TU WIEN

TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

Dissertation

Application of Semantic Web Material Libraries in AEC Context

ausgeführt zum Zwecke der Erlangung des akademischen
Grades eines Doktors der technischen Wissenschaften
unter der Leitung von

Univ. Prof. Dipl.-Ing. Dr. techn. Ardeshir Mahdavi

E259/3

Institute für Architekturwissenschaften

Abteilung Bauphysik und Bauökologie

Eingereicht an der technischen Universität Wien

Fakultät für Architektur und Raumplanung

von

Ferial Shayeganfar

Martikelnummer: 0327826

Zusammenfassung

Die Architektur-, Bauingenieur- und Konstruktions- Industrie (AEC-Industrie) umfaßt viele Wissensgebiete, welche auf verschiedene Fachkenntnisse und Arbeitsbereiche basieren. Informationsaustausch und gemeinsame Nutzung von Wissen sind die entscheidenden Faktoren, um in solch einer kollaborativen AEC-Industrie erfolgreich zu sein. Jedoch verursacht das verteilte AEC Wissen Wissensbrüche innerhalb der AEC Domänen. Denn jede Domäne hat ihre eigenen Werkzeuge und Anwendungen, was den Datenaustausch zwischen den Anwendungsgebieten erschwert. Darüber hinaus sollten Normen, Standards und Richtlinien im Bausektor kontinuierlich während des gesamten Gebäudeentwurfs und der Gesamtdauer der Konstruktion überprüft und verifiziert werden. D.h. obwohl alle Modelle eines Gebäudes dasselbe Objekt behandeln, basiert die Kommunikation zwischen den Modellen über Experten im Gebäudekonstruktionsbereich nur auf Grund ihres Wissens.

Ein weiteres wichtiges Thema im AEC-Bereich ist die richtige Auswahl von Bauprodukten und Materialien - eine Fragstellung, welche von mehreren Standpunkten aus beleuchtet werden soll. Einerseits haben die Eigenschaften von Bauprodukten einen großen Einfluss auf die Performanz von Gebäuden und den Komfort ihrer Bewohner, andererseits müssen sie relevante Normen, Richtlinien und Standards erfüllen. Anders formuliert, der Auswahlprozess für spezielle Produkte verläuft quer über die Grenzen einzelner Wissensdomänen. Domäne-Expertise wird daher benötigt, um die Eignung der ausgewählten Produkte festzustellen. Darüber hinaus müssen die Kriterien und Bedingungen für geschäftliche und wirtschaftliche Abläufe berücksichtigt werden.

Um eine lückenlose Verbindung der AEC-Werkzeuge mit Normen, Standards und Richtlinien samt Bauproduktinformationen des AEC-Bereiches zu gewährleisten, bedarf es eines effizienten und formal basierten Mediums, wie z.B. des Semantischen Webs und von Ontologien.

Das Semantische Web stellt eine kollaborative Arbeitsumgebung für den Informationsaustausch zu Verfügung, welche eine Interpretierbarkeit durch Maschinen ermöglicht. Ontologien, die Domänen-Konzepte und deren Beziehungen untereinander beschreiben, spielen eine wichtige Rolle bei der Realisierung der Vision des Semantischen Webs. Sie tragen zur Lösung von Problemen der Interoperabilität und der Kompatibilität zwischen Software-Anwendungen verschiedener Domänen bei. Dies bedeutet, dass in einer AEC-Umgebung, welche auf dem Semantischen Web basiert, das Konstruktionsmodell der beteiligten Partner miteinander interagieren kann, wo Ontologien hierbei als Kommunikationsbasis dienen.

Trotz aller potentiellen Vorteile der Semantischen Web-Anwendungen im AEC Bereich sind nur sehr wenige Forschungsarbeiten und nur eine geringe Anzahl an Prototypen verfügbar. Die AEC-Gemeinschaft zeigt sich bis jetzt als sehr zurückhaltend gegenüber der Einführung dieser neuen Technologien. Der bester Beweis für diese Aussage ist die bislang noch nicht abgeschlossene Übergangsphase von CAD-Zeichnungen zu intelligenten Gebäude-Modellen. CAD Zeichnungen werden in vielen Ländern noch immer als der de-facto Standard für den AEC-Datenaustausch eingesetzt und als solcher auch akzeptiert. Der Übergang von bestehenden zu intelligenteren Gebäude-Modellen, welche zwischen verschiedenen Domänen leicht ausgetauscht und bearbeitet werden können, stellt die nächste große Herausforderung der AEC-Industrie dar.

Im Rahmen dieser Dissertation wird die Rolle des Semantischen Webs in der AEC-Industrie genauer untersucht und eine solide Basis für eine Arbeitsumgebung, die auf einem semantischen bzw. ontologischen Ansatz basiert, geschaffen. Die vorgeschlagene Herangehensweise kann dazu dienen, die zuvor behandelten Probleme effizient zu beseitigen. Die semantische Anreicherung des Domäne-Wissens mit semantischen Web-Services ermöglicht eine Verbindung der verschiedenen AEC-relevanten Wissensbereichen. Als „Proof-of-Concept“ wird ein Prototyp für den speziellen Fall eines Dachfensters vorgestellt.

ABSTRACT

The Architecture, Engineering and Construction (AEC) industry is composed of multiple knowledge domains that are formed corresponding to the needed skills and professions. Sharing and exchanging knowledge is the key factor to success in such a collaborative environment; however, the distributed nature of AEC information has lead to knowledge gaps between AEC related domains. Each domain has its own tools and applications and the data exchange between domain applications is not straightforward. In the other words, although all the models of a building are talking about the same object, the inter-model communication is done only by the knowledge of expert building constructors. Moreover, the building code standards and regulations should be continuously tested and verified throughout the building design and construction life cycles.

Another important concern in building construction is the proper selection of building's products and materials which is a critical issue and should be seen from multiple perspectives. On one hand, product attributes will have a great effect on building performance and comfort of inhabitants, and on the other hand, its conformance with building codes and regulations should be considered. In other words, the selection process of a specific product crosses the discrete knowledge domain borders and the domain expertise is needed to confirm the fitness of selected product. Moreover, the criteria and conditions of business and economic processes need to be addressed.

In order to bridge the communication gap between AEC tools, building regulations and building products, we would need an efficient and formal communication medium such as Semantic Web and ontologies.

The design principle of the Semantic Web is to provide collaborative working environment and knowledge exchange in a way that is understandable by machine. Ontologies which describe the domain concepts and the relationships among them will play a prominent role in the Semantic Web vision. Ontologies contribute to solve the problem of interoperability and common understanding between software applications of different domains. So in a Semantic Web based AEC environment, the building models of two project partners can interact with each other using the ontologies as common understanding.

Despite all potential benefits and advantages of Semantic Web applications in the AEC field, there are very few research works and prototypes available. As a matter of fact, the AEC community is behaving conservatively about new technologies. The best proof for this statement is the transition phase from CAD drawings to smart building models, which is not completed yet. The CAD drawings are still being used in many countries and accepted as de facto standard for AEC data exchange. The transition from existing building models to smarter building models that can be easily shared and processed between different domains is the next challenge that the AEC industry will be confronted soon.

In this research the role of Semantic Web in AEC industry is exploited and a framework as solid basis for semantically-enabled working environment is proposed. The proposed framework can be used to overcome the problems stated above. The rich semantics associated with the domain knowledge together with the Semantic Web Services allow bridging the gap between discrete knowledge domains. As a proof of concept a prototype for specific case of skylight product is presented.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to my supervisor Prof. Dr. Ardeshir Mahdavi for having accepted me as a doctorate student. Without his inspiration, advices, guidelines and continuous support this thesis would not have been possible.

I'm grateful to Prof. Dr. A Min Tjoa, whose support has been irreplaceable to me in the past few years.

My sincerest thanks to my best friend and my husband, Amin Anjomshoaa, for being on my side, giving me courage to step forward in this field and helped me immensely in all steps.

I warmly thank Ms. Elizabeth Finz and Ms. Maria Schweikert for all their help and support and also Ms. Gabriela Nowotny and Mr. Christoph Grün for translation of the abstract.

I want to acknowledge also the jury member for accepting to be on my examination board and for their support and guidance and their excellent feedback throughout the process.

Last but not least, I owe my deepest gratitude to my parents and my siblings for giving me so much love and encouragement.

Table of Contents

List of Figures

List of Tables

List of Listings

List of Appendixes

Chapter 1

INTRODUCTION

Information technology (IT) tools and methods have been in use in building industry for over three decades. Utilization of IT tools such as computer-aided design applications in the building design phases is already pervasive. However CAD drawings do not suffice the requirements regarding effective building models and are not adequate for integration in business processes that are massively used in building industry today. Nowadays the Architecture, Engineering and Construction (AEC) industry requires more intelligent building models that can on one hand, capture and represent the building information efficiently, and on the other hand, get integrated into business processes cohesively. The traditional method of system integration in heterogeneous information environments starts with the standardization of data exchange methods. In AEC, the data exchange standardization efforts have lead to Industry Foundation Classes (IFC, 2008) which is a data model, developed by the International Alliance for Interoperability (IAI, 2008) to facilitate interoperability between the software used by all participants in AEC and Facilities Management (FM) industries.

The building industry is embedded in a complex web of relationships that affect building design. The building design process relies on large databases that are often managed by human knowledge and interactions. Moreover, the building processes and their related information are usually designed for a specific building construction project and many context parameters such as country-specific standards and policies should be considered during the lifetime of project. Effective mapping of relevant contextual attributes onto available building industry information has a formidable potential to improve the design process. For instance the design options and alternative materials could be more readily assessed and compared by providing semantically enriched building models to evaluation applications (e.g. performance simulation programs) (Shayeganfar et al., 2008).

At the time this research was started, a new concept called Semantic Web (Berners-Lee et al., 2001) has evolved into a powerful resource for well defined information. The Semantic Web is a web of data that will enable machines to comprehend the data. Up to now the data available on World Wide Web has been solely human understandable but by utilizing Semantic Web, it will be changed to a rich resource of information. It is important to note that despite this classical definition of semantic web, which is coupled with Internet and World Wide Web, the technology has been widely accepted and used to capture and document context information in many fields. It plays a significant role in information sharing scenarios and interoperability across applications and organizations. This added-value opens the way to integrate huge amount of data and becomes extremely useful when used by many applications that comprehend this information and bring them into play without human interaction. The shared knowledge of Semantic Web based project is captured in domain ontologies which are formal definition of concepts within a domain and the relationships between those concepts. Ontologies are generally following two patterns: first is the schema pattern where ontology describes only the data structure and second is the taxonomy pattern that captures the domain semantic. In the first pattern the knowledge is in the data and not the ontology whereas in the second pattern domain knowledge is in the ontology itself (Berners-Lee, 2006).

In AEC domain the need to a common language has lead to the development of Industry Foundation Classes (IFC, 2008) that to a great extent can be seen as ontology of AEC field. By making use of suitable ontological commitments the ontologies can successfully be integrated in AEC concerned domain using Semantic IFC. In an ontological sense the IFC presents a hybrid approach that on one hand, provides the *schema pattern* by formal definition of data structures, and on the other hand, provides the *taxonomy pattern* by capturing the domain entities and their relationships. In addition, it makes the building model available for integration in processes through specific web services to the outside world. Furthermore, the semantic web provides a uniform view of information, which is independent of its encoded language and its medium type (Internet, CD, catalogue, etc). A pre-requirement to the existence of such a library is of course a coherent and consistent ontology which can be understood and used by all processes and applications. The ontology that has been used for this thesis is created based on IFC. The use of IFC as the common template for information sharing and the application of semantic Web Services as the communication method can benefit the building industry to realize building processes and information sharing scenarios.

1.1 Motivation

One of the challenging issues during the design phase of a building is selection of the appropriate materials and components from existing products that are flooding the market. As a matter of fact, the material selection is a multidimensional problem that should be done with precise evaluation of environmental and building characteristics; among them we can mention the followings:

- Environmental impacts of different materials through building's life-cycles are not fully known to building designers. As a result it is not an easy task to find out the better material from an environmental standpoint.
- Building's operating energy is highly influenced by selected components and materials. The proper selection of components and building materials will lead to lower energy consumption.
- Building's embodied energy (the energy used in production and distribution of a product or material) of the materials that compose buildings is an important consideration to homeowners. Presently the embodied energy of building materials contributes anywhere from 15 to 20% of the energy used by a building over a 50 year period. Homeowners have tremendous influence as to what material are used and can specify those materials with low embodied energy, thus reducing the amount of fossil-fuel energy used during production (Architecture 2030, 2008)

- Selected building materials influence the work of different groups of building construction engineers. The selected material by any of these groups should be carefully examined by other groups for the potential side effects.
- Last but not the least is the financial aspects of a building construction project that should be considered in material selection processes. The better and more efficient material cannot be accepted unless the material's price meets the financial criteria. Additionally the price should be reasonable for the amount of energy that will be saved during the lifecycle of a building.

Provision of computational support for this decision making process would benefit the AEC stakeholders in view of cost reduction, energy efficiency, and occupants' comfort and productivity. Nowadays, there are many different media (catalogues, CDs, Internet, etc), which entail information on building products and their relevant technical information (Mahdavi et al., 2004). However, most of this information is neither in a standard format nor machine processable (in a semantic way). The comparison of products, their properties and study of their effects in the building's lifecycle is a time consuming process (Shayeganfar et al., 2008).

AEC Objects' data span several domains (e.g. design, specification, cost, etc.) and involve the structural, functional, and behavioural characteristics of the object (Halfawy & Froese, 2002). Every project collaborator usually considers on different views of the same object. Also, most of the building construction decisions can not be made by a single engineering role, without considering the side effect of this decision in other domains. In other words a decision making needs knowledge and expertise that is distributed among project partners and the side effects of design decisions should be considered by other relevant participants in order to achieve the building goals and required performances. The changes in the building component and products should be checked with simulation programs to make sure that the applied changes are not in conflict with building's requirements and criteria. None of these processes is possible without human interaction which cost time and money. For instance, selection of building element such as window by a partner will affect the work of others groups such as building's performance analyst and interior designer and the plan should be adopted respectively.

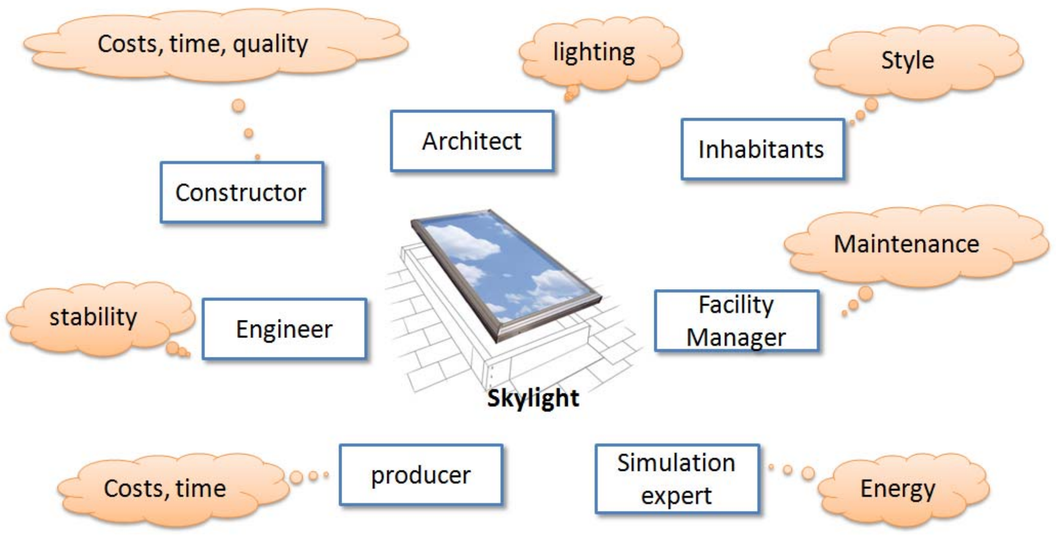


Figure 1.1: Interoperability requirements of a skylight component

In order to improve the interoperability between building model and material libraries, new methods are required that facilitate the selection of appropriate building components within the context of specific projects. These new methods should provide an automated tool support for comparing their technical characteristics and energy performance parameters, using the building simulation programs. Achieving this goal could be supported by a kind of computer-understandable data that is flexible enough to bridge the knowledge gap between building domains and material libraries.

This research presents a novel approach that uses Semantic Web technologies to enrich the building products libraries by adding ontologies and semantic meanings and this will open the door to integrate large amount of data on the web (catalogues) with simulation programs via proper product library services.

1.2 Research Question

There are several specific questions regarding the feasibility, theoretical and technical aspects of applying Semantic Web Technologies and to close the gap between building material knowledge and AEC business processes. Some of the questions that are the main focus of this research are:

- How can the semantics of building material and their associations be modelled accurately for open world interactions?
- How to define material libraries that can interact with global AEC services and share useful information about a specific material in an effective way?
- How Semantic Web Technologies can improve the quality of built environment from energy efficiency standpoint?
- How a modern Building Information Model can interact with simulation programs and material libraries in an effective way and improve the performance in interoperability between applications?
- How can building requirements and preferences be represented and how should they be taken into account in tailoring AEC services for a specific building?
- How can business processes of a building project handle the partners' interaction in a semi-automated manner?
- How can building material knowledge be extracted from existing material catalogues and prepared for usage by AEC applications such as simulation tools?

1.3 Thesis Organization

1.3.1 Chapter Summary

This thesis is comprised of three major parts: Interoperability (Chapter 2), Ontologies and Semantic Web (Chapter 3) and SkyDreamer prototype (Chapter 4). In the first part the interoperability in AEC field is discussed. The second part explores the Semantic Web technologies and their potential applications in AEC field. The last part presents the SkyDreamer prototype as a proof of concepts for the proposed approach of this thesis. The chapters are described in more detail below:

- Chapter 2: provides theoretical background and summarizes relevant research work in the area of IFC model interoperability.

- Chapter 3: gives an overview of Semantic Web technologies with a specific focus on AEC/FM application and use cases.
- Chapter 4: presents the prototype architecture and explores the different part of its architecture in detail.
- Chapter 5: presents the SkyDreamer prototype results and different use cases that can be addressed using it.
- Chapter 6: summarizes the research work presented in this thesis and presents the main results that have been concluded from the work. It also provides the future research steps.

1.3.2 Contributions

The work presented in this thesis provides some contributions to AEC building design process; the major contributions are summarized as follows:

- IFC Ontology
- SkyDreamer calculator
- Skylight Semantic Library
- IFC based web services
- A simple skylight simulation web service

Chapter 2

Interoperability

Although all AEC industries have benefited from the advantage of IT, the data structure and information models, which are used by one stakeholder are not directly usable by others, unless some common standard exists that defines the interaction protocols and common understanding of the relevant information. As a matter of fact, the need to share the information and increase the interoperability is the result of current industrialization obligation. The National Institute of Standards has estimated that data incompatibility is a 90 billion dollar problem for the manufacturing industry (Brunnermeier & Martin, 1999).

The building industry today, forces the business processes to be faster, more energy efficient, and more cost-effective. These goals cannot be achieved without increasing the interoperability between building construction domains that are at the moment mainly ill-structured and suffering from the limitations of information exchange and data sharing standards.

AEC field is a highly interdisciplinary network of knowledge and proficiencies. The expertise and knowledge required by a typical building project ranges from social-psychological to economical-technical know-hows and information that is obviously scattered among many domains. Several organizations and partners use different computer systems providing the data for design, manufacture, use, maintenance and disposal of a product. A large portion of the building construction costs are caused by splitting up of processes and flawed communication among knowledge domains. Following the traditional methods in AEC, the same information is entered many times in different systems before the building is handed over to owners and in between there are communication errors and loss of project information. The following figure shows the traditional approach of information flow between stakeholders with excessive data exchange and multiple formats.

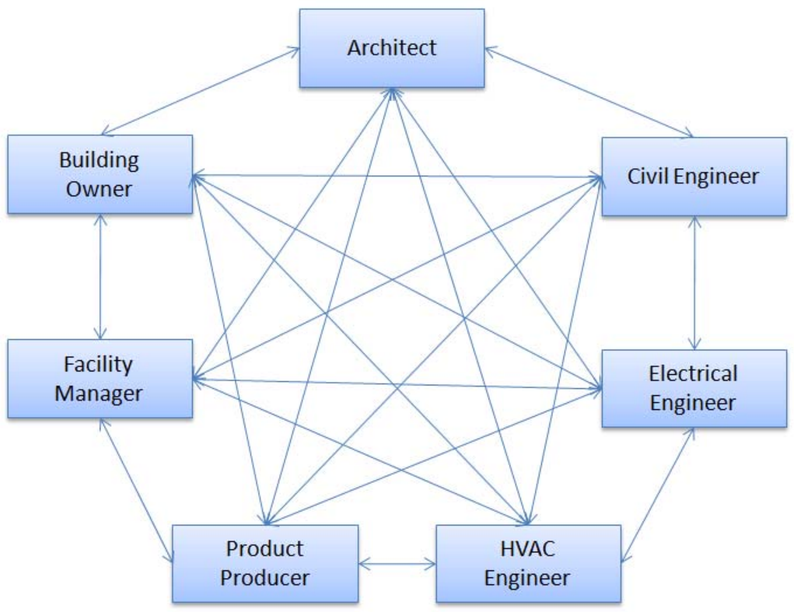


Figure 2.1: The complexity of interactions between building's stakeholders

Despite recent advances in building science and penetration of IT-based tools in the building industry to date, there is a common agreement (Augenbroe, 2001; Egan, 1998; Kieran et al., 2003) that the industry is not as efficient as expected and current practices and methods are causing major losses in productivity, material resources and liability. The following lists some of the challenges in AEC that are directly caused due to the flawed communication between knowledge domains:

- Diverse software and programs that are used to sketch, design, and document a building during its lifecycle, use dissimilar schemas to describe the building information model.
- The common part of building information model that is being used by multiple partners should be re-entered and repeated in different programs, which costs money and time.
- The re-entered building models might be slightly different and this will cause model inconsistency problems.

- The building information which is prepared by a specific software might not be usable by the other. As a result, the information should be transformed to target standard and not all transformations are able to deliver the building information entirely accurate, and without loss of information. Each system uses its own data formats and as a result the same information needs to be entered several times into various systems, which lead to data redundancy and errors. So the lack of communication or miscommunication not only increases the construction costs but also some of information may get lost.
- Even using the same program but different versions may cause some difficulties for usage of the information.
- In some cases even the hosting operating system of AEC applications may make the data unusable. For example, weather data file that is created on a German windows might not be directly usable by an English based application due to the difference in decimal point notation (in German positional numeral system unlike nations of British Empire who use dot symbol, comma symbol is used to separate integral and fractional parts of a decimal number. So for example 123.456 will be ambiguous if the notational system is not known)

Figure 2.2 depicts typical building construction phases and the information lost between phases, caused by any of above mentioned bottlenecks.

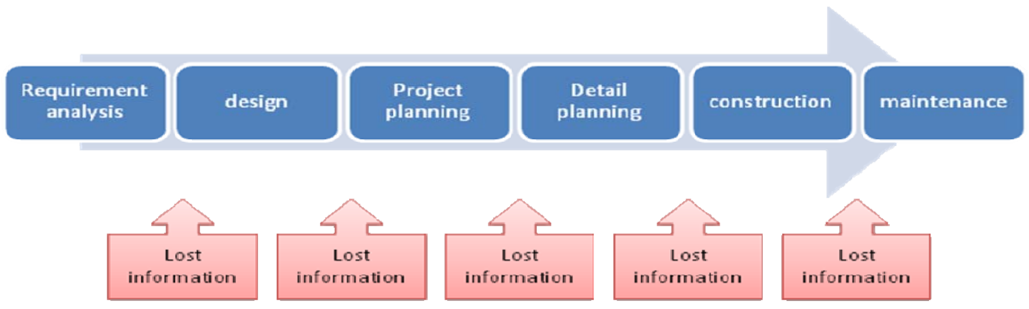


Figure 2.2: Lost of information between building construction phases

Among the major causes of the above mentioned problems, we can highlight the lack of a standard format for sharing building information, which in turn reduces the productivity and process integration. As a matter of fact we need a formal standard model that can be used for exchanging information of products during the life cycle of the built environment between systems, where the systems may be separated organizationally, geographically, or temporally (Eastman, 2008). Figure 2.3 shows the distribution of information in multiple forms during the building's lifecycle. Using a common product model by all partners, increase the quality of information and decrease the cost of preparation.

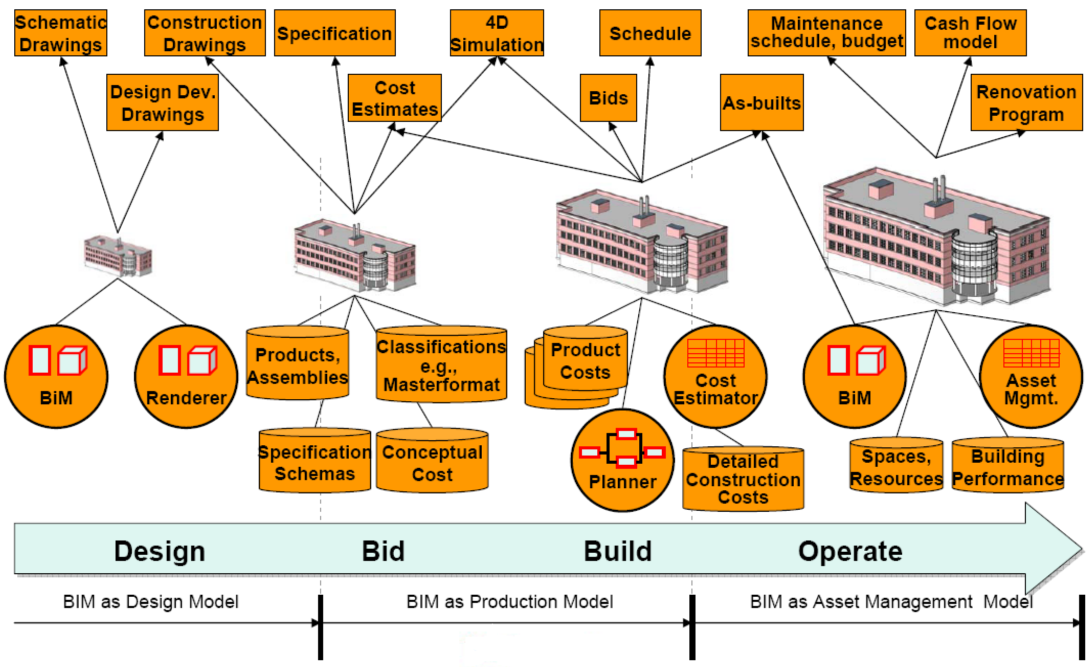


Figure 2.3: Distribution of project information in multiple forms (Grant & Ceton, 2008)

Several approaches have been proposed to overcome this situation and increase the level of support provided by the computer industry to manufacturing. At the very beginning it started with some national data exchange standards like VDAFS (Verband der Automobilindustrie - Flächenschnittstelle) in Germany (VDAFS, 2008) and Initial Graphic Exchange Specification (IGES) in the USA (IGES, 2008).

Later on the efforts were focused at international level to develop new standards for data exchange. In this chapter the international data exchange formats will be introduced briefly and afterwards the Industry Foundation Classes (IFC, 2008) which is generally accepted as the best existing data exchange standard will be explored in more details.

2.1 ISO-STEP

The "Industrial automation systems and integration - Product data representation and exchange" which is also known as STEP (the Standard for the Exchange of Product Model Data) was created in 1994/1995, under the supervision of the International Organization for Standardization (ISO) as an international product data standard (ISO 10303) for exchanging digital information between different CAD/CAM and Product Data Management systems (ISO, 1994).

STEP provides a computer-interpretable definition of the product data throughout its life cycle which is independent from any particular system and this makes it very suitable for implementing and sharing product libraries.

This standard is divided into many parts which are categorized under Environment, Integrated data models and Top parts. The Environment group contains the description methods such as EXPRESS and EXPRESS-X and also implementation methods such as STEP-FILE, STEP-XML etc. The second part of standard, which is integrated data model, contains Integrated Resources (IR), Application Integrated Constructs (AIC) and Application Modules (AM). The last part of STEP standard, which is Top Parts, consists of several hundred parts and Application Protocols (AP). Every year new parts are being added or revisions of existing parts are being released and this makes STEP the biggest standard within ISO (STEP, 2008). Table 2.1 provides a listing of design Application Protocols that are roughly categorized under Mechanical, Building Design, Manufacturing and Life cycle groups.

Mechanical Design	
Part 201	Simple 2D drawing geometry related to a product. No association, no assembly hierarchy
Part 203	Configuration controlled 3D designs of mechanical parts and assemblies.
Part 204	Mechanical design using boundary representation
Part 207	Sheet metal die planning and design
Part 209	Composite and metallic structural analysis and related design
Part 214	Core data for automotive mechanical design processes
Part 235	Materials information for the design and verification of products
Part 236	Furniture product data and project data
Building Design	

Part 202	2D/3D drawing with association, but no product structure.
Part 225	Building elements using explicit shape representation
Connectivity oriented electric, electronic and piping/ventilation	
Part 210	Electronic assembly, interconnect and packaging design.
Part 212	Electro-technical design and installation.
Part 227	Plant spatial configuration
Manufacturing APs	
Part 219	Dimensional inspection information exchange
Part 223	Exchange of design and manufacturing product information for cast parts, currently on CD level
Part 224	Mechanical product definition for process plans using machining features
Part 238	Application interpreted model for computer numeric controllers
Part 240	Process plans for machined products
Life cycle support APs	
Part 239	Product life cycle support
Part 221	Functional data and schematic representation of process plants

Table 2.1: STEP design Application Protocols

Every Application Protocol defines the meaning of STEP’s generic concepts (Conformance Classes) in a specific context of application or data exchange scenario. Each Application Protocol contains a scope that describes its purpose and the activity diagrams of the tasks that are performed by engineers within specified scope. It also includes an Application Requirement Model that describes the information requirements of those activities. The information requirements are then mapped into the common set of Integrated Resources (IR) and the result is a data exchange standard for the activities within the scope.

Nearly all major CAD/CAM system now contains a module to read and write data defined by one of the STEP Application Protocols. In the USA the most commonly implemented protocol is called AP-203. This protocol is used to exchange data describing designs represented as solid models and assemblies of solid models. In Europe a very similar protocol called AP-214 performs the same function.

The ability to support many protocols within one framework is one of the key strengths of STEP. All the protocols are built on the same set of Integrated Resources (IR's) so they all use the same definitions for the same information. For example, AP-203 and AP-214 use the same definitions for three dimensional geometry, assembly data and basic product information. Therefore CAD vendors can support both with one piece of code.

The ultimate goal of STEP was to cover the entire product life cycle, from conceptual design to final disposal. Despite the many successes of STEP, its development and deployment is not growing fast enough. Many critics point out correctly that the XML standards for e-commerce are being developed much more quickly. The real issue that stops faster STEP deployment is the commitment of those with the resources necessary to define the standards. The government does not like to pick solutions for industry, and industry does not like to fund the development of solutions that can also be used by their competitors. Consequently, much work only gets funded in situations of clear and desperate need such as when the high cost of manufacturing is causing excessive job losses (STEP development, 2008).

The Internet and the World Wide Web broke through this cycle when "killer" applications made the benefits of the new infrastructure clear and compelling for all users. AP-203 made STEP useful by allowing solid models to be exchanged between design systems. AP-238 will make STEP compelling for some users by allowing them to machine parts more efficiently. However, like the early Internet there will be alternatives that are considered more reliable by other users. The killer application that makes STEP ubiquitous has yet to be identified (STEP development, 2008).

Another challenging issue about product model data is that it is different from other kinds of e-commerce data such as invoices, receipts, etc. The traditional method for communicating product model information is to make a drawing and the traditional method to communicate an invoice is to make a form. In order to equip a drawing or 3D model with required information, many detailed and complex interrelated data structures relationships are required and this makes the STEP data exchange problem more difficult.

An XML data format is being developed for STEP but the STEP architecture requires the information requirements of an Application Protocol to be mapped into the common set of Integrated Resources. This allows all of the protocols to share the same information and is essential if all application STEP interfaces are going to share and reuse the same set of data. However, the sharing necessarily divides the original data into multiple entities that are not so easy to understand in XML or any other format. This is disappointing because one of the attractions of XML is that it is self-documenting (at least for programmers and domain experts). Therefore, a new level of documentation is required in the STEP data to show how the information requirements have been mapped. The required structures have lead to a self-documenting XML format for STEP which is called STEP-XML (ISO 10303-28, 1994).

