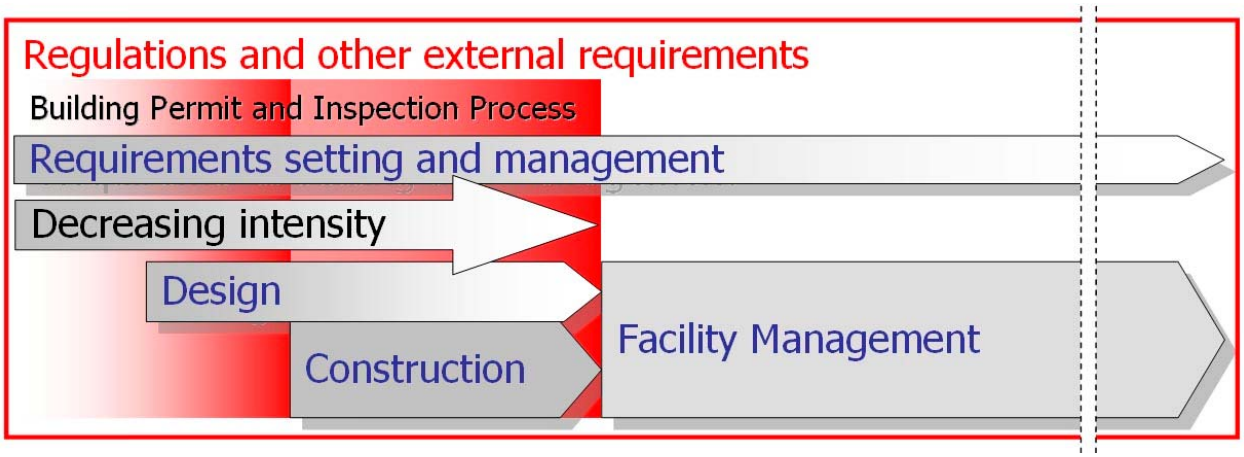

Figure 2: Parallel process view

The iterative nature of the design process and the usually large number of changes during the process increase the complexity of the problem. The *Project Team* has to make rapid decisions on how to solve a specific issue, and it is often difficult to notice all interdependencies. Thus, a solution, which is meeting one *Requirement*, can have a significant negative effect on another crucial *Requirement*. One trivial example of this is accessibility vs. access control; optimizing the accessibility to the various spaces in a building is in contradiction with access control, which demands as few access points and alternative routes as possible. Our observation is that the current process could improve significantly if:

- The *Project Team* could manage and update the evolving requirements, and
- The designers could easily find the requirements related to their on-going task

A logical solution is a data interface, a link between the *Requirements* and the design solutions, which connects the *Requirements* better to the design process and helps designers to understand the interaction between the *Requirements* and design solutions [Figure 3].

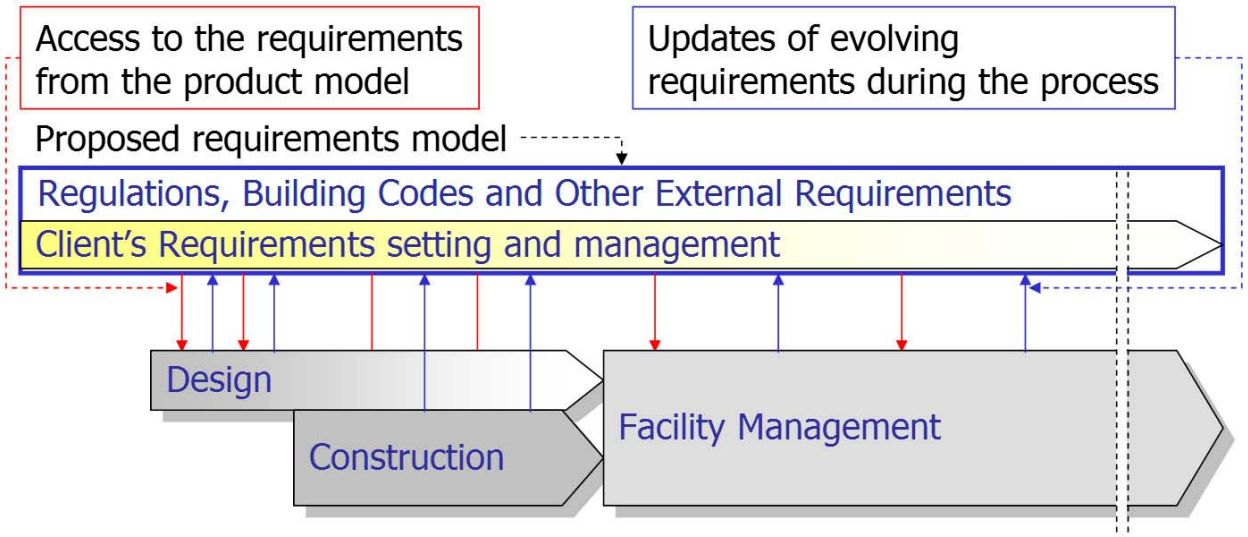

Figure 3: Proposed approach supporting interaction between *Requirements* and design solutions using linkage between *Requirements Model* and existing *Building Product Model* based design, construction and facility management software.

1.2 Shifting Focus

After conceptual design the *Requirements Documentation* is usually not used actively in the current process [Section 3.2.2, A2], and often the evolving *Requirements* are not even communicated to the whole *Project Team* [*Kagioglou et al 1998* [2]]. Thus, the changes are compared to, and decisions made based on, the previous design solutions. The current design tools do not support recording of *Client Requirements* or designers' intent in the documents. Thus, the people deciding on the changes do not always even know the original intent, and the solution can "shift away" from the original goal [Figure 4] without actual decisions to change the goal or understanding the contradiction between the proposed design and project goals.

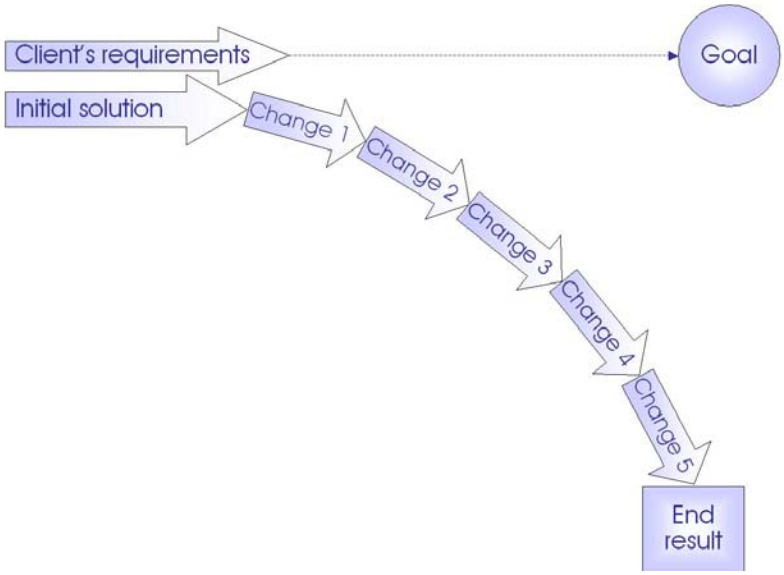

Figure 4: Shifting away from the goal

Our observation, supported by interviews and discussions with many industry experts [Discussion and interviews 2002-2003 [3]], is that to some extent this happens on most projects. This does not mean that most buildings are badly designed, or that they do not meet their overall purpose. However, we argue that they often miss some properties, which the end-users might have preferred, and that the changes of *Requirements* are not well documented. This happens because the design tools do not support such documentation, and the design process includes many trade-offs between different *Requirements*. Therefore, we suggest that the changes should be based on conscious decisions to adjust solutions [Figure 5], *Requirements* [Figure 6], or in most cases both, and that the approved updates in the *Requirements* should be recorded so that they can be checked and compared with the final building afterwards.

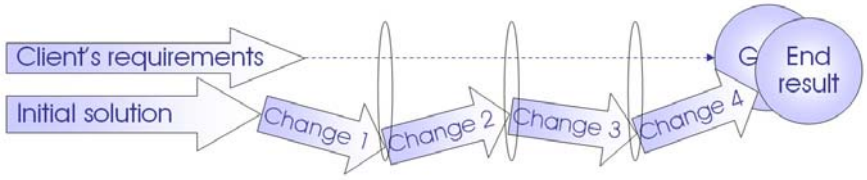

Figure 5: Adjusting design solutions

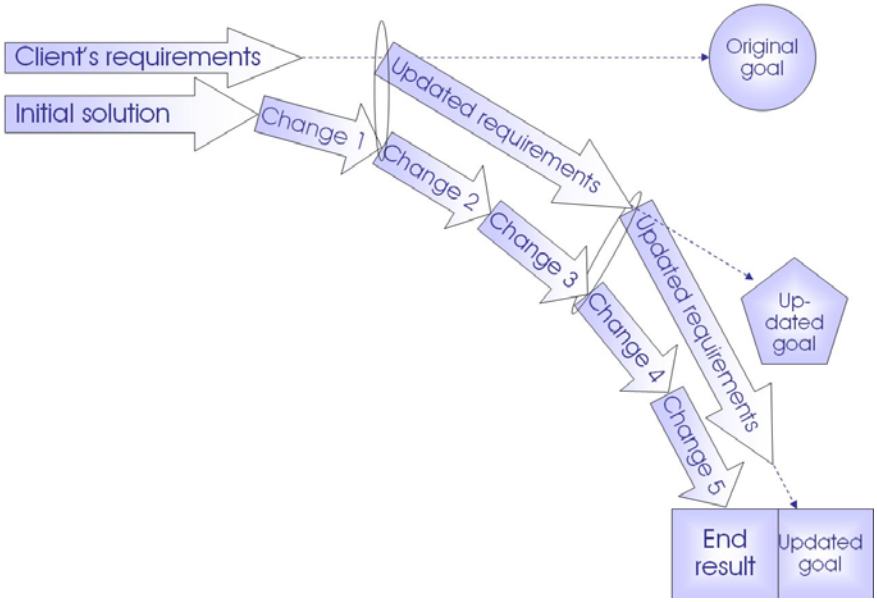

Figure 6: Adjusting requirements

1.3 Main Problems

The main problems we have identified are:

A. No connection between requirements and design documents

The current design tools do not support documentation of the reasons behind the design solutions. As described earlier, the *Requirements Documentation* is used usually only in the early design stages. Later in the process the changes are made based on the previous solution. This leads to the two main problems described above: The design can shift

away from the original goal, and the evolving *Requirements* are not updated in the *Requirements Documentation*.

B. The impact of project personnel changes and project duration

In the current process the *Requirements Changes* are not updated coherently and in an easily accessible format. In the best case, they are stored in the meeting minutes, but often only in the minds of the *Project Team* as tacit and implicit knowledge [Section 3]. Even if the changes are documented in the minutes, they are scattered and difficult to find, especially for people, who do not know exactly what and where to look for. This situation leads to significant loss of *Requirements Knowledge* if some key persons leave the *Project Team*. Long project duration has a similar impact because of personnel changes and human difficulties to remember details.

C. Impact of "middle-men" in the process

The actual end-users are not always closely involved in the design and construction process. Thus, they may lack means to follow and control what happens to their demands in the process. This emphasizes the need to have *Requirements* actively linked to the process, because it would help the designer to find the relevant *Requirements* more easily themselves. In addition, because of described inadequate documentation of the *Requirements Changes*, it is difficult to find the approved *Requirements Changes*, and the end-users may compare the building to the original, outdated *Requirements*.

D. Direct and Indirect Requirements

Most *Client Requirements* are related to spaces and in current practice recorded in the building program. However, these *Direct Requirements* often aggregate *Indirect Requirements* to the bounding elements and technical systems. Bounding elements, e.g., walls, windows, doors and slabs, can have *Requirements*, like, for example, sound or thermal insulation, security, load bearing *Requirements*, etc., inherited from the space *Requirements*. Likewise, technical systems, like mechanical, electrical and plumbing (MEP) systems or information and communication networks, can *Inherit Requirements* from the space *Requirements*. These *Indirect Requirements* can be difficult to notice or remember, because the detailed design related to them often happens late in the process, and is often done by people who were not involved in the early stages when these *Requirements* were defined, and the design documentation does not include the *Requirements Documentation*.

Our observation [Section 3] is that the effect of these factors could decrease, if the *Requirements* would be **easily available** and **actively linked** to the design solutions. Another important part of a good solution is the **appropriate level of detail**; i.e., to find exactly the relevant information for the on-going design task from the project data. This need also creates the demand to **link the *Direct and Indirect Requirements***, so that for example the wall *Requirements* caused by the related space *Requirements* can be easily found.

The existing, structured *Requirements Documentation* in the beginning of the process provides a potentially usable starting point. **The key limitation is the lack of a theory to link the *Requirements* to the *Building Product Model* based design systems.** The key elements missing are:

- Lack of a formal specification of the link between *Requirements* and *Building Product Model*, and

- Lack of a formal specification to aggregate the *Indirect Requirements* for bounding elements and technical systems from the *Direct Requirements*.

1.4 Conceptual Model Structure

To address these limitations, we propose a concept that divides a project's information model on the instance level into four separate models [Figure 7]:

- Requirements Model
- Design Model(s)
- Production Model(s)
- Maintenance Model

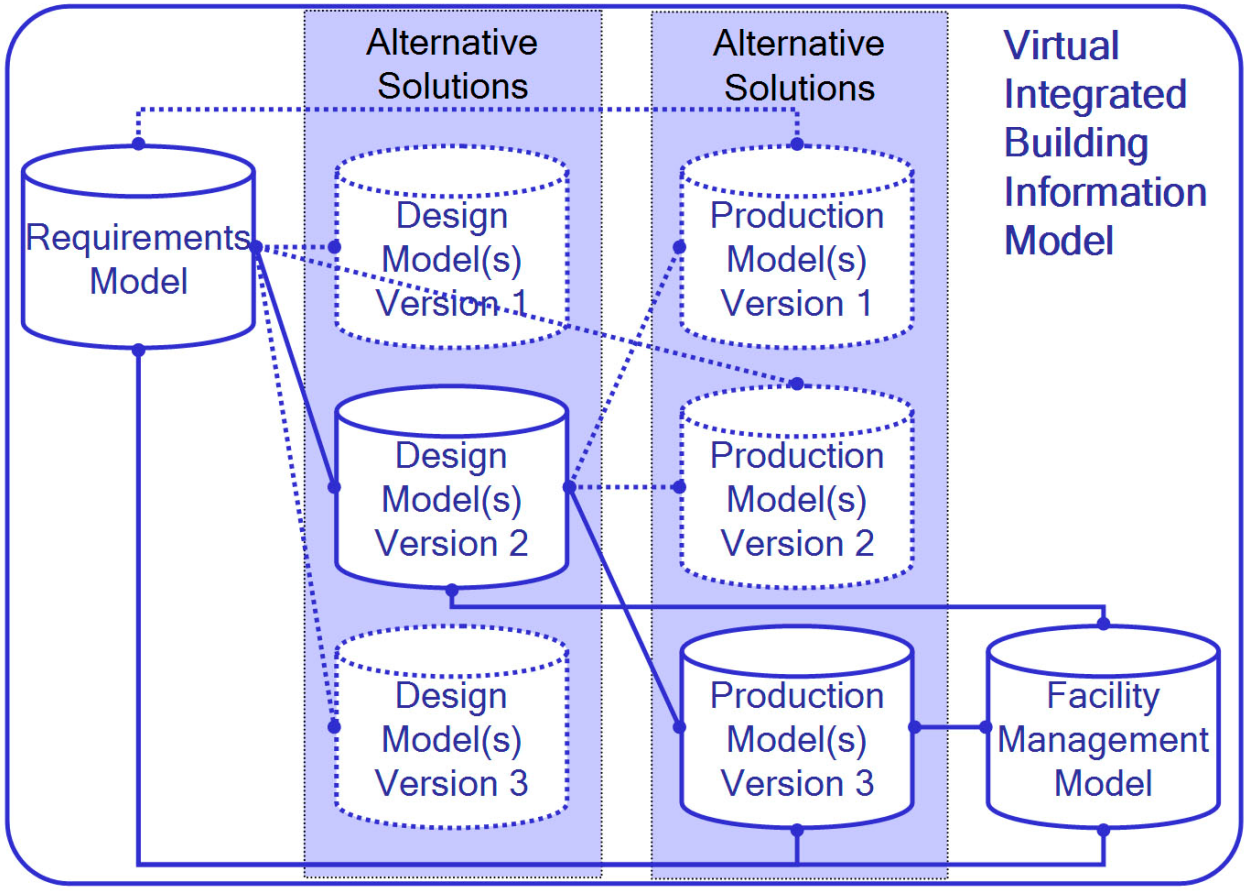

Figure 7: Model Hierarchy and Connections

There are several reasons for this proposed separation:

- Typically the *Project Team* produces several alternative design proposals, which all are expected to 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 four models should be connected into one virtual *Building Product 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 7]. This report focuses on the *Requirements Model* and its connection to the *Design Model(s)*.

- One Instantiable Requirements Entity (IRqE) can relate to a number of separate instances with identical Requirements in the Design, Production and Maintenance Models.
- The existing *Building Product Model* standard, Industry Foundation Classes (IFC), has been developed to describe mainly design solutions, and its current structure does not support *Requirements Management* well, neither does the internal structure of existing design software. The implications of this are described in more detail in Section 4.4.
- The flexibility of the *Requirements Model* structure is greater if the models are separated and connected with a "thin" link. In this case, for example, the only element needed for the link of space *Requirements* is 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.
- Another reason for the separation is to make the distinction between *Requirements* and *Properties* clear; for example sound insulation is a *Requirement* in the *Requirements Model* and a *Property* in the *Design Model*.
-

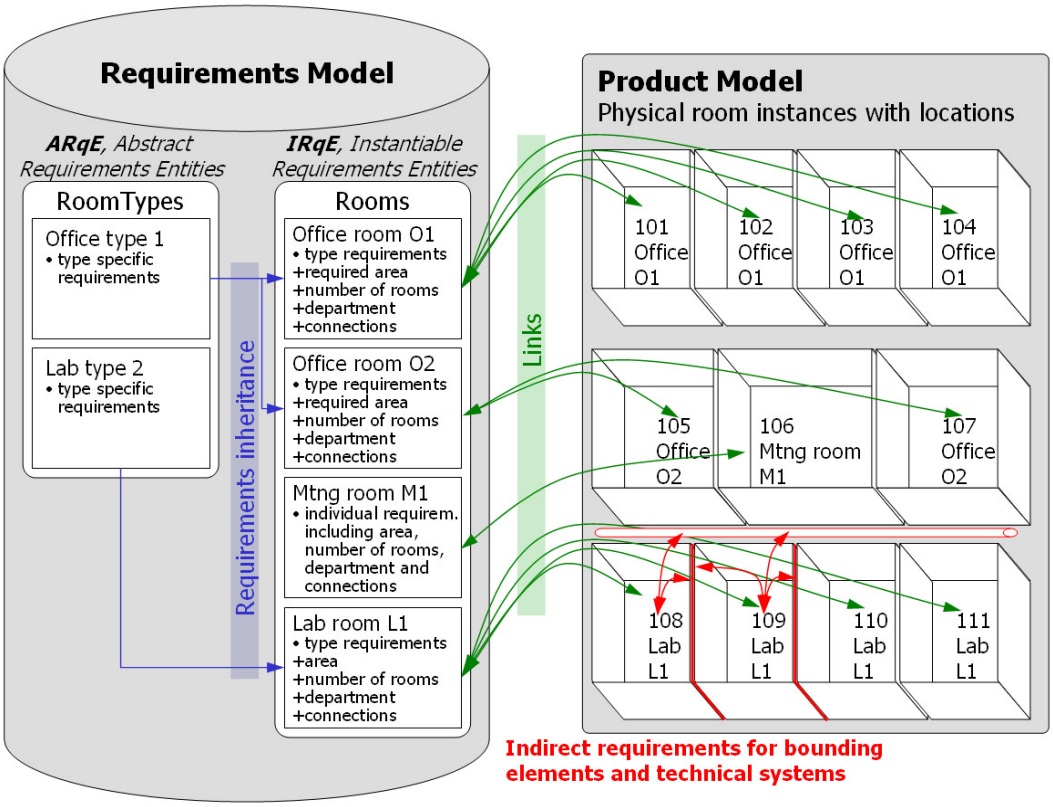

Figure 8: Proposed concept to link detailed space requirements to a building product model: Relations between requirements types (*ARqE*), requirements instances (*IRqE*), physical room instances and *Indirect Requirements*.