Perhaps a more fundamental approach to achieving smooth mappings between components is offered through the definition of ontologies (Kemmerer, 1999). ISO 10303 has been developed based upon the combined expertise of hundreds of engineers throughout the world, codifying terms familiar to them. These definitions, however, are not stated formally in logically provable forms. Thus, the possibility of ambiguity and misinterpretation exists at all levels of the standard. An ontological foundation will be needed ultimately to address rigorously issues of redundancy and misuse within the standard. A formal ontology will also address another missing piece of STEP: the vocabulary of terms used to populate the defined STEP entities. This problem is not as apparent in the traditional CAD applications of STEP where terms and values have been widely accepted for years (such as Cartesian coordinates for spatial locations). In other areas, the terms are much less certain.

Currently, a major driver for architectural change in STEP is interoperability between APs. The issue of interoperability brings up an additional point, which is the tradeoff between extensibility of the specification and guaranteed interoperability of applications using the specification. On the one hand, it would be naive to think STEP developers would have the foresight to anticipate all data elements of importance for any significant time span. Therefore, there is definitely a need to be able to expand the current STEP data structures. On the other hand, interoperability between APs is definitely threatened if expanded data structures are added outside the standard.

2.2 EXPRESS

EXPRESS (ISO 10303-11) is the data modelling language that has been developed as part of STEP standard but is used widely in other standardization, research and integration projects. The strong point of this modelling language is its implementation-independent data model. EXPRESS is a conceptual schema language that describes the specification of classes belonging to a defined domain, the information or attributes belonging to those classes (color, size, shape, etc.), the constraints on those classes (unique, exclusive, etc.) and also the relations between classes (Express-IFC, 2008). For example the EXPRESS language can be used to define an exchange model in some target domain, such as steel structures, building geometry, piping systems, and so forth.

The EXPRESS data model can be presented in two ways, textually and graphically. The graphical representation of EXPRESS data models is called EXPRESS-G and is more suitable for human use for better understanding of the underlying domain entities and their relationships. On the other hand, the textual form is used for formal verification of models and also as input for tools that accept EXPRESS data model.

The EXPRESS data model has facilitated the data modelling and data exchange scenarios. Although EXPRESS syntax is similar to a programming language, it defines only the data on which programs should operate and it cannot be used to create executable programs. A major advantage of EXPRESS language is the emergence of many software toolkits for parsing, formal syntax validation, mapping and exchange of the data models that are described in EXPRESS language. Using these tools, users are able to transform the EXPRESS model to software objects of programming languages such as C, C++ and Java. This will make it easy to integrate data models in software tools and instantiate, update, or read data in the language-implemented object model. Moreover, the EXPRESS model can be serialized in different formats such as text file, XML, and diagram. The latter serialization form makes the EXPRESS models more expressive and understandable to human users.

The basic unit of data definition in EXPRESS language is called an entity. Like modern programming object oriented programming languages, EXPRESS supports generalization and specialization of entity types. The relationships between entities is also similar to UML (Unified Modelling Language) where the relation is defined by the role played by one entity type in the definition of another.

Listing 2.1 illustrates a data model in EXPRESS language where two simple entities are related to each other through the roles they play in their relationship. The given model describes an entity called `IfcProduct` that is extending (inheriting from) `IfcObject`. Additionally the defined entity is related with an `IfcObjectPlacement` entity with its `objectPlacement` attribute.

```
ENTITY IfcProduct
  ABSTRACT SUPERTYPE OF(IfcProxy)
  SUBTYPE OF (IfcObject);
    ObjectPlacement  : OPTIONAL IfcObjectPlacement;
    Representation  : OPTIONAL IfcProductRepresentation;
  INVERSE
    ReferencedBy  : SET OF IfcRelAssignsToProduct FOR RelatingProduct;
  WHERE
    WR1  : (EXISTS(Representation) AND EXISTS(ObjectPlacement)) OR
           (EXISTS(Representation) AND
           (NOT('IFCREPRESENTATIONRESOURCE.IFCPRODUCTDEFINITIONSHAPE' IN
           TYPEOF(Representation)))) OR (NOT(EXISTS(Representation))) ;
END_ENTITY;

ENTITY IfcObjectPlacement
  ABSTRACT SUPERTYPE OF(ONEOF(IfcGridPlacement, IfcLocalPlacement));
  INVERSE
    PlacesObject  : SET [1:1] OF IfcProduct FOR ObjectPlacement;
    ReferencedByPlacements  : SET OF IfcLocalPlacement FOR PlacementRelTo;
END_ENTITY;

ENTITY IfcProductRepresentation
  SUPERTYPE OF(ONEOF(IfcProductDefinitionShape,
             IfcMaterialDefinitionRepresentation));
    Name        : OPTIONAL IfcLabel;
    Description  : OPTIONAL IfcText;
    Representations  : LIST [1:?] OF IfcRepresentation;
END_ENTITY;
```

Listing 2.1: a simple data model in EXPRESS language

The listing 2.1 also shows some other interesting features of EXPRESS language to capture the constraints. For example the EXPRESS INVERSE attribute can add constraints to relationships by making the implicit reverse relations (which are automatically resulted by relationships between two entities) explicit. This does not create a new relationship, but makes the relationship visible under a given name, and therefore allows it to be constrained. So for instance the `PlacesObject` reverse relation in `IfcObjectPlacement` simply says that there is only one `IfcObjectPlacement` instance of for each `IfcProduct`.

Another capability of EXPRESS to be considered is the concept of local constraints which are applied to every instance of a type. The constraints are called local because they are specified within definition of the type to which the constraint applies. The local constraints follow the basic pattern of a WHERE clause consisting of one or more rules, that evaluates to true, false, or unknown. In listing 2.1, `IfcProduct` product entity contains a rather complex rule that checks the existence of optional attributes and applies a constraint on product representation.

EXPRESS language has many other features and constructs that make the language powerful and flexible. In this short survey, the in-depth coverage of all EXPRESS language features is not possible, but it is worthy to mention that the language includes the support for definition of functions, procedures and rules as well as a rich set of standard constants, functions and procedures.

As mentioned before the EXPRESS language has also a graphical representation called EXPRESS-G that is more suitable for human communication. Originally the graphical representation was designed for documentation purposes and for an easier understanding of the data models by model developers and reviewers. However, it has been proved to have significant applications in model development. Moreover there is a handful of software tools that support the translation between textual and graphical representation of EXPRESS language.

To demonstrate the usefulness of the EXPRESS-G, the graphical representation of the data model of listing 2.1 has been shown in figure 2.4.

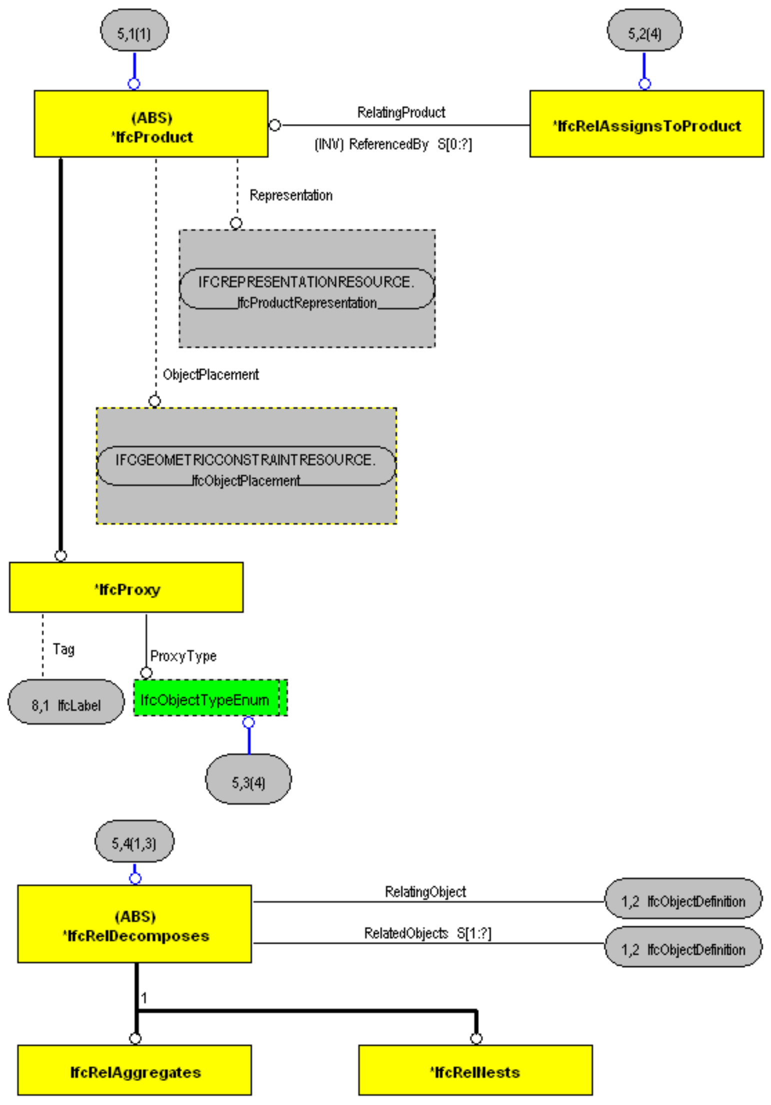


Figure 2.4: EXPRESS-G representation of a sample data model

2.3 IAI IFC Schema

STEP is used in various industries such as aerospace, automotive, shipbuilding, oil and gas, electronics architecture and building construction, and so on, but it has been clarified that the STEP standard is not specific enough to fully describe the requirements of architecture, engineering, construction and facility management (AEC/FM) field and a more domain-specific model is needed for representing building data. The need for more efficient information exchange methods has been widely recognized in AEC/FM which has lead to a large-scale international endeavour for defining a specific building information model that can be used to fully capture the building lifecycles and manage the relevant information efficiently. In this context, the International Alliance for Interoperability (IAI) which is a global coalition of industry practitioners, software vendors, and researchers (over 600 companies around the world) has introduced the Industry Foundation Classes (IFC) with the goal to provide a better interoperability throughout the AEC/FM community. This high-level, object-oriented building information model schema is closely related to the STEP standard and uses several resource definitions of STEP. Not surprisingly, it is defined in the same modelling language as STEP which is EXPRESS. The IFC classes provide support for (IFC, 2008):

- data exchange among software applications within the AEC/FM industry sector
- model-based description of spatial elements, building elements, MEP (Mechanical/Electrical/Plumbing) elements and other components that constitutes a building or facility
- shape representation of such components
- relationships of such components between each other and to the spatial and system structure
- attachment of properties, classifications, external library access, etc. to such components

The IFCs model all types of AEC/FM project information such as parts of a building, the geometry and material properties of building products, project costs, schedules, and organizations, etc. The information from almost any type of computer application that works with structured data about AEC building projects can be mapped into IFC data files. In this way, IFC data files provide a neutral file format that enable AEC/FM computer applications to efficiently share and exchange project information.

The IFCs which was initiated in 1994 have now undergone some major releases, and is being supported many commercial and open source tools. The latest official IFC release is IFC 2x Edition 3 (short IFC2x3) and is recommended for implementation.

It has been published in February 2006 and contains several improvements over the previous editions. The IFC2x3 has been the first IFC release that follows the new IFC release methodology and has been already used in some prototype implementations. The next release of IFC, IFC 2x edition 4 (IFC2x4), is also published as alpha edition and has provided a number of feature increases with some major rework and improvements of the existing IFC specification. It has been developed as the next basis for IFC enabled interoperability of Building Information Models. It is also intended that the IFC2x4 release will be submitted to the International Organization for Standardization (ISO) for approval as a full International Standard ISO16739 (IFC2x4, 2008). Each release of IFC schema introduces new classes that are needed for AEC/FM use cases. Figure 2.5 shows the growing number of IFC classes from IFC1.0 to IFC2x2.

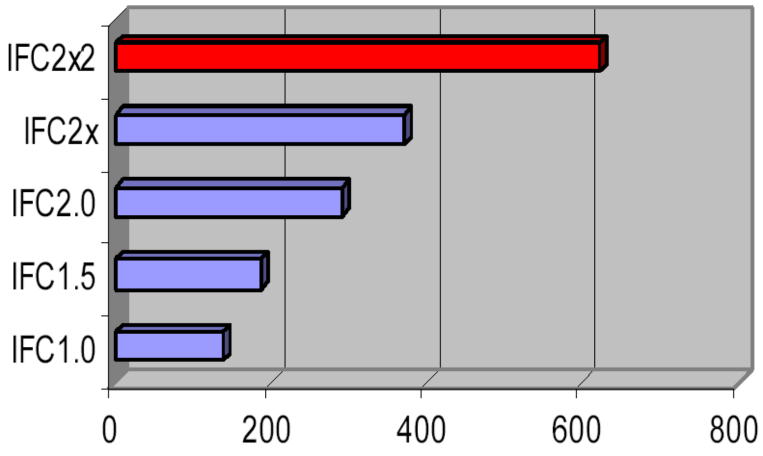


Figure 2.5: Number of IFC classes in different releases (Isermeyer, 2008)

The new IFC Classes are responsible for covering the diverse requirements of AEC/FM professions. Figure 2.6 depicts the estimated coverage of different releases of IFC (IFC1.0 to IFC2x2) in AEC/FM domains.

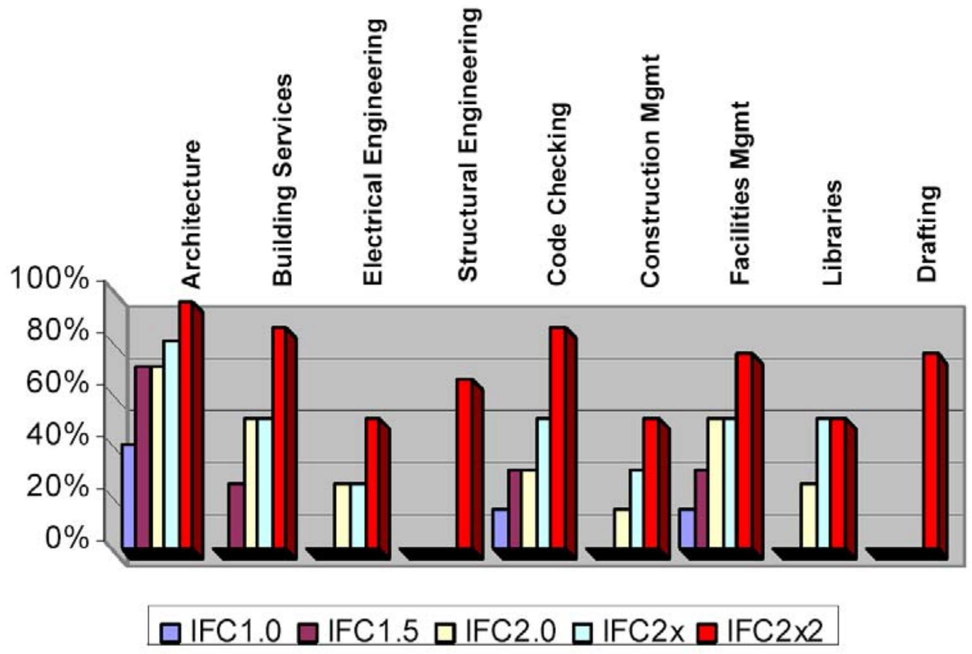


Figure 2.6: Coverage of IFC releases in different AEC/FM domains (Isermeyer, 2008)

2.3.1 IFC Architecture

Since the IFC schema should cover the interoperability of various applications in AEC/FM field, it should be able to model all relevant information. The smart way of defining such a schema is to make a “generic and compact model” rather than a “huge model” with highly complex structures. The drawback of the latter approach is that the models would be difficult to understand and virtually impossible to implement. So instead of creating a class for every type of physical objects that can be encountered, concepts of physical and other object types are generalized to provide relatively high level descriptions. For instance the IFC schema defines the abstract class of `IfcDoor` for description of building’s door components and all real world instances of door fall under this IFC “leaf node”. The detail product specification of each door will be then defined, by specialization of schema elements and adding appropriate property sets.

The IFC model schema provides an abstract modular structure for development of model components. The IFC schema architecture is broken down into four conceptual layers that provide a strict referencing principle. Each conceptual layer in turn contains a set of model schemas. The conceptual layers are organized as follows:

- The first conceptual layer contains Resource classes that are used by classes in the higher levels.
- The second conceptual layer provides a Core project model that contains the Kernel and several Core Extensions
- The third conceptual layer which is also known as interoperability layer provides several modules that contain definition of common concepts and objects. These concepts and objects are used across multiple application types or AEC industry domains.
- Finally, the Domain layer which is the highest layer in the IFC Model provides a set of modules which are tailored for the specific AEC industry domain or application type.

The layers are interrelated and higher layers are dependent on the lower. The architecture operates on a 'gravity principle'. At any layer, a class may reference a class at the same or lower layer but may not reference a class from a higher layer. References within the same layer must be designed very carefully in order to maintain modularity in the model design. Inter-domain references at the Domain Models layer must be resolved through 'common concepts' defined in the Interoperability layer (layer 3). If possible, references between modules at the Resource layer should be avoided in order to support the goal that each resource module is self-contained. However, there are some low level, general purpose resources, such as measurement and identification that are referenced by many other resources (IFC Guide, 2000).

Figure 2.7 shows the conceptual layers of IFC schema of the IFC2x3 final version.

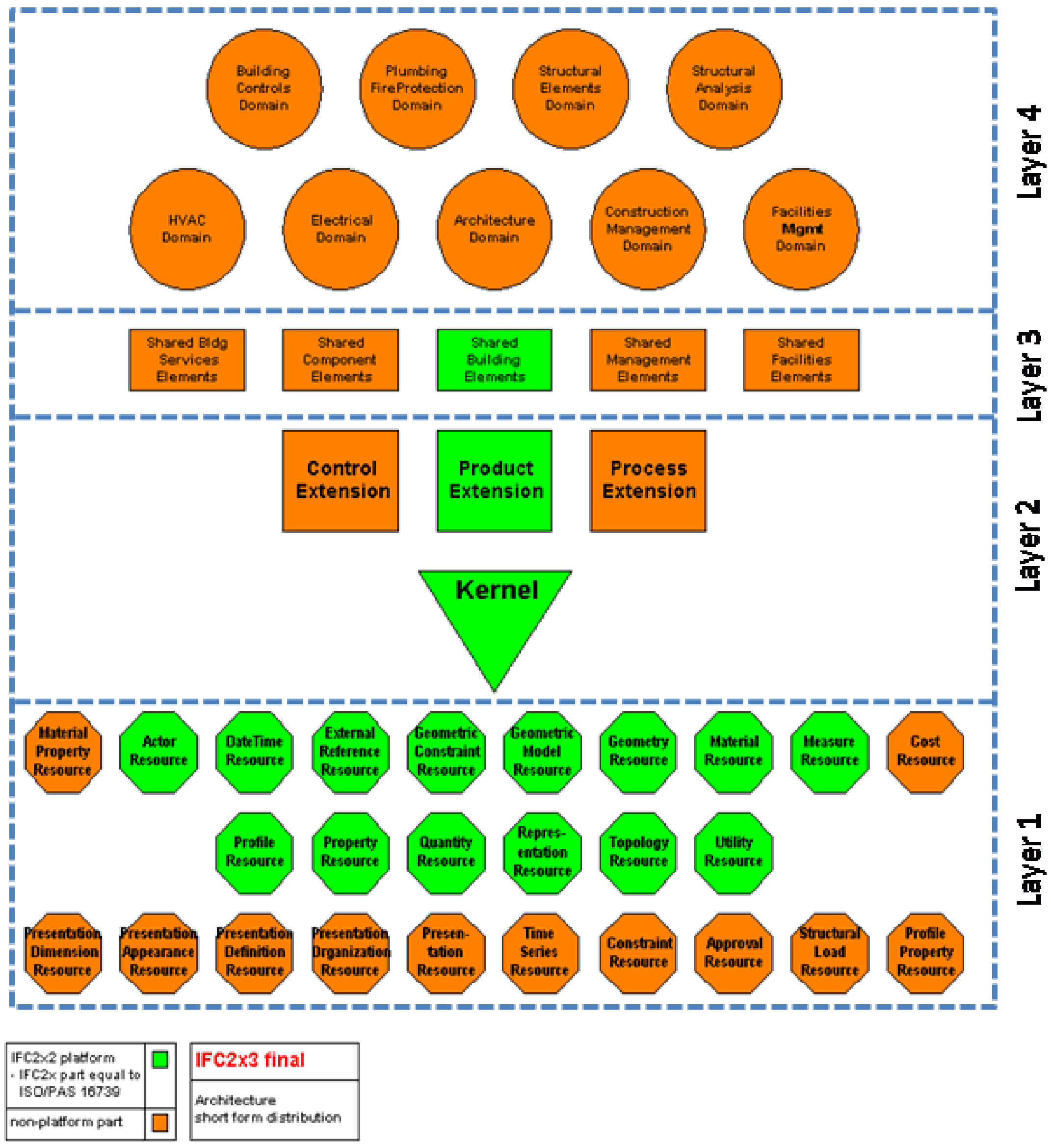


Figure 2.7: IFC architectural layers (IFC2x3, 2008)

Although the IFC information structures have provide a rich formal specification of attributes for IFC entities, but there are always some required attributes that are not currently included within IFC model. For instance, the various types of window with differing numbers of glazing panes, opening types, framing arrangements, etc. additional attributes might be required that are already specified in IfcWindow entity.

The IFC Model facilitates the definition of new attributes via the Property Definition mechanism. Property Definition is a generic mechanism that allows model-users and developers to define, connect and use data-driven, expandable properties with objects (IFC Guide, 2000). Property Definitions can be either:

- type defined and shared among multiple instances of a class, or
- type defined but specific for a single instance of a class, or
- extended definitions that are added by the end users.

Figure 2.8 shows how very highly detailed sets of properties, can be added to the model to extend the description of leaf nodes such as IfcWindow.

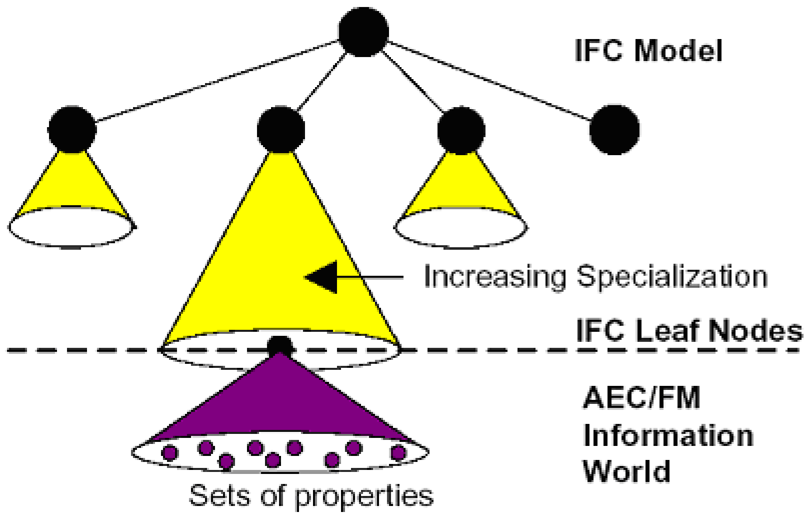


Figure 2.8: property definition in IFC model (IFC Guide, 2000)

2.3.2 IFC Certificates

The IFC specification has been widely accepted in AEC/FM community and many tools support the development of IFC compliant applications (IFC tools, 2008).

The IAI facilitates a certification process where software companies can establish a group to test the IFC interfaces. This group is facilitated by the IAI, but the responsibility for the quality of interfaces remains with the software company. The certification process is split into two phases, in step 1 the IFC interfaces are tested against an agreed set of unit test cases, whereas in step 2 the IFC interfaces are tested against selected project files coming from beta customers (IFC Certified, 2008). In this context, several prominent CAD vendors such as Autodesk, Graphisoft and

Nemetschek have participated in IFC certification program and their software is able to understand IFC and interoperate with other IFC tools. Table 2.2 lists the products that have received the IFC2x3 Step 2 certification (IFC Certified, 2008).

Product	Company	Date
ACTIVe3d	ARCHIMEN Group	13-03-07
ALLPLAN 2006.2	Nemetschek	13-03-07
ArchiCAD 11	Graphisoft	13-03-07
AutoCAD Architecture 2008 SP1	Autodesk	13-03-07
Bentley Architecture 8.9.3	Bentley Systems	13-03-07
DDS-CAD 6.4	DDS	13-03-07
Facility Online	Vizelia	22-05-07
MagiCAD	Progman	22-05-07
ESA-PT	SCIA	25-02-08
Revit Building 2008 SP1	Autodesk	13-03-07
Solibri Model Checker	Solibri	13-03-07
TEKLA Structures	TEKLA Corporation	13-03-07

Table 2.2: List of IFC2x3 Step 2 certified software

The software implementers who successfully pass the certification process can use the certification logos on their websites, printed publications and shipping-boxes. Figure 2.9 shows the first level and second level certification logos for IFC2x3.

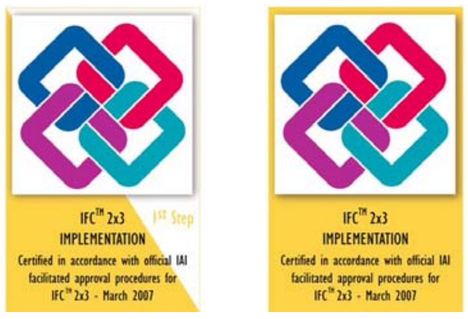

IFC™ 2x3
1st Step
IMPLEMENTATION
Certified in accordance with official IAI facilitated approval procedures for IFC™ 2x3 - March 2007
IFC™ 2x3
IMPLEMENTATION
Certified in accordance with official IAI facilitated approval procedures for IFC™ 2x3 - March 2007

Figure 2.9: First and second level IFC certification logos (IFC Certification, 2008)

2.3.3 IFC View Definition

The latest official release of IFC2x3 includes 117 defined elements, 164 enumerations, 46 select types, and 653 entities. Number of classes in IFC2x4 alpha has increased and many new classes are added.

Since IFC classes should cover a broad range of applications, the number of classes is increasing. To keep the IFC simple for targeted fields, IFC View Definition, or Model View Definition (MVD) has been defined. An IFC View Definition is a subset of the IFC schema that is needed to satisfy the exchange requirements of the AEC industry for specific application types. It represents the software requirement specification for the implementation of an IFC interface. Whereas the general exchange requirement is independent of a particular IFC release, the realization (or binding) is specific to a release.

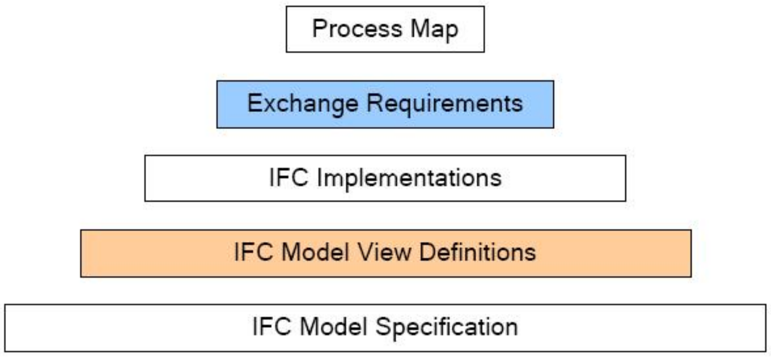


Figure 2.10: Deployment steps of an IFC based solutions (Hietanen, 2003)

The figure above shows the different steps that are needed for creating IFC based interoperable solutions that are successfully deployed in AEC/FM projects. It is like a ‘task list’ for all the things that must be taken care of. The picture is shaped like a pyramid, because the shortcomings of any level limit the possibilities of the levels above it (Hietanen, 2006).

- **IFC Model Specification** is the IFC schema and its documentation
- **IFC Model View Definitions** document how the IFC Model Specification is applied in the data exchange between different application types.
- **IFC Implementations** are the IFC import and export capabilities of software applications
- **Exchange Requirements** document the information that must be passed from one business process to enable another to happen.

- **Process Map** gives an overview of the end user process, describing its objective and describes the stages in a project at which the process is expected to be relevant.

It is important to identify some general trends related to the larger picture. In general lower levels are creating new possibilities for the levels above them. Table 2.3 lists the current MVDs that have been proposed as ideas or proposal for different interoperability goals (MVDs, 2008):

Name	Reference
Architectural design to landscape design	CRC_CI-003
Architectural design to quantity take-off - level 1	VBL-004
Architectural design to quantity take-off - level 2	GSC-002
Architectural design to quantity take-off - level 3	VBL-006
Architectural design to structural design	VBL-002
Architectural design to thermal simulation	VBL-007
Extended coordination view	ISG-001
Extensibility	VBL-003
Facility management inventory data take-over	GSC-001
GSA concept design spatial program validation	GSA-001
Indoor climate simulation to HVAC design	HUT_HVAC-001
Landscape design to road design	CRC_CI-002
Road design to landscape design	CRC_CI-001
Structural design to structural analysis	VBL-001

Table 2.3: Model View Definition proposals

2.3.4 BuildingSMART

BuildingSMART is an alliance of organisations within the construction and facilities management industries dedicated to improving processes within the industry through defining the use and sharing of information. The International Alliance for Interoperability (IAI) defines the buildingSMART as “integrated project working and value-based life cycle management using Building Information Modeling and IFCs” (IFCWIKI, 2008).

The ultimate goal of buildingSMART initiative is to share the Building Information Model (BIM) in an efficient way with AEC/FM stakeholders including architects, engineers, contractors, building owners, facility managers, manufacturers, software vendors, information providers, government agencies and more. Every BIM user on the other hand will view the BIM information from its specific point of view and integrate this specific view with internal processes. Figure 2.11 shows the specific use of shared BIM by various stakeholders.

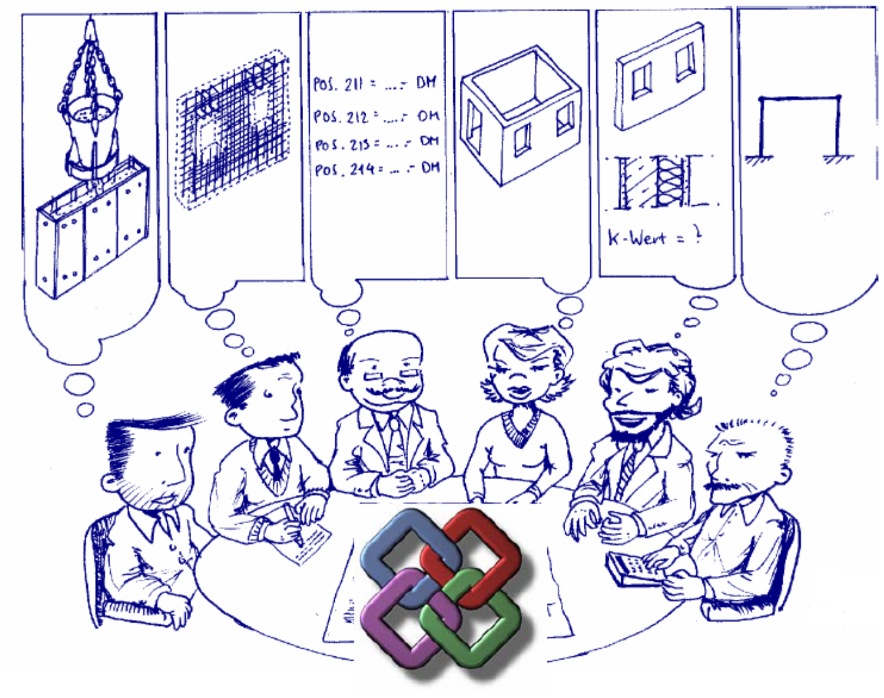


Figure 2.11: information sharing between AEC stakeholders (Junge, 2008)

The focus of buildingSMART is to guarantee lowest overall cost, optimum sustainability, energy conservation and environmental stewardship to protect the earth's ecosystem (BSA, 2008).

Building Information Models (BIMs) that conform to IFC model, build the core of this vision. BIM conveys all required information for the whole lifecycle of the building. The buildingSMART vision will be realized when the following three pillars are in place:

- IFC standard as the exchange format for sharing the information
- IFD as the reference library to define what information are being shared
- Information Delivery Manual / Model View Definition (IDM/MVD) specification to de-fine which information is being shared and when

The IFC pillar and the Model View Definition are already addressed in previous sections. In the following sections the IFD and IDM will be briefly introduced.

2.3.5 IFD: International Framework for Dictionaries

International Framework for Dictionaries (IFD) is an open library, where concepts and terms are semantically described and given a unique identification number (Bell, & Bjørkhaug, 2006). More explicitly the IFD is an ISO standard (ISO 12006-3) that is described using an EXPRESS model with a short explanation of its purpose and use. IFD libraries are more than a simple mapping of words among various languages and provide an abstraction layer that contains the conceptualization of entities. This abstraction layer will then facilitate connecting the entities via shared library concepts. For instance the word "Tür" in German is mapped to the same library concept as the English word "door". So manufacturers may introduce their products in the international market without being hampered due to language issues. The IFD library will handle the representation of products in other languages by aligning the product specifications to the IFC reference model. Moreover an entity might be interpreted differently in different countries and again an IFD can address this issue by providing an abstract reference library. An important role of IFD library is the separation of a concept from its local names and descriptions that define that concept. In IFD this is achieved by separating the concepts from the names and descriptions that are used to name and describe it. As a matter of fact, the IFD library let the concept be both described by multiple name and descriptions and also its relation to other concepts. Figure 2.12 shows the conceptual definition of a window entity and the system that are referencing this concept and its properties.

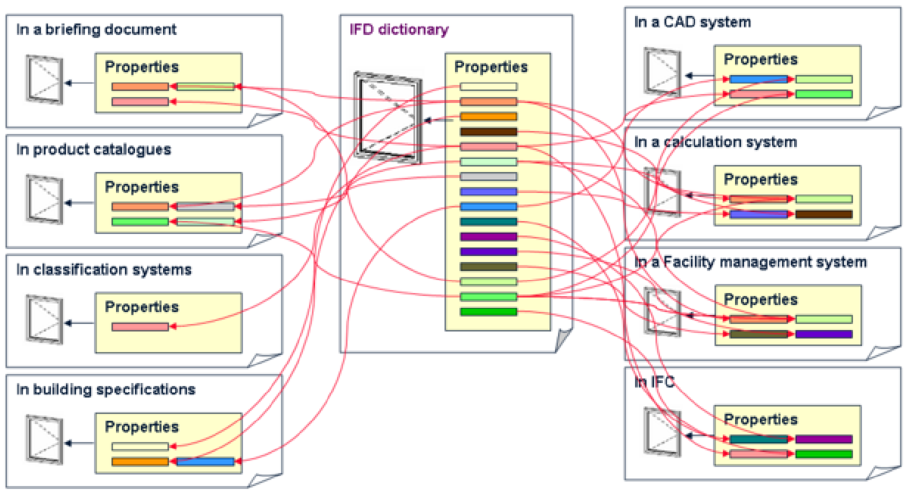


Figure 2.12: A window concept and its properties in IFD (IFD, 2008)

Several countries have started building dictionaries based of IFD. The most important libraries for building smart are BARBi (Bell et al., 2004) and LexiCon (Woestenenk, 2002), which are defined for a better communication between construction partners and for better information handling by computers. Some research has tried to harmonize IFC with IFD structure (Jansen & Wix, 2003). By using globally Unique ID (GUID), all the information in the IFC format can be tagged and the concepts may be defined in any language and can be processed by computers. It means that these GUID are used by machines to process the data and textual descriptions by humans.

Several research and projects have been done over the years on IFC model in order to develop modelling and implementation of AEC objects and to provide more integrated, interoperable and intelligent AEC objects (Halfawy & Froese, 2002). Some of them have tried to provide online product libraries for AEC industries based on IFC and present the architecture for implementation with aim to support industry practices in the production and consumption of product information (Owolabi et al., 2003). This research aims at developing a Semantic based approach to efficiently gather comprehensive knowledge of AEC domains and bridge the gap between products, building model, and AEC tools.

2.3.6 IDM: The Information Delivery Manual

In order to use Building Information Model efficiently, the communication between different stakeholders of building process should improve significantly. The quality of communication is dependent on many factors such as the on-time availability of information and also the quality of information. To achieve this goal, there should be a common understanding of the building processes, their occurrence order and also the exchanged information.

The Information Delivery Manual (IDM) is used to capture the building’s business process and to document the exact specification of the required information for a specific role at a particular time within a project.

2.3.7 IFC current status

Today, IFC model has been widely accepted in AEC/FM community and can be used in smart ways to combine the two dimensional and three dimensional geometry of a building within more complex processes. In this context, the physical information of building geometry would be coupled with building products, costs, and building processes which turns the IFC model to an excellent candidate for share and exchange of building information among IFC-Compliant application software throughout all phases of the building life-cycle. So the errors and losses of data during model transformations would be eliminated and many challenges of collaborative design are avoided.

Despite several research projects on evolution of IFC, this data model and its applications are still being developed and improved continually. For instance the interoperability of IFC models between the IFC-enabled tools is not perfect yet and by data exchange among well-known tools such as ArchiCAD, AutoCAD Architecture, and Allplan, data conflicts and incompatibilities may arise.

To clarify the issue, imagine a simple building plan with two spaces and some doors and windows is created in one of these tools and then it is exported as an IFC model. With an ideal interoperable model, the other tools are expected to interpret and render the model exactly as it was appearing in the first tool and without loss of information. However the experience has shown that in some cases this assumption is not true and the exported model might result in distortions. Figure 2.13 shows an IFC-exported model that has been opened in AutoCAD Architecture. As it is depicted in the picture, the door opening’s relation has lost its relation with door element itself. As a result resizing the door opening will not resize the door element. The same phenomenon happens to windows and window openings.

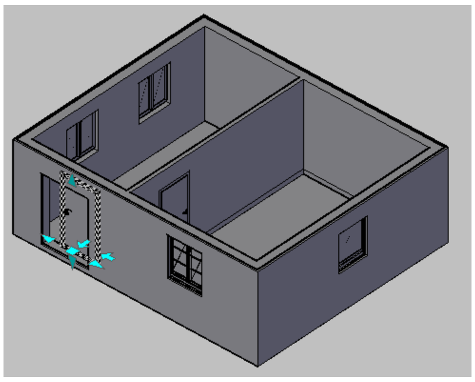


Figure 2.13: model inconsistencies among IFC-enabled tools

As mentioned above, IFC is an effort to provide a commonly accepted standard for support of information sharing between diverse applications in AEC context. In the recent years, it has been supported by many commercial and free software tools and it is clear that it will be the way ahead for more effective and structured interpretability in AEC/FM processes.

Chapter 3

ONTOLOGIES AND SEMANTIC WEB

As mentioned in the previous chapters, the goal of this research is to exploit the Semantic Web technology to bridge the knowledge gap among AEC/FM domains. Since Ontologies and Semantic Web play significant roles in the presented approach of this research, the first part of this chapter is dedicated to introduce these technologies and at the end, the specific application of Semantic Web and Ontologies in AEC/FM will be explored.

3.1 Ontology Fundamentals

The term Ontology has its origin in ancient Greek philosophy and is defined as the study of the nature of being, existence or reality and of the categories of being and their relations. Inspired by this philosophical definition, computer scientists have used this term as a formal representation of concepts within a specific domain and the relationship between those concepts. A comparison of these two definitions shows that in both philosophy and computer science, ontology is a means for the representation of entities, ideas, and events together with properties and relations in a categorized context. Despite this basic similarity, philosophers are less concerned about precise definition of the vocabularies. The computer scientists, on the other hand, started to formalize large and robust ontologies that are being exploited in many fields and used as the common understanding of domain concepts. Nowadays ontologies are playing a central role in capture of domain knowledge and joining the discrete information over organizational borders.

The basic components of ontologies are instances (individuals), concepts (classes), attributes, axioms, and relations. In the rest of this section we will introduce these concepts briefly.

Instances or individuals are representation of concrete objects such as cities, countries, and people, as well as literal values such as words and numbers. It is important to note that ontology does not have to include instances and provide a classification for them; instead ontology may define only the classification of the abstract concepts. For instance ontology can categorize people to “male” and “female” categories and do not include instances such as “Adam” and “Eve” however the provided categories can be applied to those individuals, even if those individuals are not explicitly part of the ontology.

Unlike the instance, classes or concepts provide an abstract classification for real world entities. In the previous example “male” and “female” represent the classes for two groups of objects in real world who are men and women. More formally a class can be defined as an extension or an intension. For the extension perspective, classes are abstract groups, sets or collection of objects whereas the intension definition describes them as abstract objects that are defined by values of aspects that are constraints for being member of the class. As a result a class may classify instances, other classes, or a combination of both.

An important relationship between classes is subclass or subtype which is used to create a hierarchy of classes. A subtype classes will inherit all attribute of their parents. It is important to note that in most ontologies, classes are allowed to have any number of parents which is called multi inheritance and as a result the child class will inherit all attributes of its parents. A common hierarchy building strategy is disjoint partitioning where the partition rule guarantees that a single instance cannot be in both subclasses. For instance the “human” class can be subsumed by “male” and “female” subclasses which form a disjoint partition; i.e. every human should fall only under one of these subclasses with an agreed definition of each subclass.

Ontology classes can also be described by their relation to other entities. The related entities in ontology definition are referred as attributes. An attribute can be a separate class or individual and express a specific fact about the concept that owns the attribute. For instance a “human” concept may have literal attributes such as name, age, etc and also some class attributes such as father, children, and country.

In ontology, relationships specify how object are related to other object. The relationship themselves can be seen as particular classes that clarify the relationship between concepts. Sometimes the relations are also categorized by the type of the entities that they connect to each other. For example some relationships may relate a class to other classes whereas some others relationships connect individuals.

To clarify the introduced ontology concepts and align them with AEC/FM concepts, a simple ontology has been shown in figure 3.1.

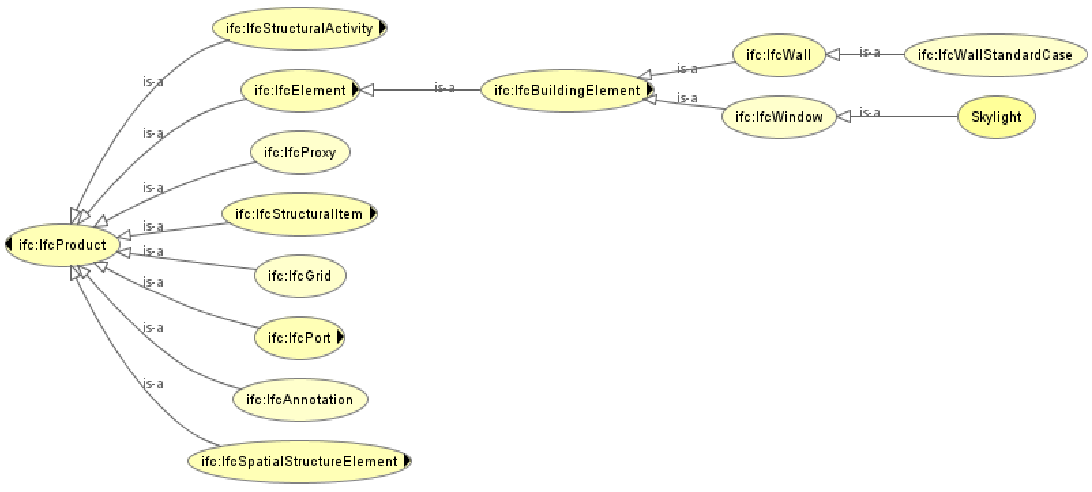


Figure 3.1: An AEC/FM related ontology

Graph visualization is a common method for illustrating ontologies. Figure 3.1 shows the graph visualization of an ontology for classes such as `IFCBuildingElement`, `IfcWindow` and `Skylight`. Please note that, some of class names are preceded by “`ifc:`” which specifies the namespace in which the class has been defined. The namespaces will be explained in more details in upcoming sections and classes are simply being referred without mentioning their namespaces. The `IfcWall` and `IfcWindow` are disjoint subclasses of `IfcBuildingElement` (the disjoint characteristic is not depicted in the figure). Also the `Skylight` class is defined as subclass of `IfcWindow`; i.e. any instance of class `Skylight` will inherit the attributes of an `IfcWindow`. The “`is-a`” relationship is a common relationship that connects the subclasses to parent classes. The only ontology elements that have not appeared in this figure are instances of specified classes which is a common approach in ontology creation. This kind of ontology is formally known as taxonomy pattern and ontology captures the data structures and relationships between concepts. As a result the domain knowledge is in the ontology itself. The other ontology pattern which is called schema pattern usually has a simple ontology with lots of instances. In this group of ontologies the domain knowledge is in the data and not ontology. An example of schema pattern is a library of different skylight types which have a small ontology for describing skylights and their physical attributes such as height, width, U-value, etc and thousands of skylight products that follow that schema. The schema ontology is a best practice for addressing the requirements of enterprise and information exchange use cases.

Besides the internal structure of ontologies, the scope of an ontology is also an important factor. From this perspective, ontologies are divided into domain ontologies and upper ontologies. Domain ontologies describe a specific domain and all concepts apply to the specific domain only. For instance the concept “Window” in an AEC ontology is interpreted as a window component, however the same concept in a user interface ontology is interpreted as a graphical presentation entity. Depending on the specific domain ontology any of these interpretations would be possible. An upper ontology or foundation ontology, on the other hand, deals with the common model of objects that is applicable to large number of domain ontologies. A well-known upper ontology is Dublin Core (Dublin Core, 2008) ontology that describes a standard for cross-domain resource description by means of simple concepts such as resource title, resource creator, publisher, format, etc. The upper ontologies facilitate the information join between multiple domains and play a significant role in information sharing.

3.2 Ontology Languages

In order to construct ontologies and encode the knowledge of a specific domain, some formal languages have been invented which are referred to as the “ontology languages”. These languages often facilitate definition of rules for knowledge processing purposes too, and are commonly built upon generalization of first-order logic (FOL, 2008) or description logic (DL, 2008). The comprehensive description of these logical frameworks is out of scope of this dissertation and in this section just some well-known ontology languages such as OWL and RDF will be introduced in more detail.

3.2.1 RDF

The Resource Description Framework (RDF) is a metadata model specification that has been designed by the World Wide Web Consortium (W3C). The first release of RDF was introduced in February 2004 as a W3C recommendation (RDF, 2004). The basic principle of RDF language is the usage of subject-predicate-object model to capture knowledge about web resources and share it for different usages. It is worthy of note that the RDF language can be used to capture knowledge of any domain and its application is not limited to web resources. In RDF terminology the model elements (subject-predicate-objects) are called triples.

For example the fact that a specific skylight component has Solar Heat Gain Coefficient (SHGC) of 0.33 will be modelled as follows:

- Subject (resource) : skylight name (for instance Velux-74 glazing)
- Predicate (property) : has SHGC
- Object (value) : 0.33

RDF triples are also visualized as a directed label graph, where subjects and objects are shown as a vertex and the predicates are depicted as an arc directed from subject to object which is labelled by predicate. Figure 3.2 depicts the previous example in graph form.

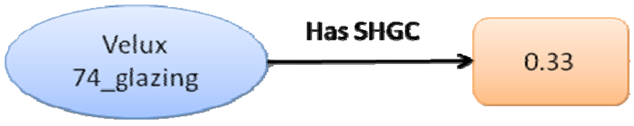


Figure 3.2: Graph visualization of a triple

An RDF document contains a number of triples and creates precise relationships between vocabulary items or resources. The RDF documents and their internal structure can be reused in other domains and this established a powerful mechanism to join the information of resource from different domains. As a result the resources can be combined and processed as if they come from a single domain. To manage the complexity of inter-domain references, all triple items should be identified by qualified names which are known as QName. A QName contains a prefix that has been assigned to a specific domain, followed by a colon, and then the local resource name. This QName's prefix which is also known as namespace is a unique identifier for a specific domain and usually refers to a Uniform Resource Identifier (URI). URIs will be explained in more details in the forthcoming sections. As a result, resources such as building, location, architect, skylight, etc and the RDF properties such as shape, color, city, title, etc are uniquely identified by a URI. The value or object part can also be identified by a URI or contain simple literal values such as 0.33, "Vienna", etc. It is important to note that some value (object) nodes have no URI or literal value; but they will add extra triples to the parent resource. This kind of node is known as blank nodes. Figure 3.3 is an improved version of skylight triple with an extra blank node for skylight producer.

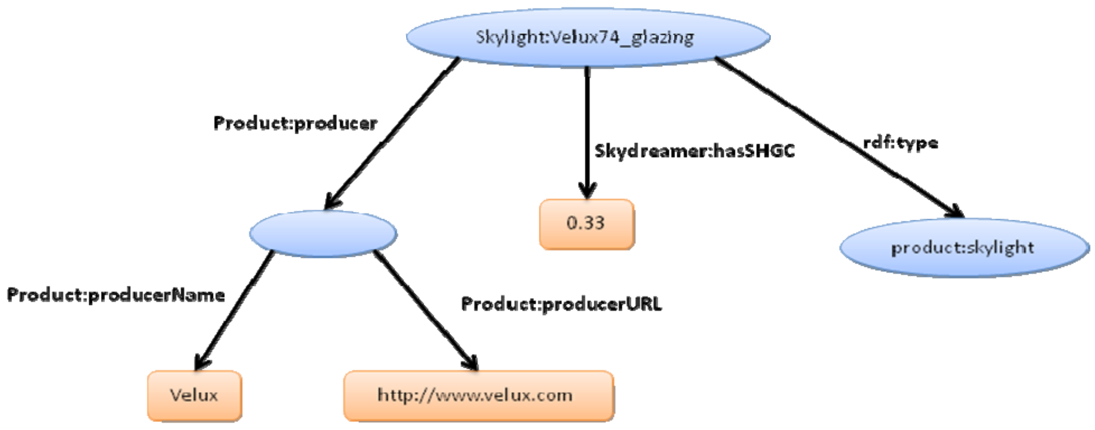


Figure 3.3: Skylight example with a blank node

As mentioned above and depicted in Figure 3.3, any of the subject, predicate, or object can be originated from various domains or equivalently from different namespaces. The textual convention for presenting the triples and name spaces is shown in listing 3.1.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix product: <http://product.info/skylights/>
@prefix skydreamer: <http://buildingsmart.at/skydreamer/>
```

```
<product:Velux-74_glazing> <rdf:type> <skydreamer:skylight>;
<product:Velux-74_glazing> <skydreamer:hasSHGC> “0.33”
```

Listing 3.1: Skylight example with various namespaces

The first three lines of listing 3.1, define namespaces for RDF, skylight products and SkyDreamer. At the end there are two triples with items that are coming from various namespaces. The triples express that the Velux-74_glazing resource which is a skylight component has the SHGC value of “0.33”.

The RDF language introduces a flexible and expressive data model. Compared to the relational knowledge representation and also traditional ontology models, RDF is simpler to comprehend both for computers and human users. In practice, RDF data is often persisted in relational database or native representations also called triple stores.

3.2.2 RDF Schema

The RDF Schema or RDFS is an extension of RDF vocabulary for building taxonomies and describing light-weight ontologies. RDFS defines the recursive concept of “rdfs:class” to declare a resource as a type for other resources. Moreover classes can be related to other classes by “rdfs:subClassOf” that declares the inheritance between parent and child classes. Another advantage of RDFS is definition of domain and range for RDF predicates via rdfs:domain and rdfs:range properties. The rdf:domain property describes the resources that can be used as subject of the selected predicate. Likewise, the rdf:range defines the object resources that can be used in triples that contain a specific predicate.

For instance, the Skylight concept from the previous example inherits the Window class and may extend the triples as shown in the listing 3.2. Please note that the notation used in the listing 3.2 is slightly different and is a shorthand substitution of the full triple notation. In this notation the triples with the same entity as subject are summarized into a single statement (for instance skydreamer:hasSHGC).

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>
@prefix ifc: <http://www.buildingsmart.at/ifc-schema#>
@prefix skydreamer: <http://buildingsmart.at/skydreamer/>

skydreamer:Skylight rdf:type rdfs:Class ;
    rdfs:subClassOf ifc:Window .

skydreamer:hasSHGC rdf:type rdf:Property ;
    rdfs:domain skydreamer:Skylight ;
    rdfs:range xsd:decimal ;
    rdfs:comment "Solar Heat Gain Coefficient" .

skydreamer:Velux-74_glazing rdf:type skydreamer:Skylight ;
    skydreamer:hasSHGC "0.33"^^xsd:decimal .
```

Listing 3.2: Skylight using DRFS extensions

The expressive power of Resource Description Framework (RDF) and RDF Schema (RDFS) is very limited. RDF is roughly limited to binary ground predicates and RDF Schema is roughly limited to a subclass hierarchy and a property hierarchy with domain and range definitions (Berners-Lee et al., 2006).

The main modelling primitives of both RDF and RDFS are concerned with the creation of taxonomies and forming the class hierarchies. However, additional features such as disjoint classes, equivalent classes, and specific type of restrictions are needed to facilitate the knowledge capture of ontology domains. These additional requirements are addressed by OWL which will be explored in next section.

3.2.3 OWL

The semantic of RDF and RDF Schema is not enough to achieve useful reasoning task on web documents and there is still the need for a more expressive language. Web Ontology Language (OWL, 2004) is another ontology language that has been developed by W3C for defining and instantiating domain ontologies. An OWL ontology (an ontology that has been defined in OWL language) may include descriptions of classes, properties and their instances. Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics (OWL guide, 2004).

Using OWL, the meaning of terms and their relationships can be represented explicitly. As explained before, such a representation of terms and their interrelationships is called ontology. Compared to RDF, OWL provides additional vocabulary for describing properties, classes, and instances among others, relations between classes: (e.g. disjoint classes), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. Therefore provides a better machine interpretability than RDF.

OWL has three increasingly-expressive sublanguages (OWL guide, 2004):

- OWL Lite that provides a classification hierarchy and simple constraint features. For example, an “Opening” entity can be classified to “Window”, “Door” and “Skylight”. Moreover it permits the cardinality constraints and can restrict the cardinality to 0 or 1.
- OWL DL (Description Logic) as the superset of OWL Lite, provides more expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems. OWL DL is so named due to its correspondence with description logics, a field of research that has studied a particular decidable fragment of first order logic. OWL DL was designed to support the existing Description Logic business segment and has desirable computational properties for reasoning systems.
- OWL Full is the super set of OWL DL and provides the maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support every feature of OWL Full.

3.2.4 Ontology Query Languages

In addition to modelling languages, Ontologies also need a query language to utilize the information represented by ontology languages for various applications. There are a couple of such query languages. In this section we will briefly introduce the SPARQL query language (SPARQL, 2008) that has been widely used in the proposed solution of this dissertation. SPARQL term is an acronym that stands for Simple Protocol and RDF Query Language. The language can be used to describe queries across RDF data sources. The SPARQL query syntax is very similar to RDF notation and the SPARQL query processor will parse the given triple pattern of query and returns the sets of triples that match that pattern.

Listing 3.3 shows a SPARQL query for extracting product name of all skylight components that has the specified values for SHGC, VT (visual transmittance) and U-Value (these attributes and their meaning will be discussed in the forthcoming sections).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ifc: <http://www.buildingsmart.at/IFC2X3.owl#>
PREFIX sl: <http://www.buildingsmart.at/Skylight.owl#>

SELECT ?sl ?name ?vt ?shgc ?ufact WHERE {

   ?sl rdf:type sl:Skylight .
   ?sl sl:hasShgc ?shgc .
   ?sl sl:hasVt ?vt .
   ?sl sl:hasUfactor ?ufact .
   ?sl ifc:IfcRoot_Name ?name .

   FILTER(?shgc <= 0.3 && ?ufact <= 0.7 && ?vt >= 0.5)
}
```

Listing 3.3: Sample SPARQL query

The given query in listing 3.3, lists the skylight and its relevant attributes such as SHGC, VT, and U-Value that fall in the provided range of values. The basic query part here is the triple patterns that combine variables (names with '?' prefix) with ontology entities. These patterns will chain together and run on RDF graph to result the set of matching triples.

3.3 Semantic Web Vision

In the previous sections Ontology concepts and relevant ontology description and ontology query languages were discussed. In this section, the Semantic Web will be introduced which has boosted many improvements in the ontology world. The Semantic Web is an evolving extension of existing World Wide Web. In contrast to the traditional web of hypertext documents, the Semantic Web is considered as a web of data, in which information is given well-defined meaning to make them interpretable and processable by machines. The Semantic Web promises to bring structure to the Web through common formats for integration and interchange of data drawn from diverse sources (Berners-Lee et al., 2005).

Semantic Web also intends to improve knowledge sharing which is the key factor to success in collaborative environments. Thanks to the development of technology, it's now easy to have access to lots of information which are available on the internet. The Internet has eliminated distances as a barrier to sharing information at a worldwide level and enables better communication. Although invention and development of internet facilitated sharing and reusing of information for the people but all this information are only human comprehensible, and they cannot be processed without human intervention. In other words, the mechanical actions of commuters will result in some results that should be interpreted by human to extract the required information.

The traditional Web is based on HTML (Hypertext Mark-up Language) which describes the way that data (web content) should be rendered on browsers, but it does not contain any means of interpretation for the semantics of web contents. As a result, the computers are unable to process the meanings and perform more complex tasks as human user does. For instance, consider the case that an architect runs a search for words "architecture" and "ontology" on a search engine. Since the computers are unable to understand the meaning of query terms and bind them to the architecture knowledge domain, the result will include many irrelevant items about ontology software components and architecture. Semantic Web tries to overcome this issue by annotating the web resources with relevant ontologies which means the resources are aligned with a known concept in an ontology that can be interpreted by computers. By adding such information one will be able to search for that specific concept and find the exact required information.

The more computers understand the meaning of data and the relationship between them, the more successful is the interoperability between applications. Imagine how humans can comprehend sentences, how the meaning of a single word differs in sentences or in a set of sentences. We learn lots of word over the years and for each of them we store a meaning in our mind. Each word has some properties and through these properties they are related to other words and meanings in our mind. For example, we hear about an opening in a wall, it is already clear that the opening will be related to a door or window with the same dimensions.

There are many languages, which are spoken around the world. Though all people have stored the same image for a specific concept in their mind, they would not understand words or sentences in a foreign language if they have not learned it before. The people should know a common language to communicate with each other. The same is true with software and applications on computers. The more complete is the definition of this language and the property of each object and the relationship between them, the more powerful and successful the communication between the applications and the exchange of information. By structuring information with such a language, not only users are able to make sense of it, but also it would be processable by machines.

Another shortcoming of the current web is the integration of data that are spread over several web pages. The only means of traditional web that connects the information together is the HTML hyperlinks and URLs. Hyperlinks are a powerful method for connecting the web pages; however it lacks the capacity for description of the resources that are being connected on World Wide Web.

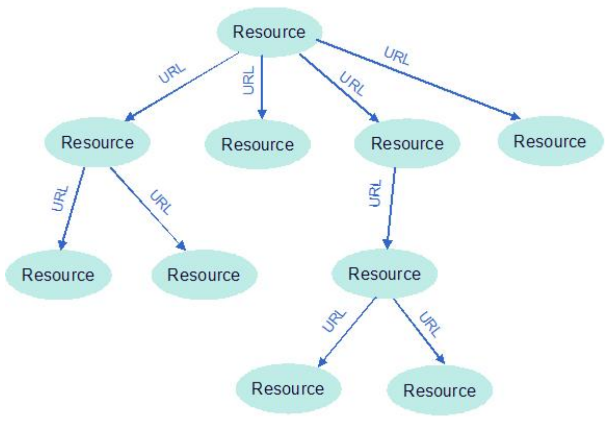


Figure 3.4: Data structure in traditional web

The objective of the Semantic Web is to provide the best framework for adding logic, inference, and rule systems to the web. If an engine of the future combines a reasoning engine with a search engine, it may actually be able to produce useful results. Achieving powerful reasoning with reasonable complexity is the ultimate goal for the Semantic Web because it will lead to machine processing and services automation on a global scale. The challenge is finding the best layering of ontology, logic, and rule mark-up languages for the Semantic Web that will offer solutions to the most useful web information processing. (Berners-Lee et al., 2006)

Then Semantic Web vision is based on multiple technologies that make realization of this vision possible. These technologies are captured in the famous Semantic Web layer cake of Tim Berners-Lee (the inventor of the World Wide Web) as shown in figure 3.5 which illustrates how Semantic Web languages are built upon XML and climbs up the mark-up language pyramid to RDF and OWL. Brief descriptions of technologies in these layers are mentioned below.

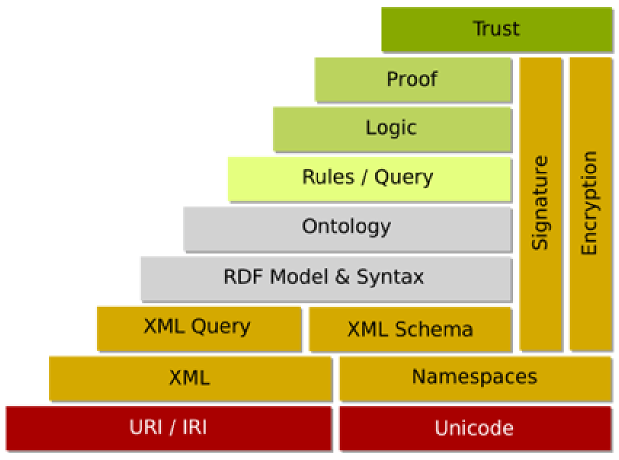


Figure 3.5: Semantic Web Cake Layer

Unicode and URI are the lowest building block of Semantic Web. Unicode is the universal standard encoding system and provides a unified system for representing textual data. The Uniform Resource Identifier (URIs), provide a standard way (using a specific protocol) for identifying any resource in the World Wide Web and is the basis of data interchange in the Semantic Web. In the previous section we have seen many examples of URIs for identifying ontology elements. Each URI belongs to a namespace (second layer of Semantic Web cake) and it's possible to have resources with the similar names but in different namespace. For instance, the "Window" is interpreted differently in a building ontology where it depicts the real work window components and in the user interface ontology where it represents the software frame in a graphical operating system.

In the second layer of Semantic Web cake, XML appears. XML is the acronym for the eXtensible Markup Language and describes a simple and flexible text format for information exchange. XML is the W3C recommendation and allows data to be exchanged between different applications. Using XML lets people to use their own tags (not predefined) and to add their arbitrary structure to their document (tree structure). It is worthy of note that, although the computer can parse and extract different parts of an XML document, it has no idea about semantic meaning of concepts and their relations. For instance, if a human user reads the XML snippet in listing 3.4, he/she will infer that "The window's frame is made of wood", but machine is unable to make this inferences due to lack of semantics. The only thing that machine will learn out of this XML snippet is that the Window element has a sub-element which is named frame_material and its value is the string “wood”.

```
...
<window>

<frame_material>wood</frame_material>
</window>
...
```

Listing 3.4: Simple XML snippet

The limitations of XML documents, for comprehensive data exchange between systems are as follows:

- Pre-arranged agreement on vocabulary is needed; i.e. a human user should configure both systems to interpret the XML document properly. As a result XML documents are reasonable for closed collaborations between partners and not for global share of resources.
- Lacks the capability to capture the semantic relation between entities. For instance the fact that “Window is-a Building element” cannot be formulated in an XML document.

Even though computers are not able to interpret semantics of XML documents, the well-defined information structure facilitates the complex data conversion and data processes. For this purpose the eXtesible StyleSheet Language (XSL, 2008) is used to describe how XML data should be transformed to other XML formats. For example an XSL document can be applied to an XML document to transform it to an HTML page (which is also an XML format) for representing the data on the web.

The Next layer of Semantic Web cake is XML Schema and XML Query. The XML Schema (XML Schema, 2008) which is also a W3C recommendation provides a means for defining the structure and data type of XML elements. The XML Query component facilitates the search mechanism on XML documents. This includes definition of query languages such as XML Path Language (XPath, 2008) and also database-like solutions for storage of XML documents and execution of XML queries.

The next two layers of Semantic Web, namely RDF model and ontology and also semantic query of the next layer have been already discussed in previous sections and as described before they heavily use the lower layers to build, capture and query the domain ontologies.

From the remaining layers, only the rules and logic layers will be explained and the higher layers that are not the main concern of this dissertation are bypassed.

Basically the semantic rules are statements in the form of an if-then (antecedent-consequent) sentence that describes the logical inferences that can be drawn from an assertion in a particular form (ontology, 2008).

Rules are one of the main concerns in Semantic Web world and in the last few years there has been a significant progress in that area. This includes standards proposals and efforts for creating rule description languages such as RuleML, SWRL, and recently W3C Rule Interchange Format. Moreover there are a handful of open source and commercial rule engines that can apply the rules to underlying ontologies and deduce new facts. This process is also known as semantic reasoning and will be explained by means of an AEC use case in the following section.

3.4 Semantic Reasoning

Semantic reasoning is the inference of logical consequences from a set of asserted facts or axioms. To demonstrate the usage of reasoning in AEC/FM field, consider a building map with a two rooms as shown in figure 3.6. By looking at this map a human can easily interpret that these two rooms are directly accessible to each other. In order to clarify the situation for a computer and enable it to interpret the same results as human, it is needed to capture the domain knowledge and then exploit a logical framework. For the first task a building ontology may be used that contains the building entities and their relations (the process of building this ontology will be explored in the forthcoming sections).

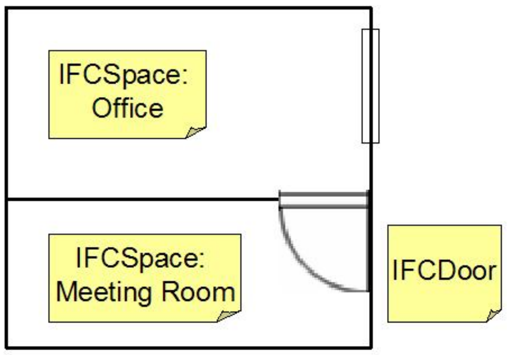


Figure 3.6: simple reasoning scenario

In order to fulfil the reasoning requirement, some rules should be defined on top of the domain ontology as shown in listing 3.5.