A further important observation is that the "*Instantiable Requirements Entities*" (*IRqE*) in the *Requirements Model* have 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 can not be defined directly for the building elements in the same manner as the space *Requirements* relate to the spaces.

1.5 POP, FFB and VDC Framework

The Center for Integrated Facility Engineering (CIFE) at Stanford University has introduced the concepts of Product-Organization-Process (POP), and is using Form-Function-Behavior (FFB) modeling for Virtual Design and Construction (VDC). This framework enables integration of different models, which are often seen as separate entities. Each of the POP elements consists of all three FFB elements, which are divided into three sub-elements; Desired, Predicted and Observed [Figure 9]. This structure provides a conceptual framework for a project ontology connecting the different views to the information. Our *Requirements Model* represents Desired Product Form connected to the Predicted Product Form (*Design Model*) in the POP ontology.

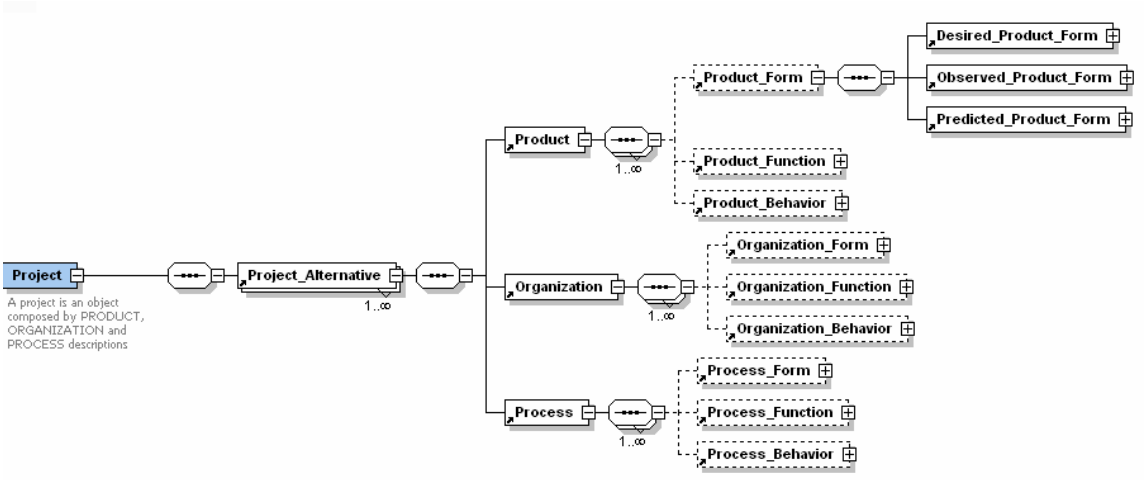

Figure 9: POP Ontology [Garcia et al, 2003 [4]]

2 Definition of the Research Scope

Our research is focusing on the *Requirements Model* and its connection to the architectural *Design Model*. The *Requirements* structure will be based on traditional building programs. The *Direct Requirements* are limited to architectural design. The aggregation of *Indirect Requirements* to the bounding elements, e.g., walls, windows and doors, from these *Direct Requirements* is within the scope of our research.

Detailed *Requirements* for other design areas, like MEP and structural engineering, are not in the scope of the research, but the connection from architectural design to these design areas will be addressed. However, only the need for such a connection from the architectural design will be analyzed and shown, but the detailed content of these *Requirements* is not in the scope of the research.

Project types in the research are limited to office and laboratory buildings. Other building types are not in the scope.

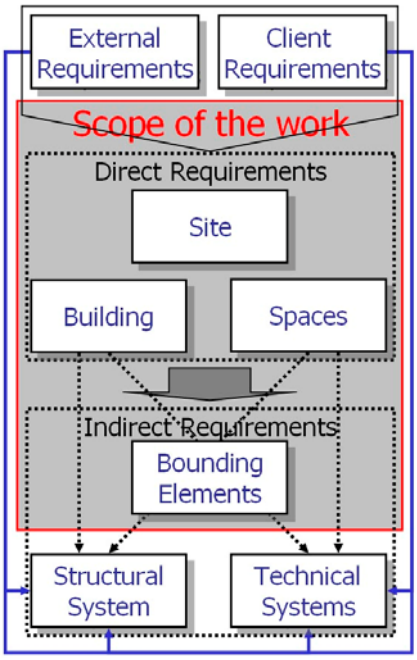

Figure 10: Scope of the work

Because many *Client Requirements* are based on descriptions, not on technical values, automated comparison of the *Requirements and Design Models* is out of the scope, though one can assume that the proposed system would enable automated checking of how well a design solution meets the *Requirements* at least to some extent.

3 Test Cases

To test the existing problems and possible solutions we studied the building programs of two real world projects, implemented some test databases in MS Access and entered the project information into the database. The two projects are the ICL Headquarters project in Helsinki built in 1994-1996 and the on-going Lucas Center Expansion at Stanford University. These two projects were selected to test the generality of the problem and proposed solution, because their characteristics are very different. The ICL Headquarters is a large office building consisting mainly of standard office rooms, but including also some special rooms and *Requirements*. The Lucas Center Expansion is a small special laboratory consisting mainly of unique rooms with very little repetition.

In the test cases the research concentrated on *Client Requirements* related to the spaces only, *External Requirements* were not in the scope at this stage.

3.1 ICL Headquarters, Helsinki

The case, which suggested the idea for the potential solution, is the ICL Headquarters project designed and built in Helsinki, Finland from April 1994 – June 1996. The project is a large office building for ~1,000 employees, including space for an extensive computer service and delivery center. The net area in the building program is ~20,000 m^2, consisting of ~800 rooms. The project's building program was done entirely in MS Excel based on a simple room type classification. In the design phase, MS Excel data was linked to AutoCAD, where rooms were represented using simple objects consisting of polylines and extended data linking the rooms in the drawings and the area *Requirements* in the MS Excel spreadsheet. During the entire design process, the *Target Values* were compared to the design solutions almost in real time, at least once a week, by exporting the actual areas into MS Excel and comparing them to the *Target Values*. However, *Client Requirements* other than area were not linked and observed using this method.

The ICL Headquarters' building program was one document. The required areas were constantly compared to actual design solutions and the requirements file was constantly updated during the design process. The *Requirements Documentation* with respect to required room areas was coherent. The only identified problem was related to the structure used in the document: All classification codes and Requirements were entered manually in each cell, which created the possibility for incoherent content and made updates more laborious. Use of references to one data source, i.e., a simple ***inheritance*** structure, would have prevented this problem.

3.2 Lucas Center Expansion, Stanford University

The structure and size of the building program of the Lucas Center Expansion project is very different from the ICL Headquarters. The Lucas Center Expansion (LCE) is a small special laboratory for Cyclotron and 7T magnetron laboratories for Stanford University. The net area is 480 m^2 including 23 rooms in the first room program (February 1st, 2002), and 1,300 m^2 including 43 rooms in the latest available documents (October 18th, 2002). The available project documentation consists of a set of design sketches, drawings and MS Excel spreadsheets of different project stages, the architect's requirements database in Claris Filemaker, meeting minutes, and technical specifications. The project was in the early construction stages when the study was done (November 2003), and the final pro-

ject documentation was available, but it was not relevant for this research because it contained only design solutions, no updates of the *Requirements*.

LCE's Project Manager (PM) and MBT Architecture's Project Architect (PA) provided some insight on the project. The basic conclusion based on these interviews is that Stanford's projects are generally well-managed and have clearly defined processes for different stages. However, as is typical in the AEC industry, the *Requirements Capturing* process is somewhat fuzzy, based strongly on meetings, where end-users and the *Project Team* are interacting trying to find solutions to specific problems. The decisions are recorded in the meeting minutes, and the room areas of each design stage are documented in MS Excel spreadsheets. The reasoning behind the changes and proposed solution becomes tacit knowledge and is "stored" only in the minds of the participants.

3.2.1 Interview with the Project Manager

On this specific project, the Project Manager recalled two major issues where the necessary *Requirements Information* was not available causing problems to the design process:

- In the first sketches the cyclotron and 7T laboratories were co-located. The reasoning for the design solution was to combine the heavy MEP systems and their spatial needs and separate them from the less demanding office and laboratory spaces. The whole *Project Team* was satisfied with the solution until the equipments' technical information from the manufacturer showed that the rooms must be as far apart from each other as possible, because of the electric and magnetic interference. This lead to a completely new design starting essentially from scratch. This could have been avoided if the necessary information had been available in the space *Requirements*.
- The other major issue was the number of fume hoods in one laboratory. The original demand for fumes was 6, then 8 and finally 12. However, at that stage 12 fume hoods were not possible because of the increasing spatial need for ducts. After some lengthy discussion, the problem was solved by having eight fume hoods of the original type and four additional bio-safe fume hoods, which circulate the air instead of exhausting it.

Both cases illustrate the need for detailed *Requirements Information* in the early design stages, and the connection between *Requirements*, spatial solutions and technical systems. The first example illustrates "inverse adjacency"; i.e., the need to know which rooms must be far apart. The second example illustrates evolving *Client Requirements*. However, these examples are just anecdotal information and based entirely on the PM's memory. The design history or the actual *Client Requirements* were not recorded in the documents in a way which would enable tracking of changes.

To the question if the PM could find the *Requirements* or design criteria to a specific space or building element, his answer was a direct and emphatic "*No*". He said that it would be a very laborious task to go through the meeting minutes trying to find the *Requirements* for any specific space or building element. An excellent practical example of this problem is a quote from the PM's secretary's email: "*I am attaching samples of programming documents per your request, but I am having a hard time finding MBT's design criteria.*" So, not only detailed *Requirements*, but even the high level documents are hard to find in the current process. Of course this is a problem, which can be solved partly just by using existing document management systems, but document based sys-

tems cannot provide formal linking mechanisms to the information content with the necessary structure, as a model based environment can do [Section 4.1].

The PM's opinions about the identified problem were:

- "The problem of *Requirements Management* is real. We have no mechanisms to record, manage and track changes in *Requirements* and especially the reasons behind them."
- "Lots of information is totally 'human dependent'. Thus, keeping the same people in the process is crucial, and for Stanford University the preferred method is to work with the same people in several successive projects."
- *"QFD* [Quality Function Deployment] *is an interesting method".* (The PM had just read an article on QFD in the Journal of Construction Engineering and Management. [5]) The PM felt that "*the main reason that it is not widely used in the construction industry is its separate software environment; there are too many software tools in the process already. If the* Requirements Management *solution would be integrated on the same platform, which is already used in the process, the usability and benefits would be much higher."*

3.2.2 Interview with the Project Architect

In an interview, the Project Architect gave the following answers:

Q1a: Could you find the answers and how much time it would take if you would have to trace back any specific *Requirements*, like, for example: "What did the client exactly require for this laboratory? When and who set the *Requirements*?"

A1a: "*We back up the design documents of every phase. It would take several hours for me to restore the backups, but then we could trace back how the design solutions developed. However, we do not record the actual Requirements Changes. The only documents where this could be found are the meeting minutes, but they do not cover all issues."*

Q1b: Could anyone else find them and how much time would it take?

A1b: *"Even in the best case it would take much more time than for me. In the worst case they could never find the right answers."*

Q2: Can you recall concrete situation, where you spent much time searching for relevant *Requirements* or where you worked with the wrong assumptions?

A2: *"Not on this project, because we have been involved from the beginning. But it happens often if the project personnel changes, because a large amount of the information is just in the head of the Project Architect."*

Q3: Do you use any other methods to communicate the *Client Requirements* to the other participants than telling what you know?

A3: *"Only in the programming phase, where we use our database tool to record some Requirements, but even then not all the details. In the design development phase there is too much work and information. We don't update our requirements database after the programming phase. The information in later phases is our design recorded in the drawings and other design documents."*

Our review of the architect's requirements database supports the statement that not all the details are recorded [Section 3.2.3].

3.2.3 Detailed Findings and Problems for the LCE project

Based on our experience and the interviews and discussions with industry experts [Discussion and interviews 2002-2003 [6]], the information management problems increase when the project size and complexity increases. The Lucas Center Expansion is a small project, and the amount of information in the Requirements Documentation was also relatively small, but despite of this the information was incomplete and contained several inconsistencies, which demonstrates that these problems occur on small and large projects.

The main problems were related to the use of two different sources of information, the PM's MS Excel spreadsheets and the PA's requirements database, and their different and partly inconsistent content. In addition, the MS Excel sheets for different stages were in separate files and the development history or reasons were not recorded even for large changes. For example the changes of the net area in different stages were very large (242%-1076%) [Table 1]. In fact, only the first version (February 1st, 2002) presents the actual *Client Requirements*; later versions summarize rather the design status in different stages.

More complex technical specifications, like MEP descriptions, have no relation to the PM's or PA's *Requirements Documentation*. "MPE Utility Planning and System Description VI, March 05, 2002" specifies clearly the *Requirements* for the two main rooms, 7T MR and Cyclotron, but the required properties for the other rooms are not easy to interpret. However, because the actual MEP systems are out of the scope for this research this was not studied in detail. It indicates though, that the *Requirements Management* problem is also related to other design areas.

Table 1: Changes of the building program summary of Lucas Center Expansion

	Feb 01 2002	*Apr 17 2002*	*Change*	*Sep 11 2002*	*Change*	*Oct 18 2002*	*Change*	*Nov 26 2002*	*Change*	*To Original*
7T MR	2 380	1 680	*71%*	1 736	*103%*	1 802	*104%*	2 011	*112%*	*84%*
Cyclotron	1 050	1 034	*98%*	997	*96%*	1 005	*101%*	2 536	*252%*	*242%*
Hot Lab	1 020	690	*68%*	1 288	*187%*	1 120	*87%*		*0%*	*0%*
Wet Labs		3 550	*1076%*	3 252	*92%*	4 326	*133%*	4 505	*104%*	*442%*
Lab subtotal	***4 450***	***6 954***	*156%*	***7 273***	*105%*	***8 253***	*113%*	***9 052***	*110%*	*203%*
Admin & Support	750	750	*100%*	750	*100%*	2 856	*381%*	4 926	*172%*	*657%*
Total	**5 200**	**7 704**	*148%*	**8 023**	*104%*	**11 109**	*138%*	**13 978**	*126%*	*269%*
Technical spaces		771		1 150	*149%*	1 162	*101%*	1 196	*103%*	*155%*
Unassigned spaces		5 195		5 234	*101%*	5 895	*113%*	5 895	*100%*	*113%*
Gross area	**10 400**	**13 670**	*131%*	**14 407**	*105%*	**18 166**	*126%*	**21 069**	*116%*	*203%*
Efficiency, real	*50%*	*56%*		*56%*		*61%*		*66%*		
	100%	***131%***		***139%***		***175%***		***203%***		

The following is a detailed list of discovered problems and contradictions:

PM's MS Excel spreadsheet:

- The information content is just ID, name, area and *Required Location* (floor) -> the file covers only area *Requirements*, all other *Requirements* are only in the architect's database. In fact, as mentioned before, even the area information is reflecting rather the design status than the *Client Requirements*.
- The same Room ID (5.10) is used for two different rooms -> Information and Communication Technology (ICT) based identification is impossible.
- Three rooms do not have IDs at all -> ICT based identification is impossible.
- In some cases a manual summary of rooms per floor exists -> summary and individual areas do not match.
- The original area *Requirements* are not stored -> changes are difficult to follow.

PA's requirements database:

- Only 1/3 of the rooms are in the database (13 of the 39 rooms in the PM's MS Excel spreadsheet).
- Area *Requirements* are not included in the database.
- The IDs are often different or missing, and the room names are often different from the names in the PM's MS Excel spreadsheet -> ICT based identification is impossible, and in some cases identification of rooms is impossible even for people, who do not have the tacit project knowledge:
- There are two different wet labs, but they do not have IDs -> it is impossible to know which is which in the other documents
- Hot labs are missing from the MS Excel file
- In some cases there are adjacency references to rooms, which do not exist in the documentation.
- There are several, slightly different ways to document the same issues:
- Room types
- Activities
- Materials
- Casework and equipment
- There are some obvious mistakes in the *Requirements*:
- The natural light *Requirements* are sometimes in unnecessary (storage rooms) or absolutely impossible places (cyclotron room). It appears to be a default value in the database, and as a consequence, it is listed for these rooms as well.
- A 1' door in the Hot Lab/Research room
- A maximum noise level *Requirement* for a storage room.

3.3 Conclusions from Both Test Cases

Based on our own experience and several interviews and discussions with industry experts [Discussion and interviews 2002-2003 [7]], the *Requirements Documentation* and process in the LCE project are a typical example of practices on current construction projects. Different parts of the *Requirements* are documented in several documents, and there is no comprehensive document containing all needed information. In addition, the names and IDs for the rooms are often ambiguous, and similar *Requirements* are formulated in different ways. This makes it difficult to connect *Requirements* to the correct room even manually, and any use of ICT to manage the relations between the *Requirements* and design solutions is impossible.

The main problem categories in the *Requirements Documentation* for the LCE project were:

- Lacking or different identifications of the rooms,
- Contradictions in the content of different documents,
- Incoherent way to describe the same *Requirements*,
- Wrong or missing information,
- Instead of actual spatial *Requirements* the documents recorded the areas of the rooms in the design solution, and
- Documents specifying detailed technical *Requirements* had no relation to the room related *Requirements Documentation*.

Though many of the mistakes in the LCE project were small, and probably caused little, if any, real problems to the people, who have been involved in the project all the time, they are a clear indication of the general *Requirements Management* problem in the current process. To anyone, who joins the project later, it is very difficult and time consuming, sometimes impossible, to find out which *Requirements* are correct and still relevant. Furthermore, someone who wonders about the growth in the size of the project will have great difficulty finding an answer in the project documents.

Though only the required area information of the ICL Headquarters project was linked with the design solution, it provided some real benefits. ICL's Project Manager states: "*Still today, over 9 years later, ICL Headquarters is the only project, where we got practically real time information comparing actual areas to the building program on a detailed level, and was able to follow constantly that the project design stayed within the allocated limits.*" In addition, despite the simple approach taken in the ICL project to only link the *Requirements* and the design information for comparing required and real areas, the coherent *Requirements Information* suggests that a link between *Requirements* and design tools and the constant use of *Requirements Information* in the process could improve *Requirements Management*.

4 Related Work and Current Limitations

As stated in Section 1.3, there is no theory, which would provide the basis to link *Requirements* to a *Building Product Model* representing a design solution. A solution to the above mentioned problems can build on the following five starting points:

- Design as an information process,
- Existing Requirements Documentation,
- Lawrence Berkeley National Laboratory's (LBNL) Building Life-Cycle Information Support System (BLISS) and Design Intent tools
- Existing *IFC Specification* and implementation, and
- Building Lifecycle Interoperable Software (BLIS) views.

Design as an information process justifies **why** the *Client Requirements* and their management should be linked to the design process. Existing *Client Requirements* provide the basic content for the *Requirements Model*, i.e., **what** should be linked. LBNL's Design Intent and BLISS tools are a reference for *Requirements Management* in the MEP area. The existing IFC structure describes what is available and what is missing in the *Building Product Model* based design software; **to which** *Requirements* can be linked, and the existing implementations and BLIS views provide the technical platform; **how** to establish the link. The existing elements are *Requirements Documentation* and *Building Product Model* based design software, the main limitation is the lack of a method to link these and handle the relation between *Direct and Indirect Requirements* [Section 1.4 and Figure 15].

4.1 Information Processing and Management in Design

Froese (2002 [8]) describes the design process as an information processing activity: "*All design and management tasks are primarily information-based activities; they take certain information as inputs, create new information about the project, and produce some type of information as a result*". In the beginning of the process, the inputs are *Client Requirements* and *External Requirements*, like, for example, site *Requirements*, building codes and other regulations, and the *Project Team's* knowledge [*Kamara et al, 2003* [9]]. Later in the process the previous design solutions - modified information - are increasingly used as inputs, while the use of *Client Requirements* - original information - decreases [Section 3.2.2, A2]. As described in Section 1.2, incremental changes based on previous solutions without comparison to the original *Requirements* can gradually lead to an end-result, which differs significantly from the *Requirements* without conscious decisions to change the scope of the project. This is the key observation behind the *Requirements Management* problem, which is the basis for this work. However, there is little research of this problem related to building programs in the building design process.

Efficient and appropriate information management is crucial for the success of the project [*Best and De Valence 1999* [10]*, Kamara et al 2003* [11]]. The information processing needs in complex building projects are very high and the increasing demands to fast-track the process make the information management an even more crucial issue [*Eastham 2002* [12]*, Confederation of Finnish Construction Industries 2003* [13]]. The development of ICT has provided significant help to many of the problems related to information management.

However, many design tools were developed to automate drafting, instead of serving information management. The drawing based documents are human, not computer interpretable, which sets serious limitations to the reuse and linkage of the information represented in the documents [*Froese 2002* [14]]. Froese identifies two basically different approaches to the information management problem: Internet-based collaboration, and Model-based approaches. Internet-based collaboration is mainly based on electronic versions of the traditional human-readable documents, while the model-based approach is based on a different abstraction of a real building having a defined semantic content, which is also computer interpretable.

Another approach to the design as an information process is the Active Design Documents (ADD) concept [*Garcia et al, 1993* [15]], which demonstrated an automated approach to record the design intent in preliminary routine design. The main focus in the ADD research was on designers' decisions, while our research is focusing to the management of *Client Requirements* and the connection between the *Requirements* and design solutions.

Our research is based on these two observations:

- The need to manage *Requirements Information* during the design process, and
- The possibility to link *Requirements* to the information content in the *Design Model*.

Because the semantic content of the *Building Product Model* enables a meaningful connection between *Requirements* and project, site, building, spaces, building elements and systems, our research builds on the model-based approach and existing *Building Product Models* [Sections 4.4 and 4.5].

4.2 Requirements Capturing and Documentation

Requirements Capturing is a wide area, starting from high level strategic views of real estate portfolios ending with detailed technical specifications. Our research scope covers only a small subset of this area; *Requirements* related to architectural design. These *Requirements* are in practice captured mostly by interviewing the *Clients*, owners and end-users of the building, and they are documented in the building program.

Some *Requirements* are common to practically all buildings, like, for example, required area, activities in the space, connections to other spaces, etc. Some *Requirements* are specific only to some building types, like, for example, exact limits for minimum and maximum temperatures and moisture, which are common for laboratories, museums, demanding technical facilities, etc., but not defined for most buildings. We argue that these different types of *Requirements* can not be standardized. Thus the goal of our research is to identify a reasonable set of *Requirements* within the defined scope and create a flexible framework, *Requirements Model Specification*, which enables additional project-specific *Requirements* in the project's *Requirement's Model* [Sections 6.1 and 6.2].

Furthermore, the source of requirements is varying, in many cases the original *Client Requirements* are fuzzy, and designers turn them into more accurate *Requirement Descriptions* or *Requirement Attributes*. These varying needs make it difficult, if not impossible, to define a perfect method to capture *Requirements* or define their content, i.e., a comprehensive set of *Requirements*. However, *Requirements Capturing* is not in the scope of our research. It starts from the assumption that *Client Requirements* are de-

fined using some method, and in some structured form, which has a relation to project, site, building, spaces and their Requirements.

There are several structured *Requirements Capturing and Documentation* methods; Quality Function Deployment (QFD), Client Requirements Processing Model (CRPM), Total Quality Management (TQM), Failure Mode and Effects Analysis (FMEA), etc. [*Kamara et al, 2003* [16]].

QFD includes many *Requirements Management* features, and it is widely used in other industries. However, it is not commonly used for building projects. There are several identified reasons for this. It has been observed that the effectiveness of QFD diminishes downstream, e.g., in actual design and planning stages, phase 3 and 4 in Figure 11, which are the stages of building design and construction activities [*Evbuomwan, 1994* [17]]. Prasad [*1996* [18]] argues that this makes QFD less likely to deal with complex products and conflicting *Requirements*, like buildings. Furthermore, the latest AEC related QFD research [*Syed et al, 2003* [19]] finds the method potentially useful for defining strategic goals, but not for detailed *Requirements*. An interesting observation about QFD was the LCE's PM's comment of the need to integrate the tools into today's practice, instead of trying to bring a new software platform to the process [Section 3.2.1], which supports the basic idea of our research; linking *Requirements* and existing design software.

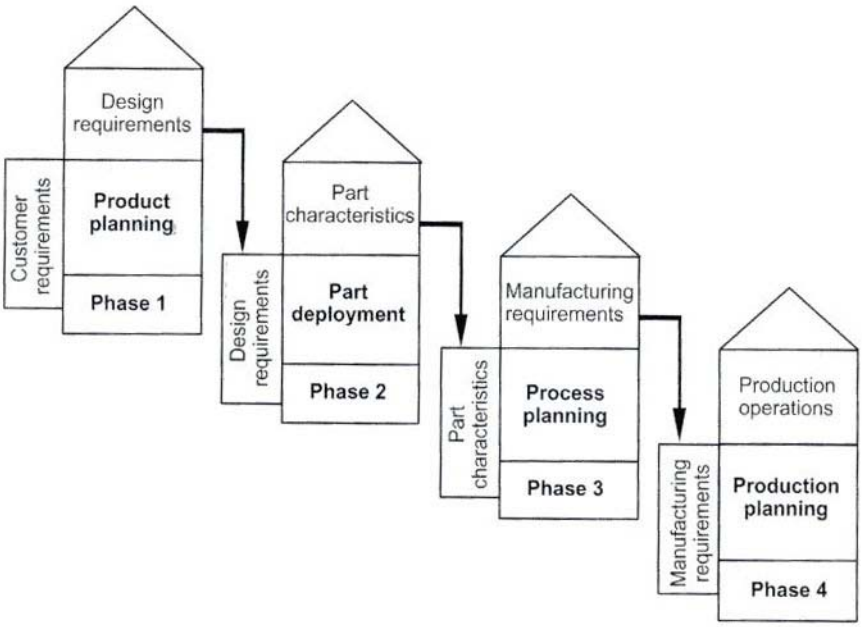

Figure 11: The four stages of the QFD process [*Kamara et al, 2003* [20]]

CRPM has been developed to improve the client *Requirements Capturing* process [*Kamara et al, 2003* [21]]. Its main focus is in high level *Requirements*, but its most detailed level could be a useful source for room related *Client Requirements*. However, the method is new and not widely tested or used. More important from our research viewpoint is that traditional building programs provide at least the same information related to the spaces than the CRPM. However, a direct data link from CRPM, or some other existing *Requirements Capturing* tool, to our *Requirements Model* is a related research topic, but it is not in the scope of our research.

As stated above, the most common method to document *Client Requirements* is the traditional building program, and we have chosen it as the starting point for our *Requirements Model Specification*. In addition, our argument is that as long as the information content is the same, *Client Requirements* can be managed using the proposed method regardless of the capturing method; the purpose of required area, minimum temperature, access control, etc., is the same if they are defined with QFD, CRPM or any other method. The important issue is the relevant content, and though it can not be fully standardized, as described above, one of our contributions is to define a concept and method to link different types of *Requirements* to the *Building Product Model* [Figure 24].

The focus of our research proposal – detailed *Direct and Indirect Requirements*, and their connection to the *Design, Production and Maintenance Models* – is specific to the construction industry. Thus, our argument is that because of the specific product structure and different processes the existing *Requirements Capturing* methods used in other industries are not directly applicable to the identified problem.

4.3 LBNL's Requirements Management Research

In the technical systems area the research to capture and manage the *Requirements* has been more active than in the research related to *Requirements* for architectural design. Lawrence Berkeley National Laboratory has carried out several projects, in which the main focus was in building performance and especially in the energy efficiency [*LBNL 1995-2003* [22]]. Two main efforts have been Building Life-Cycle Information Support System (BLISS) and Design Intent Tool. As described in the Sections 4.3.1 and 4.3.2, these projects do not provide a direct basis to our proposed research, but the work at LBNL in this area is related to our research. Thus, collaboration with LBNL's development is an important part of our project, and Dr. Vladimir Bazjanac from LBNL's Environmental Energy Technologies Division is one of the authors of this report.

4.3.1 Building Life-Cycle Information Support System

BLISS development was aiming to be consistent with the IFC specification, and according to the BLISS web site the project goals were partly overlapping with our research [*LBNL BLISS, 1997* [23]]: "*The goal of the BLISS effort is to create a software infrastructure that can be used for information sharing across disciplines and can be used to link interoperable software tools throughout the building life cycle. The project has three major elements: (1) to specify the distributed software architecture, (2) to develop a life-cycle building model database schema, and (3) to develop a mechanism to capture and update "design intent" throughout the life cycle. The distributed systems architecture describes how various software components communicate, and the building model schema specifies the structure and semantics of the database.*"

However, the results have not been published, and the project has finished without the intended link between the design intent and design software. Another quote from the BLISS web site: "*An initial version of the BLISS infrastructure will be built as an extension of the Building Design Advisor data model. Intended extensions to this model include data for documenting design intent, in the form of performance metrics, and time-series data for documenting actual building performance over time. An initial implementation of BLISS is expected to be developed during 1997.*" The website is still accessible (September 2004), but the link to software tools points to a non-existing page.

4.3.2 *Design Intent Tool*

The Design Intent Tool is publicly available software, including some parts of the earlier BLISS development mentioned above. Quote from LBNL's website [*LBNL DIT, 2003* [24]]: *"This database tool provides a structured approach to recording design decisions that impact a facility's performance in areas such as energy efficiency. Using the tool, owners and designers alike can plan, monitor and verify that a facility's design intent is being met during each stage of the design process. Additionally, the Tool gives commissioning agents, facility operators and future owners and renovators an understanding of how the building and its subsystems are intended to operate, and thus track and benchmark performance."*

As described, the implementation focuses on energy efficiency, but the overall goal, managing the *Requirements* through the design, construction and maintenance processes, is the same as in our research, though the application area is different. The tool consists of a database solution enabling flexible documentation of objectives, strategies, metrics, responsible agent, etc. for the MEP systems [Figure 12]. However, the tool concentrates on *Requirements Documentation* only, and does not have a link to the design solution, which is the core element for our research.

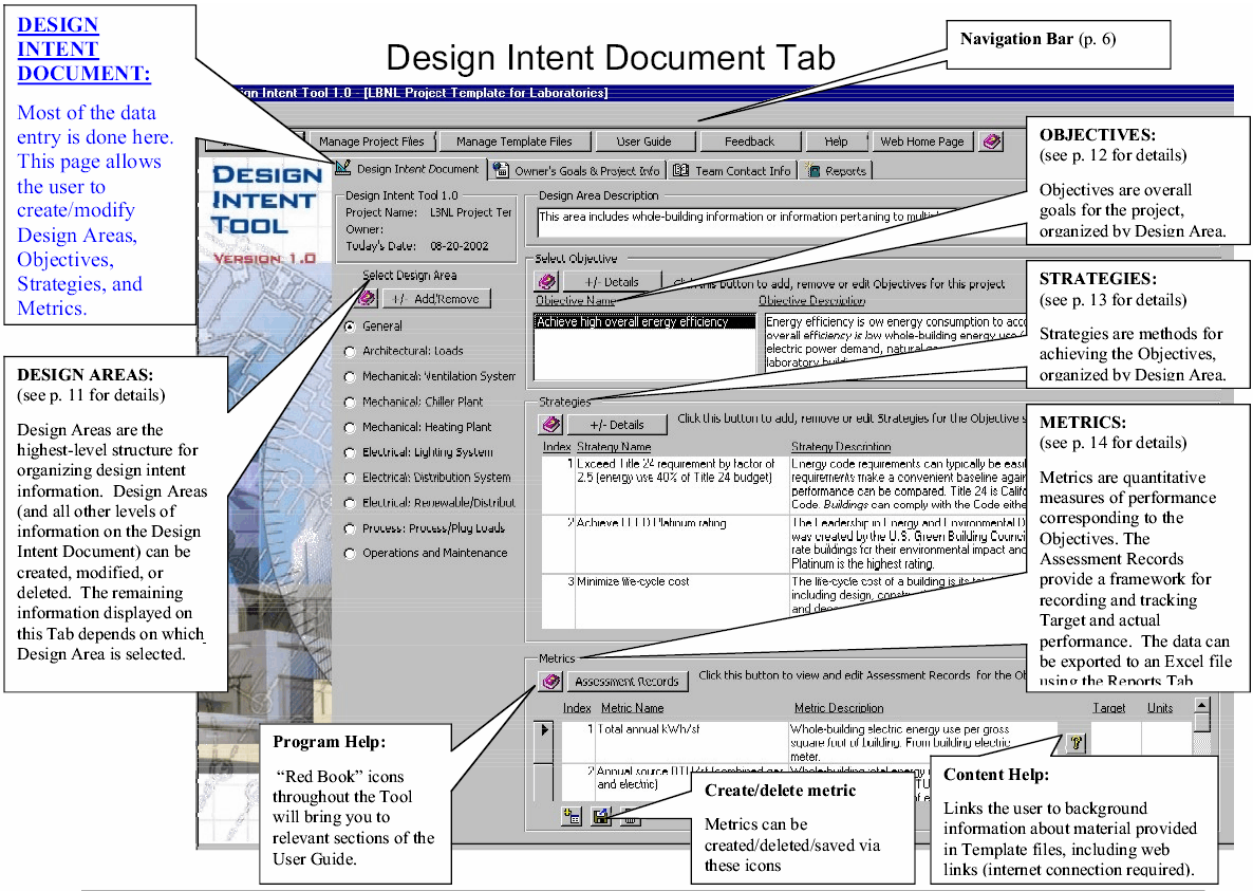

Figure 12: Design Intent Tool's User Interface, © LBNL 2002 [25]

4.4 Current Status of *Building Product Models*

The key element in our research is a link between *Client Requirements* and design solutions. As described in Section 4.1 the link cannot be based on traditional, human readable documents. Its prerequisite is a semantic *Building Product Model*, which consists of building objects like spaces, walls, doors and windows, etc.

The International Alliance for Interoperability (IAI) has developed a *Building Product Model* standard for the AEC/FM industry. IAI has produced several versions of this standard called Industry Foundation Classes (IFC). The IFC2x Platform Specification was also officially accepted as a Publicly Accessible Specification ISP/PAS 16793 by ISO in November 2002 [*IAI NA 2003* [26]]. This means that the *IFC Specification* is not only a de-facto standard, but official standard, for *Building Product Models*, and thus the logical basis for the *Building Product Model* related part of our research. The *IFC Specification* and its implementation provides sufficient information content for the objects related to the *Requirements* relevant for our research; rooms and their related bounding elements, walls, windows and doors.

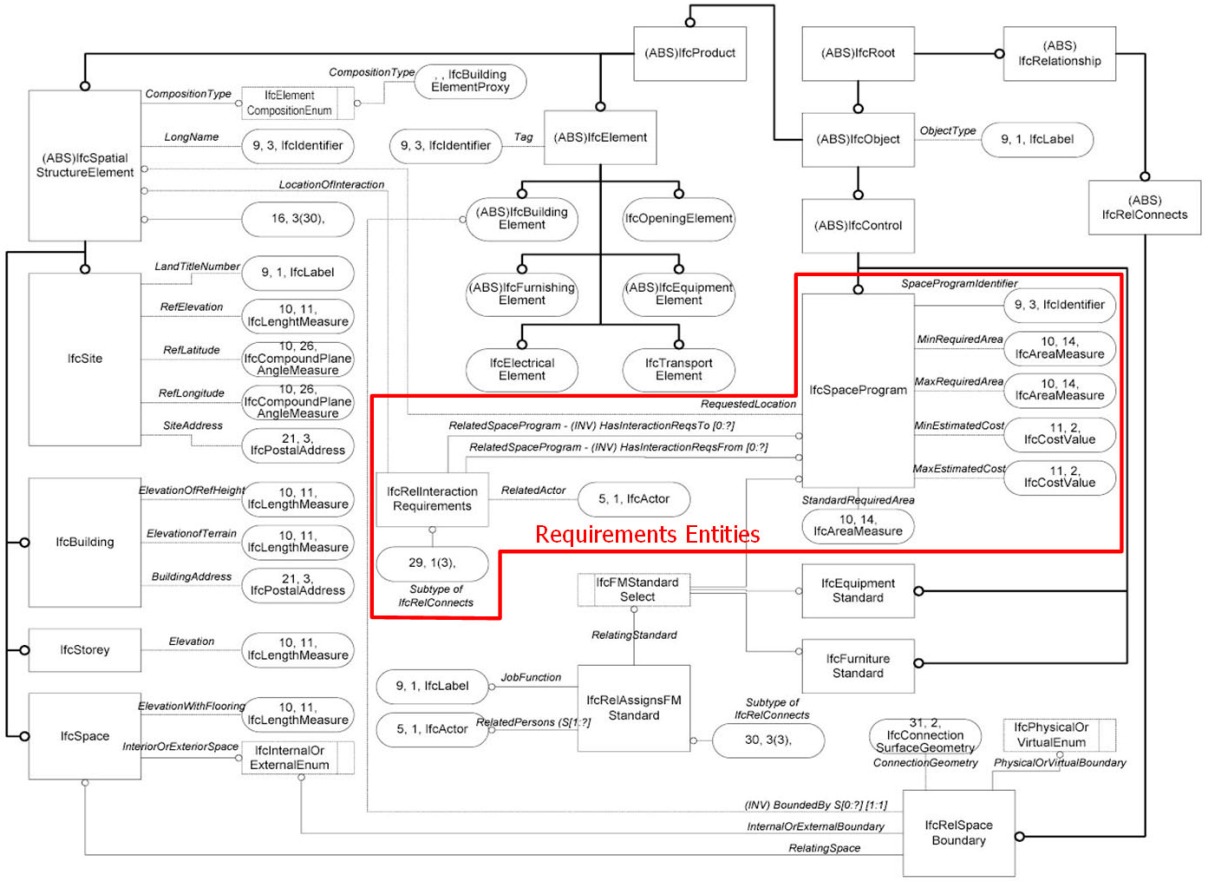

Figure 13: IFC 2x requirements elements and their relations

The limitation is in *Requirements Management*. The main focus of IFC development has been on design view; i.e., the *Specification* includes extensive building geometry representation and many other design properties for building objects, but it does not support other phases of the process well. The current *IFC Specification* contains only limited support for *Requirements* [Figure 13]. This part of the *Specification* has been implemented in one software only, Alberti. This software does not support *Requirements* other

than minimum and maximum areas and physical connections between rooms, and is not a suitable tool for large projects [*SPADEX project report 2002* [27]]. The main reason for this is the complexity of defining the connections, and the attempt to automate the creation of space layout, which is extremely complicated if the number of the spaces is large.

There are two on-going projects trying to expand the *IFC Specification* to the *Requirements* level [*IAI projects 2003* [28]]. These projects are:

- Portfolio and Asset Management: Performance Requirements (PAMPeR, IAI FM-9)
- Early Design (ED, IAI AR-5).