```
Rule 1 : [
   hasDirectAccess: (?s1 ifc:hasDoor ?d1)
   (?s2 ifc:hasDoor ?d1)
   notEqual(?s1,?s2)
                  -> (?s1 ifc:hasDirectAccess ?s2)
]
```

Listing 3.5: Simple XML snippet

The above mentioned rule, simply describes the fact that two rooms are directly accessible to each other, if they are sharing a door. Moreover the rule explicitly avoids the direct access of a room to itself by applying the “not equal” rule for any two given spaces. After applying this rule some new triples for the adjacent rooms with direct access to each other will be added to ontology.

3.5 Semantic Web Services

In Information Technology world, the problem of linking applications from heterogeneous environments, which is more or less similar to AEC knowledge domain problems, is addressed using Service Oriented Architecture (SOA) and Web Services. In AEC field also there are some research works toward realizing SOA environments where, there is no need to install and upgrade several tools (e.g. energy, acoustic, structural, cost analysis) on desktop computers. Instead, all applications are available on the web as web services and they would communicate freely with each other and project participants would access to a much broader network of technical resources than is possible today (AECBytes, 2008). A complementary standard to Web Services is Web Service Description language (WSDL) which is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information (WSDL, 2008). End-user / system can use the service description to communicate the service correctly, however selecting suitable services needs the human interaction, i.e. a person should explore the description of web services and make the decision about services that would best suit the requirements of each single use case. The missing element in this scenario is again the semantic information about services and the fact that the WSDLs describe the web services syntactically and not semantically.

In order to overcome this deficiency, web services are additionally equipped with semantic definitions that are shared among the participants of defined domain by means of Semantic Web technologies. Semantic Web Services on the simplest case may define the service's input and output parameters by mapping them to a known shared ontology. Figure 3.7 depicts a simple service that makes the costs calculations for lighting and the service profile is annotated by an IFC ontology.

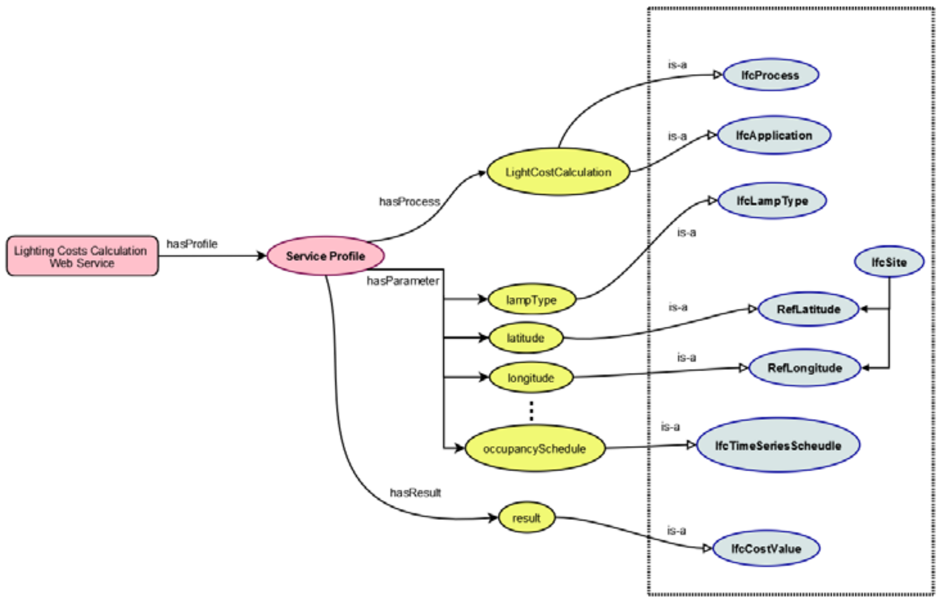


Figure 3.7: Aligning the web service parameters with an IFC-based ontology

3.6 Semantic Web Tools

Several tools are in use for creation of ontologies and metadata. In this section, the tools that have been used for the proposed solution of this dissertation will be introduced briefly.

3.6.1 Protégé Ontology Editor

Protégé (Protégé, 2008), is a free open source ontology editor and a knowledge-base framework based on Java. Ontologies can be modelled in two main ways i.e. protégé-frames and protégé-OWL editors. Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and XML Schema. Plug-ins can be used to change and extend the behaviour of Protégé. In this research, protégé has been used for creating and editing the required ontologies.

3.6.2 Jena Semantic Web API

Jena (Jena, 2008) is an open source Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. Jena is grown out of work with the HP Labs Semantic Web Programme. The Jena framework includes:

- An RDF API
- Reading and writing RDF in RDF/XML, N3 and N-Triples
- An OWL API
- In-memory and persistent storage
- SPARQL query engine

3.6.3 Joseki

Joseki (Joseki, 2008) is a WebAPI for the remote query and update of RDF models and comprises of the following components:

- A client API providing convenient access to the update and query operations of the RDF WebAPI protocol.
- An RDF server, which can run embedded in an application, as a standalone program, or as a web application inside a suitable application server. It provides the operations of query and update on models it hosts.

Joseki supports the SPARQL Protocol and the SPARQL RDF Query language (SPARQL, 2008).

3.7 Applications of Ontology in AEC/FM field

In order to provide interoperability, it is needed to provide a common syntax for machine understandable statement, creating common vocabularies agreeing on a logical language using the language for exchanging proofs (Studer et al., 2003). This requires formal and explicit specifications of domain models, which in the semantic web terminology are called ontologies. Ontologies will present the domain concepts and the relationships between those concepts, by providing a controlled vocabulary of concepts. As the first step toward using Semantic Web technologies, there should be a unified ontology that reflects the requirements of AEC/FM domain and empower the computers to interpret domain concepts in a formal way.

At the time this dissertation started, there was no significant research that practically deals with application of Semantic Web in the AEC/FM context. As Semantic Web technologies gets elaborated steadily and attracts more attention in scientific community, the AEC/FM industries have also slowly started to take advantage of these technologies and recently, many research activities are trying to exploit ontology applications and utilize its potentials to address the interoperability issues. One of the recent research works in this context is the European Semantic Web-based Open Engineering Platform project (SWOP, 2008). The project introduces a Product Modeling Ontology (PMO) that aims to be a fully generic, freely reusable, “upper ontology” specified in the OWL language (Böhms et al., 2008). PMO is supposed to contain, in a necessary and sufficient way, all constructs to define any end-user product ontology, modelling all relevant end-user’s product classes, properties and interrelationships (in particular specialization and decomposition) together with cardinalities, data types, units and default values. Another interesting point in the SWOP project is that the product configurations are optimized by applying Genetic Algorithm so that the resulting product is not just a valid solution, but even near an optimal solution.

Generally, the Semantic Web approaches in AEC/FM are generally aiming to:

- facilitate machine understanding of domain knowledge as well as their complex interdependencies
- bridge the gap between knowledge domain and across heterogeneous construction documents
- ease the integration tasks and reuse of knowledge with loss of information
- providing customised views for project partners without altering the shared model (ontology mapping)

In this section the potential semantic resources to establish AEC/FM ontologies will be discussed and at the end the ontology creation process of this dissertation will be explored. While in the other knowledge domains the ontologies should be created from scratch, the AEC/FM owns some potential resources such as domain dictionaries, schemas and taxonomies that can be transformed to a formal ontology structure. The advantage of this approach is that the created ontology remains compatible with all existing domain models that have been built on top of these domain dictionaries. The most notable efforts include the following (Rezgui, 2006):

- The BS6100 Glossary of Building and Civil Engineering terms, produced by the British Standards Institution (BSI), the independent national body responsible for preparing British Standards. This is a rich and complete glossary that provides a comprehensive number of synonyms per term that can contribute towards any ontology development effort in the sector;
- The bcXML (eConstruct, 2001) is an XML vocabulary developed by the eConstruct IST project for the construction industry. The bcXML provides the foundation for the development of the bcBuildingDefinitions taxonomy, which can be instantiated to create catalogue contents. Through bcXML,

eConstruct has enabled the creation of “requirements messages” that can be interpreted by computer applications and then find suitable products and services that meet those requirements;

- The ISO 12006-2 (1999) is concerned with current classification needs and builds on the experience of developing and using conventional classification systems;
- The IFC model, developed by the IAI, has produced a specification of data structures supporting an electronic project model enabling data sharing across software applications; and
- The OmniClass Construction Classification System (OCCS) developed in Canada by the Construction Specifications Institute, addresses the Construction industry’s information management needs through a coordinated classification system.

Among these semantic resources, IFC deserves a particular attention due to the wide acceptance in AEC/FM domains. In this dissertation the IFC has been used as the basis for establishing the required domain ontologies. There are similar research works that have explored the challenges of converting IFC to web ontology languages (Schevers & Drogemuller, 2005), however these approaches are missing the proper integration with the real world building model and providing a mechanism to add class instances (building components) to abstract ontology model. The process of creating this ontology and mapping the building model to ontology schema is as follows:

Step 1: The IAI distributes the IFC specifications as EXPRESS (ISO 10303-11) files. This files are transformed it to OWL by a translator called e2ont (e2ont, 2008). The resulting OWL file (IFC OWL Schema) that contains some errors and logical inconsistencies needed to be refined and corrected to be usable in the proposed prototype.

Step 2: To combine the OWL schema of previous step with real world building models, the schema classes should be instantiated accordingly. The OWL schema together with instances and their relationships constitute the ontology of the corresponding building model. In order to obtain instances, the IFC model of the building is required. This model can be generated using the handful of tools that support IFC format. In the case of this dissertation, the ArchiCAD design tool has been used that supports the IFC via its IFCXML Add-in. The building model is exported as IFCXML (ISO 10303-28) which facilitates the data processing and model translations.

Step 3: In order to translate the building model to OWL compliant instances, an XSL transformation is used that reads the IFCXML file and convert it to OWL instances. This transformation needs the appropriate XSL style sheet that explicitly defines the mapping for each building element. In the use case of this dissertation, a small portion of building model namely the space dimensions and skylights are required and the transformation is done via specific building model parser that has been implemented as java classes.

Step 4: finally the instances will be added to IFC OWL Schema and use case specific requirements can be implemented using the Semantic Web Rules. The resulting model can be put to work by a Semantic Web reasoning engine such as Jena Reasoner for further queries and inferences.

Figure 3.8 depicts the above mentioned steps and how a building ontology model is formed.

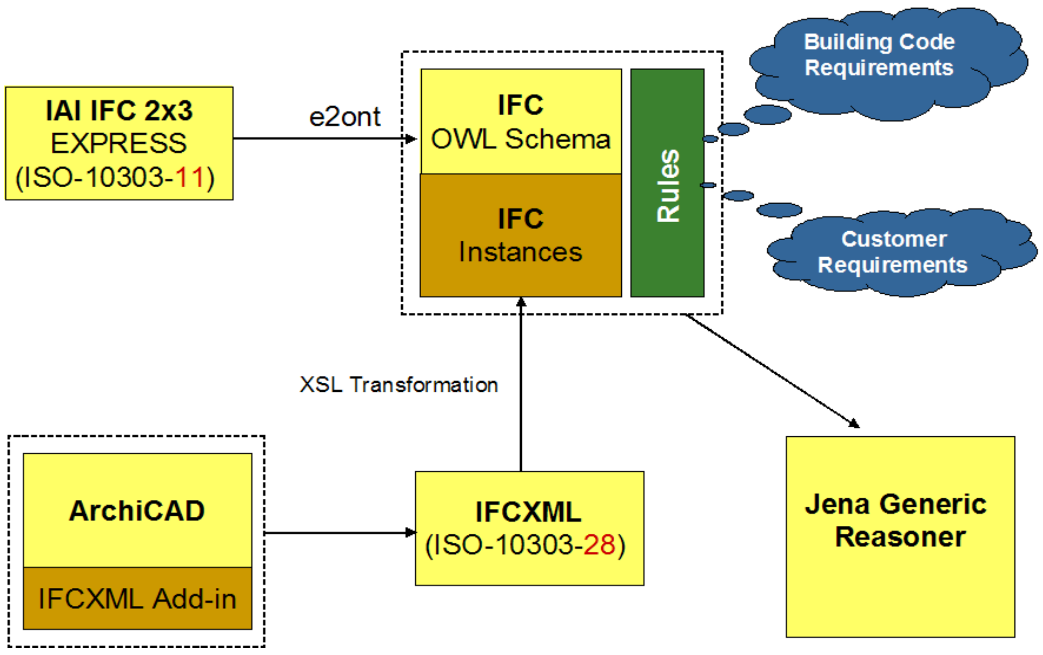


Figure 3.8: Building ontology model

The details of the described method is not explained here, however the interested reader may follow these steps and explore the corresponding file formats in appendices.

Figure 3.9 shows a simplified building ontology model to demonstrated different parts of this ontology. Please that for sake of simplicity, this ontology is not IFC-complaint, however from the conceptual point of view there is no difference between this ontology and an IFC-compliant one. The black boxes in this figure show the classes that are being translated from EXPRESS distribution (STEP 1). The red boxes depict the instances that are added to ontology model by applying the XSL transformation to IFCXML format of corresponding building (STEP 2 and 3). The arrows between instances and classes are tagged with “io” tag which stands for “is of” type and show the instantiation of corresponding classes. Finally the blue arrows show the inferences that are added to the model via semantic rules.

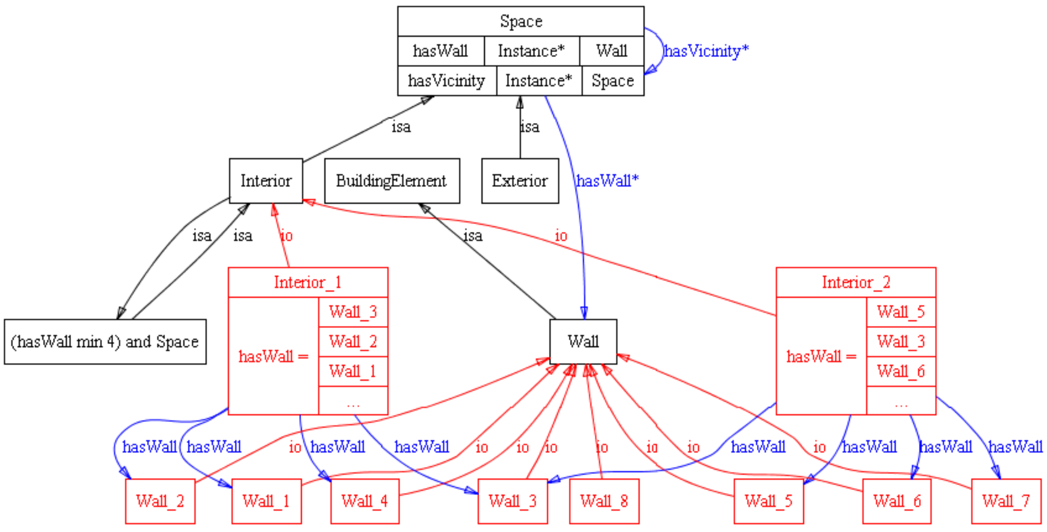


Figure 3.9: graph view of ontology model

Chapter 4

SKYDREAMER PROTOTYPE

The ultimate goal of this research as stated before is to envision, design and prototypically implement a versatile information retrieval and management environment to bridge the gap between disperse knowledge domains of Architecture, Engineering and Construction by using Ontologies and Semantic Web technology. The AEC knowledge domains are formed corresponding to the needed skills and professions. Despite several efforts, there is still no efficient information communication between domain applications. The building design process relies on large databases that are often managed by human knowledge and interactions. However, information from such databases can only be utilized in specific projects if contextual parameters such as country-specific standards and policies are considered. Effective mapping of relevant contextual attributes onto available building industry information has a formidable potential to improve the design process and design options and alternatives could be more readily assessed and compared by providing semantically enriched building models to evaluation applications (e.g. performance simulation programs).

During the design phase, architects and engineers must make some critical decisions about building components and materials to be used. Provision of computational support for this decision making process would benefit the AEC stakeholders in view of cost reduction, energy efficiency, and occupants' comfort and productivity.

To illustrate such a multidimensional decision making problem, this research will focus on the example of a specific decision making scenario, namely the selection of skylight product. According to this scenario, the selection of a proper skylight product is dependent on multiple factors such as client requirements, space functions, visual and thermal requirements, structural constraints, design concepts, and budget. Any decision toward selecting a specific skylight product must not only comply with the applicable requirements and criteria, but also be evaluated in view of resulting performance (energy, daylight, etc.). Note that the specific application in this case (skylights) and the associated computational tools (energy calculator) serve here as illustrative instances. As such, similar processes can be implemented for other – and more realistic – application scenarios in building design and construction domain.

Today, there are many different media (catalogues, CDs, internet), which entail information on building products and their relevant technical information (Mahdavi et al., 2004). However, most of this information is neither in a standard format nor machine processable (in a semantic way). The use of IFC as the common template for information sharing (Halfawy et al., 2002) and the application of Semantic Web Services as the communication method can address this problem.

4.1 Use case scenario

One of the most important aspects of a building design is lighting, which is a combination of electric lighting and daylighting. Achieving efficient daylighting in a building depends on many factors such as climatic condition, solar orientation, sky conditions, the orientation and location of fenestration and the size and material of windows and transparent media. An efficient daylighting affects light quality and illumination levels of indoor spaces. Moreover it is a significant architectural feature in interior design that on one hand, enhances the appearance of building's spaces and on the other hand, provides a comfortable and pleasant environment that guarantees the satisfaction of occupants.

There are variety of daylight-related devices such as windows, skylights, light wells, light shelves, etc that may be used to support effective internal illumination and also saving energy for cooling/heating of building’s spaces. For instance, a correct design of side-lighting and/or top-lighting will lead to energy saving by providing adequate daylight illumination in buildings so that electric lighting can be dimmed or switched off by means of an integrated lighting control. The daylighting does not have a big impact on the saved cooling energy explicitly; however it helps to avoid the unnecessary heat that would be caused by electrical lighting. In other words, to achieve the same level of illumination, daylighting would need less energy compared to electric lighting. Also in the mild or warm season the airflow through windows can provide natural ventilation, cool the rooms and reduce the need for air conditioning which means the costs of air conditioning system can be reduced.

But on the other side, the improper selection of daylighting solution can increase the energy consumption. For instance, due to an improper configuration (material, position size, etc) of windows/skylights the building may face increased heat loss in the cold season and high gain in the warm season and consequently the HVAC system must work excessively and consume more energy to keep the indoor climate within the comfort range. In other words the selection of fenestration is a very sensitive issue. On the one hand the solar heat flow through the fenestration can provide heating in the cold season, however in the warm seasons the undesired solar heat will raise the temperature of spaces and cause discomfort. In this case, the annual cooling requirement and its costs, which may be higher than heating costs, will defeat the benefits of the heat gained in the cold season.

For the use case of this dissertation, a top-lighting solution has been selected and the skylight design challenges will be addressed. Figure 4.1 depicts the basic parts of a skylight component.

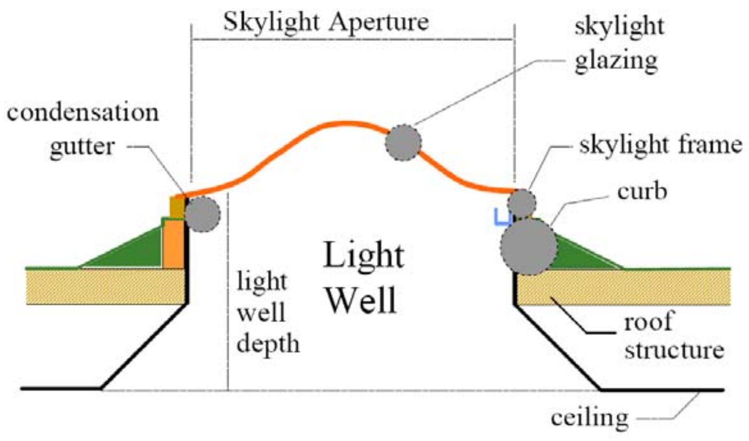


Figure 4.1: skylight system components

Selection of the appropriate fenestration (skylight in the case of this dissertation) in today’s market with so many available choices of products and manufacturers is a challenging task. There are wide variety of skylights with different sizes, shapes (range from simple rectangles to complex polygons, etc), frame materials and glazing materials in variety of shapes (e.g. dome, flat), colors (from clear and white to bronze and gray colors), thickness and number of layers. All these factors significantly affect the amount of light coming through the skylight and the efficiency of saving energy. In addition to skylight physical attributes that were mentioned above, the available illumination for daylighting depends also on the climate condition of building location. As a result the skylight component of a building will not have the same effect in another building with different climate condition and the climate considerations should be considered to select the skylight's physical attributes such as glazing and frame material and also choice of electrical lighting control. For instance, a transparent glazing material may be acceptable in a cloudy climate, whereas a diffusing material is necessary for sunny weather with on-off electrical control whereas for a location with low daylight availability may be more satisfied with dimming control.

Glazing properties have highly impacts on the efficiency of skylight systems. To select an energy efficient skylight product, some important factors should be considered that are explained blow.

U-Value: defines how well a product prevents heat from entering the building. The U-value ranges generally between 0.20 and 1.20. The lower the U-value, the better a product is at keeping heat in and also provides a better thermal isolation. This factor allows designers to compare the insulating properties of different skylights. To increase the insulating value of skylights many manufacturers offer products with double or triple glazing layers including gas fills or low-emittance (low-E) coatings. It is worthy to note that manufacturers provide the U-factor of a skylight either for just the glazing material part or for entire skylight component including glazing, frame and the spacer material. Table 4.1 lists the U-Value for some sample glazing and frame materials (ASHRAE Handbook, 1993).

Glazing material / frame material	Alum. no thermal break	Alum. with thermal break	Wood or Vinyl
Single Glass	1.30	1.07	n/a
Double Glass, ½" air space	0.81	0.62	0.48
Double glass, low-e, (E*=0.2), ½" air space	0.70	0.52	0.39
Double glass, low-e, (E*=0.1), ½" air space	0.67	0.49	0.37
Double glass, low-e, (E*=0.2), ½" space with argon	0.64	0.46	0.34
Triple glass, low-e, on two panes, ½" paces with argon	0.53	0.36	0.23
Quadruple glass, low-e (E=.01) on two panes, ¼" spaces with krypton	n/a	n/a	0.22

Table 4.1: Window U-Values for some glazing and frame materials

Solar Heat Gain Coefficient (SHGC): defines how much of the solar spectrum is transmitted through the glazing material. The SHGC range between 0 and 1 and the lower the SHGC, the better a product is at blocking unwanted heat gain. Additional glazing layers, or tinted glazing or using reflective coatings reject more solar heat radiation and reduce SHGC.

Visual Transmittance (VT or Tvis): defines how much light is transmitted through the glazing. The VT ranges between 0 and 1 and the higher the VT, the higher the potential for daylighting.

Table 4.2 shows the VT and SHGC values for different glazing type, material and colour (Skylighting Guidelines, 2008).

Glazing Type	Glazing Layers	Color	VT	SHGC
Acrylic/fiberglass	Single-glazed	Clear	0.92	0.77
		Med White	0.42	0.33
		Bronze	0.27	0.46
	Double-glazed	Clear	0.86	0.77
		Med White	0.39	0.30
		Bronze	0.25	0.37
Fiberglass	Insulated	Crystal	0.30	0.30
	translucent	White	0.20	0.23
	U-0.24	Bronze	0.10	0.16
Polycarbonate	Single-glazed	Clear	0.85	0.89
		Bronze	0.50	0.69
		Med White	0.37	0.50
	Double-glazed	Clear	0.73	0.75
		Bronze	0.43	0.58
		Med White	0.32	0.43
Glass	Single-glazed	Clear	0.89	0.82
		Bronze	0.55	0.64
		Green	0.74	0.59
	Double-glazed	Clear	0.78	0.70
		Bronze	0.48	0.51
		Green	0.66	0.47
	Double-glazed, low-e	Clear	0.72	0.57
		Bronze	0.45	0.39
		Green	0.61	0.39
	Triple-glazed, low-e	Clear	0.70	0.53
		Bronze	0.42	0.37
		Green	0.61	0.38

Table 4.2: SHGC and VT for different glazing types, layers, and colors

The above mentioned parameters are commonly available on the skylight components in the market and standard organizations together with manufacturers assess the parameter values and make product certifications for a fixed environmental condition.

Once daylight has passed through the skylight glazing, it can be controlled and diffused by the shape and reflective properties of light wells, shading devices, and the characteristics of rooms' surfaces.

Light wells are used to distribute and reflect the light and bring it to the rooms. They can be designed in different shapes and in wide variety of surface and colours. For instance a white matt surface provides uniform light distribution while coloured wells will distribute the light evenly but reduce its intensity. Other vertical surfaces like horizontal shading under the skylight in the light well would control the amount of daylight reaching the room. Also the surfaces of room itself have an impact on distribution of lights. The light coloured surfaces help to distribute the brightness but the matte surfaces diffuse the light further. In addition to the described factors integration with electric lighting and heating system plays also an important role in design of an efficient skylight system.

A balance of all these parameters is essential to supply the visual and thermal comfort of the building occupants. The skylight is a good use case where many different skills and professions (energy analyst, interior designer, etc) should collaborate to make the best fitting selection. Figure 4.2 shows some of the important factors and processes that should be considered during the skylighting process.

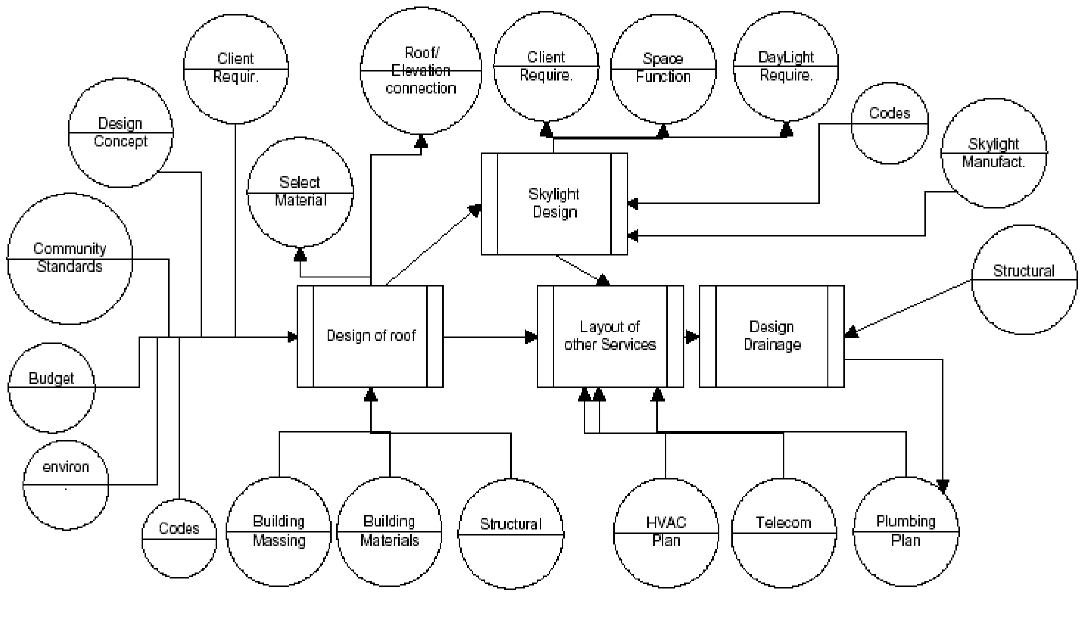


Figure 4.2: Process Diagram: Roof Design

The selected scenario is the multi dimensional problem of finding the best matching skylight component with optimum glazing areas for a specific space in a building, which provides uniform, low glare interior illumination, and achieves better heating/cooling and lighting performance.

4.2 Proposed solution

To address the skylight design requirements, the SkyDreamer prototype has been designed and implemented. SkyDreamer is a modern web 2.0 application that bridges the gap between BIM, energy simulation services and product libraries using semantic technologies in order to find the most efficient skylight components from the energy consumption point of view. The SkyDreamer has the following five basic parts:

- Semantic repository stores the skylight product information in a semantic way. The Skylight ontology extends the core IFC2X3. The semantic repository can then be queried via Joseki Servlet (Joseki Servlet, 2008).
- Web extraction component parses the product information pages from the available sources on the web and stores them according to the skylight ontology. In the present case study, a plug-in for “Certified Product Directory” (CPD, 2008) has been implemented that lists the certified products categorized by type and producer.
- Building's navigator facilitates the selection of desired space from the building’s hierarchy of zones and spaces.
- Calculator receives the building model in IFCXML format and calculates the energy use implications of the selected skylight component (extracted from Semantic-based IFD library) for lighting, heating, and cooling.
- User interface that mediates between the end user and other system components such as calculator and semantic repository.

Figure 4.3 shows the SkyDreamer system components and how different components are plugged together to realize the proposed scenario. In the rest of this chapter the SkyDreamer components will be introduced in more details.

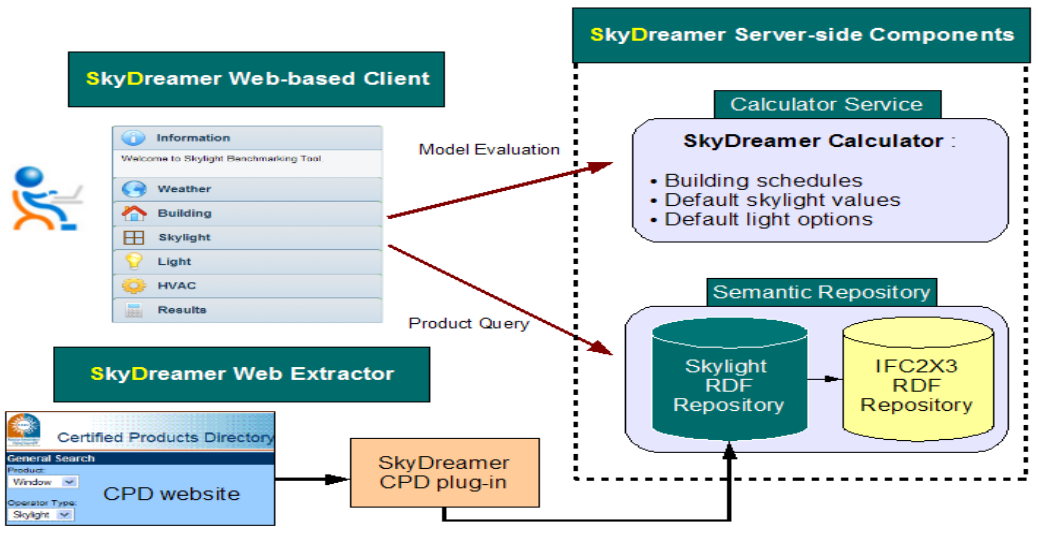


Figure 4.3: System Components

In order to calculate the energy efficiency of a building with a specific skylight element, some basic configuration is required. This configuration includes the following items:

- The building model in IFCXML format
- The weather file of the building location
- Building properties such as building type (residential, office, etc)
- HVAC options
- Generic skylight features such as glazing type and glazing layers, etc.

The details of this configuration will be explained in the forthcoming sections; here the bird view of task sequences will be briefly explained. The SkyDreamer configuration is provided via the user friendly web interface and the required geometry information such as space dimensions and also number of skylights and their dimensions are automatically extracted from the building model. Based on the given skylight specification such as glazing characteristics, the user will run the simulation process for a producer-neutral configuration. In the next step the user will be able to look for real products from the semantic repository and repeat the simulation for the specific skylight component.

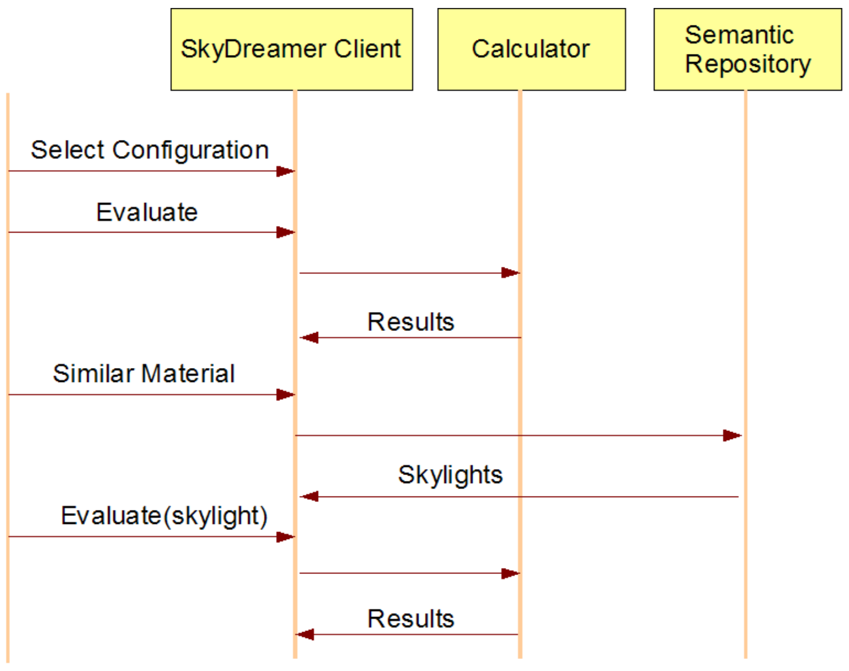


Figure 4.4: Sequence diagram of SkyDreamer prototype

4.2.1 Ontology component

To establish a Semantic IFD Repository, two basic parts are needed, namely: An ontology schema that describes the required elements and the instances. In the previous chapters it was explained how IAI's IFC have been used to establish the schema of the upper ontology. Ontology component library is considered to share the skylight product information in a comprehensive and processable way to other IFC compliant applications. This idea is fully compatible with Semantic Web concepts, where ontologies represent the shared knowledge of a specific domain.

A challenging issue in creating the ontologies based on IFC ontology is that the IFC standard does not explicitly define all building elements. Basically, the IFC model is a generic object oriented model that can be extended to define new elements. For example, an `ifcWindow` needs to be extended for creating a skylight with required collection of properties. Thanks to the dynamic mechanisms of Semantic Web, a new skylight ontology has been created that uses the IFC upper ontology concepts and extend it to define new classes such as skylight.

Moreover, some new property sets was required that are specific to skylight products. Definition of these property sets has also followed the IFC ontology schema by instantiating the `ifcPropertySet.` However the organization of property sets were not suitable for the required queries. A disadvantage of IFC's generic model is the fact that the IFC-based elements are not semantically well-organized and the format is more appropriate for object oriented computer processes. As a result, the human user who needs to query the model needs to build complex queries to extract the required information. Again, the Semantic Web methods have been used to address this issue by semantic inference. In other words, the Semantic Web rules have been used to make a shortcut and attach the properties directly to skylight component via a new predicate (attribute). In this way the implicit property values that where previously connected through a long chain of classes, are summarized in terms of explicit attributes that are connected directly to skylight instances. Figure 4.5 shows the initial situation where for example SHGC attribute value needs to be queried for a specific skylight and the new inferred predicate `hasSHGC`, which simplifies the access to attribute values. Accordingly, to query all the skylights that have a specific SHGC value, the user should trace the tree from skylight (right hand side) up to the root and then the property sets (on the left hand side) and finally the specific property value pairs. However by means of the inferred predicate the formulation and execution of queries will improve significantly.

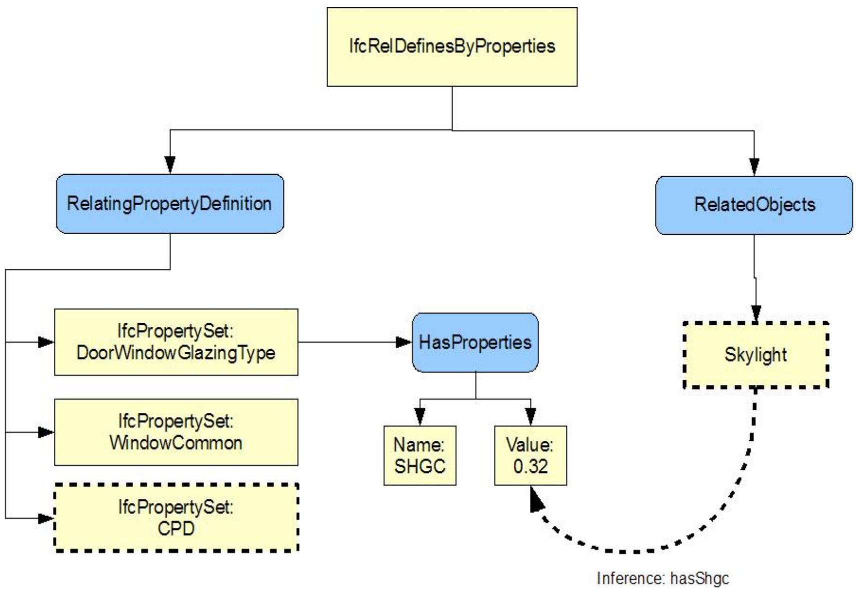

IfcRelDefinesByProperties
RelatingPropertyDefinition
RelatedObjects
IfcPropertySet: DoorWindowGlazingType
IfcPropertySet: WindowCommon
IfcPropertySet: CPD
HasProperties
Name: SHGC
Value: 0.32
Skylight
Inference: hasShgc

Figure 4.5: Semantic inference for simplifying access to skylight attributes

Basically the inference process in Semantic Web is implemented by loading the raw model (IFC ontology in the present case) and applying the inference rules by means of a Semantic Inference Engine. The inference result which is called the inferred model will be then used for further queries. Semantic inference is a heavy and time-consuming process and in case of huge ontologies such as IFC ontology, some techniques are used to perform the inference efficiently. For instance instead of running the inference and creating the in-memory inferred model for each query, the result model can be persisted and used for further queries. Furthermore the ontology models can be also stored in a relational database and accessed via an appropriate semantic driver that translates the semantic queries to relational queries and returns the resulting records as semantic information.

An alternative method to inference is to create the inferred data explicitly for the newly added instances. This is especially helpful for highly structured ontologies such as skylight ontology, that the inferred data can be generated and added to the persistent model together with every new skylight instance. Listing 4.1 shows how the new inferred data will make the shortcut for SHGC.