The focus in these two projects is in capturing and documenting the *Requirements*, not in linking the requirements to design solutions, which is the focus of our research [Figure 14]. In addition, the *Requirements* are stored in PropertySets, which have certain limitations (Appendix 1, A1)

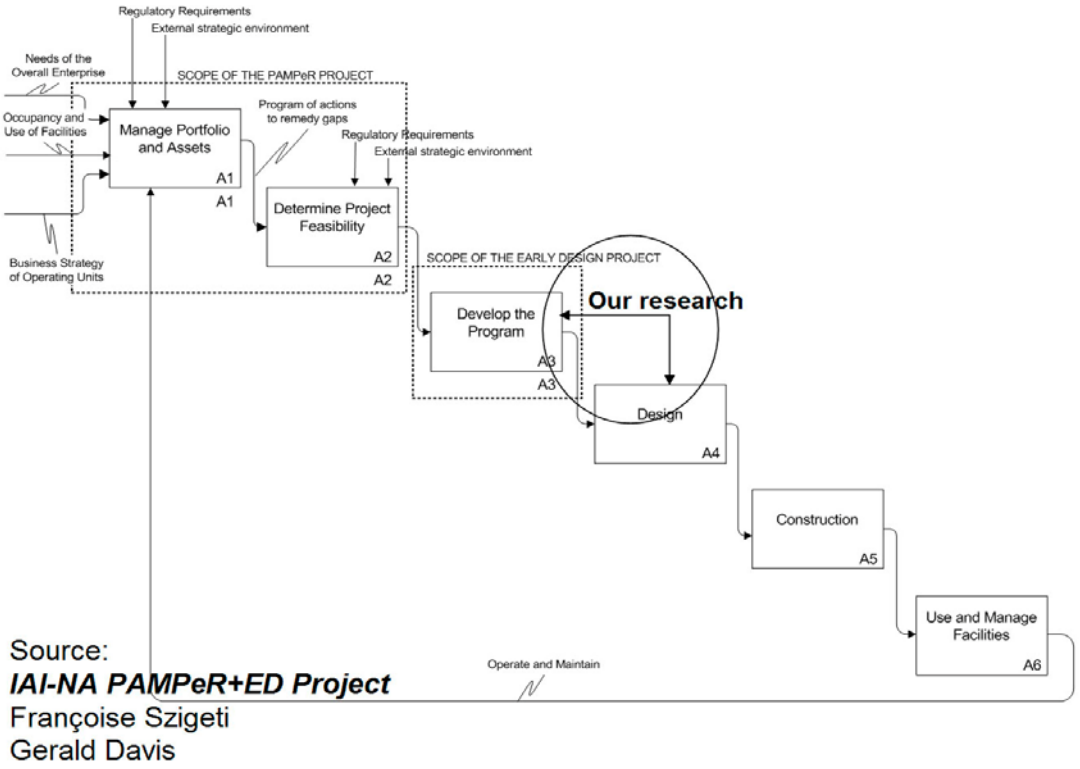

Figure 14: PAMPeR/ED project scope and relation to our research

In addition, our argument is that including the *Requirements* in the design objects on the instance level in the *Building Product Model* is not a feasible solution [Sections 1.4 and Appendix 1, A1]. IFC implementation is already very demanding and this has created the need to develop easier methods to use the *IFC Specification* [SABLE 2003 [29] and Appendix 1, A4]. We argue also that on the instance level the *Requirements Model* and *Design Model* are two different levels of abstraction [Section 1.4]. Thus, combining them into the same objects on the instance level would make both the implementation of the *IFC Specification* and the project's information management more complicated. Our proposed solution is a link between *Requirements Objects* in the *Requirements Model* and objects in *Design, Production and Maintenance Models*, separating the *Requirements* and solutions at the concept level to individual objects [Figure 8]. This approach also matches research on representing form, function and behavior (FFB) [Section 1.5].

At the moment, there are no *Requirements Capturing* tools supporting IFC, though one object based briefing study has been recently executed [*Ekholm and Lehtonen 2002* [30]], and at least one prototype software for early design exists [*BLIS 2003* [31]].

However, the semantic structure of the IFC specification and its current implementations provide the basis for our proposed solution to link the *Requirements Model* and *Design, Production and Maintenance Models* as described in Section 1.4. The needed elements from the *Design, Production and Maintenance Models* are identifiable objects, which can be linked to the *Requirements Objects*, and the identification of related objects, which can be affected by the *Indirect Requirements*, like, for example, bounding elements of the space and technical systems serving the space.

For practical software implementation, the preferred solution to implement the interface between the *Requirements Model* and the existing *Building Product Model* applications is to use model server technology and some standardized API, like the SABLE interface described in Appendix 1, A4. Using a standardized API would make the implementation easier and provide a connection to several design software. However, this is not a crucial issue for our research; the connection between *Requirements, Design, Production and Maintenance Models* can be implemented on three levels [Figure 15]:

- Using the SABLE application interface,
- Using one of the IFC Model Servers, e.g., IMSvr, WebSTEP, etc., or
- Directly with some *Building Product Model* based software, e.g., ArchiCAD, AutoCAD ADT, MS Visio, etc.

4.5 BLIS Views

Because of the complexity and ambiguity of the EXPRESS based *IFC Specification* the Building Life Cycle Interoperable Software (BLIS) group has developed the view concept to support IFC based information exchange [*BLIS 2003* [32]]. It is possible to implement the *IFC Specification* in several ways and any individual software product supports only some parts of the *Specification*. However, the information exchange must be based on the same content and interpretation of the *Specification* in the *Building Product Model*, what is the needed information content for a certain task, and how to present the geometry, properties, etc. needed in that task. The current views are:

- Architectural Design -> Quantities Take Off / Cost Estimating
- HVAC System Design -> Quantities Take Off / Cost Estimating
- Architectural Design -> Thermal Load Calculations / HVAC System Design
- Client Brief / Space Layout -> Architectural Design
- CAD View

The relevant view for our research is “Client Brief / Space Layout -> Architectural Design” (CB/SL-AD). The following descriptions in italic are quotes from the BLIS website [*Hietanen, 2003* [33]]:

“The view for 'Client Brief / Space Layout -> Architectural Design' defines the subset of the IFC model that is used for transferring spatial data from the client brief to architectural design applications.

The Client Brief application can be anything from a simple spreadsheet to a real application, the important thing is that it can output the requirements captured in the client brief

into IFC format. Architectural design applications can import the resulting IFC file and start the actual design process designing the spaces, walls, doors, windows, etc. There can also be a special space layout program between the Client brief and the architectural design application.

The first level of functionality is to be able to generate a 'space skeleton' that matches the requirements set in the client brief, e.g., the right number of spaces of the right types and areas. The second level is to actually store the requirements in the design application and to be able to give feedback about how the design meets the requirements."

As described in Section 1.4 our proposed solution for the second level of functionality is based on separating the *Requirements Model* from the *Design Model* instead of storing the requirements in the design application. This approach has been discussed with the BLIS technical team and accepted for the basis to expand the current CB/SL-AD view. This approach also enables the use of a Model Server database as the repository for both the *Requirements and Design Model* information [Figure 27, Option #2].

4.5.1 List of supported concepts in the current CB/SL-AD view

The BLIS views consist of concepts; functional units isolated from the *IFC Specification*. The views are built by combining the relevant concepts using them like "building blocks". The following lists of BLIS concepts in the CB/SL-AD view for IFC 2.0 are grouped based on the relevance for the PREMISS *Requirements Model* and the link between it and the *IFC Specification*. A short explanation is in the brackets after the concept name:

Concepts, which should be part of the link:

- *Building* [Some *Requirements* are linked to the *Design Model* on the building level]
- *Containment* [For example, space can be a container for its furniture and equipment.]
- *Dynamic property assignment* [The mechanism to assign property objects or property sets to objects dynamically, i.e., without changing the IFC schema.]
- *Project* [Some *Requirements* are linked to the *Design Model* on the project level.]
- *PropertySet system* [This is used to dynamically attach properties to IFC objects; e.g. implementation of attributes, which are not defined in the IFC specification. However, this method causes problems, which are discussed in Appendix 1, A1]
- *SimpleProperty* [This defines a simple property that can be used in a PropertySet. The 'SimpleProperty' has a name and a value.]
- *Site* [Some *Requirements* are linked to the *Design Model* on the site level.]
- *Space* [Central element for the *Requirements Model* and the link to the *Design Model*.]
- *Space program properties* [Central element for the *Requirements Model* and the link to the *Design Model*. This is a property set called Pset_SpaceProgramCommon and in *IFC Specification* 2.0 it consists of ProgramSpaceDescription, ProgrammedFloorArea and Function. In the *IFC Specification* 2x2 the content is changed to cover the following attributes: Location *(Required Location)*, FunctionRequirement, SecurityRequirement, PrivacyRequirement, LightingRequirement, FFETypeRequirement, EmployeeType, OccupancyType and OccupancyNumber, StandardRequiredArea is added to the IfcSpaceProgram object.]

- *Space type* [Central element for the *Requirements Model* and the link to the *Design Model*. This is a property set called SpaceCommon and consisting of Description.]
- *Unit assignment* [IfcUnitAssignment defines whether the units are metric or imperial.]
- *Units [metric]* [Defines the metric units used in the project.]

Concepts, which need to be considered:

- *Building story* [Though unusual, can be a relevant level for *Requirements*.]
- *Organization* [Currently used for project participants in *IFC Specification*. However, this might also be relevant for end-users of the building.]
- *Owner history* [Defines the ownership of the objects in the *Design Model*. This concept can be used if the original space objects are automatically generated from the *Requirements*.]
- *Person* [Currently used for project participants in *IFC Specification*. However, this might also be relevant for end-users of the building.]

Concepts, which are not useful:

- 2D placement [Geometrical Locations are not Requirements.]
- 3D placement [Geometrical Location]
- Absolute placement [*Geometrical Location*]
- *Actor role* [A predefined list (enumeration) of construction process participants, not part of *Client Requirements*.]
- Address [A Project Attribute, but not a Requirement.]
- *Bounding box geometry* [Geometrical representations are not *Requirements*.]
- Extruded solid: arbitrary [Geometrical representation]
- Geometric representation [Geometrical representation]
- Polyline [Geometrical representation]
- Profile: arbitrary [Geometrical representation]
- Relative placement [*Geometrical Location*]
- Site address [A Project Attribute]

Our research will expand this view with several identified *Requirements Objects*, which can be expanded further using PropertySets for additional, project-specific *Requirement Attributes or Descriptions* [Sections 4.2, 6.1 and 6.2]. These *Requirements Objects* can be used as a quick implementation method using PropertySets for the *Requirements Objects* as a whole. This issue and related problems are discussed in detail in Appendix 1, A1. The expanded view is one of the scientific and practical contributions of our research and can serve as a basis for an official extension of the IFC specification [Section 7].

In the *IFC Specification* the identification of objects is based mainly on the Global Unique ID (GUID), which can be problematic for several reasons discussed in detail in Appendix 1, A2. Because of these problems the pilot implementation was based on the idea of using the Description attribute in the SpaceCommon PropertySet to store the RoomID as the link between the *Instantiable Requirements Entities* (*IRqE*) and space objects in the *Design Model*. However, the same concept can be implemented in several ways, and in the next stages of our research alternative linking methods will be discussed in detail with the BLIS group.

5 Pilot Implementation and First Tests

The main conclusions from the two pilot projects described in Section 3 are:

- The *Requirements* are not well documented and managed during the design process, and
- An active link between the *Requirements* and design tools could improve the process.

The first pilot implementation was limited to *Client Requirements* related to the spaces, and the purpose was to test the general idea to link *Requirements* to the objects in the *Design Model*. The points of departure for a technical solution to address these issues in the pilot implementation were:

- The room related *Client Requirements* are defined and documented in the beginning of the process,
- The existing *IFC Specification* contains the necessary elements to link room related *Client Requirements* to the *Building Product Model*,
- The existing *IFC Specification* provides a connection between the rooms and bounding elements, and
- The existing IFC implementations provide a platform, which can be used as a technical basis for the pilot implementation to test the solution.

To explore the possible solutions to manage *Client Requirements*, we used rapid prototyping and implemented some different database structures to find a usable solution to document the room related *Client Requirements* in a structure, which:

- Provides solutions to the problems identified in the LCE project [Sections 3.2.3 and 3.3],
- Supports *Inheritance* of the room type requirements (*Abstract Requirements Entities, ARqE*) to rooms (*Instantiable Requirements Entities, IRqE*) [Figure 8], and
- Enables a link between the *Requirements Model* and the existing *Building Product Model* [Figure 8].

As defined in Section 2, the scope of our research is limited to *Requirements* related to rooms for office and laboratory buildings. On the detail level the *Requirements* for different projects cannot be fully standardized [Section 4.2], but the framework, i.e., the *Requirements Model Specification*, must be project independent. However, the *Requirements Model* for a project can have project-specific *Requirements* [Section 4.5.1].

As described in Sections 1.3, 7 and Appendix 1, the goal of this research is to improve the design process by providing a method to update and manage *Client Requirements* coherently, and give direct access from design software to the *Client Requirements* related to the on-going design task.

5.1 Requirements Database Tests with LCE Project Data

The user interface and database structure of the first pilot implementation were based mainly on the room program documents of Stanford's Lucas Center Expansion. The implementation was made in a MS Access 2002 database. The main criteria for the database structure were to provide a solution to the identified problems:

- Unique IDs for the rooms; i.e., *IRqE* and all the rooms in the *Design, Production and Maintenance Models* referencing it must share the same ID -> unambiguous identification.
- Use of *Abstract Requirements Entities* (*ARqE*) and *Inheritance* -> efficient and easy maintenance and updating of repetitive *Requirements*.
- Use of user-definable enumeration (list of values) instead of free text -> coherent content.
- No default values, which might inadvertently set wrong *Requirements*.
- Functionality to compare area *Requirements* with areas in design documents.
- Functionality to link external documents to the *Requirements Database*, e.g., to include also complex *Requirement Descriptions*, not only short text and numerical *Requirements*.

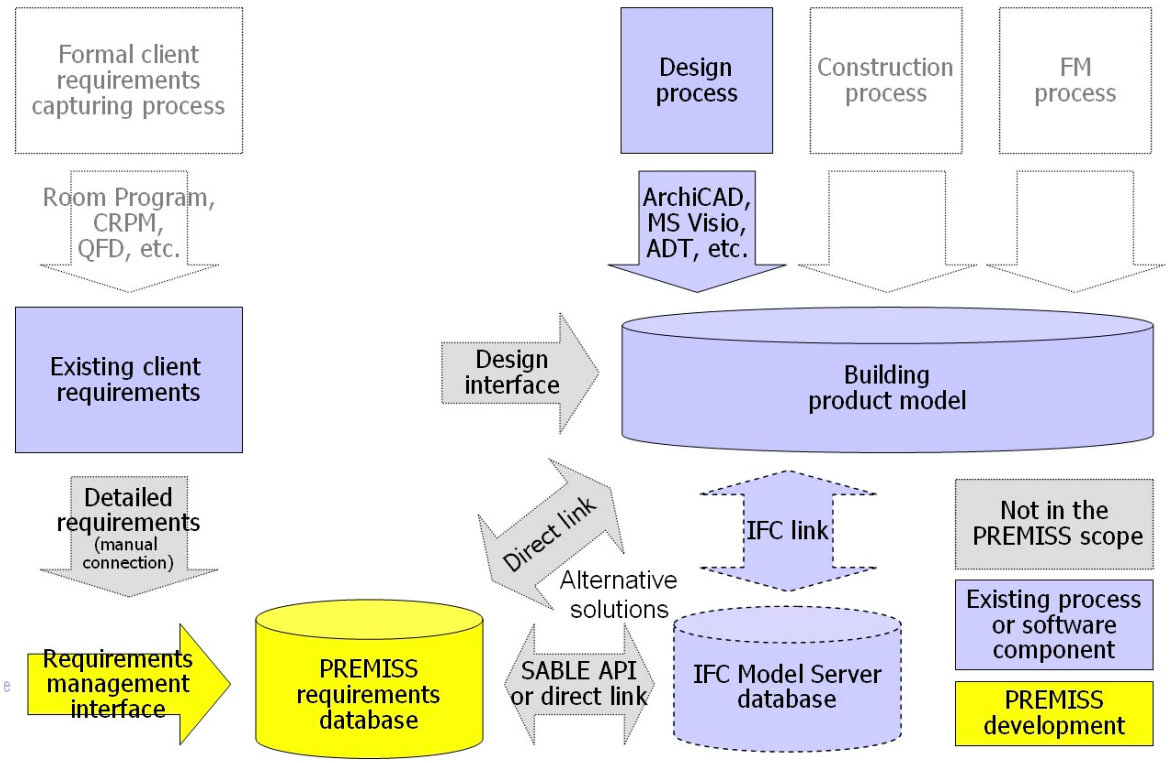

Figure 15: PREMISS pilot implementation and relations to existing solutions

We tested several database structures in the first pilot implementation, mainly to find possible solutions for a structure and user interface, which could support *Inheritance* from room types (*ARqE*) to rooms (*IRqE*) and *Multi-Value Requirements* (*MVR*). The final, and at this phase sufficient, structure for the first test case, Lucas Center Expansion, is presented in Figure 16, which also illustrates the terms "*Multi-Value Requirement*" (MVR) and "*Single-Value Requirement*" (*SVR*).

As introduced in Figure 8 the key idea is the use of two main tables: "RoomTypes" (*ARqE*) and "Rooms" (*IRqE*). "RoomType" is an *Abstract Requirements Entity* and "Room" is an *Instantiable Requirements Entity* in the *Requirements Database*. Both have the same fields and references (*Shared Properties*, *ShP*) with the following exceptions:

- “Room” can reference a “RoomType” to *Inherit* its *Requirements*, but the opposite relation is not possible,
- “Room” can have a relation to department and other room(s), but “RoomType” cannot have these relations (*Instance-specific Properties, ISP*)
- The “Rooms” table contains a “NumberOfInstances” and “RoomName” fields, which are not in the “RoomTypes” table (*ISP*)
- Only “RoomType” has RoomTypeDescription and RoomTypeDoc fields, (*Type-Specific Properties*, *TSP*)

The *Requirements* used in the implementation are only one example of possible *Requirements*, and do not cover all possible building types or use cases. However, they can be categorized in two main groups:

- *Single-Value Requirements* (*SVR*), which can have only one value or reference for each room, like, for example, noise level, maximum number of occupants, maximum temperature, etc.
- *Multi-Value Requirements* (*MVR*), which can have a number of different values or references in each room, like, for example, activities, equipment, windows, etc.

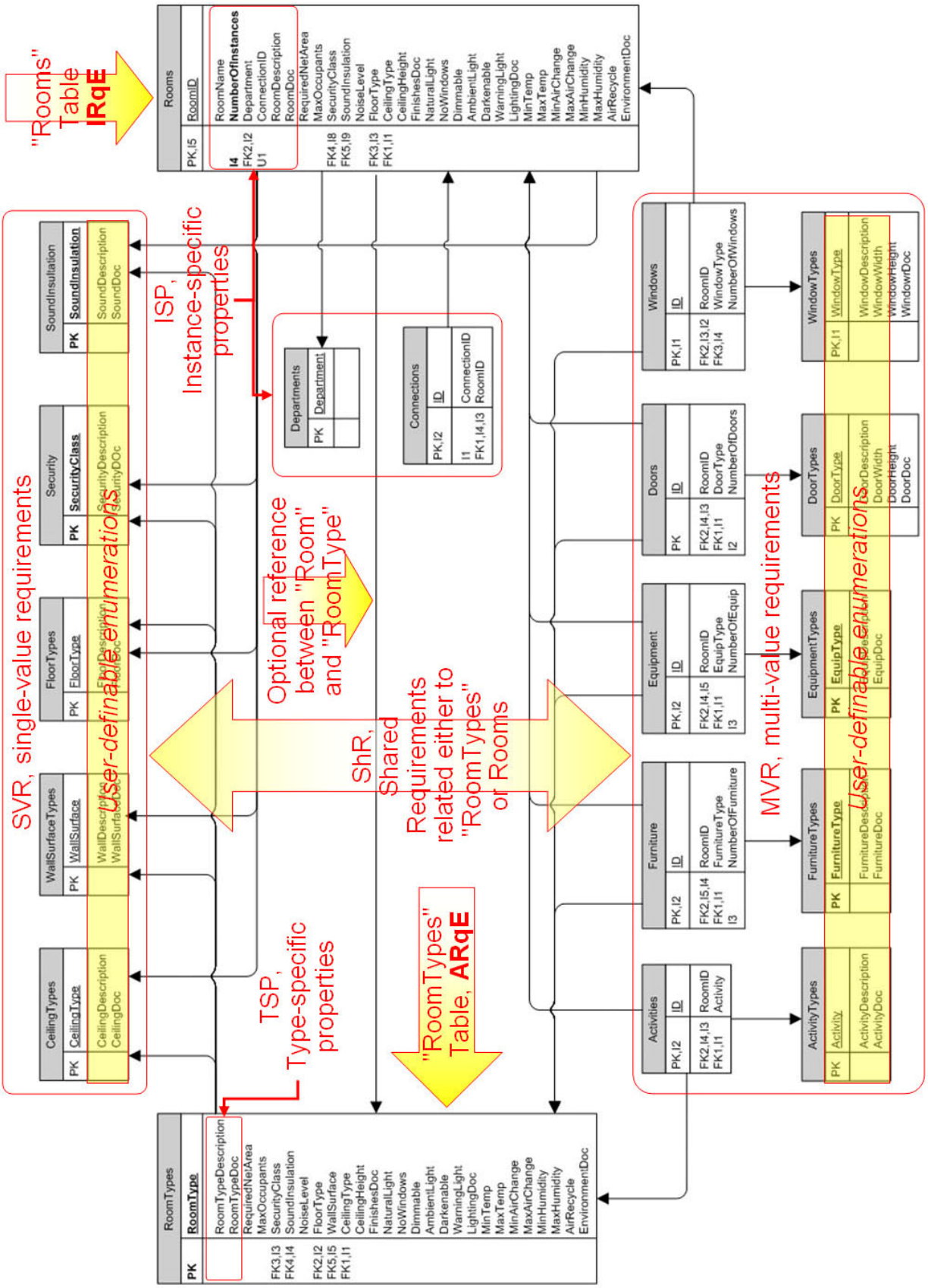

Figure 16: Database structure for the LCE project

For the following reasons this separation of *SVR* and *MVR* types is an important issue, and it defines the basic structure of the *Requirements Database*:

1) If all *Requirements* would be defined and implemented as *SVR* types, the database structure would not allow use of an unlimited number of requirements for each room, which is necessary for some requirement types as described above.

2) If all requirements would be defined and implemented as *MVR* types, the possibility to give multiple values for all properties could cause contradicting *Requirements*, like several different maximum temperatures. In addition, the database structure would be more complicated, which could create performance problems, and the user-interface to the data would be more difficult to understand and slower to use, if all values were in sub-tables.

Figure 17 shows the 1_to_1 and 1_to_many relations in the first pilot database. "RoomType" and "RoomID" are the key links between different tables.

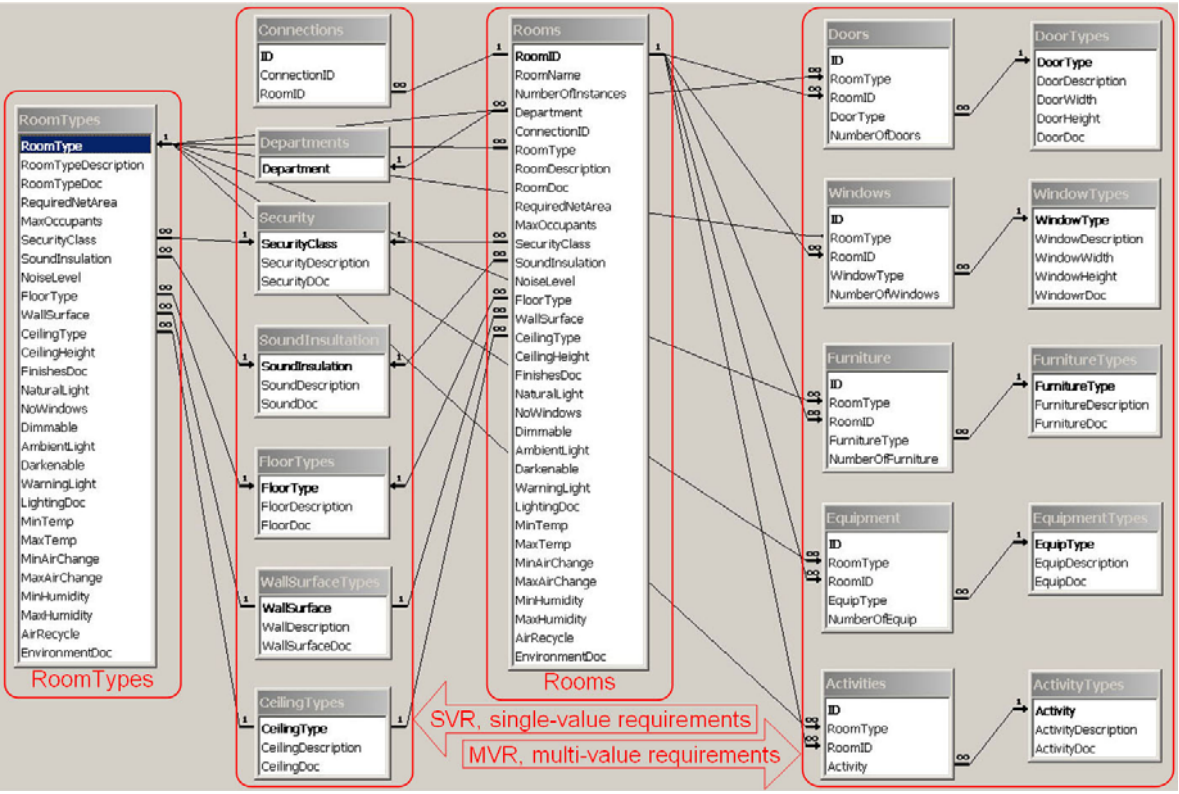

Figure 17: Relations in the LCE database

The proposed structure forces the user to define unique IDs for each *Instantiable Requirements Entities* (in this case Rooms), and all the "free text" *Requirements*, like departments, adjacent rooms, equipment, activities, etc. are based on user-definable lists (enumerations), which prevents slightly different descriptions of the *Requirements* or references to non-existing rooms; all these problems were identified in the LCE project data. The RoomTypes were not used in the LCE project database, because the rooms are not based on any repeating types; all rooms are defined as separate instances.

5.2 Test and Results with ICL Requirements Data

When starting to populate the database with the ICL project data, one observation came up almost immediately; "RequiredNetArea" and "MaxOccupants", which were located in both the "RoomTypes" and "Rooms" tables in the LCE test, would had demanded extensive duplication of similar type definitions with different area and occupant values. Thus, the database structure was changed so that these *Requirements* were removed from the "RoomTypes" table and changed to *Instance-Specific Properties* in the "Rooms" table [Figure 18].

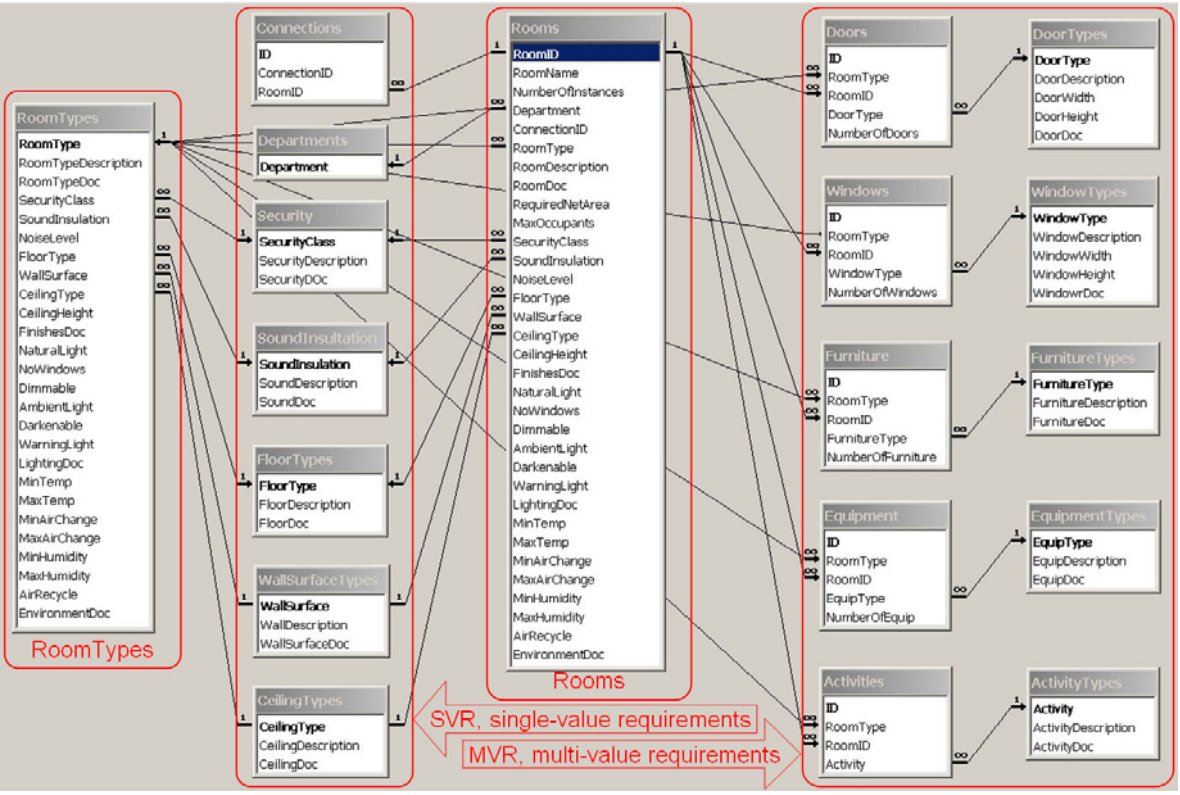

Figure 18: Relations in the ICL database

Otherwise the same database structure, which was used in the LCE project test, also worked for the ICL Headquarters project and enabled recording of the *Requirements* in a usable format; 782 physical room instances are stored in 186 *Instantiable Requirements Entities* (Rooms) based on 51 *Abstract Requirements Entities* (RoomTypes). The maximum number of type references is 16, the average 3.8 and the median 2. The population of the database took about 3 hours, which can be seen as a reasonable effort.

Our conclusion from the rapid prototyping phase is that the final database structure is sufficient proof of concept for further research.

5.3 Connection to a *Building Product Model*

The actual connection of the *Requirements Database* to the *Building Product Model* based design software has not yet been implemented, only a mock-up presenting the idea of a connection from the design application to the Access database was developed. By selecting objects in the design software, e.g., rooms and bounding elements, the user can see all the related *Requirements* in the *Requirements Database* [Figures 19 - 22].

"RoomID" is the connecting element between the *Requirements Database* and the *Building Product Model*. The room instances in the *Building Product Model* are connected directly, but the bounding elements related to the rooms are identified in the *Building Product Model* and the connection to the *Requirements Database* is based on the "RoomID" of identified rooms.

The user interface mock-up in Figures 19 - 22 demonstrates how to access the *Requirements Database* from the design software by adding a *Requirements View* to its user interface. Depending on the use scenario, the modifications of the *Requirements* from the design interface can be either allowed or denied; in some projects the *Client* might delegate the *Requirements Management* to the designers, in some projects it might be the task for the PM or the *Client's* own representative. The selected database approach enables independent access control for the *Requirements Database*.

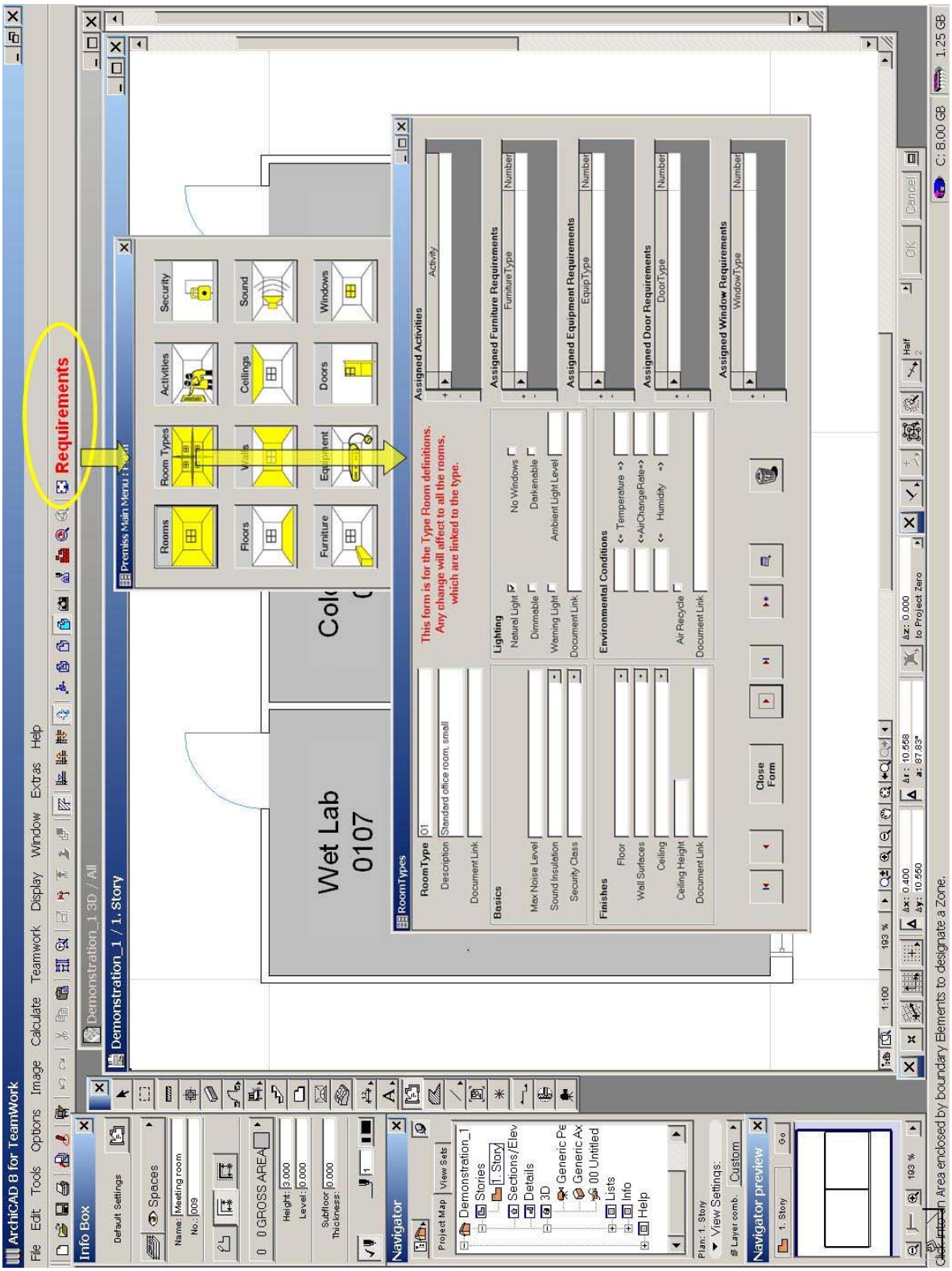

Figure 19: Requirements management UI, mock-up version: Main Interface
Access from design software to definitions in the *Requirements Database*, like Rooms, RoomTypes, Activities, Security, Equipment, etc.

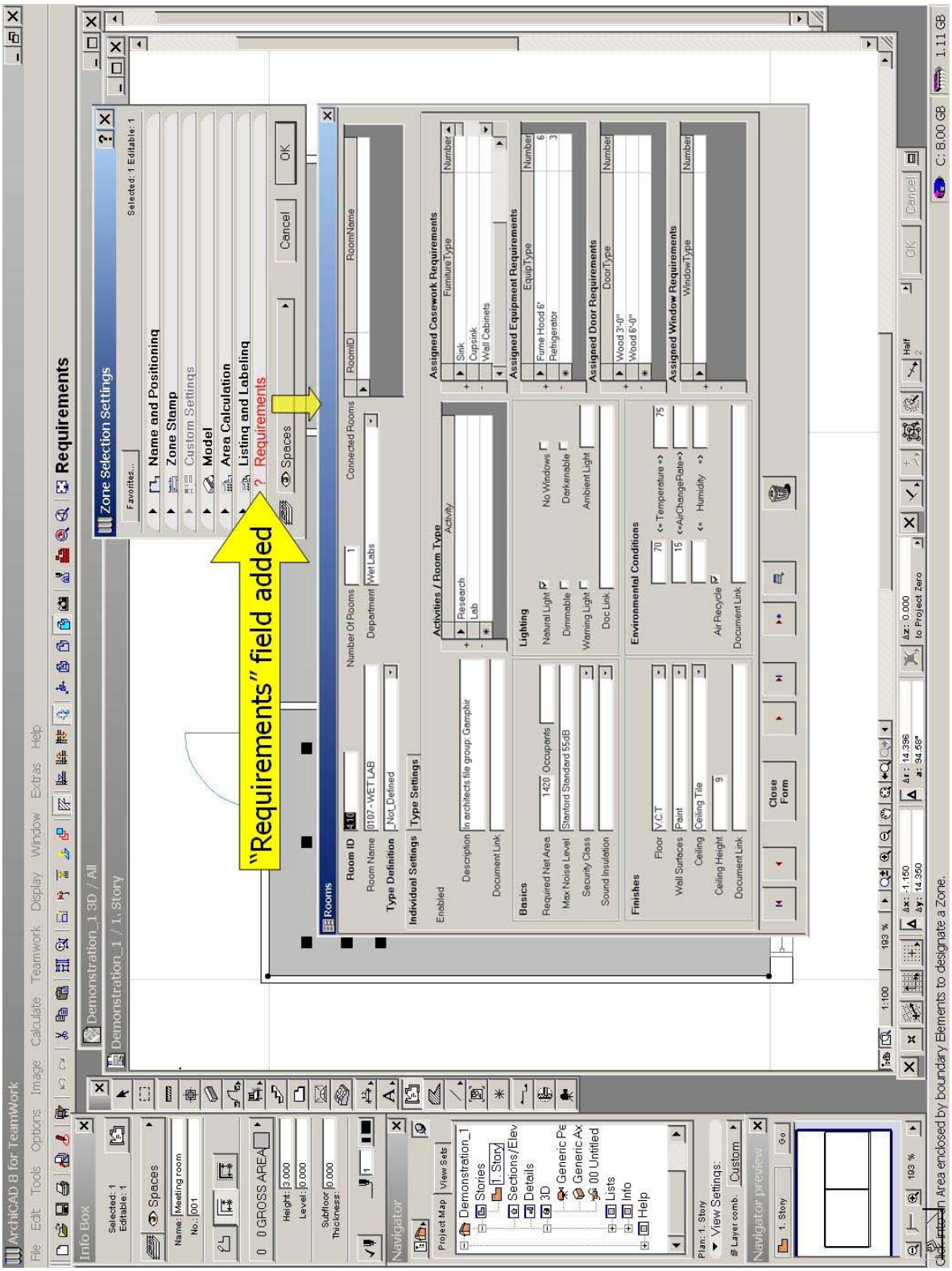

Figure 20: Room requirements UI, mock-up version: Room

By selecting a room and then the *Requirements View*, the software shows all the defined *Requirements* for the selected room.