```
<Skylight rdf:ID="skylight_3001"> ❶
       <ifc:IfcRoot_Name rdf:datatype="&xsd;string">
           PEL-N-68-00005
        </ifc:IfcRoot_Name>
       <ifc:IfcRoot_OwnerHistory rdf:resource="#dummyOwnerHistory"/>
       <ifc:IfcRoot_GlobalId rdf:resource="#dummyUniqueId"/>
       <hasUfactor rdf:datatype="&xsd;float">0.56</hasUfactor>
       <hasVt rdf:datatype="&xsd;float">0.65</hasVt>
       <hasShgc rdf:datatype="&xsd;float">0.32</hasShgc> ❷
</Skylight>

<ifc:IfcRelDefinesByProperties rdf:ID="rel_3035"> ❸
       <ifc:IfcRelDefinesByProperties_RelatingPropertyDefinition
                    rdf:resource="#pset_wc_3004"/>
       <ifc:IfcRelDefinesByProperties_RelatingPropertyDefinition
                    rdf:resource="#pset_gt_3015"/> ❹
       <ifc:IfcRelDefinesByProperties_RelatingPropertyDefinition
                    rdf:resource="#pset_cpd_3034"/>
       <ifc:IfcRelDefines_RelatedObjects rdf:resource="#skylight_3001"/> ❺
       <ifc:IfcRoot_OwnerHistory rdf:resource="#dummyOwnerHistory"/>
       <ifc:IfcRoot_GlobalId rdf:resource="#dummyUniqueId"/>
</ifc:IfcRelDefinesByProperties>

<ifc:IfcPropertySet rdf:ID="pset_gt_3015"> ❻
       <ifc:IfcRoot_Name rdf:datatype="&xsd;string">
             Pset_DoorWindowGlazingType
        </ifc:IfcRoot_Name>
       <ifc:IfcPropertySet_HasProperties rdf:resource="#val_3006"/>
       <ifc:IfcPropertySet_HasProperties rdf:resource="#val_3008"/>
       <ifc:IfcPropertySet_HasProperties rdf:resource="#val_3010"/>
       <ifc:IfcPropertySet_HasProperties rdf:resource="#val_3012"/>
```

```
        <ifc:IfcPropertySet_HasProperties rdf:resource="#val_3014"/> ❼
        <ifc:IfcRoot_OwnerHistory rdf:resource="#dummyOwnerHistory"/>
        <ifc:IfcRoot_GlobalId rdf:resource="#dummyUniqueId"/>
</ifc:IfcPropertySet>

<ifc:IfcPropertySingleValue rdf:ID="val_3014"> ❽
        <ifc:IfcProperty_Name rdf:resource="#SolarHeatGainTransmittance"/>
        <ifc:IfcPropertySingleValue_NominalValue rdf:resource="#nval_3013"/>
</ifc:IfcPropertySingleValue>

<ifc:Ifc_NominalValue rdf:ID="nval_3013"> ❾
        <ifc:Ifc_PositiveRatioMeasure rdf:datatype="&xsd;float">
             0.32
        </ifc:Ifc_PositiveRatioMeasure>
</ifc:Ifc_NominalValue>
```

Listing 4.1: adding inferred data for skylight's SHGC

The content of this listing has been created automatically by web extractor component for each skylight and depicts the textual serialization of the structure shown in figure 4.5. The web extractor component will be introduced in the next section; here only the explicit semantic inference method will be explored. As it is obvious from this listing, most of elements are defined in `ifc` namespace which is connecting the resources to IFC upper ontology. The listing defines a skylight instance, with unique identifier skylight_3001 (at point 1). As mentioned before IFC 2x3 does not have a specific class for Skylight, and the Skylight class has been added in Skylight ontology by extending the `ifcWindow` class. As the above listing shows the Skylight class is not prefixed with ifc namespace, which means it is a new class defined in ontology's default namespace (skylight). To connect an entity to its properties in IFC world, the `IfcRelDefinesByProperties` should be used (point 3) that in present case connects the skylight (point 5) instance to a specific IFC property set called `Pset_DoorWindowGlazingType` (points 4 and 6). This property set referes to a property value (point 7) which is called `SolarHeatGainTransmittance` (point 8) with the `Ifc_PositiveRatioMeasure` value of 0.32 (point 9). This long chain of references has been simplified by adding the `hasShgc` attribute directly to skylight (point 2).

Since the IFC does not contain all required properties for skylights, the extra properties can be added by user-customized property sets and defining the desired attributes to this property sets. The drawback of this approach is that the new IFC product and the models that are using this product might not be understood by other stakeholders that do not use skylight ontology. In skylight case a new property set namely `Pset_CertifiedProductsDirectory` has been defined that contains all required attributes for skylight components and a reference to this property set is added to each skylight in familiar way (line after point 4, `#pset_cpd_3034`).

Especially, it would be interesting if the new product and its specifications are machine processable without human interaction. This is the point where Semantic Web technology and ontologies might be helpful. It means, IFC plays its traditional role as a standard information sharing model, while the Semantic Web extends and enriches the IFC model in a machine-processable way.

As a matter of fact, Semantic Web and ontology engineering issues are the challenging field of this thesis and are expected to bridge the gaps between specifications, regulations and AEC applications

Finally, the inferred skylight ontology will be stored in an intermediate OWL file that will be imported in the backend relational database to facilitate the semantic queries. In this ontology, IFC plays its traditional role as a standard information sharing model, while the Semantic Web extends and enriches the IFC model in a machine-processable way. For the use case in this dissertation, the Jena2 Database Interface and MySQL database have been used. The defined ontology (including instances) has been shared with other SkyDreamer components as a web service. This feature is provided by Joseki Servlet that accepts the semantic queries in SPARQL syntax and returns the result in different formats such as RDF and JSON. Figure 4.6 shows the basic architecture of ontology component.

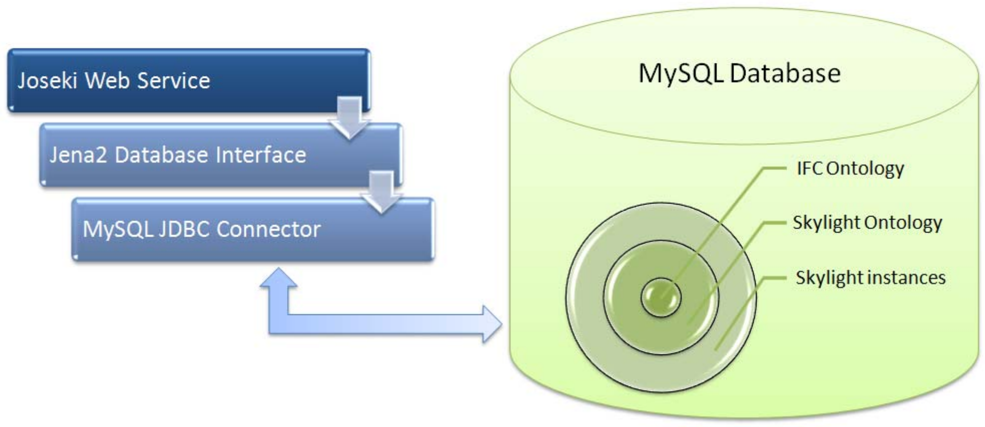
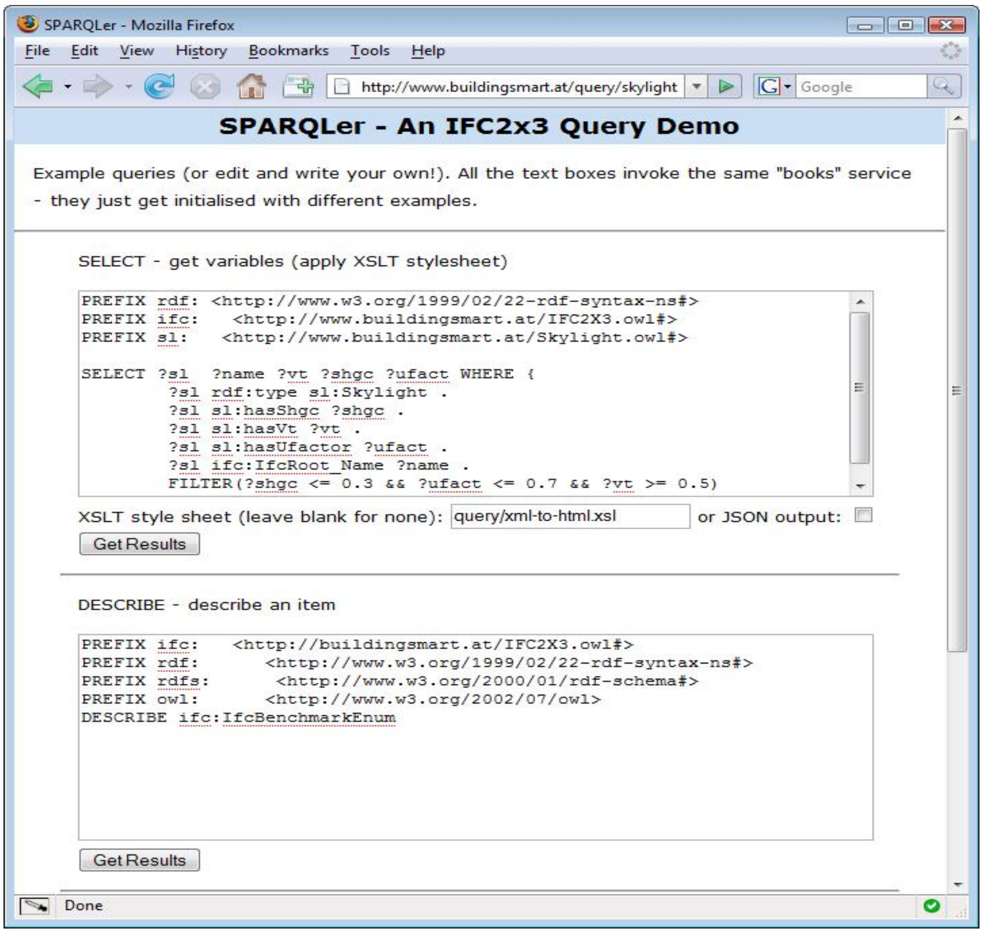


Figure 4.6: architecture of ontology component

Joseki Web Service can be also called via a web interface to answer the semantic queries directly. A customized version of Joseki query interface has been created for semantic repository. Figure 4.7 shows the SPARQL query interface for skylight semantic repository (IFC Ontology + Skylight Ontology + Skylight instances).


SPARQLer - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.buildingsmart.at/query/skylight
Google
Done

SPARQLer - An IFC2x3 Query Demo

Example queries (or edit and write your own!). All the text boxes invoke the same "books" service - they just get initialised with different examples.

SELECT - get variables (apply XSLT stylesheet)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ifc:   <http://www.buildingsmart.at/IFC2X3.owl#>
PREFIX sl:   <http://www.buildingsmart.at/Skylight.owl#>

SELECT ?sl  ?name ?vt ?shgc ?ufact WHERE {
        ?sl rdf:type sl:Skylight .
        ?sl sl:hasShgc ?shgc .
        ?sl sl:hasVt ?vt .
        ?sl sl:hasUfactor ?ufact .
        ?sl ifc:IfcRoot_Name ?name .
        FILTER(?shgc <= 0.3 && ?ufact <= 0.7 && ?vt >= 0.5)
```

XSLT style sheet (leave blank for none): query/xml-to-html.xsl or JSON output: ☐

Get Results

DESCRIBE - describe an item

```
PREFIX ifc:   <http://buildingsmart.at/IFC2X3.owl#>
PREFIX rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:      <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl:      <http://www.w3.org/2002/07/owl>
DESCRIBE ifc:IfcBenchmarkEnum
```

Get Results

Figure 4.7: Joseki skylight query interface

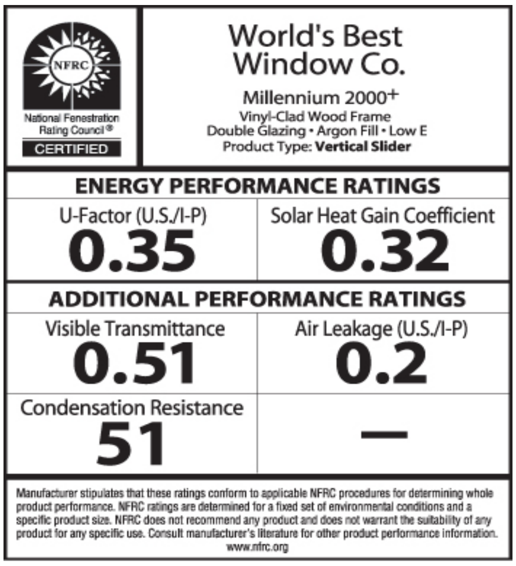
4.2.2 Web extractor component

As explained before, the huge amount of information available on the web today, is an important resource for AEC/FM businesses. There are thousands of online catalogues and directories for all building components. In the first generation of World Wide Web, information is merely formatted for presentation purposes and to extract required information, human user should read and analyze the relevant information resources. The primary goal of Semantic Web is to change this situation and make this content readable and processable by computers in a way that extracted information can be integrated into relevant business processes. In future web, the web pages will include descriptive information that are open to software agents and client applications that request the data. An example of such descriptive language is RDFa (RDFa Primer, 2008) specification that embeds the semantic information into HTML page. This aspect of Semantic Web vision has not been widely implemented yet and still there are large amount of web data that are presented in traditional web pages. One of the approaches that facilitate the transition from traditional web to Semantic Web is the use of targeted page parsers (adaptors) that read the traditional web pages and programmatically add the semantics to raw data. Examples of such adapters are Web-Harvest (Web-Harvest, 2008) and Lixto Solutions (Lixto Solutions, 2008). Both are general configurable parsers for web pages. The result of these adaptors can be then stored semantically for further usage. It is also important to note that these adapters are dependent on the structure and style of the web pages, and the website changes will invalidate the information extraction configurations and as a result the adapter should be configured to cope with website changes.

For the use case of SkyDreamer prototype, a customized adapter has been developed that adds skylight products (instances) to the semantic repository. The web extractor component is a simple Java application that runs periodically and synchronizes the repository contents with the website's information. The information resource that has been used for establishing the semantic material repository is the National Fenestration Rating Council’s website (NFRC, 2008) based in United States. NFRC is a non-profit organization that was formed in response to energy crisis of 1970s and administers a uniform, independent rating and labeling system for the energy performance of windows, doors, skylights, and attachment products. The most important artifact of NFRC is a Certified Product Directory (CPD) that affords query of certified products by providing its manufacturer and performance criteria. This product directory will benefit both designers and manufacturers:

- Designers could adopt the appropriate building components and evaluate their designs while considering alternative products. This would result in more efficient buildings. An interesting use case of this kind is the selection of energy efficient components for building envelope elements.
- Building product manufacturers could introduce the technical specifications of their products that will be queried by consumers and building designers

In the specific case of skylights, CPD includes information such as U-value, SHGC, VT, frame type, sash type, glazing layers and gas fill. The CPD manufacturers may use the NFRC energy performance label that reflects the energy performance characteristics of a product. The NFRC label can be used to determine how well a product will perform in conjunction with cooling systems in summer and heating systems in winter to keep building convenient for inhabitants. Moreover it gives additional information about how effective is a product keep out wind and resist condensation. This information can reliably be used by builders and consumers to compare products and find the best fitting door, window, or skylight for buildings. Figure 4.8 shows a typical NFRC energy performance label of a window with the specified U-value, SHGC, and VT values. These values, which play a significant role in behavior of window and skylight products are also used by SkyDreamer prototype for calculation of energy performance of the zones that are benefiting from skylight products.



NFRC
National Fenestration Rating Council®
CERTIFIED

World's Best Window Co.
Millennium 2000+
Vinyl-Clad Wood Frame
Double Glazing • Argon Fill • Low E
Product Type: **Vertical Slider**

ENERGY PERFORMANCE RATINGS

U-Factor (U.S./I-P)	Solar Heat Gain Coefficient
0.35	0.32

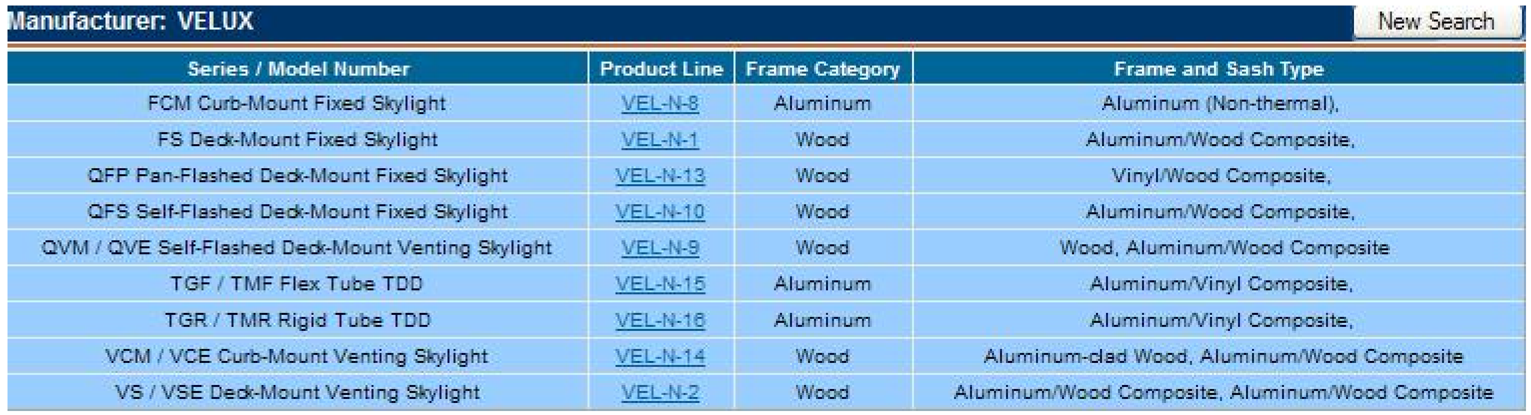
ADDITIONAL PERFORMANCE RATINGS

Visible Transmittance	Air Leakage (U.S./I-P)
0.51	0.2
Condensation Resistance 51	—

Manufacturer stipulates that these ratings conform to applicable NFRC procedures for determining whole product performance. NFRC ratings are determined for a fixed set of environmental conditions and a specific product size. NFRC does not recommend any product and does not warrant the suitability of any product for any specific use. Consult manufacturer's literature for other product performance information.
www.nfrc.org

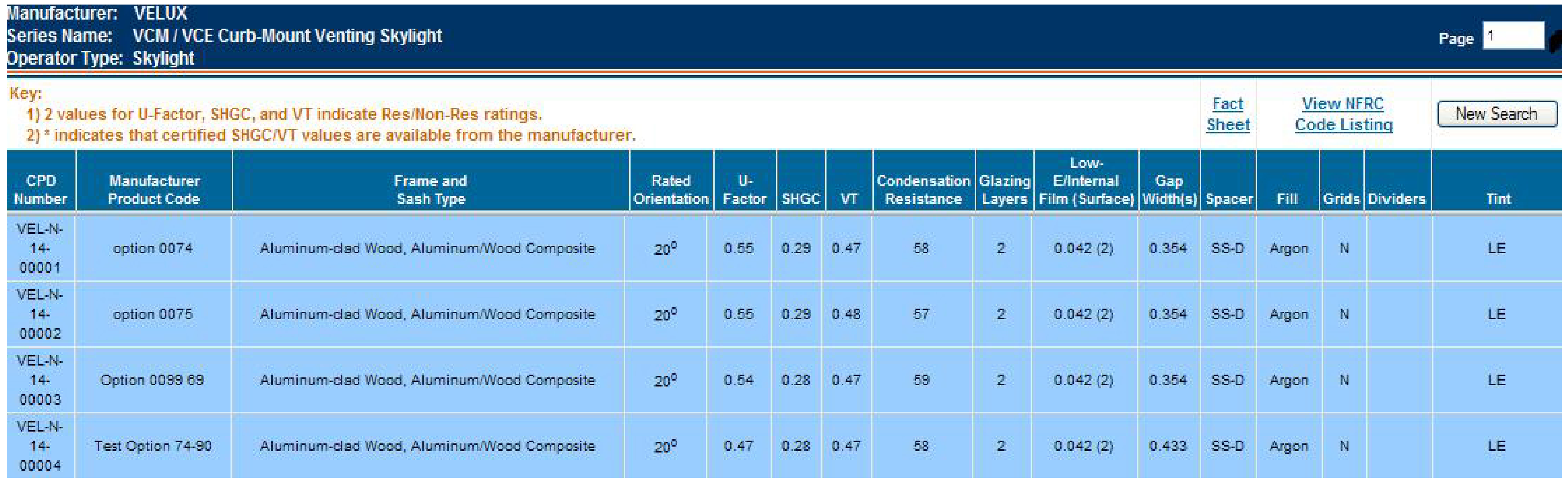
Figure 4.8: NFRC energy performance label

The SkyDreamer's web extraction component parses the relevant web pages of NFRC website and generates the intermediate RDF files that will be later on added to skylight repository. The web extractor starts with the manufacturer page and then drills down to the product lines and individual products. At the product level, web extractor captures the detailed physical and energy performance characteristics of products. Figure 4.9 shows the typical production lines of a specific manufacturer and also the product details of a selected production line.



Manufacturer: VELUX

Series / Model Number	Product Line	Frame Category	Frame and Sash Type
FCM Curb-Mount Fixed Skylight	VEL-N-8	Aluminum	Aluminum (Non-thermal),
FS Deck-Mount Fixed Skylight	VEL-N-1	Wood	Aluminum/Wood Composite,
QFP Pan-Flashed Deck-Mount Fixed Skylight	VEL-N-13	Wood	Vinyl/Wood Composite,
QFS Self-Flashed Deck-Mount Fixed Skylight	VEL-N-10	Wood	Aluminum/Wood Composite,
QVM / QVE Self-Flashed Deck-Mount Venting Skylight	VEL-N-9	Wood	Wood, Aluminum/Wood Composite
TGF / TMF Flex Tube TDD	VEL-N-15	Aluminum	Aluminum/Vinyl Composite,
TGR / TMR Rigid Tube TDD	VEL-N-16	Aluminum	Aluminum/Vinyl Composite,
VCM / VCE Curb-Mount Venting Skylight	VEL-N-14	Wood	Aluminum-clad Wood, Aluminum/Wood Composite
VS / VSE Deck-Mount Venting Skylight	VEL-N-2	Wood	Aluminum/Wood Composite, Aluminum/Wood Composite



Manufacturer: VELUX
Series Name: VCM / VCE Curb-Mount Venting Skylight
Operator Type: Skylight

Key:
1) 2 values for U-Factor, SHGC, and VT indicate Res/Non-Res ratings.
2) * indicates that certified SHGC/VT values are available from the manufacturer.

CPD Number	Manufacturer Product Code	Frame and Sash Type	Rated Orientation	U-Factor	SHGC	VT	Condensation Resistance	Glazing Layers	Low-E/Internal Film (Surface)	Gap Width(s)	Spacer	Fill	Grids	Dividers	Tint
VEL-N-14-00001	option 0074	Aluminum-clad Wood, Aluminum/Wood Composite	20^0	0.55	0.29	0.47	58	2	0.042 (2)	0.354	SS-D	Argon	N		LE
VEL-N-14-00002	option 0075	Aluminum-clad Wood, Aluminum/Wood Composite	20^0	0.55	0.29	0.48	57	2	0.042 (2)	0.354	SS-D	Argon	N		LE
VEL-N-14-00003	Option 0099 69	Aluminum-clad Wood, Aluminum/Wood Composite	20^0	0.54	0.28	0.47	59	2	0.042 (2)	0.354	SS-D	Argon	N		LE
VEL-N-14-00004	Test Option 74-90	Aluminum-clad Wood, Aluminum/Wood Composite	20^0	0.47	0.28	0.47	58	2	0.042 (2)	0.433	SS-D	Argon	N		LE

Figure 4.9: NFRC skylight production lines (left) and product details (right)

The complete listing of web extractor component is not included here, but the snippet of the code that extracts skylight characteristics from the web page cells is given in listing 4.2.

```
public void getProductInfo(Vector<String> cell){

   for(int i =4 ; i < cell.size(); i+=16){

        SkylightProduct skyproduct = new SkylightProduct();

        skyproduct.setModelNumber(cell.elementAt(0));
        skyproduct.setProductLine(cell.elementAt(1));
        skyproduct.setFrameCategory(cell.elementAt(2));
        skyproduct.setSashType(cell.elementAt(3));
        skyproduct.setCPDNumber(cell.elementAt(i));
        skyproduct.setManufacturerProductCode(cell.elementAt(i+1));
        skyproduct.setFrameAndSashType(cell.elementAt(i+2));
        skyproduct.setRatedOrientation(cell.elementAt(i+3));
        skyproduct.setUFactor(cell.elementAt(i+4));
        skyproduct.setShgc(cell.elementAt(i+5));
        skyproduct.setVt(cell.elementAt(i+6));
        skyproduct.setCondensationResistance(cell.elementAt(i+7));
        skyproduct.setGlazingLayers(cell.elementAt(i+8));
        skyproduct.setLowE(cell.elementAt(i+9));
        skyproduct.setGapWidth(cell.elementAt(i+10));
        skyproduct.setSpacer(cell.elementAt(i+11));
        skyproduct.setFill(cell.elementAt(i+12));
        skyproduct.setGrids(cell.elementAt(i+13));
        skyproduct.setDividers(cell.elementAt(i+14));
        skyproduct.setTint(cell.elementAt(i+15));

        saveRDF(skyproduct.toRDF());

   }

}
```

Listing 4.2: Parsing product details by web extractor

As mentioned in the previous section, IFC does not include all the required properties of skylights but the data structure of IFC model can be extended to define additional properties. For this purpose a new property set, `Pset_CertifiedProductsDirectory`, has been defined that contains all additional skylight properties. This property set will be later on added to relevant objects through an objectified relationship called `IfcRelDefinedByProperties`. Listing 4.3 shows how properties such as gap-width, spacer, grids and dividers are added to skylight instance in IFC-friendly style.

```
if (gapWidth > 0) {
  long id = addSingleValue("GapWidth", ""+gapWidth, "Ifc_Real");
  props.append(
     "\n\t<ifc:IfcPropertySet_HasProperties rdf:resource=\"#val_"+id+"\"/>");
}

if (spacer.length() > 0) {
  long id = addSingleValue("Spacer", spacer, "Ifc_Label");
  props.append(
      "\n\t<ifc:IfcPropertySet_HasProperties rdf:resource=\"#val_"+id+"\"/>");
}

if (grids.length() > 0) {
  long id = addSingleValue("Grids", grids, "Ifc_Label");
  props.append(
     "\n\t<ifc:IfcPropertySet_HasProperties rdf:resource=\"#val_"+id+"\"/>");
}

if (dividers.length() > 0) {
  long id = addSingleValue("Dividers", dividers, "Ifc_Label");
  props.append(
     "\n\t<ifc:IfcPropertySet_HasProperties rdf:resource=\"#val_"+id+"\"/>");
}

// PropertySet: Pset_CertifiedProductsDirectory
long pset_cpd = getNextGlobalId();
sb.append("\n<ifc:IfcPropertySet rdf:ID=\"pset_cpd_"+pset_cpd+"\">");
sb.append("\n\t<ifc:IfcRoot_Name
rdf:datatype=\"&xsd;string\">Pset_CertifiedProductsDirectory</ifc:IfcRoot_Name>");

sb.append(props);

sb.append("\n\t<ifc:IfcRoot_OwnerHistory rdf:resource=\"#dummyOwnerHistory\"/>");
sb.append("\n\t<ifc:IfcRoot_GlobalId rdf:resource=\"#dummyUniqueId\"/>");
sb.append("\n</ifc:IfcPropertySet>\n");

//RelDefineByProperties
sb.append("\n<ifc:IfcRelDefinesByProperties rdf:ID=\"rel_"+getNextGlobalId()+"\">");
sb.append("\n\t<ifc:IfcRelDefinesByProperties_RelatingPropertyDefinition
rdf:resource=\"#pset_wc_"+pset_wc+"\"/>");
sb.append("\n\t<ifc:IfcRelDefinesByProperties_RelatingPropertyDefinition
rdf:resource=\"#pset_gt_"+pset_gt+"\"/>");
sb.append("\n\t<ifc:IfcRelDefinesByProperties_RelatingPropertyDefinition
rdf:resource=\"#pset_cpd_"+pset_cpd+"\"/>");
sb.append("\n\t<ifc:IfcRelDefines_RelatedObjects
rdf:resource=\"#skylight_"+skylightId+"\"/>");
sb.append("\n\t<ifc:IfcRoot_OwnerHistory rdf:resource=\"#dummyOwnerHistory\"/>");
sb.append("\n\t<ifc:IfcRoot_GlobalId rdf:resource=\"#dummyUniqueId\"/>");
sb.append("\n</ifc:IfcRelDefinesByProperties>\n");
```

Listing 4.3: Creating product instances by web extractor

The web extractor component is also able to keep the repository information up-to-date by parsing the relevant web pages and incorporate the new added information. However there are some concerns in altering the skylight repository items. Since the skylight repository is being used for energy performance analysis the change of sensitive data such as SHGC, VT, and U-value will invalidate the energy analysis results. One of the mechanisms to avoid this situation is to apply versioning information to product repository and by each update assign a new version number to the modified object while keeping the outdated versions. So the previous analysis remains valid regarding to an older version of the product information. In Semantic Web world this issue is addressed under ontology versioning title, which is more relevant for the ontology schema. For the SkyDreamer case, we have ignored the ontology versioning because it does not a primary challenge of this dissertation. As a result, updates simply overwrite the existing information and all energy simulations will be done based on the current stand of repository.

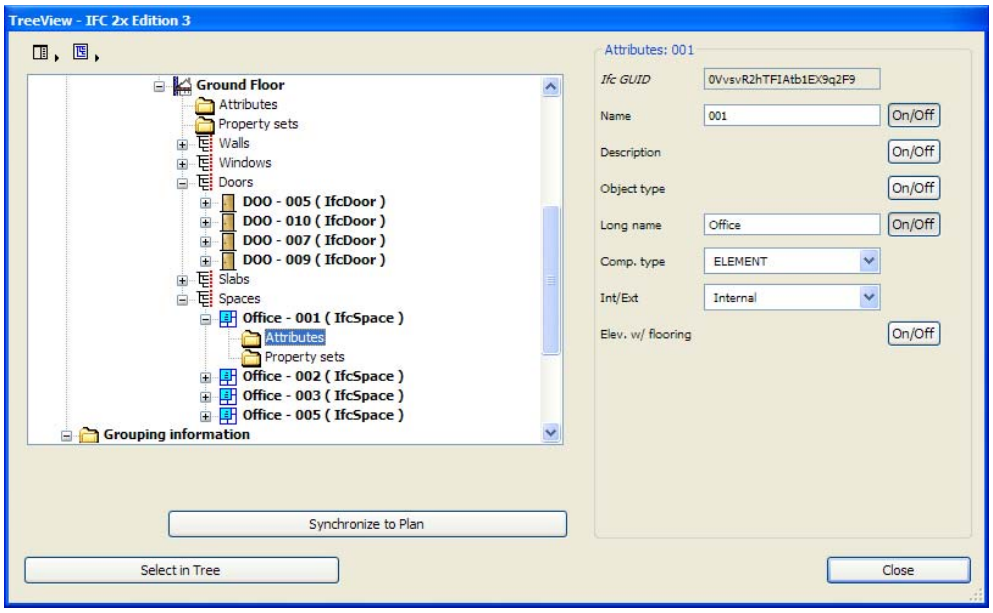
4.2.3 Building Parser and Building Navigator

In the building energy simulation programs, building is usually divided into some logical or spatial divisions that are known as zones. Each zone might be a single space or a grouping of similar spaces that are dedicated to a particular activity and/or usage. Due to different occupancy and comfort requirements, each zone has a equipped with different HVAC procedures to afford the inhabitant’s comfort standards. For instance, an apartment may be divided into living and sleeping zones and the temperature of each zone should be selected accordingly for different occupancy models and also different times of the day.

The IAI defines a zone (IfcZone) as an aggregation of spaces, partial spaces or other zones. Zone structures may not be hierarchical, i.e. one individual space (IfcSpace) may be associated with zero, one, or several IfcZone's.

Zones play an important role in building energy calculation and in case of SkyDreamer prototype; the building model should deliver the zone information and corresponding spaces. Additionally, the energy calculation component needs to know the space area, number of skylights, and also the total skylight area. This information will be used to calculate skylight to floor ratio which is an important factor for building energy calculation.

The IFC building information model contains all the required information. Some of this information such as zones, spaces, and areas are explicitly defined in the model, but others such as skylight to floor ratio should be assessed by building navigator component. In order to make model parsing easier, SkyDreamer uses the IFCXML form of building model which can be easily parsed and queried by XML libraries such as dom4j (dom4j, 2008). This model may be created by any of IFC-enabled design tools such as ArchiCAD, where IFC add-on will help to define IfcSpaces and associate them to corresponding IfcZones. In case the uploaded model contains no IfcZone, building navigator component will add a default zone and list all spaces under this zone. Figure 4.10 depicts the tree view of an IFC model with some IfcSpaces, which may be later on grouped together and form zones. After parsing the model, the tree structure of building model will be created starting with zones and corresponding spaces and then the available skylights in each space will be added to space nodes.



TreeView - IFC 2x Edition 3

```
Ground Floor
    Attributes
    Property sets
    Walls
    Windows
    Doors
        DOO - 005 ( IfcDoor )
        DOO - 010 ( IfcDoor )
        DOO - 007 ( IfcDoor )
        DOO - 009 ( IfcDoor )
    Slabs
    Spaces
        Office - 001 ( IfcSpace )
            Attributes
            Property sets
        Office - 002 ( IfcSpace )
        Office - 003 ( IfcSpace )
        Office - 005 ( IfcSpace )
Grouping information
```

Attributes: 001

Ifc GUID	0VvsvR2hTFIAtb1EX9q2F9
Name	001
Description	
Object type	
Long name	Office
Comp. type	ELEMENT
Int/Ext	Internal
Elev. w/ flooring	

Figure 4.10: IFC tree view and spaces in ArchiCAD

At runtime, user uploads the IFCXML file and the model parser component will assess the building structure tree. Furthermore, the parser will automatically detect the length and area units of any given model (by reading the model's `IfcSIUnit` for length and area) and convert them to metric system for further usages in energy calculator component. When parsing is finished, user will be able to navigate through the resulting tree and select the required space for energy calculations. In the next step, building navigator will calculate the corresponding skylight to floor ratio for the selected space. Some important parts of the model parser and model navigator source code are available as an appendix (see Appendix 2).

4.2.4 Skylight Energy Calculator

As explained above, skylights can provide sufficient uniform illumination and save a great deal of energy when they are designed and selected carefully. Oversized skylighting system may increase the cooling loads in summer and waste of heating in winter adding to energy consumption of building, also the cost for installation and maintenance of undersized skylight system my be higher than its energy saving cost.

There are many large single-story buildings such as retail spaces, grocery stores, restaurants, schools and offices that can benefit from skylighting to gain more daylighting and reduce the need for electric lighting. The earlier daylighting is considered in design process, the greater is the success in integration with other architectural, electrical, and mechanical components and in providing most daylighting with highest potential for energy saving.

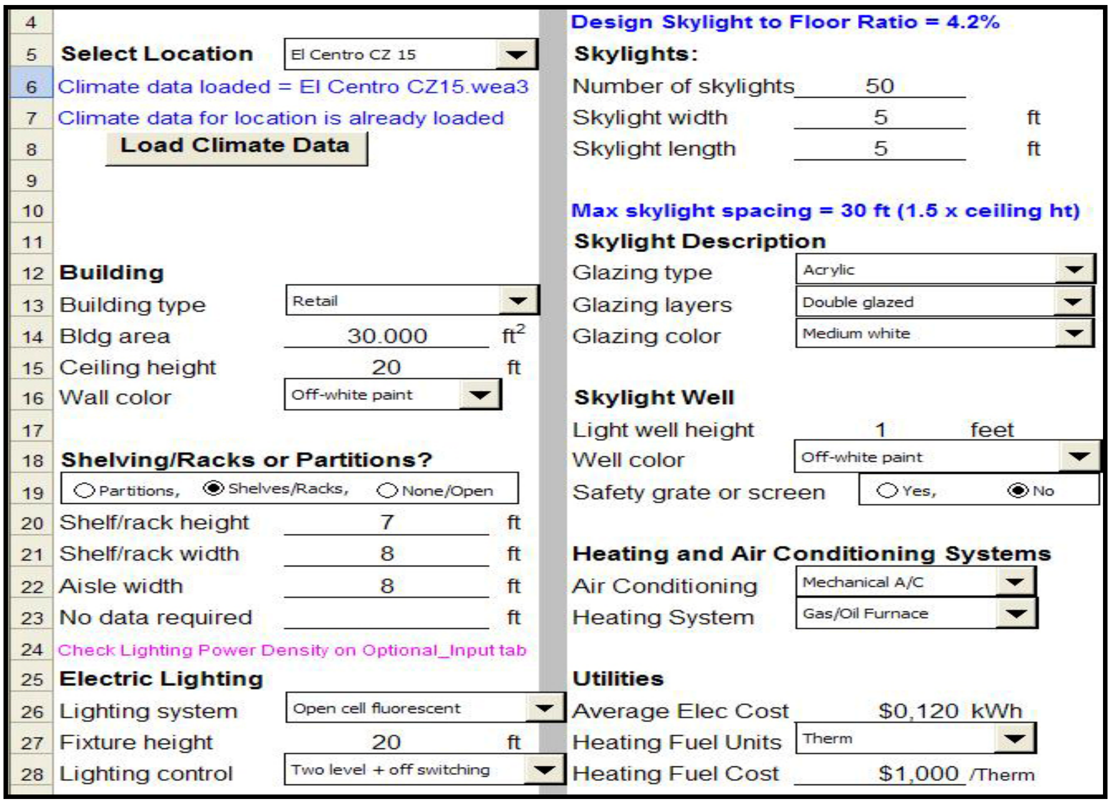
To analyse the efficiency of the selected skylight and improve the energy performance of the buildings, a skylight calculator is needed. This skylight calculator should receive the information from different resources such as building model, skylight characteristics and location information and make the necessary calculations for each selected configuration. The result will assist designers and consumers to select better skylight from the energy consumption point of view.

The SkyDreamer's calculator is based on the logic of SkyCalc (SkyCalc, 2008) spreadsheet that calculates the energy loads of different skylights with respect to climate zone and helps to find the optimal sizing of skylights to maximize energy or cost savings. SkyCalc uses simple data inputs (either common defaults or user-supplied data) to describe a building and analyze possible skylighting strategies. It calculates the lighting and whole-building energy impacts of each design, and produces graphs and charts that describe annual energy-use patterns. SkyCalc runs on older versions of Microsoft Excel and the core calculations are either distributed among spreadsheet cells, or coded in VBA (Visual Basic for Applications). This imposes some limitation toward extending and reuse of skylight calculations. The SkyDreamer calculator component on the other hand is a modern Java based web service with a well-defined interface that can be used in heterogeneous environments. Moreover, the calculation is able to receive the required input data from IFC BIM and semantic skylight repository. More precisely, this web service can be seen as a semantic web service that is documentable by skylight ontology. As a result such web services can then be located, selected, employed, composed, and monitored automatically by the processes that need those services.

In the next section, SkyCalc tool and its basic inputs and outputs are explored and at the end the SkyDreamer calculator is introduced in more details.

4.2.5 SkyCalc Tool

As explained before, SkyCalc is a free and simple computer tool that calculates the energy loads of single storey buildings for different skylights configurations by considering the corresponding climate zone. It helps building designers determine the optimum skylighting strategy that provides the highest HVAC and lighting energy saving for a given building. The SkyCalc is implemented as a Microsoft Excel spreadsheet that runs on a personal computer. The program input is divided into basic and optional inputs whose values should be provided in corresponding input worksheets. Figure 4.11 depicts some of the basic inputs of ScyCalc tool such as climate data, building type, and skylight characteristics.



4				Design Skylight to Floor Ratio = 4.2%		
5	**Select Location**	El Centro CZ 15		**Skylights:**		
6	Climate data loaded = El Centro CZ15.wea3			Number of skylights	50	
7	Climate data for location is already loaded			Skylight width	5	ft
8	Load Climate Data			Skylight length	5	ft
9						
10				**Max skylight spacing = 30 ft (1.5 x ceiling ht)**		
11				**Skylight Description**		
12	**Building**			Glazing type	Acrylic	
13	Building type	Retail		Glazing layers	Double glazed	
14	Bldg area	30,000	ft^2	Glazing color	Medium white	
15	Ceiling height	20	ft			
16	Wall color	Off-white paint		**Skylight Well**		
17				Light well height	1	feet
18	**Shelving/Racks or Partitions?**			Well color	Off-white paint	
19	Partitions, [✓ Shelves/Racks,] None/Open			Safety grate or screen	Yes, [✓ No]	
20	Shelf/rack height	7	ft			
21	Shelf/rack width	8	ft	**Heating and Air Conditioning Systems**		
22	Aisle width	8	ft	Air Conditioning	Mechanical A/C	
23	No data required		ft	Heating System	Gas/Oil Furnace	
24	Check Lighting Power Density on Optional_Input tab					
25	**Electric Lighting**			**Utilities**		
26	Lighting system	Open cell fluorescent		Average Elec Cost	$0.120	kWh
27	Fixture height	20	ft	Heating Fuel Units	Therm	
28	Lighting control	Two level + off switching		Heating Fuel Cost	$1.000	/Therm

Figure 4.11: SkyCalc basic inputs

The SkyCalc spreadsheet contains also some default tables for different skylighting systems and building operation types. Moreover the climate data of some US cities are already available with the program. If the desired location is not in the pull-down menu, the weather data can be uploaded. This data tells the calculator the amount of daylight and the heating and cooling conditions in the location of building and how much cooling and heating is needed to keep the comfort requirements. SkyCalc is using a specific weather file format that can be created using eQuest 3.61 (eQuest, 2008) simulation program. So theoretically it is possible to create the weather file for any location in the world, provided that the DOE2 compatible weather file is available.

Depending on the skylight characteristics such as glazing type, glazing layers and glazing colour, the SkyCalc considers predefined values for VT, SHGC and U-value from the skylight default tables and uses these values for energy calculations. SkyCalc does hour by hour analysis, using the DOE 2.1 simulation program as a pre-processor, and Typical Meteorological Year (TMY) weather data.

The output of SkyCalc is both in tabular and in graph forms and provide the lighting and estimates the annual energy (kWh/year) and cost savings for lighting, heating, and cooling compared to no-skylight status. The amount of energy saving of a given skylighting is also presented over a range of skylight-to-floor area ratios. This has been shown in figure 4.12, where current design’s energy efficiency is compared for different skylight-to-floor ratios.

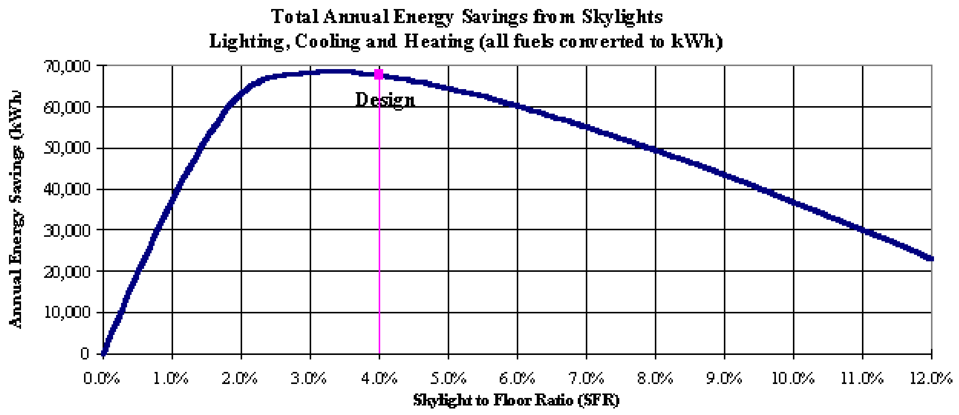
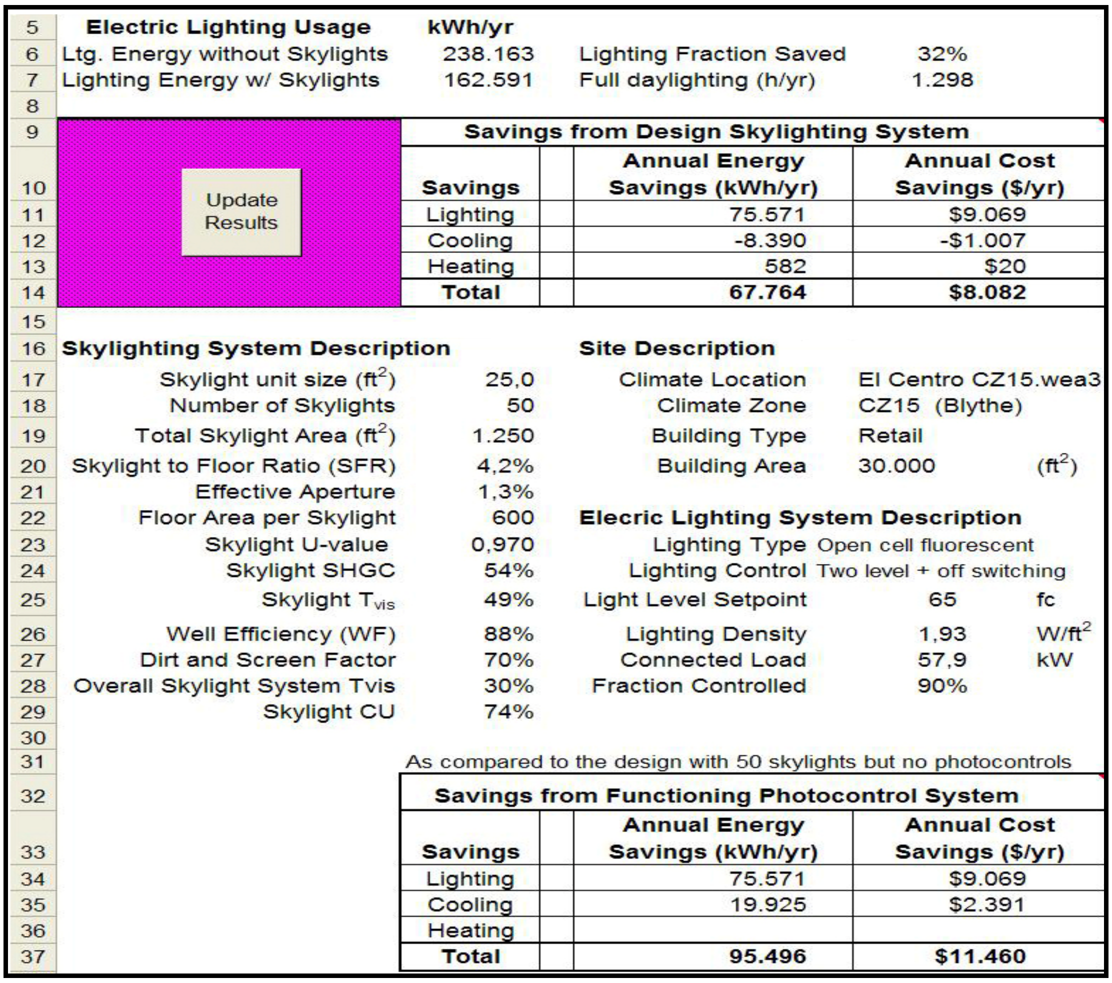


Figure 4.12: Annual energy saving graph calculated by SkyCalc

Please note that figure 4.12 clearly shows the importance of correct selection of skylight sizes. For bigger skylight-to-floor ratios the amount of energy that is used to keep the building cool in warm seasons is getting more and more and cuts the lighting and heat advantages in colder seasons. Another important factor that plays a significant role in the skylight selection is the climate condition. So for example a skylight that is ideal for a specific climate condition will not deliver the same energy efficiency level in other climates.

Figure 4.13 shows the tabular output of SkyCalc results. Please also note that the calculated annual cost saving are calculated based on a default energy price and user can customize the given prices.



5	**Electric Lighting Usage**	**kWh/yr**		
6	Ltg. Energy without Skylights	238.163	Lighting Fraction Saved	32%
7	Lighting Energy w/ Skylights	162.591	Full daylighting (h/yr)	1.298
8				

		Savings from Design Skylighting System		
9				
10	Update Results	**Savings**	**Annual Energy Savings (kWh/yr)**	**Annual Cost Savings ($/yr)**
11		Lighting	75.571	$9.069
12		Cooling	-8.390	-$1.007
13		Heating	582	$20
14		**Total**	**67.764**	**$8.082**
15				

16	**Skylighting System Description**		**Site Description**		
17	Skylight unit size (ft^2)	25,0	Climate Location	El Centro CZ15.wea3	
18	Number of Skylights	50	Climate Zone	CZ15 (Blythe)	
19	Total Skylight Area (ft^2)	1.250	Building Type	Retail	
20	Skylight to Floor Ratio (SFR)	4,2%	Building Area	30.000	(ft^2)
21	Effective Aperture	1,3%			
22	Floor Area per Skylight	600	**Elecric Lighting System Description**		
23	Skylight U-value	0,970	Lighting Type	Open cell fluorescent	
24	Skylight SHGC	54%	Lighting Control	Two level + off switching	
25	Skylight T_{vis}	49%	Light Level Setpoint	65	fc
26	Well Efficiency (WF)	88%	Lighting Density	1,93	W/ft^2
27	Dirt and Screen Factor	70%	Connected Load	57,9	kW
28	Overall Skylight System Tvis	30%	Fraction Controlled	90%	
29	Skylight CU	74%			
30					
31			As compared to the design with 50 skylights but no photocontrols		

	Savings from Functioning Photocontrol System		
32			
33	**Savings**	**Annual Energy Savings (kWh/yr)**	**Annual Cost Savings ($/yr)**
34	Lighting	75.571	$9.069
35	Cooling	19.925	$2.391
36	Heating		
37	**Total**	**95.496**	**$11.460**

Figure 4.13: SkyCalc table results

The SkyCalc results provide a good estimation for selection of better skylights, however the tool is not being updated by software maintainers and also lacks the integration possibilities with BIM and real world product libraries.

4.2.6 SkyDreamer Calculator

The SkyDreamer's calculation web service is supported by a Java library that has been implemented based on the SkyCalc energy calculator spreadsheet. In this section this energy calculation methodology will be explored in more details.

As mentioned before the skylights will affect the building energy consumption in many ways. First of all the skylights will reduce the need to electric lighting and consequently prevent the heat gains of electric lighting when daylighting controls are used. Skylights also increase the solar gain and also the thermal transmittance of roofs. Other factors such as climate information, building occupancy and operating process equipments will also play an important role in building energy simulation and should be considered in calculations of the heating or cooling load of the selected building.

The required information of SkyDreamer's calculation web service is derived from the following resources:

- The building information that are provided by end-user either by direct input in web interface or extracted from the provided building model
- Default occupancy schedules, comfort ranges, and producer-neutral skylight performance characteristics which are included in calculation web service statically
- The hourly climate data of selected building location which has been generated by any DOE-2 building energy simulation program for a reference building
- And finally the java methods that provide the required data for energy calculation algorithm

The DOE-2 reference building has the following characteristics (SkyCalc Guide, 2008):

- 100X100 square foot building with a 20 foot ceiling, task height is 2.5 feet above the floor
- 4% skylight to floor ratio (SFR), shading coefficient = 1.0, effective aperture = 2 % (overall Tvis = 0.5)

- Roof U-Value = 0.057, Skylight overall U-Value = 1.0 Btu/hr•°F•ft2, adiabatic walls
- Both lighting and equipment power density are 1.5 W/ft2
- Occupant density is 1 person per 100 square foot
- Set points: Cooling = 72 degrees Fahrenheit (22.22 degrees Celsius) and , Heating = 68 degrees Fahrenheit (20 degrees Celsius)
- All schedules (lighting, occupancy, process loads) are set at 100% for 24 hours per day and 7 days per week
- Daylighting controls are disabled

The SkyDreamer calculator uses these values internally and converts them to metric system for sake of consistency with other information resources such as building model. SkyDreamer calculator uses the lumen method algorithm (IES, 1993) to calculate the coefficient of utilization for skylights. The key assumptions of this method are (SkyCalc Guide, 2008):

- The skylights are completely diffusing (Lambertian distribution) and uniformly spaced
- Each surface in the room is diffusely reflecting
- Each major surface of the room is uniformly illuminated

An important factor in daylighting using skylights is the Effective Aperture (EA) of a skylight. Effective Aperture that describes the fraction of daylight is transmitted through the roof and is calculated as follows:

EA = SFR * Overall Visible Light Transmittance * Well Efficiency

Where,

- **SFR**: is the Skylight to Floor Ratio for the given space
- **Well Efficiency**: is the fraction of light entering the well that leaves the light well. Well efficiency is calculating using the lumen method (IES, 1993) from the skylight dimensions, well depth and well reflectance. Skylight well is modelled as a space having the same geometric relationships as the light well

with a 99% reflective ceiling, a 0% reflective floor, and wall reflectance matching that of the reflectance of the light well walls.

- **Overall Visible Light Transmission**: this accounts for overall visible transmittance of skylight glazing and diffusers or lenses in the light well. Also the dirt build-up on the skylight and in the light well should be considered. The SkyDreamer calculator used a 70% dirt factor as recommended in the IESNA handbook (IES, 1993) for horizontal glazing in clean areas.

Listing 4.4 shows the Java method that is responsible for calculation of the Effective Apertures of a skylight. As the first step (point 1), this method will look for the Visible Light Transmittance (Tvis) which is a property of the glass or plastic glazing material. The Visible Light Transmittance is the fraction of the light that is transmitted through the glazing. This value is usually found in the manufacturer's catalog for the skylight products. It is important to note that the provided Tvis might only be given for the glazing material and does not include the effects of the framing. In case of NFRC (NFRC energy performance labels as described in 4.2.2), ratings also include the effects of the framing. As mentioned before the other factor that affects the Overall Visible Transmission is the Dirt Factor (DF) that has been applied by applying the recommended dirt light loss factor value in IESNA handbook (IES, 1993) for horizontal glazing in clean areas which is 70% of the screen or safety grate factor (point 3).

In order to calculate the Well Efficiency, the `getEffectiveAperture` method calls another method called `CU` that performs the Lumen method calculation for the light well (point 2). The `CU` method considers the light well as a space having a 99% reflective ceiling (parameter 5) and a 0% reflective floor (parameter 7).

```
public double getEffectiveAperture() {
  double vt = getSkylightTvis();❶
  double we = InputData.CU(0,width,length,lightWellHeight,     ❷
                           0.99,Light.getColorReflectance(wellColor),0);
  double dirtFactor = 0.70 * getGrateFactor();❸
  return getSFR() * vt * we * dirtFactor;
}
```

Listing 4.4: Calculation of Effective Aperture of skylights

Listing 4.5 shows the CU method that displays an important role in SkyDreamer calculator. The method basically calculated the Coefficient of Utilization for any given space and different luminaries as light source (first parameter). SkyDreamer calculator uses the user selected lighting option and the internal default tables to get the corresponding luminaries intensities. In the previous case (calculation of Well Efficiency) it was applied on skylight well to calculate the fraction of light that will leave the light well.

```
public static double CU(int L_ints_index, double w, double l, double h,
                        double ceilRefl, double wallRefl, double floorRefl) {

  double[] kGN = new double[10];
  double PhiD = 0;
  double PhiU = 0;
  double DG = 0;

  double G = RCR(w, l, h);

  // Index = 0 -> Lambertian Distribution
  if ( L_ints_index > 0) {

    double PhiN;
    double FluxD = 0;
    double FluxU = 0;

    // Zonal Multiplier Equation Constants Fig 9-27 IESNA handbook 8th Ed.
    kGN[1] = 1;
    kGN[2] = Math.exp(-0.041 * Math.pow(G , 0.98));
    kGN[3] = Math.exp(-0.07 * Math.pow(G , 1.05));
    kGN[4] = Math.exp(-0.1 * Math.pow(G , 1.12));
    kGN[5] = Math.exp(-0.136 * Math.pow(G , 1.16));
    kGN[6] = Math.exp(-0.19 * Math.pow(G , 1.25));
    kGN[7] = Math.exp(-0.315 * Math.pow(G , 1.25));
    kGN[8] = Math.exp(-0.64 * Math.pow(G , 1.25));
    kGN[9] = Math.exp(-2.1 * Math.pow(G , 0.8));
```

```
      double[] intens_range = Light.getLuminaireIntensity(L_ints_index);

      for (int theta = 1; theta < 10; theta++){
        PhiN = intens_range[theta-1] * (Math.cos((theta - 1) * Math.PI / 18) –
               Math.cos(theta * Math.PI / 18));

        FluxD += PhiN;
        DG += kGN[theta] * PhiN;
      }

      FluxD = FluxD * 2 * Math.PI;
      DG *= 2 * Math.PI / FluxD;

      // This is equivalent to dividing the summation of kGN*PhiN by total lumens
      // and fraction downward (eq 9-68)
      for (int theta = 10; theta <= 18; theta++){

        FluxU = FluxU + intens_range[theta-1] * (Math.cos((theta - 1) *
                Math.PI / 18) - Math.cos(theta * Math.PI / 18));
      }

      FluxU = FluxU * 2 * Math.PI;
      //FluxT = FluxD + FluxU;
      // Phi = Fraction of flux in Down and Up directions
      // IES has normalized intensities to 1000 lamp lumens
      // FluxT is total flux leaving the fixture
      PhiD = FluxD / 1000;
      PhiU = FluxU / 1000;
    } else {
      //Lambertian Distribution - perfect diffuser
      PhiD = 1;
      PhiU = 0;
    }

    double result = 0;
    if (G < 0.01){
      result = (PhiD + ceilRefl * PhiU) / (1 - ceilRefl * floorRefl);
    } else {
      double F = Fexch(w, l, h);
      //Lambertian Distribution
      if (L_ints_index == 0) {
        DG = F;
      }

      double C1 = (1 - wallRefl) * (1 - Math.pow(F , 2)) * G / (2.5 * wallRefl *
                  (1 - Math.pow(F , 2)) + G * F * (1 - wallRefl));

      double C2 = (1 - ceilRefl) * (1 + F) / (1 + ceilRefl * F);
      double C3 = (1 - floorRefl) * (1 + F) / (1 + floorRefl * F);

      double C0 = C1 + C2 + C3;

      double CU1 = 2.5 * wallRefl * C1 * C3 * (1 - DG) * PhiD /
                   (G * (1 - wallRefl) * (1 - floorRefl) * C0);

      double CU2 = ceilRefl * C2 * C3 * PhiU / ((1 - ceilRefl) *
                   (1 - floorRefl) * C0);
      double CU3 = (1 - floorRefl * C3 * (C1 + C2) / ((1 - floorRefl) * C0)) *
                   DG * PhiD / (1 - floorRefl);
      result = CU1 + CU2 + CU3;
    }
    return result;
  }
```

Listing 4.5: method to calculate Coefficient of Utilization

The CU method makes some method calls to fulfill the calculation tasks. Some of these methods will be explored in more details.