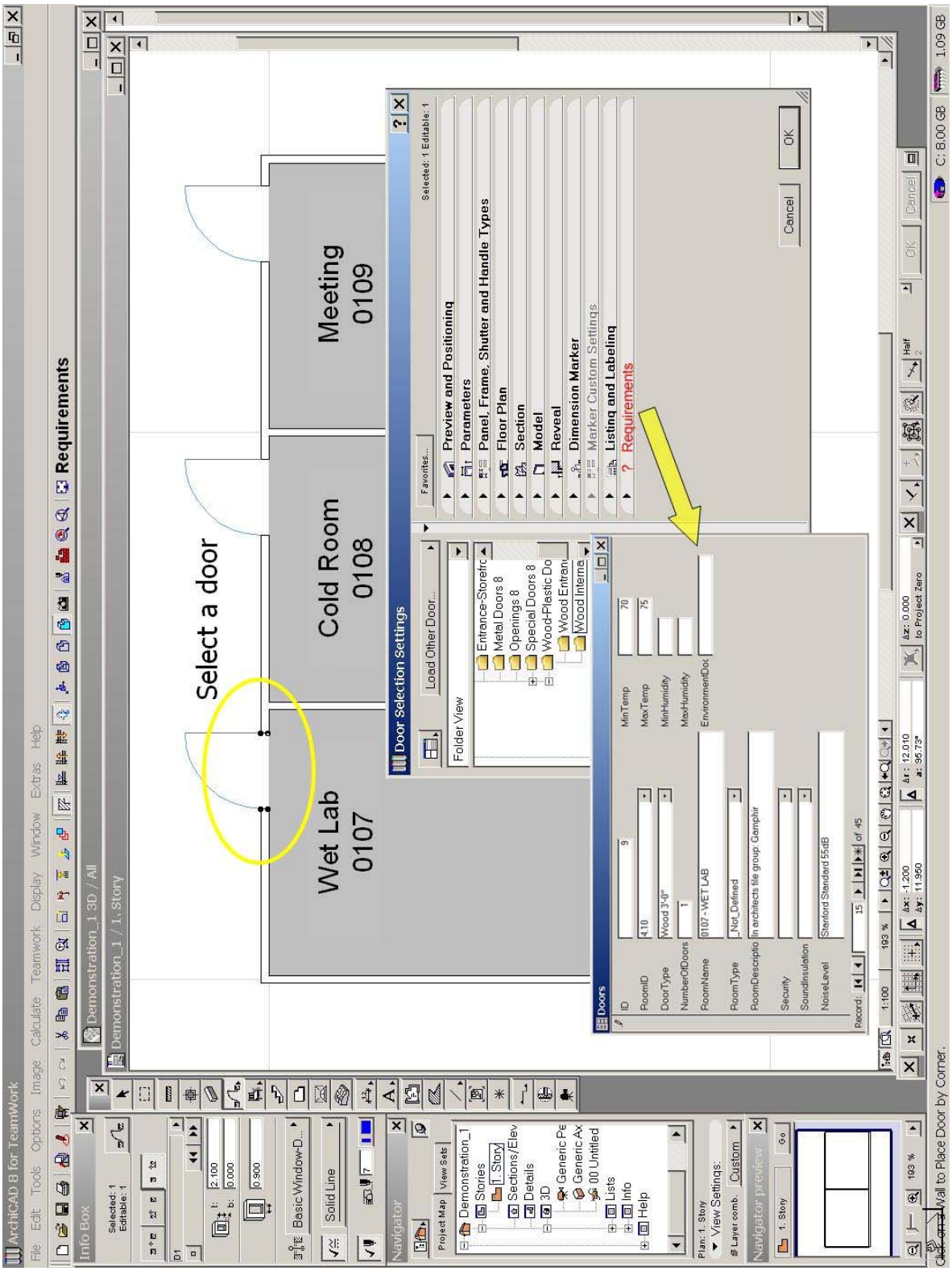

Figure 21: Door requirements UI, mock-up version: Door

By selecting a door and then the *Requirements View*, the software shows all door related *Requirements* [Table 2] from the related room(s).

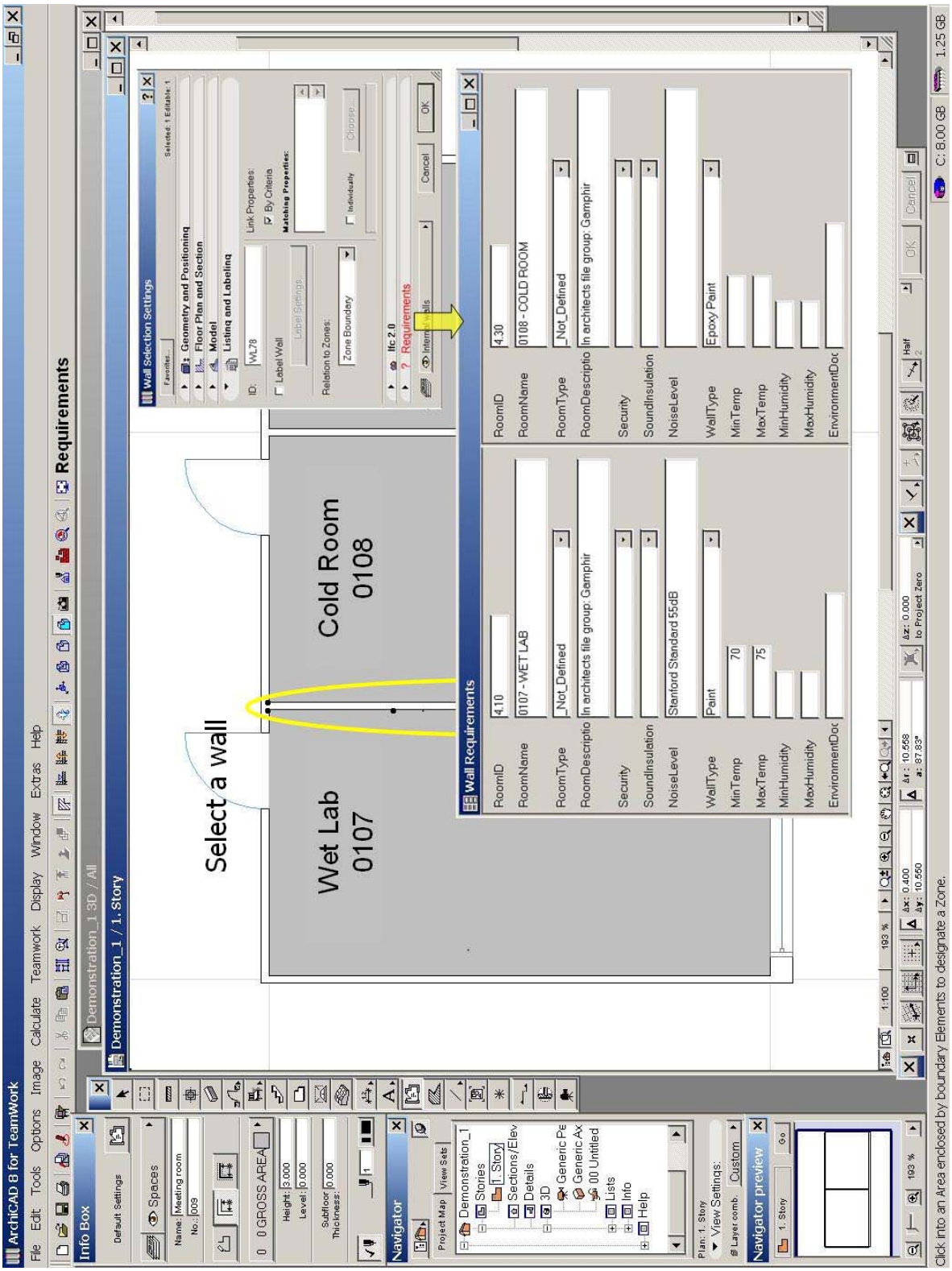

Figure 22: Wall requirements UI, mock-up version: Wall

By selecting a wall and then the *Requirements View*, the software shows all wall related *Requirements* [Table 2] from the related room(s).

6 Research Questions and Method

Based on investigations carried out so far and discussed in Sections 0-5 this work addresses the following research questions:

RQ1: How to formalize a *Requirements Model Specification* for *Client Requirements* in a building project?

The method to answer RQ1 consists of three steps:

- Detailed analysis of *Client Requirements* and principal solutions for *External Requirements* [Section 6.1],
- Development of a *Requirements Model Specification* for these *Requirements* [Section 6.2],
- Validation of the proposed *Requirements Model Specification* in a workshop with industry and academia experts [Section 6.4].

RQ2: How can the relation between this *Requirements Model* and *Design Model* be formalized?

The method to answer RQ2 consists of three steps:

- Development of an interface between the proposed *Requirements Model Specification* and current *IFC Specification* [Section 6.2],
- Definition of an expanded view for implementation of the proposed *Requirements Model Specification* and *IFC Specification* [Section 6.3], and
- Validation of the proposed interface and implementation view in a workshop with industry and academia experts [Section 6.4].

6.1 Detailed Analysis of Client Requirements

The first stage to answer RQ1 is to analyze 3 additional building programs to test:

- The generality of the *Client Requirements* categories, i.e., does the *Requirements Model Specification* have a category for all identified *Requirements*?
- Relevant *External Requirements*, i.e., on which level the *External Requirements* might be linked and managed in a project-specific *Requirements Model*?
- The generality of the types of projects, i.e., is the *Requirements Model Specification* reasonably useful in the projects, which are within the scope of our research?
- Useful level of detail, i.e., which *Requirements* should be in the *Requirements Model Specification*, and which should be project-specific additions?

After the two initial tests we have grouped the room related *Client Requirements* into the preliminary main sets presented in Figure 23 and Table 2. Based on these results we propose the conceptual structure presented in Figure 24. Table 2 also contains information on how often these *Requirements* were used in the first two test projects. Our preliminary analysis of five building programs and about 250 different *Requirements* confirm some of the findings noticeable already on Table 2:

- There are only very few *Requirements*, which are defined in all projects; most *Requirements* are project-specific.
- In typical *Requirements Capturing* systems many of the pre-defined *Requirements* are not used for most projects.

This raises interesting questions about the information content and user interface in *Requirements Management* software linked to the *Design Model*:

- Should the number of pre-defined *Requirements* be very limited? Users would add new *Requirements* based on the project's needs, which would require them to define also the links to the *Design Model* and make the effects on the *Indirect Requirements* explicit?
- Or, should the *Requirements Management* software have a large number of different requirements, which are used only seldom? The resulting complexity of the underlying *Requirements Model Specification* would need to be handled by a hierarchical user interface, for example.

The optimal solution is most probably somewhere between both extremes. We will attempt to answer these questions in our final analysis of building programs. The results will provide the basis for the final content of the *Requirements Model Specification*; identification of *Requirements*, which satisfy the typical *Requirements Documentation* needs for the project types within the scope of our research, and which *Requirements* should be project-specific additions [Sections 4.2 and 6.2].

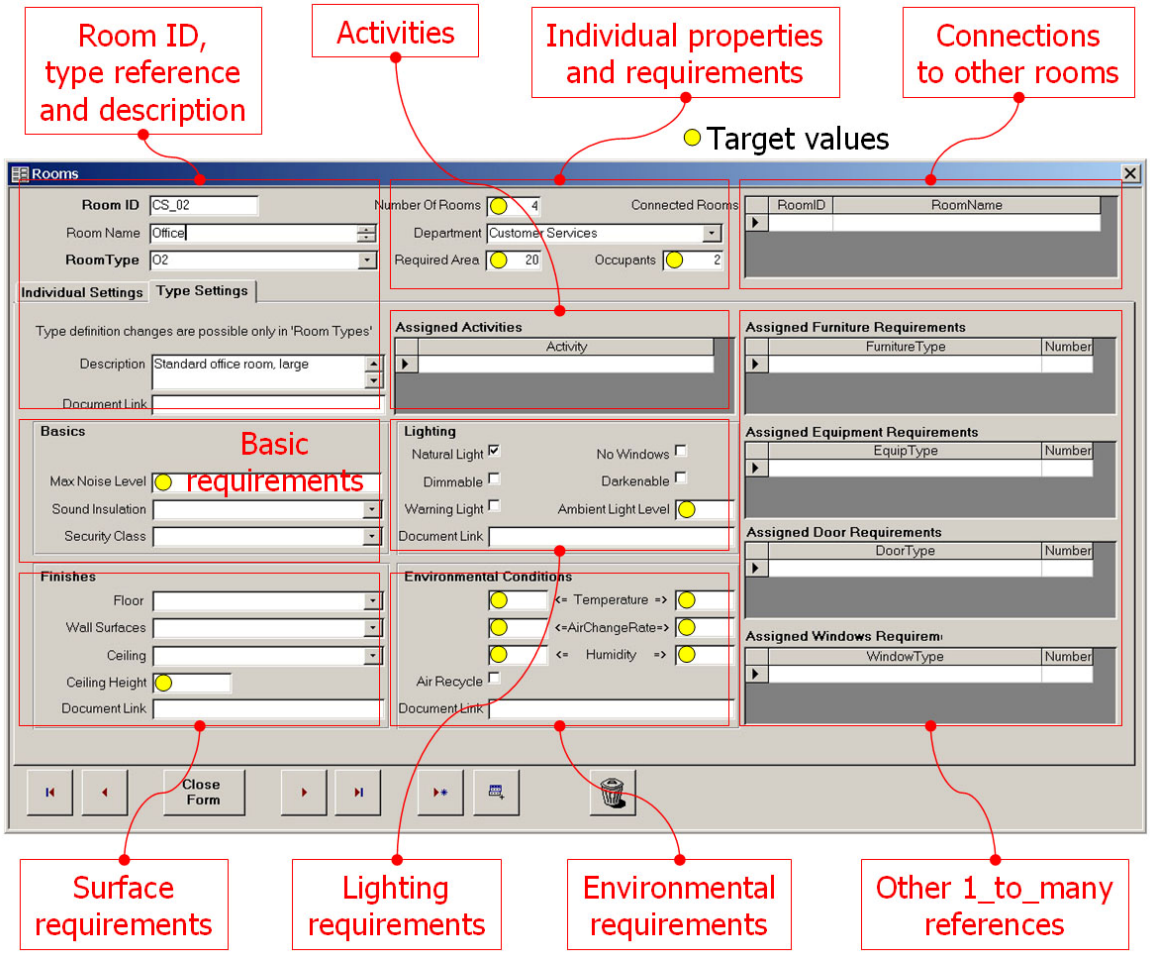

Figure 23: Form showing the requirements groups in the current pilot implementation

Table 2: Database elements, types and usage in test projects

Property	Requirement	Room	RoomType	SVR	MVR	Data Type	Bounding element	Systems	*LCE, PM*	*LCE, PA*	*ICL*	Average
Identification and overall definition												
RoomID		m		x		UID, string	x	x	*62%*	*92%*	*100%*	88%
RoomName		o		x		String			*100%*	*100%*	*100%*	100%
RoomType		o	m	x		UID, string			*46%*		*100%*	73%
RoomDescription		o	o	x		String						
Document		o	o	x		Hyperlink						
Individual properties and requirements												
Department		o		x		Enum			*92%*	*100%*	*100%*	98%
	NumberOfRooms	m		x		Integer			*100%*	*100%*	*100%*	100%
	RequiredArea	o		x		Real				*100%*	*100%*	100%
	MaxOccupants	o		x		Integer		x			*100%*	50%
Basic Properties												
	MaxNoiseLevel	o	o			Integer		x	*38%*			19%
	SoundInsulation	o	o			Enum	x	x				
	SecurityClass	o	o			Enum	x	x				
Connections, activities, furniture, equipment, doors and windows												
	Connections	o			x	Ref to UID			*46%*		*28%*	37%
	AssignedActivities	o	o		x	Enum list		x	*85%*			42%
	Furniture	o	o		x	Enum list			*62%*		*1%*	31%
	Equipment	o	o		x	Enum list		x	*38%*		*3%*	21%
	Doors	o	o		x	Enum list	x	x	*100%*			50%
	Windows	o	o		x	Enum list	x	x				
Finishes												
	Floor	o	o	x		Enum			*92%*			46%
	Walls	o	o	x		Enum			*100%*			50%
	Ceiling	o	o	x		Enum			*100%*			50%
	Ceiling height	o	o	x		Real		x	*92%*			46%
Document		o	o	x		Hyperlink						
Lighting												
	NaturalLight	o	o	x		Yes/No		x	*77%*			38%
	NoWindows	o	o	x		Yes/No	x	x				
	Dimmable	o	o	x		Yes/No		x				
	Darkenable	o	o	x		Yes/No		x				
	WarningLight	o	o	x		Yes/No		x				
	AmbientLightLevel	o	o	x		Real		x				
Document		o	o	x		Hyperlink	x	x				
Environmental Conditions												
	MinTemperature	o	o	x		Real	x	x	*46%*			23%
	MaxTemperature	o	o	x		Real	x	x	*46%*		*2%*	24%
	MinAirChangeRate	o	o	x		Real		x	*92%*			46%
	MaxAirChangeRate	o	o	x		Real		x				
	MinHumidity	o	o	x		Real	x	x				
	MaxHumidity	o	o	x		Real	x	x				
	AirRecycle	o	o	x		Yes/No		x	*62%*			31%
Document		o	o	x		Hyperlink	x	x				

m = mandatory field
o = optional field

6.2 Development of the *Requirements Model Specification*

Development of the final *Requirements Model Specification* will be based on the analysis described in Section 6.1. The status after the first phase of our research is the *Requirements Database* structure presented in Figure 16 and Figure 18. Based on the current observations and this structure we propose the *Conceptual Model* for room related *Client Requirements* presented in Figure 24. The final *Requirements Model Specification* will also include other types of *Requirements* and it will be the main scientific contribution of our research.

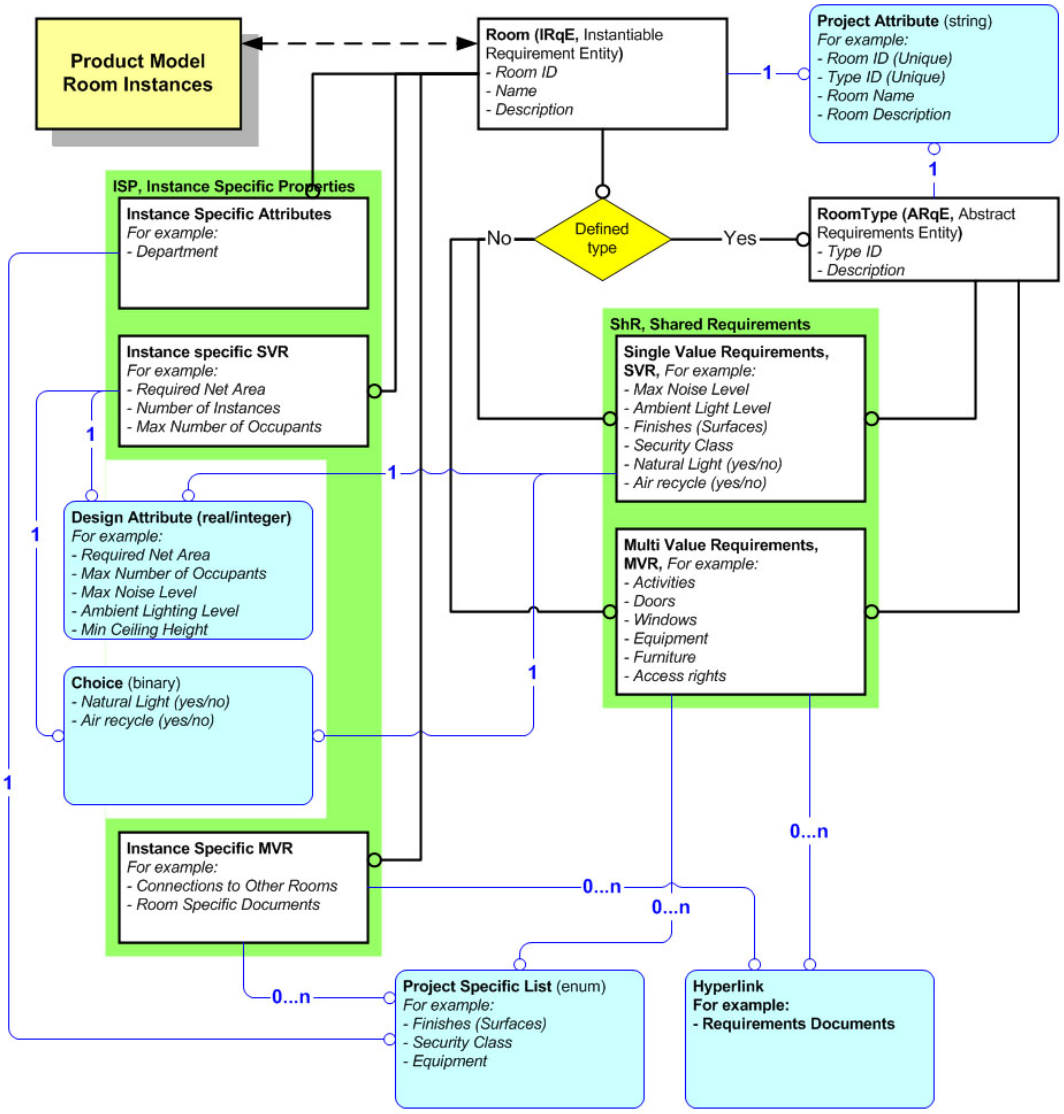

Figure 24: Basic structure of the proposed *Conceptual Model* for room related *Client Requirements*

The main ideas in the *Conceptual Model* for room related *Client Requirements* are:

- Use of RoomType to define *Requirements*, which are identical for several Rooms in the *Requirements Database*
- Separation of the requirements which are always *Instance-Specific Properties* (*ISP*) and which can be *Shared Properties* (*ShP*) defined either at the Room or RoomType level.

- Separation of the *SVRs* and *MVRs*, as described in Section 5.1.
- Flexible framework, which enables additional project-specific requirements attributes [Sections 4.5.1 and Appendix 1, A1]

6.3 Expanded View for the Implementation of the IFC Specification

As explained in Section 4.5, the Building Life Cycle Interoperable Software (BLIS) group has developed the view concept to support IFC based information exchange, and our research will expand the existing "Client Brief / Space Layout -> Architectural Design" view. The content of the view will be based on the *Requirements Model Specification* described in Section 6.2. The expanded view will be the basis for the implementation to link the *Requirements Model* and the *Building Product Model* based software, and it will be one of the scientific and practical contributions of our research [Section 7].

6.4 Expert Workshop to Validate the Requirements Model Specification and Interface to the Building Product Model

The pilot implementation in the first phase of the research demonstrated, that in principle the *Requirements Model* for room related *Client Requirements* is implementable. In the second phase the scope will be extended to cover other types of *Requirements,* and the focus of this work is in the development of the *Requirements Model Specification*. Implementation into software is not in the scope of the project. The validation criteria for the *Requirements Model Specification* are:

- Usefulness: does the *Requirements Model Specification* address relevant factors of the identified problem and could its implementation into a tool improve the current process?
- Generality: does the *Requirements Model Specification* cover a reasonable part of the identified problem?
- Implementability: is the proposed *Requirements Model Specification* possible to implement?

However, there is no objective method to measure or validate the usefulness or generality of a *Conceptual Model*, like our proposed *Requirements Model Specification*. However, we believe that the expansion of the existing *IFC Specification* by developing a *Requirements Model Specification* is a valid scientific and practical contribution. Thus, the validation process for the model will be a workshop for industry and academia experts to obtain a wide expert view on how our *Requirements Model Specification* and its interface to the *Building Product Model* will meet the three validation criteria.

7 Summary of the Anticipated Contributions and Related Future Research

The goal of our research is to develop and test a method to create an active link between *Requirements* and *Building Product Model* based design tools. The scientific contributions of our research are:

- Documentation of the *Requirements Management* problem in the design process based on two case studies,
- Documentation and analysis of the different *Requirements* types based on five case studies,
- Requirements Model Specification based on the analysis,
- Specification for a link between Requirements Objects and objects in the Design, Production and Maintenance Models,
- Specification for the aggregation of *Indirect Requirements* from the *Direct Requirements*,
- Extended view “Room Program -> Architectural Design” for the implementation of *Requirements Model Specification* and *IFC specification*.

Our main contribution from a practical perspective is that the *Requirements Model Specification* enables the development of *Requirements Management* software, which can link the *Requirements* and design solutions and improve *Requirements Management* during the design and construction process. Our view is that our research is also creating the basis for many interesting future research topics, like, for example:

Different building types and process phases: As defined in Section 2, the scope of our research covers a few building types only. Our intuition is that the same *Conceptual Model* could be applied to most buildings, but because of the different *Requirements* the database and user interface implementation might be different. In addition, our research covers only a short period of the process, design, the use of the *Requirements Model* in other parts of the process, like, construction, FM, etc., is not covered in detail, though the same principles are possibly applicable. All these are possible topics for future research.

Technical systems and other design areas: As described in Section 4.3, the designers’ role in defining detailed *Requirements* for technical systems is more dominant than in architectural design, and the research in this area provides another view on *Requirements Management* in the AEC industry. However, there is currently no link between the room related *Client Requirements* and technical *Requirements* for systems. Our research is building a link between *Requirements* and the *Design Model* and identifies some connections to technical systems, but a formal link between the technical system *Requirements* and *Design, Production and Maintenance Models* will need further research, as do links in other design areas, like structural engineering.

Requirements history: One interesting related research area is the *Requirements* history; how the *Project Requirements* evolved during the process. Our research proposes a *Requirements Model Specification*. It will provide a conceptual basis to store all the *Requirements Changes* during the process in the database. How to implement such a historic perspective of *Requirements Management* in detail and which functionalities the user interface would need, are interesting areas for further development.

Verification: Some of the *Client Requirements* are verbal descriptions and only human interpretable (*Requirement Descriptions*), but some have an exact content (*Requirement Attributes* in our *Requirements Model Specification*. The possibility to use these "exact" *Requirements Attributes* for automated verification, i.e., how well the design meets the *Requirements*, is a potential usage of the *Requirements Model*. Verification of the "fuzzy" *Requirement Descriptions* must include designers' interaction, but the designer's or project manager's confirmation that the *Requirements* are met, could be part of the database and serve as a formal project history.

Appendix 1: Some Implementation Issues Related to the *IFC Specification*

The following issues are not crucial for our research; the implementation of a Requirements Management software and the link between Requirements and Design Objects can be done in several methods. However, the issues which came up in the rapid prototyping phase are documented in this appendix as a guideline for the future implementation.

A1 PropertySet Mechanism

In the *IFC Specification* 2x2 the Pset_SpaceProgramCommon has been expanded to cover the following attributes: Location (*Required Location*), FunctionRequirement, SecurityRequirement, PrivacyRequirement, LightingRequirement, FFETypeRequirement, EmployeeType, OccupancyType and OccupancyNumber. StandardRequiredArea is part of the IfcSpaceProgram object. The detailed analysis of the current requirements part of the IFC Specification will part of the phase 2 of our research. However, there are some problems related to the use of PropertySets and IfcSpaceProgramObject for Requirements Management:

- The only level in the *Design Model* where IfcSpaceProgramObject can be linked are the spaces, Requirements related to other levels of detail, like, for example, Project, Building, Site, etc., can not utilize the Pset_SpaceProgramCommon PropertySet.
- Each Pset_SpaceProgramCommon PropertySet is related to one IfcSpaceProgramObject in the *Building Product Model*. This means that the *Requirements*, which are shared with several spaces, must be multiplied to all instances. This can cause serious problems for *Requirements Management* when the *Requirements* evolve and must be updated.

We believe that all repeated information should be stored in one instance in the *Requirements Model* if possible. Thus, the *Requirements* should be separate objects, which can be linked to each other in the *Requirements Model* and related objects in the *Design, Production and Maintenance Models*.

A2 Object Identification and Link between Different Models

As documented in Section 1.4, 4.5.1 and 5.3, one RoomID in the *Requirements Database* can refer to multiple instances in the *Building Product Model*. This means that the link between the objects in the Requirements Model and Design Model must allow multiple references. In the pilot implementation this was done by storing the RoomID of the *Instantiable Requirements Entity* (*IRqE*) in the Description attribute of the SpaceCommon PropertySet in the linked space objects in the *Design Model*.

However, the issue of object identification is a wider problem in information sharing in the *Building Product Model* environment. The usual method to identify objects is based on Global Unique ID (GUID), which is a perfect solution if all the software used in the project can use it similarly, and if the designers do not delete and add the objects in the *Design Model* if they could make the changes by editing existing objects. Unfortunately this is not the case in many projects, and the GUID based identification

is a very woundable method to link objects in different *Models*. When ever an object is replaced by a new object the link is broken, and unfortunately some software change the GUIDs when the model data is exchanged even if the objects are not changed in the original *Model*.

The end-user behavior can be influenced mainly by education, but if the linking method is based on user-definable, understandable mechanism instead of highly abstract software generated GUID, it is easier for the end-users to understand and remember the importance of correct editing methods when working with the *Models*.

If the identification and link is based on values stored in some attributes of the Models, the software products will not change the information as they can change the GUIDs in the data exchange. However, the requirement for this method is that all the software used in a project include the necessary attributes to store and handle the link information.

As our conclusion we propose that the link between the *Requirements and Design Models* should be based on some other identification methods than the GUID.

A3 Automated Generation of Room Objects from the Room Program

Linking the *Requirements Objects* with the *Design Objects* can be an extensive task depending on the size of the *Models*. The error possibilities in such tasks are also relatively high. The possibility to generate the room objects automatically from the room program would solve both problems. Technically the task is not difficult; it can be based on the required area in the Requirements Model and some parameters defining the generated shape and location of the rooms. At least two such applications already exist; both are using MS Excel based room program. The first application, KIVI, was implemented by the first author of this report 1992-1994, and it was based on the extended data possibilities of the AutoCAD blocks and polyline objects. The first project were the application was used was the ICL Headquarters [Section 3.1]. The second application, Space Layout Editor, was implemented by Jiri Hietanen in 2000 [34], and it is based on MS Visio and IFC data. Both applications generate initial room objects into the design software where they can be edited by the designer.

One important part of the solution is that the automated generation of the room objects also enables automated linkage between the *Requirements and Design Models*. This is also related to the issues discussed in the previous section (A3); how to identify the objects and maintain the link, although the automated generation can use either GUIDs or user-definable attributes as the link information.

A4 Model Server Technology

As described in Section 4.4, the main requirement for the pilot implementation was identifiable space objects, which can be linked to the *Requirements Objects*, and recognition of the bounding elements related to the space objects. The linkage between *Requirement Objects* and objects in the *Design, Production and Maintenance Models* can be done using several methods. Though the implementation is not in the scope of our research, this section gives a brief overview to the latest IFC implementations to explain the technical options for implementation.

IFC file exchange is now supported by several commercial software vendors [*BLIS 2003* [35] *and IAI ISG 2003* [36]]. However, IFC based file exchange is an insufficient solution for real projects [*Kam and Fischer, 2002* [37]]. The key problems are:

- The different information content in different software -> It is impossible to maintain all the data when transferring the *Building Product Model* between different software applications, and
- The lack of partial model exchange -> This causes two main problems:
- The *Building Product Models* are large, which makes the file exchange of the whole model time-consuming. However, usually only a small part of the model has changed and transferring the whole model would not be needed, if partial exchange was available.
- Versioning and controlling user rights are practically impossible.

Also the complexity of the *IFC Specification* is one bottleneck for implementation, and easier access to the model data using simple queries would improve the usability of the *IFC Specification*. Thus, several projects have been developing IFC Model Servers since 2001 [*IMSvr 2002* [38]*, WebSTEP 2002* [39]*, and EPM 2003* [40]]. All Model Servers provide partial model exchange and simple query access to the model using standard technologies like XML, SOAP and STEP [*Adachi, 2002* [41]*, Hemiö, 2002* [42]].

However, from the implementation viewpoint the different application interfaces to different Model Servers are a problem, because it either limits the use to one server or requires implementation of several application interfaces. Thus, a standardized application interface is needed, and the SABLE project is currently working to develop it based on SOAP [*SABLE 2002* [43], Figure 25 and Figure 26].

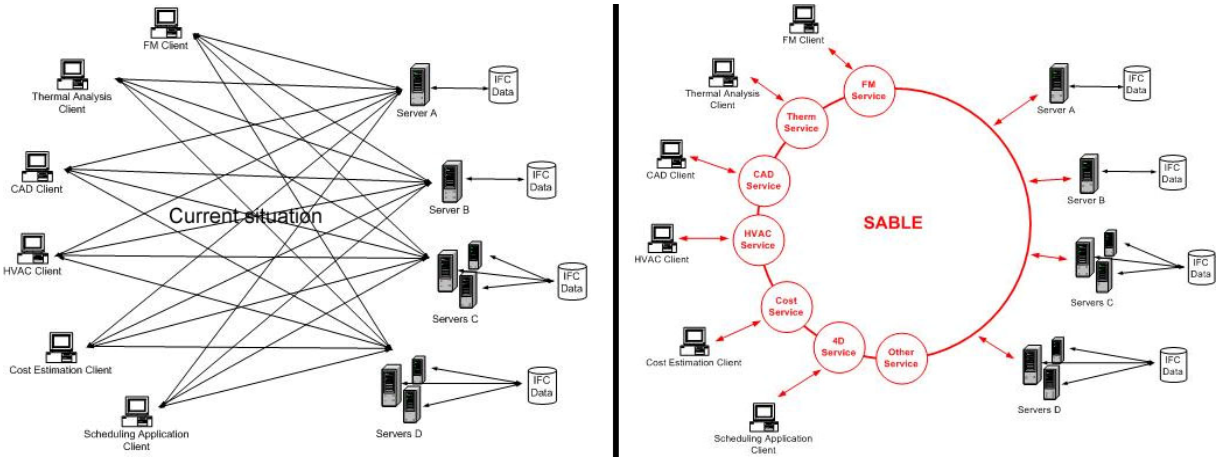

Figure 25: SABLE: advantage of a standardized interface approach *[© BLIS & SABLE]*

The best technical solution to implement the interface between the *Requirements Model* and the *Building Product Model* would be to use a standardized API, like the SABLE interface. A standardized API would make the implementation easier and provide connections to several software products, including other design software if further research projects proposed in Section 7 or commercial software development use the same structures. The standardization of the software interfaces as well as the standardization of data structures is crucial for the development and use of interoper-

able software. However, as described in Section 4.4, this is not a crucial issue for our research.

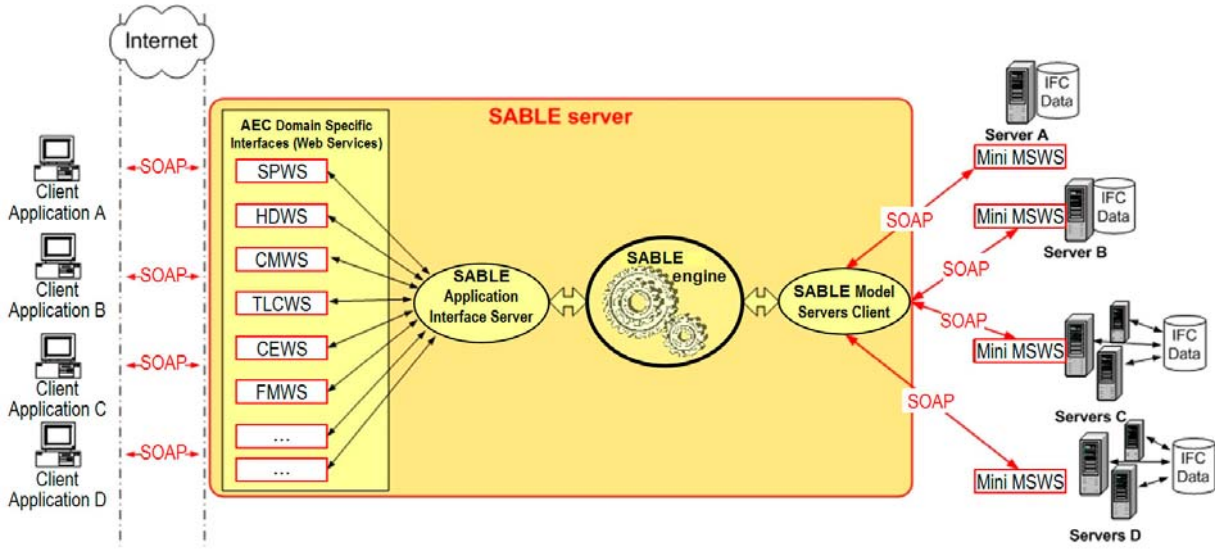

Figure 26: SABLE architecture *[© BLIS & SABLE]*

The proposed *Requirements Model Specification* can be implemented in Product Model Server environment in two different ways [Figure 27]:

- Option #1: The *Requirements Model* is stored in a separate database which has its own user interface (UI), and the connection from the *Requirements Model* to the *Building Product Model* is through a Domain Specific API. In this option, the *Requirements Management* software is a "stand-alone" application and the connection to the Model Server is needed only when the connection between *Design Model* and *Requirements Model* is used. However, this means that the *Requirements Management* UI in the design software must be able to connect to the *Requirements Database*, when the user wants to see the *Requirements* related to his design tasks.
- Option #2: The *Requirements Model* is stored in a Model Server database. In this option the *Requirements Management* software's UI communicates with the *Requirements Database* through the Domain Specific API in the same way as design software's *Requirements Management* UI. The benefit of this approach is that all the project information is stored and accessible on the same server and using same methods. Thus, option #2 is significantly better solution to the *Requirements Management* problems discussed in this report than option #1, where the connection between *Requirements and Design Models* is less integrated. However, even option #1 would a clear improvement to the current situation, where the connection is totally missing, and useful solution if the integrated Model Server platform and standardized API is not available.

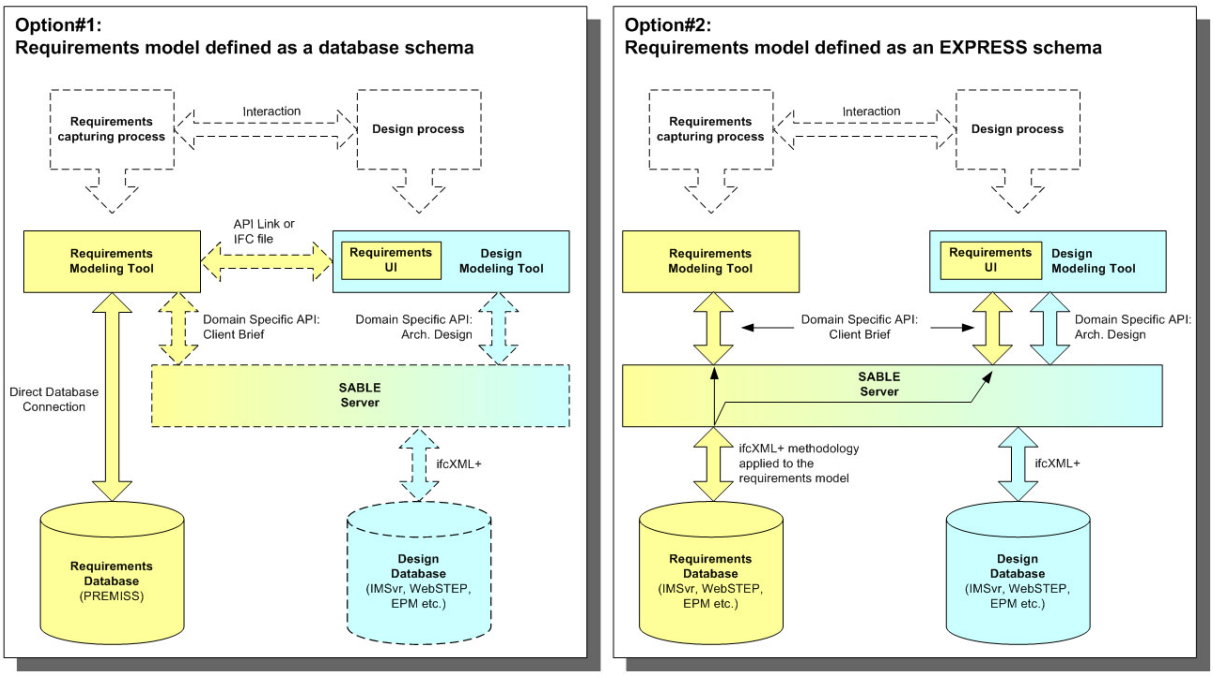

Figure 27: PREMISS and SABLE connection options *[© Jiri Hietanen, 2003]*

Appendix 2: Terminology

The following list defines the key terms and abbreviations used in this report. When used in the defined meaning, these terms are formatted in *Underlined Italic* in this document.

Abstract Requirements Entity **(*ARqE*):** A type definition, like room types, in the *Requirements Model*. *ARqE*s do not have direct links to the objects in the *Design Model*, they relate only to the *Instantiable Requirements Entities* (*IRqE*). One *ARqE* can be linked to several *IRqE*s, c.f. Figure 8.

Building Product Model: A computer-interpretable description of a building structured according to some *Building Product Model Specification*, like, for example, the *IFC Specification*.

Client: Building owner and end-user(s) of the building, who participate in the *Requirements Capturing* and/or *Requirements Management* by defining *Requirements*. Other project stakeholders, like the community, are assumed to communicate with the project through the *Client*(s) and *Project Team*.

Client Requirement **(*CR*):** A detailed *Requirement*, which defines some *Client* need, provides useful information for design decisions, and can be linked to object(s) in the *Design, Production and Maintenance Models* on some level, e.g., project, site, building, space, envelope, etc. *CRs* can be either *Requirement Attributes or Requirement Descriptions.* The first pilot implementation of this research discussed in Section 5 focused on *CRs*, which have connection to spaces.