First of all, the CU method needs the Cavity Ratio of the given space. The Cavity Ratio is a single value that describes the proportions of the given space and is usually referred to as Room Cavity Ration (RCR) for a room space, or Well Cavity Ration (WCR) for the skylight well space. Listing 4.6 shows the method that calculates the Cavity Ratio of the given space.

```
public static double RCR(double W, double l, double d) {
  return 5 * d * (W + l) / (W * l);
}
```

Listing 4.6: Room Cavity Ratio (identical to the well cavity ratio)

Another required value for calculation of Well Efficiency is the Form Factor for two equal sized parallel rectangles. Listing 4.7 shows the implementation of Form Factor as presented in IESNA handbook (IES, 1993).

```
public static double Fexch(double width, double length, double depth){

  double x = length / depth;
  double y = width / depth;

  double F1 = 2 / (Math.PI * x * y) * Math.log(Math.pow((1 + Math.pow(x , 2)) *
     (1 + Math.pow(y , 2)) / (1 +Math.pow(x , 2) + Math.pow(y , 2)) , 0.5));

  double F2 = 2 / (Math.PI * x) * Math.pow(1 + Math.pow(x , 2) , 0.5) *
     Math.atan2(y, Math.pow(1 + Math.pow(x , 2) , 0.5));

  double F3 = 2 / (Math.PI * y) * Math.pow((1 + Math.pow(y , 2)) , 0.5) *
     Math.atan2(x, Math.pow((1 + Math.pow(y , 2)) , 0.5));

  double F4 = -2 / (Math.PI * x) * Math.atan2(y, 1) - 2 / (Math.PI * y) *
     Math.atan2(x,1);
```

```
    return F1 + F2 + F3 + F4;
  }
```

Listing 4.7: Form factor for two equal sized parallel rectangles

The illuminance of the building with skylights is related to illuminance of the reference building (included in DOE-2 weather file) by means of simple ratios by comparing the Effective Apertures and Coefficient of Utilization that describes what fraction of light exiting from the bottom of the light well reaches the work surface.
Thermal losses due to skylights are modeled using a simple UA (conductance area product) equation. This steady-state heat transfer method does not consider thermal storage of heat in the mass of the building. In contrast, the solar heat gain model, which scales the hourly solar loads from the DOE-2 reference building by the relative area of the skylights and their solar heat gain coefficient, does reflect the thermal capacitance of the reference building. The other thermal loads of occupancy and equipment are also added in to arrive at the total zone heating or cooling load for that hour.
At this point, the zone loads for each hour and the sum of electricity consumption for electric lighting for both the base case building without skylights and the area under skylight, designed building have been stored. The maximum cooling load is also stored for sizing of the heating and air conditioning systems.
A HVAC system model then evaluates the energy consumption required by the hourly building loads. This model allows the user to specify an outside air economizer that can displace some or the entire cooling load when the outside air is sufficiently cool. This model also varies the heat pump efficiency depending upon outside temperature. (As the outside temperature drops, the heat pump needs more electricity per Btu of heat generated.) (SkyCalc Guide, 2008).

The SkyDreamer calculator extends the SkyCalc in many directions. First of all the new calculator is designed as a java implemented library that facilitates the use of energy calculation services both as a published web service for internet computing and also as a standalone java library for local java applications. Moreover the maintenance of the calculator web service is much easier compared to SkyCalc's legacy code where energy calculation logic is distributed among cell calculations and VBA calculation modules. So for instance the calculation results can be tested automatically by means of automated test methods such as JUnit testing framework (JUnit, 2008) for java applications.

From the AEC/FM perspective, the new energy calculator service is fully integrated with modern BIM concepts and also can repeat the energy calculation for the product alternatives that are semantically described in a product repository. The BIM integration facilitates the data entry process greatly and the required information is directly extracted from building model by building parser component. As a result, the calculation component can be fed on the fly from building model and no extra user interaction is needed. Some other advantages of SkyDreamer calculator over ScyCalc tool are as follows:

- SkyCalc uses United States customary units but SkyDreamer uses metric System.
- SkyCalc calculates the energy saving compared to the no-skylight status but SkyDreamer calculates the annually required energy consumption (KWh/year) for the selected skylight product.
- SkyCalc provides only the generic skylight component but SkyDreamer is dealing with real world products that are semantically stored in a product repository.
- SkyDreamer is equipped with a Web 2.0, elaborated user interface that make the user interaction much easier. The Web 2.0 interface will be explored in the next section.

The SkyDreamer calculator is a rather complex component with multiple packages and java classes, however it provides a simple API for the end-users to configure and call the calculation service. In order to use this service, first a use case configuration with the following items is provided:

- The weather file of the building location
- The building geometry
- Building properties such as building type (residential, office, etc)
- Lighting type and corresponding control types

- HVAC system types for cooling and heating
- Generic skylight characteristics

Listing 4.8 shows a typical use case configuration which is done programmatically by a java client.

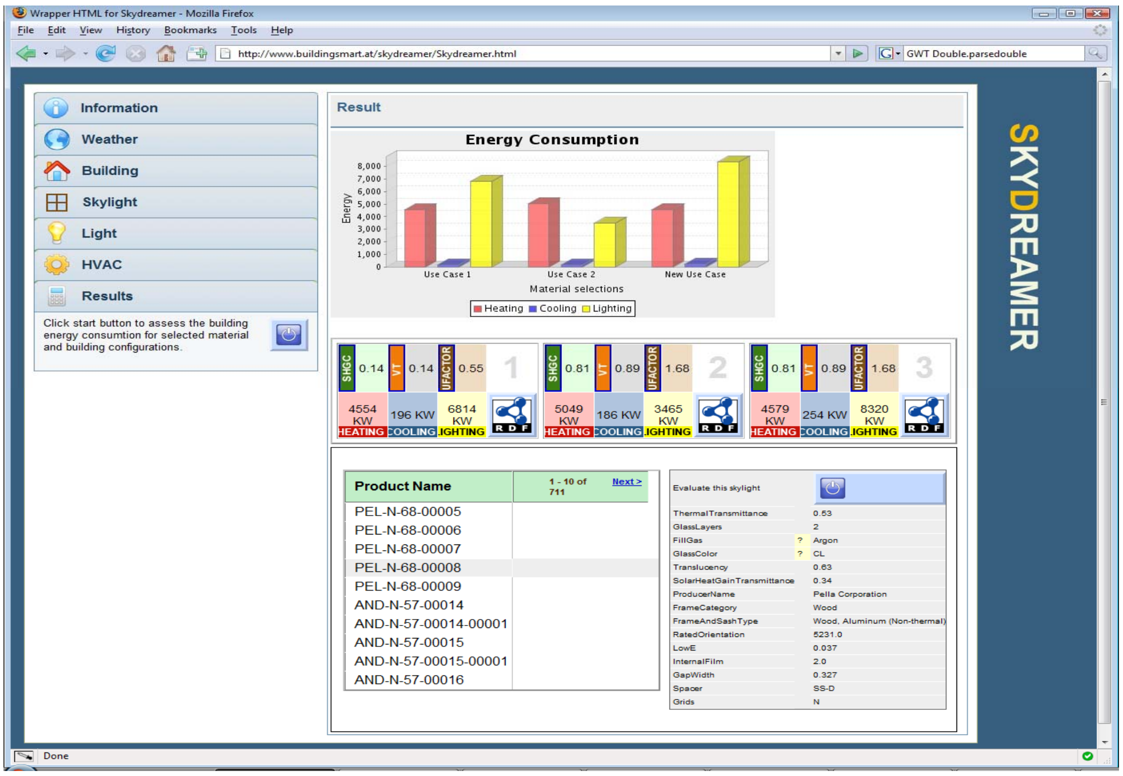
```java
// weather configuration
Weather wea = new Weather("Frankfurt.wea3");

// building configuration
Building bldg = new Building(Building.CLASSROOM_K12_12MON,
                             10,20,200, Light.PAINT_CONCRETE );

// light configuration
Light light = new Light(Light.HIGHT_BAY_METAL_HALIDE,
Light.CNTRL_TWO_LEVEL,Light.SHELVING_NONE_OPEN);
light.setLightHeight(20);

// HVAC configuration
Hvac hvac = new Hvac( Hvac.COOLING_MECHANICAL, Hvac.HEAT_GAS_OIL_FURNACE);

// skylight configuration
Skylight skylight = new Skylight(
               Skylight.DOME_SHAPE,
               Skylight.ACRYLIC,
               Skylight.SINGLE_GLAZED,
               Skylight.CURB_WOOD,
               Skylight.FRAME_METAL,
               Skylight.COLOR_OPTION1);

// (theLength, theWidth, theWellHeight, skyNum)
skylight.setDimensions(2, 2, 1, 4);
skylight.setWellColor(Light.PAINT_BRIGHT_COLORS);
skylight.setLight(light);
skylight.setDiffuser(true);

// evaluating the overall configuration
SkycalcBean result = Calculator.evaluate(bldg, skylight, wea,hvac);

// printing out the results
System.out.println("Required cooling  : " + result.getCool());
System.out.println("Required heating  : " + result.getHeat());
System.out.println("Required lighting : " + result.getLightPower());
```

Listing 4.8: Use case configuration programmatically

4.2.7 User Interface

The SkyDreamer system is implemented as a web based system that accepts the user input and accesses the backend services for fulfilling the user requests. In order to simplify the user interaction, the front end has been implemented using a modern Web 2.0 concept that recently has set a new trend in Rich Internet Application (RIA) world. It makes better use of the client machine’s processing power and at the same time pushes the SOA paradigm to its limits. At the moment most SOAs are conceptually trapped inside an organizations' intranet and Web 2.0 envisions building collective intelligence and mashed up functionality based on web services. In this environment, Internet will play the role of a global operating system that hosts the web services. In other words the Web 2.0 is a step toward the global cloud computing idea where business services are presented on Internet and developers should select and weave them together to create new compound services.

Thus Web 2.0 is much more than adding a nice facade to old web applications rather it is a new way of thinking about software architecture of RIAs. In comparison to traditional web applications, the application logic of modern Web 2.0 applications tends to push the interactive user interface tasks to the client side. The client components on the other hand negotiate with remote services that deal with user events. Figure 4.14 shows a screenshot of the SkyDreamer user interface.



Click start button to assess the building energy consumtion for selected material and building configurations.

	SHGC	VT	UFACTOR	Heating	Cooling	Lighting
1	0.14	0.14	0.55	4554 KW	196 KW	6814 KW
2	0.81	0.89	1.68	5049 KW	186 KW	3465 KW
3	0.81	0.89	1.68	4579 KW	254 KW	8320 KW

Product Name	1 - 10 of 711 Next >
PEL-N-68-00005	
PEL-N-68-00006	
PEL-N-68-00007	
PEL-N-68-00008	
PEL-N-68-00009	
AND-N-57-00014	
AND-N-57-00014-00001	
AND-N-57-00015	
AND-N-57-00015-00001	
AND-N-57-00016	

Evaluate this skylight		
ThermalTransmittance		0.53
GlassLayers		2
FillGas	?	Argon
GlassColor	?	CL
Translucency		0.63
SolarHeatGainTransmittance		0.34
ProducerName		Pella Corporation
FrameCategory		Wood
FrameAndSashType		Wood, Aluminum (Non-thermal)
RatedOrientation		5231.0
LowE		0.037
InternalFilm		2.0
GapWidth		0.327
Spacer		SS-D
Grids		N

Figure 4.14: SkyDreamer screenshot

The Web 2.0 processes generally use Ajax (Asynchronous JavaScript and XML) development technique for creating interactive web applications. The main intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes so that the entire web page does not have to be reloaded each time the user requests a change. This is intended to increase the web page's interactivity, speed, functionality, and usability. In AEC/FM context where use cases should be configured with huge amount of input information and the longtime wizards and processes are required, the Ajax technology may provide significant advantages. For instance, Ajax allows every element within a web interface to be individually and quickly updated without affecting the rest of the interface. At the time of writing this dissertation, the SkyDreamer is the only available Web 2.0 application in AEC field on Internet but due to the great advantages of Web 2.0 it is very viable that new applications arrive very soon.

To implement Ajax based applications, multiple development frameworks is available. For case of SkyDreamer, the Google Web Toolkit (GWT, 2008) has been used which is an open source Java software development framework that allows web developers to develop Ajax applications in Java and benefit the Java best practices at implementation phase. However at runtime, the GWT compiler will translate the Java code to browser-compliant JavaScript and HTML that only needs a web server for launching the application pages and as a result no java-enabled server components will be needed at runtime.

Chapter 5

RESULTS

In the previous chapters, it was explained how different components interact with each other and fit into the SkyDreamer architecture to address the dissertation goals. In this chapter different use cases of SkyDreamer prototype will be explored and the project results will be discussed. The SkyDreamer prototype has been fully implemented based on Open Source Software and open standards and the web application is available as a free service at http://www.pixdeal.com/skydreamer. Interested readers are invited to test it and explore the use cases online.

As explained before, the SkyDreamer prototype aims to bridge the gap between BIM, product library and simulation services. So the system will need the relevant information about building, building context and also the thermal attributes of skylight products to run energy calculations. The required configuration information should be provided first via SkyDreamer's web interface. This configuration consists of the following items:

- Building envelope including the size and number of skylights for a selected space
- Weather data for building's location
- Heating, cooling, and lighting systems
- Common skylight options such as glazing type, layers and colour

After assessing the use case configuration, users will be able to:

- make energy calculations for a producer-neutral configuration and present the results graphically

- repeat the calculation for alternative use case configurations (different skylight glazing options, different weather data, different HVAC options, etc) and compare the energy consumption results
- configure the building model with real world skylight products that are coming from Semantic Skylight Repository and benchmark the energy consumption results of different use case configurations.

5.1 Use Case Configuration

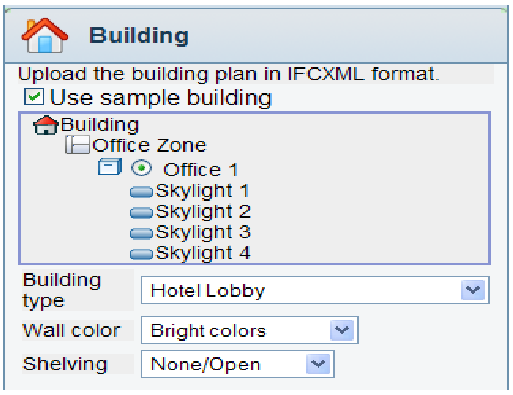
In this section, different use case configuration options and user interactions with the system will be explored in more details and then the prototype results and also the optimum skylight strategy options will be discussed.

5.1.1 Building configuration

In order to calculate the energy efficiency of a building some basic building information such as building function, building geometry, and material is required. In the present scenario, the user first provides the building model, which is used to extract spaces and skylights information. The SkyDreamer prototype facilitates building's data entry by extraction of the relevant data from building model, which should be provided as an IFCXML file. So the end user can easily design the building using any IFC-enabled CAD Software and export the building design in IFCXML format. This file will be then submitted to the SkyDreamer system, for further processing. Alternatively, user may choose a default building model incorporated in the system. This will help the users to test the skylight without having to provide the building model in IFCXML format that might be difficult for some users.

Subsequently, the uploaded building model is parsed and user will be able to navigate through zones and spaces to choose the target space. After selection of the target space, SkyDreamer will also calculate the "skylight to floor area ratio" (SFR) which will be used for energy calculation in the following steps. The energy calculation algorithm will later on calculate the energy consumption for the selected building space. Figure 5.1 shows the building configuration section where the

"Office 1" space of the built-in sample map has been selected for further calculations. The building navigator also depicts the skylight to space assignments and for the given case of figure 5.1, the "Office 1" space has four skylights. The other building calculation options are building type, wall colour and shelving option that will affect the energy calculation algorithm. For instance, the selected building type option will determine the building's lighting, occupancy, and process load schedules as well as the thermal and visual comfort ranges, which are available as part of the SkyDreamer calculator component.

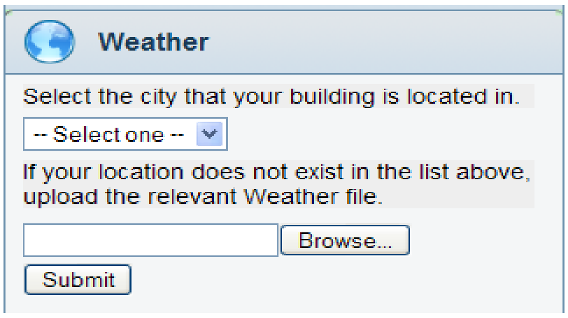

Building
Upload the building plan in IFCXML format.
☑ Use sample building
Building
Office Zone
Office 1
Skylight 1
Skylight 2
Skylight 3
Skylight 4

Building type	Hotel Lobby
Wall color	Bright colors
Shelving	None/Open

Figure 5.1: Building navigator

5.1.2 Weather configuration

The weather information is another required configuration that affects the building energy calculations. The weather information specifies the available daylight, the solar heat intensity, and needed amount of heating and cooling. SkyDreamer uses SkyCalc weather files (.wea3) that can be generated using eQuest 3.61 (eQuest, 2008). So the end-user will be able to create his/her own weather files by feeding the eQuest with the DOE2 compatible weather files that are available for many locations. Alternatively SkyDreamer provides a list of pre-processed weather files for selected locations that can be simply selected by the end-user without dealing with complexity of weather file formats. Figure 5.2 shows the weather information panel that accepts either a pre-processed weather file from the given list or a weather file that should be uploaded to the system.


Weather

Select the city that your building is located in.

-- Select one --

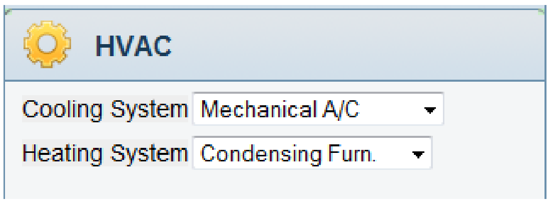
If your location does not exist in the list above, upload the relevant Weather file.

Figure 5.2: Selection of weather file

5.1.3 HVAC and lighting configurations

In addition to the building model information mentioned above, SkyDreamer needs to know the properties of building's lighting and HVAC systems.

For heating and cooling systems, SkyDreamer provides a list of options that contains the most common heating and cooling systems. The selected option will be used to estimate the amount of energy needed to keep the building comfort ranges for inhabitants for each day based on the given weather file. Figure 5.3 shows the HVAC panel in SkyDreamer prototype.

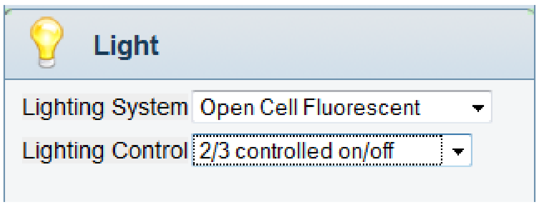

HVAC

Cooling System Mechanical A/C

Heating System Condensing Furn.

Figure 5.3: Selection of HVAC options

For lighting calculation, the SkyDreamer calculator assumes a default lighting power density based on building characteristics and building type. Moreover the lighting system and lighting control options in combination with weather data are used to estimate the required energy for lighting. Setting the Lighting Control option to "No Daylight", means the user wants to evaluate only the skylights' energy-related implications without daylighting controls. Figure 5.4 shows the light panel in SkyDreamer prototype where an "open cell fluorescent" with "2/3 controlled on/off" has been selected.



Light

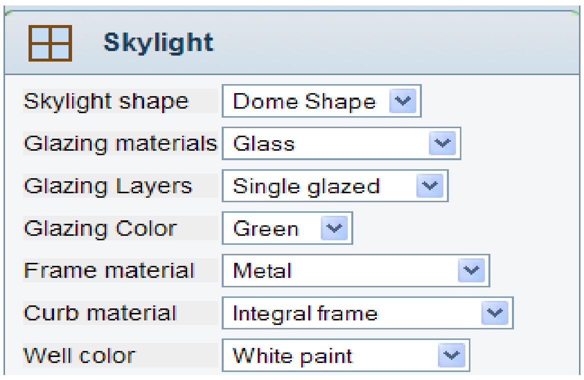
Lighting System: Open Cell Fluorescent

Lighting Control: 2/3 controlled on/off

Figure 5.4: Selection of light options

5.1.4 Skylight configuration

Next, the user needs to provide the physical characteristics of the skylights and light wells by choosing a generic product from the list. Based on this selection, initial skylight's thermal and optical properties are assigned to the use case. These values can later be interactively modified by the user. The system identifies a real product with the corresponding user-desired property, and re-computes the performance indicators. Figure 5.5 shows the required properties that should be set for a generic skylight component.



Skylight

Skylight shape	Dome Shape
Glazing materials	Glass
Glazing Layers	Single glazed
Glazing Color	Green
Frame material	Metal
Curb material	Integral frame
Well color	White paint

Figure 5.5: Generic skylight configuration

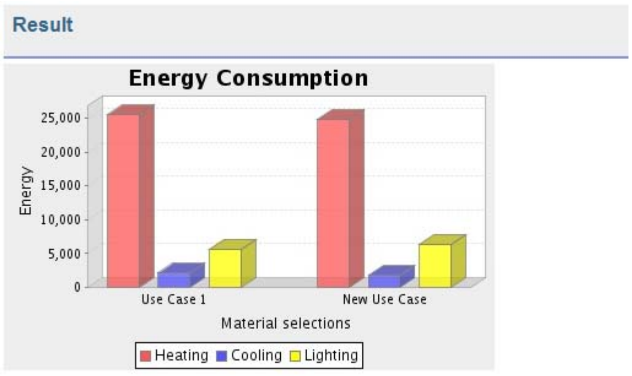
5.2 Simulation process

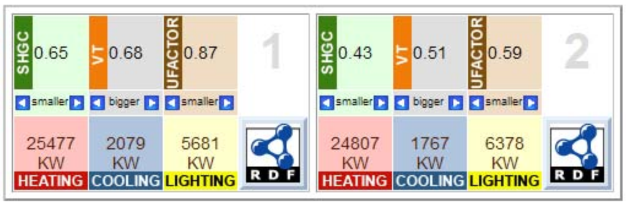
After providing the building configuration, SkyDreamer calculates energy demand for heating and cooling. The result of simulation is also presented in graphic form that is suitable for benchmarking. After running the simulation for the first time, the user will be able to change the building configuration and run the simulation again (see Figure 5.6). As a result, this online simulation tool helps the designers to easily evaluate the effect of their design decisions.

It is important to note that the currently implemented SkyDreamer's energy calculator is a simple one (single-storey building, rectangular floor plan) and serves for demonstration of system's capabilities. Also the applied calculation method will not accurately model clear skylights, non-uniform spacing, or high partitions and in such cases a more sophisticated simulation program would be required.

Figure 5.6 shows the energy calculation results which consist of two parts:

- Benchmark part (upper part) that can be easily used to compare different use case configurations. This part depicts the yearly required heating, cooling and lighting energy for each use case.
- Dataset part (lower part) that includes the numerical form of the benchmark part plus the thermal attributes of the selected skylight component (SHGC, VT, and U-value). The thermal attributes for the producer-neutral run, are selected from SkyDreamer defaults and after the selection of a skylight component, the real product values are selected. The dataset part is also used to formulate the product query (semantic query) that will return the set of similar products in sense of energy performance from semantic product repository. The next section explores the selection of skylight components in more details.





	SHGC	VT	UFACTOR	Heating	Cooling	Lighting
1	0.65 smaller	0.68 bigger	0.87 smaller	25477 KW	2079 KW	5681 KW
2	0.43 smaller	0.51 bigger	0.59 smaller	24807 KW	1767 KW	6378 KW

Figure 5.6: Simulation results

The skylight benchmarking results can be interpreted differently based on the different building requirements. For instance, figure 5.6 shows the results for two different use case configurations. From the heating and cooling perspective, the second use case is a better choice because:

- U-value is smaller so keep more heat inside the home and reduce excess work for heating systems
- SHGC is smaller so it reduces summer cooling and overheating

And from the lighting perspective, use case one provides a better performance because the skylight's Visual transmittance is large and consequently the benefit of natural day lighting is maximized.

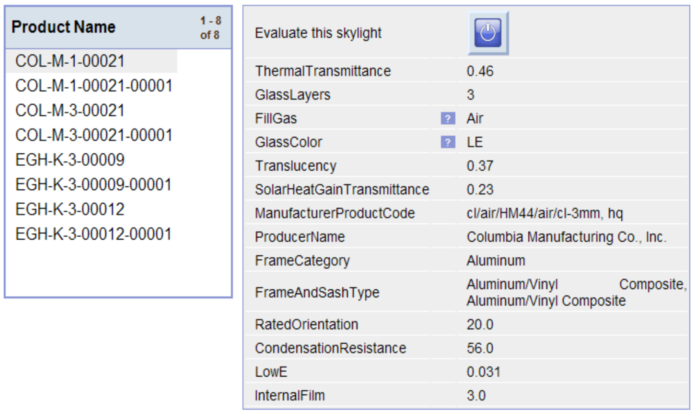
By comparison of results and also considering the local energy prices for heating, cooling and lighting, architect will select the better choice.

5.2.1 Selection of Skylight component

After running the simulation with generic skylight components, it is finally possible to select actual skylight products from the semantic repository and re-run the simulation. The selection criteria are the SHGC, VT and U-value of the skylights. User can select his/her choice by altering the selection condition (larger, smaller) for each of these three parameter and the system will translate user's query into ontology query language SPARQL (SPARQL, 2008) that runs against the semantic repository. As soon as the result is displayed, user can navigate through the results set and select the appropriate skylight component and repeat the simulation process.

Figure 5.7 shows a sample query result that is rendered as HTML (the original query results are RDF triples). The query result panel consists of a list of products on the left hand side and the product details for the selected item on the right hand side. As explained above, the product data are translated from resulting RDF triples which might also include the definition of some items. For instance the "FillGas" and "GlassColor" attribute values have an additional description that will be rendered as a help tool-tip (this value will be displayed when the mouse is over the help button in front of these attributes). After selection of the desired item, user may click on the calculation button (top right corner of

figure 5.7) to re-run the simulation.



Product Name 1 - 8 of 8

- COL-M-1-00021
- COL-M-1-00021-00001
- COL-M-3-00021
- COL-M-3-00021-00001
- EGH-K-3-00009
- EGH-K-3-00009-00001
- EGH-K-3-00012
- EGH-K-3-00012-00001

Evaluate this skylight	
ThermalTransmittance	0.46
GlassLayers	3
FillGas ?	Air
GlassColor ?	LE
Translucency	0.37
SolarHeatGainTransmittance	0.23
ManufacturerProductCode	cl/air/HM44/air/cl-3mm, hq
ProducerName	Columbia Manufacturing Co., Inc.
FrameCategory	Aluminum
FrameAndSashType	Aluminum/Vinyl Composite, Aluminum/Vinyl Composite
RatedOrientation	20.0
CondensationResistance	56.0
LowE	0.031
InternalFilm	3.0

Figure 5.7: Skylight product query results

Chapter 6

Outlook

The vital need for integration in Architecture, Engineering and Construction domains has forced the emergence of smart building information models that can efficiently capture the building information and, at the same time, be uniformly integrated in business processes. The smart BIM initiatives have been already started and many researchers are now focusing on new concepts such as IFC, IFD, and BuildingSMART to improve the information exchange in AEC/FM domains and address different requirements of building industry.

In this context, Semantic Web technologies can also be used as an enabler resource to increase the interactivity among different domains and professions of building industry. This new approach can cross the AEC domain borders by effective mapping of relevant contextual attributes onto available building industry information.

At the time this research was started, very few applications of Semantic Web in AEC field were known, however this situation has changed recently and a handful of national and international projects have employed the Semantic Web technologies for AEC/FM purposes. Nevertheless, since both BIM and Semantic Web technology are rather new research fields, more research work is required to mature the technology for real world applications.

6.1 Summary and Discussion

The research presented in this thesis is aiming at demonstrating the uniform integration of different resources in AEC/FM fields. More specifically, it shows how elaborate semantic technologies can be used to bridge the knowledge gap among manufacturers' data, building information models, and simulation web services (see Figure 6.1). A semantically enriched process helps the designer to find the desired product through automatic access to the building product libraries.

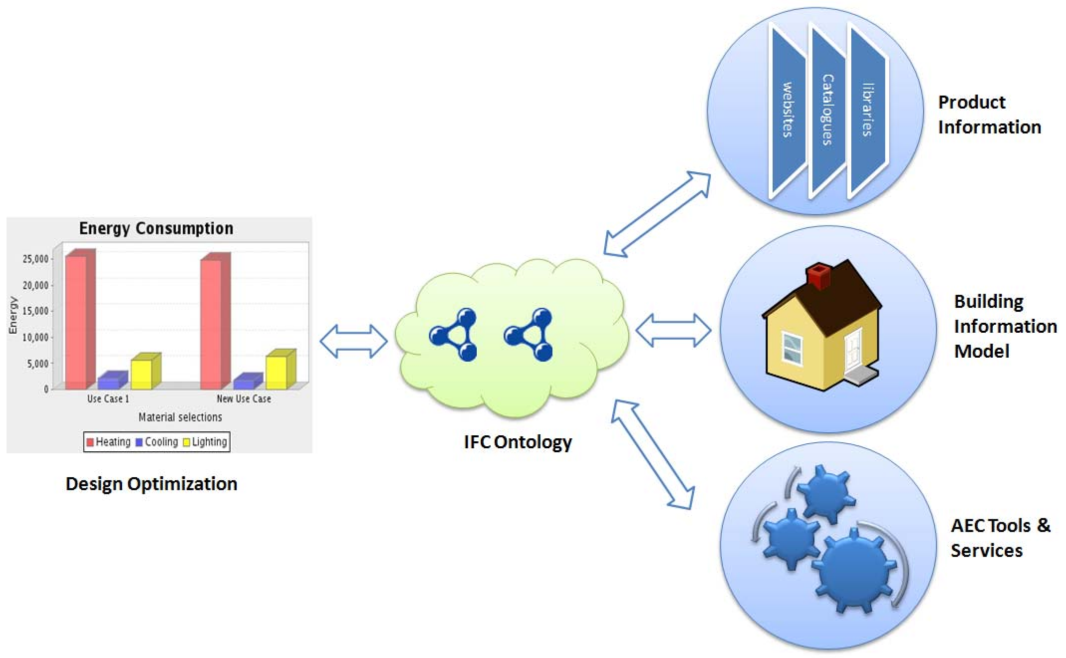


Figure 6.1: Bridging the AEC gaps using Semantic Web technology

In this section the research questions that were raised at the beginning of this thesis will be revisited to show how the proposed solution will address the challenging issues.

How can the semantics of building material and their associations be accurately modelled for open world interactions?

To establish a Semantic repository of building material, two basic parts are needed, namely: An ontology schema that describes the required elements and the product instances. In the proposed approach the IAI’s IFC have been used to establish the schema of the upper ontology which keeps the solution compatible with the current advances in AEC industry. Moreover, the generated upper ontology is empowered by best practices of the Semantic Web world and its added values such as semantic storage, semantic extension, semantic reasoning. The second part of the semantic repository, concerning product instances are extracted from the relevant websites and added to the ontology schema using a modular approach. The Semantic product repository can then interact with relevant AEC services and BIM via its IFC-compliant ontology.

How to define material libraries that can interact with global AEC services and share useful information about a specific material in an effective way?

As explained in the previous answer, both semantic product repository and AEC services use the same upper ontology that has been derived from IAI’s IFC. In other words, the rich semantics associated with the domain knowledge together with the Semantic Web Services allow bridging the gap between services of discrete knowledge domains.

How Semantic Web Technologies can improve the quality of built environment from energy efficiency standpoint?

The SkyDreamer prototype has specifically targeted the building’s energy efficiency. It shows how Semantic Web technology can be used to bridge the gaps between product libraries, BIM, and simulation services. As a result the building designers will be able to improve the quality of their building models by comparing the performance of alternative products and choosing the better options.

How a modern Building Information Model can interact with simulation programs and material libraries in an effective way and improve the performance in interoperability between applications?

The building information model of the proposed solution, which is also IFC-complaint, can be directly integrated with product library and also AEC services. In the SkyDreamer scenario much information has been directly or indirectly extracted from the model's IFCXML file format. Some of this information such as zones, spaces and areas are explicitly defined in model, but others such as skylight to floor ratio should be assessed by building navigator component. The IFC upper ontology will again play a significant role to connect the BIM elements to product library and AEC services.

How can building requirements and preferences be represented and how should they be taken into account in tailoring AEC services for a specific building?

In the proposed solution all elements are mapped to an IFC-complaint upper ontology where Semantic Web technologies support precise definition of requirements and preferences. As a matter of fact, the dynamic and extendable nature of Semantic Web will facilitate the information exchange with relevant AEC services. Moreover, the semantic reasoning approaches can be applied to building model and secondary IFC-complaint ontologies that describe the building requirements (such as building codes, national standards, etc) to check the design integrity during the lifetime of a building construction project.

How can business processes of a building project handle the partners' interaction in a semi-automated manner?

The semantic-enabled AEC services of each project partner, would allow the automatic discovery and composition of relevant services. Creating the semantic profile of these services based on IFC upper ontology can be seen as the first step toward integration of the business services across the AEC domains.

How can building material knowledge be extracted from existing material catalogues and prepared for usage by AEC applications such as simulation tools?

Different approaches of automatic information extraction in AEC context has been investigated in this research work. For the specific case of the SkyDreamer prototype a web extractor component extracts the skylight information from specific websites and stores them in a semantic repository. The SkyDreamer prototype also shows how a simulation service such as skylight calculator can interact with a semantic repository to retrieve the required product information that are used in energy simulation web service.

6.2 Future Work and Conclusion

While the difficulties in sharing and processing of the required information in AEC domain are generally known, effective solutions have not been found. Partial progress has been made in the area of building modeling to address the knowledge sharing challenges by advances such as development of IFCs (industry-foundation classes), IFDs, and the BuildingSMART initiative. The emerging area of semantic web promises a developmental potential that is, as of now, almost entirely untapped. The combined adaptation of these technologies in the context of this research work has shown the effectiveness of the presented approach to bridge the knowledge gap between BIMs (Building Information Models), product information resources and AEC tools and services. The proposed approach is specifically promotes the building energy efficiency use cases by providing a robust and uniform environment that links the material information to building simulation tools and do energy benchmarking for alternative materials.

In future the proposed solution of this research work can be extended and improved as follows:

- The Semantic Repository used in the SkyDreamer prototype would be extended to cover a more representative set of building components. Future research in this area should address the following issues:
    - Extending the base building ontology and instantiation of the base classes

    - Development of the semantic library to contain all building products and components to provide an effective searchable building products library.
    - Providing an easy to use mechanism for manipulation of product library and also the automatic update of product library (for producers)
    - Providing appropriate Web Services to query the product repository
    - Presenting the product data via a multi-lingual portal
- The current energy simulation web service is implemented based on a simple method. In future more elaborated AEC tools and services can be communicated to cover a brighter range of building construction use cases.
- Likewise, it would be interesting to explore in more detail how powerful analysis and evaluation tools can be offered as web services to semantically enrich building models.

Appendixes

Appendix 1: Part of the “IFCXML to OWL” XSL style sheet

```
<?xml version="1.0"?>
<xsl:stylesheet
       xmlns:ifc="http://www.iai-international.org/ifcXML2/RC2/IFC2X2_FINAL"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xlink="http://www.w3.org/1999/xlink"
       xmlns:ex="urn:iso.org:standard:10303:part(28):version(2):xmlschema:common"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:j.0="http://sample.org#"
       version="1.0">

<xsl:output method="xml" indent="yes"
            media-type="text/plain" encoding="ISO-8859-1"/>

<xsl:template match="ex:iso_10303_28">
       <xsl:apply-templates />
</xsl:template>

<xsl:template match="ex:uos">
       <rdf:RDF><xsl:apply-templates /></rdf:RDF>
</xsl:template>

<xsl:template match="ifc:IfcOrganization">
<j.0:IfcOrganization>
       <xsl:attribute name="rdf:ID"><xsl:value-of select="@id"/></xsl:attribute>

    <j.0:IfcOrganization_Id>
      <j.0:IfcIdentifier>
       <xsl:attribute name="rdf:ID">
           IfcOrganization_<xsl:value-of select="ifc:Id"/>
        </xsl:attribute>
      </j.0:IfcIdentifier>
    </j.0:IfcOrganization_Id>

    <j.0:IfcOrganization_Name>
      <j.0:IfcLabel>
       <xsl:attribute name="rdf:ID">
           IfcOrganization_<xsl:value-of select="@id"/>
        </xsl:attribute>
              <rdfs:comment
                rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                     <xsl:value-of select="xsl:Name"/>
              </rdfs:comment>
      </j.0:IfcLabel>

    </j.0:IfcOrganization_Name>

    <j.0:IfcOrganization_Description>
      <j.0:IfcText>
       <xsl:attribute name="rdf:resource">
           IfcOrganization_<xsl:value-of select="ifc:Description"/>
       </xsl:attribute>
      </j.0:IfcText>
    </j.0:IfcOrganization_Description>

</j.0:IfcOrganization>
</xsl:template>
```

```
<xsl:template match="ifc:IfcApplication">
<j.0:IfcApplication>
       <xsl:attribute name="rdf:ID"><xsl:value-of select="@id"/></xsl:attribute>

    <j.0:IfcApplication_ApplicationDeveloper>
       <xsl:attribute name="rdf:resource">#<xsl:value-of
select="ifc:ApplicationDeveloper/@ref"/></xsl:attribute>
    </j.0:IfcApplication_ApplicationDeveloper>

    <j.0:IfcApplication_Version>
      <j.0:IfcLabel>
       <xsl:attribute name="rdf:ID">
          Version_<xsl:value-of select="@id"/>
       </xsl:attribute>
       <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
              <xsl:value-of select="ifc:Version"/>
       </rdfs:comment>

      </j.0:IfcLabel>
    </j.0:IfcApplication_Version>

    <j.0:IfcApplication_ApplicationFullName>
       <j.0:IfcLabel>
       <xsl:attribute name="rdf:ID">
          ApplicationFullName_<xsl:value-of select="@id"/>
       </xsl:attribute>
       <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
              <xsl:value-of select="ifc:Description"/>
       </rdfs:comment>
      </j.0:IfcLabel>
    </j.0:IfcApplication_ApplicationFullName>

    <j.0:IfcApplication_ApplicationIdentifier>
       <j.0:IfcIdentifier>
       <xsl:attribute name="rdf:ID">
            <xsl:value-of select="ifc:ApplicationIdentifier"/>
       </xsl:attribute>
      </j.0:IfcIdentifier>
    </j.0:IfcApplication_ApplicationIdentifier>

</j.0:IfcApplication>
</xsl:template>
.
.
.

</xsl:stylesheet>
```

Appendix 2: IFCXML Model Parser

```
package org.ferial.model;

import java.io.FileInputStream;
import java.io.FileNotFoundException;

import java.util.HashMap;
import java.util.Iterator;
import java.util.List;

import org.dom4j.Attribute;
import org.dom4j.Document;
import org.dom4j.DocumentException;
import org.dom4j.Element;
import org.dom4j.Namespace;
import org.dom4j.Node;
import org.dom4j.QName;
import org.dom4j.io.SAXReader;

public class IfcXMLParser {
    private Document doc = null;

    private static double lengthCoefficient = 1;
    private static double areaCoefficient;

    private static final String ROOT = "//iso_10303_28/uos/";

    private double skylightArea = -1;
    private double floorArea = -1;

    private HashMap<String, Window> skylights = new HashMap<String, Window>();
    private HashMap<String, Space> ifcSpaces = new HashMap<String, Space>();
    private HashMap<String, Zone> ifcZones = new HashMap<String, Zone>();


    /**
     * @param fileName
     * @throws FileNotFoundException
     */
    public IfcXMLParser(String fileName)  {

      SAXReader reader = new SAXReader();

      try {
        doc = reader.read(new FileInputStream(fileName));
      } catch (DocumentException e) {
        e.printStackTrace();
      } catch (FileNotFoundException e) {
        e.printStackTrace();
      }

      removeNamespaces(doc.getRootElement());
      setLengthCoefficient();
      setAreaCoefficient();
      parseSpaces();
      attachSpaceWindows();
      parseZone();
    }

    /**
     * removes name space for ifcXML model to ease XPATH queries
     * @param elem
     */
    private void removeNamespaces(Element elem) {
      setNamespaces(elem, Namespace.NO_NAMESPACE);
    }

    private void setNamespaces(Element elem, Namespace ns) {
      setNamespace(elem, ns);
      setNamespaces(elem.content(), ns);
    }

    private void setNamespace(Element elem, Namespace ns) {
      elem.setQName(QName.get(elem.getName(), ns, elem.getQualifiedName()));
    }

    private void setNamespaces(List l, Namespace ns) {
      Node n = null;
      for (int i = 0; i < l.size(); i++) {
        n = (Node) l.get(i);
        if (n.getNodeType() == Node.ATTRIBUTE_NODE)
          ((Attribute) n).setNamespace(ns);
          if (n.getNodeType() == Node.ELEMENT_NODE)
           setNamespaces((Element) n, ns);
      }
    }

    /**
     * parse all zones in the model
     */
    private void parseZone(){
      List<Element> zones = doc.selectNodes(ROOT+"IfcZone");
      for (Iterator<Element> iter = zones.iterator(); iter.hasNext();) {
        Element zone = iter.next();
        String zoneID = zone.attributeValue("id");
        String name = zone.element("Name").getText();
        Zone ifcZone = new Zone(zoneID, name);
        ifcZones.put(zoneID,ifcZone);
      }
      addZoneSpaces();
    }

    /**
     * parse all spaces in a zone
     */
    private void addZoneSpaces(){
      List<Element> elms =
         doc.selectNodes(ROOT+"IfcRelAssignsToGroup[RelatingGroup/IfcZone]");

      for (Iterator<Element> iter0 = elms.iterator(); iter0.hasNext();) {
        Element elm = iter0.next();
        Element zone = elm.element("RelatingGroup").element("IfcZone");
        String zoneID = zone.attributeValue("ref");
        List<Element> spaceList = elm.element("RelatedObjects").elements();

        for (Iterator<Element> iter1 = spaceList.iterator(); iter1.hasNext();) {
          Element space = iter1.next();
          String spaceID = space.attributeValue("ref");
          ifcZones.get(zoneID).addSpace(spaceID);

        }
      }
    }

    private void parseSpaces(){
      List<Element> spaces = doc.selectNodes(ROOT+"IfcSpace");

      for (Iterator<Element> iter = spaces.iterator(); iter.hasNext();) {
        Element space = iter.next();
        String spaceID = space.attributeValue("id");
        String name = space.element("Name").getText();
        ifcSpaces.put(spaceID, new Space(spaceID,name));
      }
    }

    /**
     * parse all windows in a space
     */
    private void attachSpaceWindows(){
      findSkylights();
      List<Element> boundries = doc.selectNodes(ROOT+
          "IfcRelSpaceBoundary[RelatedBuildingElement/IfcWindow]");
      for (Iterator<Element> iter = boundries.iterator(); iter.hasNext();) {
        Element elm = iter.next();
        String boundryID = elm.attributeValue("id");
        Element space = elm.element("RelatingSpace").element("IfcSpace");
        Element window = elm.element("RelatedBuildingElement").
          element("IfcWindow");

        if ( skylights.containsKey(window.attributeValue("ref")))
          ifcSpaces.get(space.attributeValue("ref")).
             addWindow(skylights.get(window.attributeValue("ref")));
      }
    }

    private double calculateFloorArea(String spaceID) {
      Element elm = (Element) doc.selectSingleNode(
        ROOT+"IfcRelDefinesByProperties[RelatedObjects/IfcSpace/@ref='"+
        spaceID + "']");

      String id = elm.element("RelatingPropertyDefinition").
        element("IfcElementQuantity").attributeValue("ref");
      Element elm1 = (Element) doc.selectSingleNode(
        ROOT+"IfcElementQuantity[@id= '"+ id + "']");
      String id1 = elm1.element("Quantities").element("IfcQuantityArea").
        attributeValue("ref");

      Element elm2 = (Element) doc.selectSingleNode(
        ROOT+"IfcQuantityArea[@id= '"+ id1 + "']");
```

```
          Element sub2 = elm2.element("AreaValue");

          return  Double.parseDouble(sub2.getText()) * areaCoefficient;
        }

        public void findSkylights(){
          List list = doc.selectNodes(ROOT+"IfcWindow");
          for (Iterator<Element> iter = list.iterator(); iter.hasNext();) {
            Element win = iter.next();
            String id = win.attributeValue("id");

            Element elm = (Element) doc.selectSingleNode(ROOT +
              "IfcRelFillsElement[RelatedBuildingElement/IfcWindow/@ref = '"+
              id + "']");

            if (null == elm) {
              String name = win.element("Name").getStringValue();

              Element sub1 = win.element("OverallHeight");
              double height = Double.parseDouble(sub1.getStringValue()) *
                lengthCoefficient;

              Element sub2 = win.element("OverallWidth");
              double width = Double.parseDouble(sub2.getStringValue()) *
                lengthCoefficient;

              Window skylight = new Window(id, name, height, width);
              skylights.put(id, skylight);
            }
          }
        }

        /**
         * set length unit conversion coefficient
         */
        private void setLengthCoefficient() {
          lengthCoefficient = 1;
          Element elm = (Element) doc.selectSingleNode(
            ROOT+"IfcSIUnit[UnitType ='lengthunit']");

          Element sub = elm.element("Prefix");
          String id = elm.attributeValue("id");

          // it is millimeter
          if (null != sub) {
            lengthCoefficient = 0.001;
          }

          Element elm1 = (Element) doc.selectSingleNode(
            ROOT+"IfcMeasureWithUnit[UnitComponent/IfcSIUnit/@ref = '"+ id + "']");

          if (null != elm1) {
            lengthCoefficient = 0.0254;
          }
        }

        /**
         * set area unit conversion coefficient
         */
        private void setAreaCoefficient() {
          areaCoefficient = 1;
          Element elm = (Element) doc.selectSingleNode(
            ROOT+"IfcSIUnit[UnitType ='areaunit']");
          String id = elm.attributeValue("id");

          // it is SQUARE_FOOT
          Element elm1 = (Element) doc.selectSingleNode(
            ROOT+"IfcMeasureWithUnit[UnitComponent/IfcSIUnit/@ref = '"+ id + "']");

          if (null != elm1) {
            areaCoefficient = 0.09290304;
          }

        }
        /**
         * @return skylight to floor ratio
         */
        public double getSkylightToFloorRatio() {
          return (skylightArea / floorArea);
        }

        /*
         * return the building structure as a JSON tree
         */
        public String getStructureTree() {
          StringBuffer sb = new StringBuffer();
          sb.append("{\"areaCoefficient\":" +
                    areaCoefficient+ ",\"lenghtCoefficient\":" +
                    lengthCoefficient+ ",\"zones\":[");

          for (Iterator<String> iter0 = ifcZones.keySet().iterator();
               iter0.hasNext();) {
            String key = iter0.next();
            Zone z = ifcZones.get(key);

            sb.append(z.toStrig());

            for (Iterator<String> iter1 = z.getSpaceIterator(); iter1.hasNext();) {
              String st1 = iter1.next();
              sb.append("{\"area\":"+
                   calculateFloorArea(ifcSpaces.get(st1).getSpaceID()) +",");
              sb.append(ifcSpaces.get(st1).toStrig()+",");
              ifcSpaces.remove(st1);
            }
            sb.deleteCharAt(sb.length()-1);
            sb.append("]},");
          }

          if(!ifcSpaces.isEmpty()){
            sb.append("{\"ID\":\"i00000\",\"name\":\"Default Zone\",\"spaces\":[");
            for(Iterator<String> iter0 = ifcSpaces.keySet().iterator();
                 iter0.hasNext();) {
              String key = iter0.next();
              Space s = ifcSpaces.get(key);
              sb.append("{\"area\":"+ calculateFloorArea(s.getSpaceID()) +",");
              sb.append(s.toStrig()+",");
            }
            sb.deleteCharAt(sb.length()-1);

          }else
            sb.deleteCharAt(sb.length()-1);
            sb.append("]}");
            return sb.toString();
          }

        /*
         * Testing the parser locally
         */
        public static void main(String[] args) {
          IfcXMLParser parser = null;
          parser = new IfcXMLParser("test_building.ifcxml");
          System.out.println(""+parser.getStructureTree());
        }
}
```

Abbreviations:

AEC: Architecture, Engineering, and Construction

AIC: Application Integrated Constructs

Ajax: Asynchronous JavaScript and XML

AM: Application Modules

AP: Application Protocols

BIM: Building Information Model

BSI: British Standards Institution

CAD / CAM: Computer-Aided Design / Computer-Aided Manufacturing

CPD: Certified Product Directory

FM: Facilities Management

GUID: Globally Unique ID

GWT: Google Web Toolkit

HTML: Hypertext Mark-up Language

HVAC: Heating, Ventilating and Air Conditioning

IAI: International Alliance for Interoperability

IDM: Information Delivery Manual

IFC: Industry Foundation Classes

IGES: Initial Graphic Exchange Specification

IR: Integrated Resources

ISO: International Organization for Standardization

IT: Information Technology

JSON: JavaScript Object Notation

MEP: Mechanical / Electrical / Plumbing

MVD: Model View Definition

NFRC: National Fenestration Rating Council

OWL: Web Ontology Language

PMO: Product Modelling Ontology

RDF: Resource Description Framework

RIA: Rich Internet Application

SFR: Skylight to Floor Ratio

SHGC: Solar Heat Gain Coefficient

SOA: Service Oriented Architecture

SPARQL: Simple Protocol and RDF Query Language

STEP: Standard for the Exchange of Product Model Data

SWOP: Semantic Web-based Open Engineering platform

TMY: Typical Meteorological Year

UML: Unified Modeling Language

URI: Uniform Resource Identifier

VBA: Visual Basic for Applications

VDAFS: Verband der Automobilindustrie – Flächenschnittstelle

VT: Visual Transmittance

W3C: World Wide Web Consortium

WSDL: Web Service Description Language

WWW: World Wide Web

XML: eXtensible Mark-up Language

XSL: eXtensible Stylesheet Language

Bibliography

AECBytes (2008). AECBytes newsletter. http://www.aecbytes.com/newsletter/issue_8.htm, last visited December 2008.

Architecture 2030 (2008). Material selection and embodied energy. http://www.architecture2030.org/regional_solutions/materials.html, last visited December 2008.

ASHRAE Handbook (1993). Fundamentals, American Society of Heating, Refrigerating, and Air Conditioning Engineers, Inc., Atlanta, GA.

Augenbroe, G. (2001). Building simulation trends going into the new millennium. Proceedings of the seventh international IBPSA conference, Rio de Janeiro, Brazil, pp. 15 – 28.

Bell, H., Bjørkhaug, L., Rønning, J. (2004). ICT-Platform for Object Oriented Knowledge in the Building and Construction Industry. B4E Conference Maastricht, the Netherlands.

Bell, H., Bjørkhaug, L. (2006). A buildingSMART ontology. Proceedings of the European Conference on Product and Process Modeling, ECPPM-2006, Valencia, Spain, pp. 185-190.

Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. Scientific American, May, 284(5):34-43, http://www.sciam.com/article.cfm?id=the-semantic-web&print=true.

Berners-Lee, T. (2006). Artificial Intelligence and the Semantic Web. http://www.w3.org/2006/Talks/0718-aaai-tbl/, AAAI, 18 July 2006.

Berners-Lee, T., Gödel, K., Turing, A. (2006). Thinking on the Web. ISBN-13: 978-0471768142, Publisher: Wiley-Interscience (September).

Böhms, H.M., Bonsma, P., Bourdeau, M., Josefiak, F. (2008). Semantic Product modelling with SWOP's PMO., ECPPM 2008, eWork and eBusiness in Architecture, Engineering and Construction, pp. 95-104.

Brunnermeier, S. B., Martin, S. A. (1999). Interoperability Cost Analysis of the U.S. Automotive Supply Chain. Research Triangle Institute Project Number 7007-03.

BSA (2008). BuildingSMART Alliance. http://www.buildingsmartalliance.org/, last visited December 2008.

CPD (2008). Certified Products Directory. http://cpd.nfrc.org/login.asp, last visited.

DL (2008). Description Logic. http://en.wikipedia.org/wiki/Description_logic, last visited December 2008.

Design skylights (2008). Design skylights with suspended ceilings. http://www.energydesignresources.com/docs/db-04-skysuspceil.pdf, last visited December 2008.

dom4j (2008). Dom4j XML library. http://www.dom4j.org/, last visited December 2008.

Dublin Core (2008). Dublin Core Metadata Initiative. http://dublincore.org/documents/usageguide/, last visited december 2008.

e-COGNOS (2008). e-COGNOS project - IST-2000-28671. http://e-cognos.cstb.fr/, last visited December 2008.

Eastman, C., Teicholz, P., Sacks, R. and Liston, K. (2008). BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors. John Wiley & Sons, Inc., New Jersey.

eConstruct (2008). Final edition of the bcXML Specifications. http://www.econstruct.org/6-public/bcxml_cd/publicdeliverables/d103_v2.pdf , last visited December 2008.

Egan, J. (1998). Rethinking Construction, Department of the Environment. London, UK.

Express-IFC (2008). The EXPRESS Definition Language for IFC Development. http://www.iai-international.org/Model/documentation/The_EXPRESS_ Definition_Language_for_IFC_Development.pdf, last visited December 2008.

FOL (2008). First-Order logic. http://en.wikipedia.org/wiki/First-order_logic , last visited December 2008.

Fowler, J. (2008). STEP for Data Management, Exchange and Sharing. http://www.pdtsolutions.co.uk/book/STEPbook.pdf, last visited December 2008.

Grant, R., Ceton, G. (2008). OmniClass and IFD Library. http://www.omniclass.org/ CSI_OmniClass-IFD_2008.pdf, last visited December 2008.

Gruber, T.R. (1993). Towards principles for the design of ontologies used for knowledge sharing. In N. Gaurino and R. Poli, editors, Formal Ontology in conceptual analysis and knowledge representation, Deventer, the Netherlands.

Gu, T., Wang, X. H., Pung, H. K., Zhang, D. Q. (2004). An Ontology-based Context Model in Intelligent Environments. Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference.

GWT (2008). Google Web Toolkit. http://code.google.com/webtoolkit/, last visited December 2008.

Halfawy, M.R., Froese, T. (2002). Modeling and implementation of smart AEC objects: an IFC perspective, Proceedings of the international council for research and innovation in building and construction. Aarhus School of Architecture, Aarhus, Denmark, vol. 1, pp. 45–52.

Harrison, D., Donn, M., Skates, H. (2003). Applying web services within the AEC industry: enabling semantic searching and information exchange through the digital linking of the knowledge base. Proceedings of the CIB W78's 20th International Conference on Construction IT, Construction IT Bridging the Distance, CIB Report 284, ISBN 0-908689-71-3, New Zealand, 23-25 April, pp. 145-153. http://itc.scix.net/cgi-bin/works/Show?w78-2003-145.

Hietanen, J. (2003). The Interoperability Pyramid. http://www.iai-international.org/ software/MVD_060424/IAI_ IFCModelViewDefinitionFormat.pdf, last visited December 2008.

Hietanen, J. (2006). IFC Model View Definition Format, Version 1.0, International Alliance for Interoperability.

IAI (2008). International Alliance for Interoperability. http://www.iai-international.org/, last visited December 2008.

IES (1993). IES Lighting Handbook, 8th Edition, IES of North America, 1993.

IFC (2006). Industry Foundation classes, Release 2x3, International Alliance for Interoperability. http://www.iai-international.org/Model/R2x3_final/index.htm.

IFC (2008). Industry Foundation Classes. http://www.iai-tech.org/, last visited December 2008.

IFC Certified (2008). IFC Certified Software. http://www.ifcwiki.org/index.php/ IFC_Certified_Software, last visited December 2008.

IFC Certification (2008). IFC Certification Logos. http://www.iai.hm.edu/ how-to-implement-ifc/certification, last visited December 2008.

IFC Guide (2000). IFC Technical Guide. International Alliance for Interoperability, October 2000.

IFC model (2008). Industry Foundation Classes release 2x technical guide. http://www.iai-nternational.org/Model/documentation/IFC_2x_Technical _Guide.pdf.

IFC Property Set (2008). http://www.iai-international.org/Model/documentation/R151/Online_Documents/documents/IfcPropertyTypeResource-151.html, last visited December 2008.

IFC tools (2008). IFC tools for developers. http://www.iai-international.org/software/Tools%20for%20IFC%20developers.html, last visited December 2008.

IFC2x3 (2008). IFC/ifcXML Specifications, http://www.iai-international.org/Model/IFC(ifcXML)Specs.html, last visited December 2008.

IFC2x4 (2008). IFC/ifcXML Specifications. http://www.iai-international.org/Model/IFC(ifcXML)Specs.html, last visited December 2008.

IFCWIKI (2008). Basic Information. http://www.ifcwiki.org/index.php/Basic_Informations, last visited December 2008.

IFD (2008). International Framework for Dictionaries. http://dev.ifd-library.org/index.php/Ifd:IFD_in_a_Nutshell.

IGES (2008). Initial Graphic Exchange Specification in the USA. US Product Data Association (USPRO), IGES 5.3 (ANSI-1996). http://www.uspro.org/documents/IGES5-3_forDownload.pdf

Isermeyer, U. (2008). Datenaustausch auf neuem Niveau. http://www.ingware.ch/download/IFC2-AxisVM8.pdf, last visited December 2008.

ISO 10303-1 (1994). Industrial automation systems and integration Product data representation and exchange - Overview and Fundamental Principles, International Standard. ISO TC184/SC4.

ISO 10303-28 (1994), Industrial automation systems and integration, Product data representation and exchange, Part 28: Implementation methods: XML representations of EXPRESS schema and data.

Jansen, P., Wix, J. (2003). Harmonization IFC and IFD. IAI industry day, Washington, 14 May.

Jena (2008). Jena Framework. http://jena.sourceforge.net/, last visited December 2008.

Joseki (2008). Joseki WebAPI. http://www.joseki.org/, last visited December 2008.

Junge, R. (2008). Semantic Product Modeling. As presented in http://www.ingware.ch/download/IFC2-AxisVM8.pdf, last visited December 2008.

JUnit (2008). JUnit testing framework. http://www.junit.org , last visited December 2008.

Lima, C., Fies, B., Zarli, A., Bourdeau, M., Wetherill, M. & Rezgui, Y. (2002). Towards an IFC-Enabled Ontology for the Building and Construction Industry: The e-COGNOS Approach'. Proceedings of the European Conference on Information and Communication Technology Advances and Innovation in the Knowledge Society eSM@RT 2002 in collaboration with CISEMIC 2002 Conference, Salford (UK), vol. 1, pp 254-264.

Lixto Solutions (2008). http://www.lixto.com, last visited December 2008.

Lixto Trasformation Server (2008). http://www.lixto.com/lixto_transformation_server, last visited December 2008.

Kemmerer, S. J. (1999). STEP, the Grand Experience, NIST Special Publication 939. National Institute of Standards and Technology, Gaithersburg, MD.

Kieran, S. and Timberlake, J. (2003). Refabricating Architecture. How manufacturing methodologies are poised to transform building construction. McGraw-Hill, ISBN-13: 978-0071433211.

Mahdavi, A., Suter, G., Häusler, S., Kernstock, S. (2004). An inquiry into building product acquisition und processing. "eWork and eBusiness in Architecture, Engineering and Construction: Proceedings of the 5th ECPPM conference" (Eds: Dkbas, A. – Scherer, R.). A.A. Balkema Publishers. ISBN 04 1535 938 4. pp. 363 – 370.

MVDs (2008). Model View Definitions. http://www.blis-project.org/IAI-MVD, last visited December 2008.

NFRC (2008). National Fenestration Rating Council. http://www.nfrc.org, last visited December 2008.

Ontology (2008). Ontology in information science. http://en.wikipedia.org/wiki/Ontology_(computer_science), last visited December 2008

OWL (2004). Web Ontology Language. http://www.w3.org/2004/OWL, last visited December 2008.

OWL guide (February 2004). Web Ontology Language guide. W3C Recommendation, http://www.w3.org/TR/owl-ref/, last visited December 2008.

Owolabi, A., Anumba, C.J., El-Hamalawi, A. (2003). Architecture for implementing IFC-based online construction product libraries. ITcon Vol. 8, Special Issue IFC - Product models for the AEC arena, pg. 201-218, http://www.itcon.org/2003/15

Protégé (2008). Stanford's Protégé Ontology Editor Tool. http://protege.stanford.edu , last visited December 2008.

RDF (2004). Resource description framework, Concepts and Abstract Syntax. W3C Recommendation. http://www.w3.org/TR/rdf-concepts, last visited December 2008.

RDFa Primer (2008). Bridging the Human and Data Webs. http://www.w3.org/TR/xhtml-rdfa-primer, last visited December 2008.

Rezgui, Y. (2006). Ontology-Centered Knowledge Management Using Information Retrieval Techniques. Journal of Computing in Civil Engineering, Vol. 20, No. 4, July/August 2006, pp. 261-270, (doi 10.1061/(ASCE)0887-3801(2006)20:4(261)).

Schevers, H., Drogemuller, R. (2005). Converting the Industry Foundation Classes to the Web Ontology Language. In Proceedings of the First international Conference on Semantics, Knowledge and Grid. SKG. IEEE Computer Society, Washington, DC, 73. DOI= http://dx.doi.org/10.1109/SKG.2005.59.

Shayeganfar, F., Mahdavi, A., Suter, G., Anjomshoaa, A. (2008). Implementation of an IFD library using semenatic web technologies: A case study. ECPPM 2008 eWork and eBusiness in Architecture, Engineering and Construction, pp. 539 – 544.

SkyCalc (2008). A Microsoft Excel® based skylight design tool. Heschong Mahone Group, Inc. (HMG): http://www.h-m-g.com/projects/skylighting/skycalc.htm or http://www.energydesignresources.com, last visited December 2008.

SkyCalc Guide (2008), SkyCalc user's Guide. http://www.energydesignresources.com/Portals/0/documents/Manuals/sg-6-skycalc.pdf, last visited December 2008.

Skylighting Guidelines (2008). A detailed guide for skylight design. http://www.energydesignresources.com, last visited December 2008.

SPARQL (2008). SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/, last visited December 2008.

STEP (2008). Standard for the Exchange of Product model Data. http://en.wikipedia.org/wiki/STEP_(ISO_10303) , last visited December 2008.

STEP development (2008). http://www.steptools.com/library/standard/step_5.html, last visited December 2008.

Studer, R., Volz, R., Stumme, G., Hotho, A. (2003). Semantic Web - State of the art and future directions. In journal Special Issue on the Semantic Web,Vol. 3, pp. 5-9.

SWOP (2008). Semantic Web-based Open Engineering Platform. http://www.swop-project.eu, last visited December 2008.

Turk, Ž., Dolenc, M., Nabrzyski, J., Katranuschkov, P., Balaton, E. , Balder, R., Hannus, M. (2004). Towards Engineering on the Grid. eWork and eBusiness in Architecture, Engineering and Construction Taylor & Francis Group, London, pp. 179-186.

Van Rees, R. (2003). Clarity in the usage of the terms ontology, taxonomy and classification. Proceedings of CIB W78 conference, Auckland. http://itc.scix.net/data/works/att/w78-2003-432.content.pdf, last visited December 2008.

VDAFS (2008). Verband der Automobilindustrie – Flächenschnittstelle. http://www.alias.com/eng/support/studiotools/documentation/DataTransfer/appendix5.html, last visited December 2008.

Web-Harvest (2008). Web Data Extraction Tool. http://web-harvest.sourceforge.net/, last visited December 2008.

Woestenenk, K. (2002). The LexiCon: structuring semantics. Proceedings of CIB W78 conference on Distributing Knowledge in Building, Aarhus, Denmark, Vol 2, p. 241-247.

WSDL (2008). Web Services Description Language 1.1. http://www.w3.org/TR/wsdl, last visited December 2008.

XML Schema (2008). XML Schema. http://www.w3.org/XML/Schema, last visited December 2008.

XPath (2008). XML Path Language. http://www.w3.org/TR/xpath, last visited December 2008.

Yang, Q. Z., Zhang, Y. (2006). Semantic Interoperability in Building Design: Methods and Tools, Computer-Aided Design. vol. 38(10), pp. 1099-1112, October.

Curriculum Vitae

Personal Information

Name	Ferial Shayeganfar
Address	Kammelweg 10/212, A-1210 Vienna, Austria
E-mail addresses	ferial@ifs.tuwien.ac.at ,
	shayeganfar@gmail.com
Date of birth	16.07.1975

Education

2004 - Present	Doctoral study at the department of Building Physics and Building Ecology, faculty of Architecture, Vienna University of Technology
1994 - 2001	Master of Science from Azad University of Architecture, Tehran

Work Experience

2004 - Now	Institute of Software Technology & Interactive Systems, Vienna University of Technology; Project Assistant
2002 - 2003	Tarhsazan Inc. (Tehran); Architect
2000 - 2002	Hatra Inc. (Tehran); Architect
1995 - 1999	Artabin Consultant Engineers (Tehran); Designer

Publications

Shayeganfar, F., Anjomshoaa, A. (2009). Exploitation of Semantic Building Model in Indoor Navigation Systems, EGU 2009, to be presented in European Geosciences Union General Assembly, April 2009.

Shayeganfar, F., Mahdavi, A., Suter, G., Anjomshoaa, A. (2008). Implementation of an IFD library using semantic web technologies: A case study, ECPPM 2008 eWork and eBusiness in Architecture, Engineering and Construction, pp. 539 – 544.

Shayeganfar, F., Anjomshoaa, A., Tjoa, A. (2008). A Smart Indoor Navigation Solution based on Building Information Model and Google Android, Computers Helping People with Special Needs, Springer, pp. 1050 – 1056.

Nguyen, M., Tjoa, A., Anjomshoaa, A., Shayeganfar, F. (2006). Utilising Web Service Based Business Processes Automation by Semantic Personal Information Management Systems - The SemanticLife Case, 6th International Conference Practical Aspects of Knowledge Management (PAKM2006), pp. 1 – 10.

Tjoa, A., Anjomshoaa, A., Nguyen, M., Shayeganfar, F. (2006). Using Semantic Personal Information Management Systems - The Semantic Life Case, Practical Aspects of Knowledge Management, Springer, pp. 1 – 12.

Tjoa, A., Anjomshoaa, A., Karim, S., Shayeganfar, F. (2006). Exploitation of Semantic Web Technology in ERP Systems, Research and practical issues of enterprise information systems, Springer, pp. 417 – 427.

Tjoa, A., Wagner, R., Andjomshoaa, A., Shayeganfar, F. (2005). Semantic Web: Challenges and New Requirements, DEXA Workshop, IEEE Computer Society Press,Copenhagen, Denmark, pp. 1160 – 1163.