Conceptual Model: In this report the term "*Conceptual Model*" is used for model structures, which are rather illustrations of the principle than actual specifications. C.f. *Model* and *Specification*.

***Design Model (DM)*:** A *Building Product Model* representing a design solution. Several *Design Models* can be linked to one *Requirements Model*. C.f. *Maintenance, Production and Requirements Models*, and Figure 7.

***Direct Requirement (DR)*:** A *Requirement* set and managed by the *Client* or his appointed representative in the *Project Team*; for example room properties like required area, needed equipment or allowed minimum and maximum temperatures. *DRs* can be either *Requirement Attributes or Requirement Descriptions*.

External Requirement **(*ER*):** A *Requirement* set for a building project by external sources, like, for example, building codes, local regulations, permitting authorities, neighbors, etc. *ER*s can be either *Requirement Attributes or Requirement Descriptions.*

Geometrical Location: The location of a building element in the *Design, Production and Production Models*. These locations can be defined in different coordinate systems either as an absolute location or as a relative location to another element. They specify the exact place of the element in the model. C.f. *Required Location.*

IFC Specification*:* Industry Foundation Classes, *Building Product Model Specification* defined by the International Alliance for Interoperability.

Indirect Requirement (IR): A *Requirement* for the objects on the same or lower level of detail in the *Design Model* derived from or related to some *Direct Requirement*. For example, wall properties, which fulfill some *Requirements* set for a room, like, for example, sound or thermal insulation. *IR*s can be *Requirement Attributes* or *Requirement Descriptions*.

Inheritance: In this report the term "*Inheritance*" is used in a wider meaning than in Object Orientated programming. When an *Instantiable Requirements Entity* (*IRqE*, for example, a room), inherits the *Requirements* from an *Abstract Requirements Entity* (*ARqE*, for example, RoomType) it is not a sub-class of the *ARqE*, but it will have all the requirements defined in the *ARqE*.

Instance-Specific Property (ISP): A Requirement or Project Attributes, which relate only to the Instantiable Requirements Entities (IRqE) in the Requirements Model, c.f. ShP, TSP, IRqE and ARqE.

Instantiable Requirements Entity (***IRqE***)**:** A *Requirements* instance, like requirements for a room, in the *Requirements Model*. *IRqE*s have direct links to the objects in the *Design Model*. One *IRqE* can be linked to several objects in the *Design Model*, cf. *ARqE* and Figure 8.

Location: In this report location has two different meanings. *Required Location* and *Geometrical Location*.

Maintenance Model: A *Building Product Model* representing the as-built building. C.f. *Design, Production, Maintenance and Requirements Models*, and Figure 7.

Model: An instantiated representation of an entity based on some *Model Specification*; for example, the *Requirements Model* contains the *Requirements* of a project structured according to the *Requirements Model Specification*. Likewise the *Design Model* contains project's design objects structured according to some *Building Product Model Specification*, for example, the *IFC Specification*.

Model Specification: Formal definition of a model structure, like Requirements Model Specification, Building Product Model Specification, etc. C.f. Model.

Multi-Value Requirement (***MVR***)**:** A *Requirement*, which can have several different values or references for one *Instantiable Requirements Entity* (IRqE), like activities, equipment, windows, etc. for a room, cf. *Single-Value Requirements*.

Production Model: A Building Product Model representing a production solution. Several Production Models can be linked to one Requirements Model and/or Design Model. C.f. Design, Maintenance and Requirements Models, and Figure 7.

Project Attribute (PA): In the PREMISS-model the *Project Attributes* are attributes, which do not define actual *Requirements*, but serve as identifiers, names or other information of the *Requirements Objects*, like project name, ID and name of a room, etc. C.f. *Requirement Attribute.*

Project Requirements: Requirements for a specific project; usually created in Requirements Capturing and updated in Requirements Management processes.

Project Team: Group of people actively producing, managing and using information in the design and construction process, including, typically, project managers, architects, and engineers.

Property: Attribute or feature of an object in *Design, Production and Maintenance Models*, like area of a room, thermal insulation of a window, color of a wall, etc. A single property or a group of properties can meet one or more *Requirements* in the *Requirements Model*.

Required Location: Defines the client need for a location of a space or group of spaces, usually in relation to other adjacent spaces or a specific story, part of the *Requirements Model*. C.f. *Geometrical Location*.

Requirement Attribute: *Requirement* having a specific numeric *Target Value*, which can be verified from the design model by calculations, simulation results or other computational methods, like required area, minimum or maximum temperature, ceiling height, connections to other rooms, maximum noise level, etc. C.f. *Requirement Description* and *Project Attribute.*

Requirement Description: *Requirement* defined by a verbal description, and thus needing human interpretation, c.f. *Requirement Attribute.*

Requirement: Statement of quality or desired property of the building or its parts. The possible *Requirements* depend on building type and *Client* needs, and, as shown by this research, the list can not be standardized. Thus, the *Requirements Model Specification* must be a flexible framework which enables additional project-specific *Requirements*.

Requirements Capturing: The process defining original *Project Requirements* before the design process, c.f. *Requirements Management*.

Requirements Changes: Changes made to the *Project Requirements* in the *Requirements Management* during the design, construction or maintenance process after the *Requirements Capturing* phase.

Requirements Database: *Requirements* organized into a database structure. In this report the formatted term "*Requirements Database*" refers specifically to the rapid prototype (PREMISS pilot implementation).

Requirements Documentation: All documents containing any portion of *Project Requirements*, like, for example, building program, environmental goals, meeting minutes, etc.

Requirements Information: The information content of the Requirements Documentation.

Requirements Knowledge: The explicit information in the *Requirements Documentation* and the implicit and tacit knowledge of the *Project Requirements* in the *Project Team*.

Requirements Management: The process to update the project Requirements after the Requirements Capturing process.

Requirements Model: A *Model* representing the requirements of a specific project based on the *Requirements Model Specification*. C.f. *Maintenance, Production and Requirements Models*, and Figure 7.

Requirements Object: An objectified set of *Requirements* in the *Requirements Model*. One *Requirements Object* can be linked to several objects in the *Design Model*, and can be expanded using the PropertySet mechanism. C.f. Appendix 1, A1

Requirements View: A functionality proposed in our report to show the *Requirements* linked to a specific object in the *Design Model* in design software. C.f. Section 5.3.

Shared Properties (ShP): Requirements or Project Attributes, which can relate either to the Abstract Requirements Entities (ARqE) or to the Instantiable Requirements Entities (IRqE) in the Requirements Model, c.f. ISP, TSP.

Single-Value Requirements (***SVR***)**:** *Requirements*, which can have only one value or reference for one *Instantiable Requirements Entity (IRqE)*, like, for example, noise level, maximum number of occupants, maximum temperature, etc. for a room, cf. *Multi-Value Requirements*.

Specification: C.f. Model Specification.

Target Values: "Specific values that define the solution space for design attributes (e.g. 5,000 m2 for gross floor area or 10% of gross floor area as circulation space)" [*Kamara et al, 2003*]. In the PREMISS model all *Requirement Attributes* have *Target Values*. In the pilot implementation, the attributes, for which the data type is integer or real, are *Target Values* [*Table 2*].

Type-specific Properties (TSP): Requirements or Project Attributes, which relate only to the Abstract Requirements Entities (ARqE) in the Requirements Model, c.f. ISP, ShP, IRqE and ARqE.

Appendix 3: Authors

Arto Kiviniemi, Chief Research Scientist at VTT (Technical Research Centre of Finland). Arto earned a Master of Science in Architecture in Helsinki University of Technology, and he worked as a designer from 1972 to 1996, winning 12 first prizes in architectural competitions as the main assistant of Arto Sipinen. Arto has been involved in R&D of Construction Information Technology since 1986, and 1990-1996 he also taught CAD at Tampere University of Technology. In 1997 Tekes (Technology Agency of Finland) invited him to become the Program Manager for a large Finnish R&D program; "Vera - Information Networking in the Construction Process 1997-2002" consisting of over 160 research projects. Arto has been actively involved also in the International Alliance for Interoperability (IAI); Chairman of the International Council and Executive Committee 1998-2000, Deputy Chairman 2000-2002, and Chairman of the IAI Nordic Chapter 1996-1998 and 2000-2002. He was also a co-founder and the first Chairman of the Building Lifecycle Interoperable Software (BLIS) association in 2000 Arto has presented over 30 keynote and invited lectures in international seminars and conferences since 1994. Currently he is also the Chairman of the Steering Committee of the Centre for Research and Innovation in Salford University (UK), a member of the Scientific Committee of BuildingEnvelopes.org of the Center for Design Informatics at Harvard University (USA) and member of scientific or organizing committees of several international conferences. In January 2003 he started his Ph.D. studies in Stanford University at the Center for Integrated Facility Engineering (CIFE), and this working paper is based on his thesis proposal, which was accepted on December 15, 2003.

Martin Fischer, Ph.D., Associate Professor of Civil and Environmental Engineering and (by courtesy) Computer Science; Director, of Stanford Center for Integrated Facility Engineering (CIFE). His research interests include 4D modeling tools and interfaces, linking design and construction with construction method models, and prototyping products and processes for concurrent engineering. Teaching activities focus on project management techniques and the design of construction processes.

Vladimir Bazjanac, Ph.D., Staff Scientist, Lawrence Berkeley National Laboratory, University of California. Dr. Bazjanac is the U.S. government technical representative in the International Alliance for Interoperability (IAI). He has been involved with the IAI since its beginning in 1994 and is the leader of the IAI Technical Advisory Group, and a member in the IAI International Management Committee and in the IAI Software Implementation Support Group. Dr. Bazjanac is a former long-time faculty member in the Department of Architecture, University of California at Berkeley. He has won national architectural design, industry and scientific awards; has published over 70 articles and papers related to the AEC/FM industry on design theory, simulation and information technology, and has lectured at major universities and professional societies in both Americas, Europe, Australia and Asia.

Boyd Paulson, Ph.D., holds the endowed Charles H. Leavell Professorship of Engineering in Stanford University's Graduate Program in Construction Engineering and Management. He is the author or co-author of 2 books and over 100 papers. Dr. Paulson's research and teaching interests are primarily in the design and construction of affordable housing. Related interests include computer applications in construction,

equipment and methods for construction field operations, and international construction. He has had numerous research projects sponsored by the National Science Foundation, the U.S. Department of Transportation, and others. Dr. Paulson's past professional activities include Chairman of ASCE's Committee on Professional Construction Management and the ASCE Task Committee on Computer Applications in Construction, Vice Chairman of ASCE's Construction Research Council, and Chairman of the ASCE Construction Division Executive Committee.

References

[*Adachi, 2002*] Adachi, Yoshinobu: Introduction of IFC Model Server, 2002, website, last accessed on September 29th, 2004: http://cic.vtt.fi/vera/Seminaarit/2002.04.24_IAI_Summit/Adachi.pdf

[*Best and De Valence, 1999*] Best, Rick and De Valence, Gerard: Building in value, page 8, Arnold 1999, ISBN 0 340 74160 0

[*BLIS 2003*] BLIS SW website, last accessed on September 29th, 2004:

- Main page http://www.blis-project.org/BLIS_Product_Public.html
- View definitions http://www.blis-project.org/views/index.htm and http://www.blis-project.org/IFCR2_Concept_block_approach_000524_jh.pdf

[*Bruce and Cooper, 2000*] Bruce, Margaret and Cooper, Rachel: Creative Product Design: Figure 1, John Wiley & Sons 2000, ISBN 0-471-98720-4

[*Eastham, 2002*] Eastham, G. M: ECI Fast Track Projects Study "The Effective Management of Fast Track Projects", Executive summary, pp. 1-6, European Construction Institute, July 2002

[*EPM 2003*] EDM (Express Data Manager), EPM Technology, last accessed on September 29th, 2004: http://www.epmtech.jotne.com/products/index.html

[*Ekholm and Lehtonen, 2002*] Ekholm, Anders and Lehtonen, Riikka: Creative construction Briefing with Object-Oriented Technology and IFC, eSM@rt 2002 Conference Proceedings Part A, University of Salford, ISBN 0902896415

[*Evbuomwan, 1994*] Evbuomwan, PhD Thesis, City University of London, 1994. Indirect reference from Kamara et al, 2003: Capturing client requirements in construction projects, page 42.

[*Froese, 2002*] Froese, Thomas: Current Status and Future Trends of Model Based Interoperability, eSM@rt 2002 Conference Proceedings Part A, University of Salford, ISBN 0902896415

[*Garcia et al, 1993*] Garcia, Ana Cristina; Howard, H. Craig; Stefik, Mark J.: Active Design Documents: A New Approach for Supporting Documentation in Preliminary Routine Design, 1993, Stanford University, CIFE Technical report TR 82

[*Garcia & al, 2003*] Garcia, Ana Cristina; Kunz, John, Ekström, Martin and Kiviniemi, Arto: Building a Project Ontology with Extreme Collaboration and Virtual Design and Construction, 2003, Stanford University, CIFE Technical report TR152

[*Hemiö, 2002*] Hemiö, Tero: A Project Model Server Approach, WebSTEP, last accessed on September 29th, 2004: http://cic.vtt.fi/vera/Seminaarit/2002.04.24_IAI_Summit/WebSTEP.pdf

[*Hietanen, 2000*] Hietanen, Jiri: Space Layout Editor Demo, last accessed on September 29th, 2004: http://www.blis-project.org/demos/01_BLIS_Helsinki_SLE.zip

[*Hietanen, 2003*] Hietanen, Jiri: BLIS view definitions, website, last accessed on September 26th, 2004: http://www.blis-project.org/views/index.htm

[*IAI ISG 2003*] IAI ISG (Implementation Support Group) website, last accessed on September 29th, 2004: http://www.iai.fhm.edu/ImplementationOverview.htm

[*IAI NA 2003*] IAI North American Chapter website, last accessed on September 29th, 2004:

- IAI NA News website: http://www.iai-na.org/news/120402.php
- IAI NA Projects website: http://www.iai-na.org/technical/projects.php

[*IMSvr 2002*] IMSvr IFC Model Server project, Yoshinobu Adachi, Secom/VTT 2001-2002, web site last accessed on September 29th, 2004:
http://cic.vtt.fi/projects/ifcsvr/index.html and
http://cic.vtt.fi/vera/Projects/e_ifc_model_server.htm

[*Kagioglou et al, 1998*] Kagioglou, Michael; Cooper, Rachel; Aouad, Ghassan; Hinks, John; Sexton, Martin and Sheath, Darryl: A Generic Guide to the Design and Construction Process Protocol; University of Salford 1998, ISBN 0-902896-17-2

[*Kam and Fischer, 2002*] Kam, Calvin; Fischer, Martin: PM4D Final Report, CIFE Technical Report 143, available also at
http://www.stanford.edu/group/4D/download/c1.html

[*Kamara et al, 2003*] Kamara, John M.; Anumba, Chimay J.; Evbuomwan and Nosa F. O.: Capturing client requirements in construction projects, Thomas Telford Ltd 2003, ISBN 0 7277 3103 3

[*LBNL 1995-2003*] Lawrence Berkeley National Laboratory, all sites last accessed on September 29th, 2004:

- Simulation Research Group 2003, http://simulationresearch.lbl.gov/
- [*LBNL BLISS 1997*] Building Life-Cycle Information Support System (BLISS), http://eetd.lbl.gov/BTP/CBS/BPA/bliss.html
- [*LBNL DIT 2003*] Design Intent Tool DIT, http://ateam.lbl.gov/DesignIntent/home.html
- Hitchcock R.J.: Improving Building Life-Cycle Information Management Through Documentation and Communication of Project Objectives, Proceedings of Modeling of Buildings through their Life-Cycle, August 21-23, pp. 153-162. CIB Proceedings Publication 180, 1995, http://eetd.lbl.gov/btp/papers/37602.pdf
- Hitchcock, R.J.: Standardized Building Performance Metrics – Final Report, 2003

[*Prasad, 1996*] Prasad, B.: Concurrent Function Deployment – An Emerging Alternative to QFD: Conceptual Framework. Advances in Concurrent Engineering: Proceedings of CE 96 Conference, USA pp. 105-112. Indirect reference from Kamara et al, 2003: Capturing client requirements in construction projects, page 42.

[*RT, 2002*] Confederation of Finnish Construction Industries RT, Technology Strategy, 2002

[*SABLE 2003*] Simple Access to Building Lifecycle Exchange project, Houbaux, Patrick and Hemiö, Tero, Eurostep, last accessed on September 29th, 2004:

- Main pages http://www.blis-project.org/~sable/
- Motivation http://www.blis-project.org/~sable/about/why.html

[*Stanford 2001*] The Project Delivery Process at Stanford, Process Phase and Control Summaries, Stanford University Capital Planning & Management, Volume 1, Version 1.0, September 2001

[*Syed et al, 2003*] Syed M. Ahmed; Li Pui Sang and Zeljko M. Torbica: Use of Quality Function Deployment in Civil Engineering Capital Project Planning, Journal of Construction Engineering and Management, ASCE, Volume 129, Number 4, July/August 2003, Pages 358-368, ISSN 0733-9364

[*WebSTEP 2002*] WebSTEP IFC Model Server project, Karstila, Kari and Hemiö, Tero; Eurostep, last accessed on September 29th, 2004: http://cic.vtt.fi/projects/ifcsvr/index.html and http://cic.vtt.fi/vera/Projects/e_ifcnextstep.htm

---

[1] Stanford 2001: Page 8

[2] Kagioglou et al, 1998: Section 1, page 1:11

[3] Discussions and interviews 2002-2003: Robin Drogemuller/CSIRO, Stephen Hagan/GSA, Jiri Hietanen/TUT, Reijo Kangas/Tekes, Auli Karjalainen/Senaatti, Markku Kaskimies/Pöyry, Tapio Koivu/VTT, Tuire Kujala/Engel, Jarmo Laitinen/TUT, Pekka Metsi/Pöyry, Olli Nummelin/YIT, Vesa Pirinen/YIT, Sointu Rajakallio/Pöyry, Ilkka Romo/RT, Ben Schwegler/Disney Imagineering, Richard See/Microsoft, Mika Soini/NCC, Riitta Takanen/NCC, Juha Tammivuori/Skanska, Eija Virtasalo/Tekes, John Voller/ BV Solutions Group, Inc

[4] Garcia et al, 2003: Project Ontology, figure 1, page 7

[5] Syed et al, 2002: Pages 358-368

[6] Discussions and interviews 2002-2003, see reference 3

[7] Discussions and interviews 2002-2003, see reference 3

[8] Froese, 2002: Section 2.1, pages 199-200

[9] Kamara et al, 2003: Section 1.3.3, page 6

[10] Best and De Valence, 1999: Section 1.4, page 9

[11] Kamara et al, 2003: Section 7.4, page 156

[12] Eastham, 2002: Executive Summary, pages 3 and 5.

[13] RT, 2002

[14] Froese, 2002: Section 2.2, page 200

[15] Garcia et al, 1993

[16] Kamara et al, 2003: Section 3, pages 35-60

[17] Evbuomwan, 1994: Currently through Kamara et al, 2003: Section 3.4.4 page 42

[18] Prasad, 1996: Currently through Kamara et al, 2003: Section 3.4.4 page 42

[19] Syed et al, 2002: Pages 358-368

[20] Kamara et al 2003:, Figure 3.1, page 39.

[21] Kamara et al, 2003

[22] LBNL, 1995-2003

[23] LBNL BLISS, 1997

[24] LBNL DIT, 2003

[25] LBNL DIT, 2003

[26] IAI NA 2003: Internet

[27] SPADEX, 2002: Section 5.4, page 11

[28] IAI NA, 2002: Internet

[29] SABLE 2002: Internet

[30] Ekholm and Lehtonen, 2002: Pages 229-234

[31] BLIS 2003: Space Layout Editor, qPartners

[32] BLIS 2003: Internet

[33] Hietanen, 2003

[34] Hietanen, 2000

[35] BLIS 2003: Internet

[36] IAI ISG 2003: Internet

[37] Kam and Fischer, 2002: Section 6, pages 36-41

[38] IMSvr 2002: Internet

[39] WebSTEP 2002: Internet

[40] EPM 2003: Internet

[41] Adachi, 2002

[42] Hemiö, 2002

[43] SABLE 2002: Internet