Distributed IT for Integration and Communication of Engineering Information for Collaborative Building Design

IBRAHIM FAHDAH

Thesis Submitted to the University of Nottingham for the Degree of Doctor of Philosophy

March 2008

Abstract

In recent years, the rapid development of new information technologies has significantly impacted on the product development process as strategic means to gain competitive advantage in a global market. In the engineering domain, powerful computer-based tools such as Computer Aided Design systems enable engineers to perform various design tasks and realise product concepts in the early phase of the product development process. However, the increasing complexity of modern products as well as the globalization of product development further necessitate distributed and collaborative design environments. This is where different computer systems and dispersed specialists in similar or different disciplines need to collaboratively be involved in shared design activities. Therefore, the integration and communication of engineering information are two of the most key technical factors in ensuring successful collaboration.

The current application of information technology in supporting collaboration during the design process is limited to either a document-based or a common format-based exchange level. These methods provide relatively simple forms of collaboration compared with desired distributed and collaborative design environments that can deliver more effective ways of collaboration.

The work detailed in this research investigates the advantages of using modern distributed information technologies alongside a suitable framework and a product model to support multi-disciplinary collaborative design. The work also involves exploring other important issues related to real-time collaborative design environments. These are design transaction management, access control, communication, and version management.

The research work employs modern technology and distributed computing to enhance the processes of collaborative building design. The research proposes a framework and a product model to extend the functionalities of stand-alone and single-user design systems to facilitate synchronous collaborative design where distributed designers can work

concurrently on a centralised shared model and carry out all necessary communication and data exchanges electronically. The implemented framework proposes a data transaction management approach that ensures efficient concurrent access to the model data and maintains data consistency. The framework also employs software agents to automatically access and operate on the information exchanged among the collaborators. The proposed product model in this work extends an adopted model to support access right control and version management. The work is implemented in an experimental software as a client-server model. .Net technology is used for implementing the framework and the product model and virtual reality technology is used to allow for intuitive interaction with the system.

The research concludes that the utilisation of the modern distributed technologies can effectively induce change in the design process toward a more collaborative and concurrent design. As demonstrated within this work, these technologies with a suitable system design can meet the main requirements of a real-time collaborative building design system.

Keywords:
Collaborative design, Agent technology, Integration, Information communication, Distributed technology, Concurrent design, Product modelling, Steel structures.

Acknowledgement

This work has been carried out at the School of Civil Engineering, University of Nottingham, UK. I am grateful for the financial support provided by Damascus University, Syria.

First and for most I wish to express my deep gratitude to my supervisor, Dr Walid Tizani, who initiated this research idea and has provided great support throughout the research work. He was more than just a supervisor, he was a friend.

I would like also to extend my thanks to all my family and friends. My deep gratitude and love to my Mother for her love, care, and encouragement.

I would like to express deep gratitude to my dear wife Sawsan. I will never forget the lonely evenings you endured during the last couple of years. Your love, patience, support are undoubtedly a big part of this achievement. Last, but not least I would like to thank my daughter, Sarah, for all her troubles and distractions that she brought to the family but greater than that love and happiness.

Finally, I believe that I have been given the power to carry out this work and for that I shall say thanks.

Table of Contents

List of Figures

List of Tables

Acronyms / Abbreviations

2D	Two Dimensional
3D	Three Dimensional
AEC	Architecture Engineering and Construction
AI	Artificial Intelligence
CAD	Computer Aided Design
CE	Concurrent Engineering
CIMSteel	Computer Integrated Manufacturing of Construction Steelwork
CIS	Construction Integration Standards
DCOM	Distributed Object Component Model
IAI	International Alliance for Interoperability
ICT	Information and Communication Technologies
IDEF0	Integration DEFinition language 0
IFC	Industry Foundation Classes
IGES	Initial Graphics Exchange Specification
IA	intelligent Agent
ISO	International Organisation for Standards
IT	Information Technology
MAS	Multi-Agent system
OOP	Object Oriented Programming
SBM	Single Build Model
STEP	Standard for Exchange of Product Data
VBPM	Virtual Building Product Model
VP	Virtual Prototype
VR	Virtual Reality
VRML	Virtual Reality Modelling Language
WWW	World Wide Web

Chapter 1.

Introduction

1.1. Research Overview

The construction industry is one of the largest economic sectors of any industrialised country. The performance of the construction industry is very important to the governments as well as to those within the industry. In the UK for example, the output of the construction industry in 2005 was £107.01bn (KeyNote, 2006). Unlike many other industries, such as the automotive and aerospace that have been able to achieve significant improvements in productivity and quality over the last few decades, the construction industry has been much slower in its adoption new technologies for process improvement (Egan, 1998).

The current problems of the construction industry are quite well known and documented (NIST, 2004). The construction industry suffers fragmentation and it was not able to combine high quality with productivity, client satisfaction and profitability. Other related problems are lack of industrial standards, unsafe working conditions and insufficient ability to evaluate environmental impacts of building materials, products and production methods (Egan, 1998). The fact that the construction industry is behind other industries in modernising its current method of work by adopting and using modern technologies as a major catalyst for improving, especially the design process, was significant for the problem formulation of the research.

The design process in a construction project is a collaborative process. It is a distributed task across multiple functional perspectives addressing interrelated issues of a single product. The fact that many disciplines are involved in the design process, maintaining consistency among the participants is a real challenge since each participating party has their unique competence and responsibilities. The process is also iterative in nature to specify the 'product' that best meets the client's brief, and that ensures safety during construction and use. It is agreed that design decisions taken at the early design stages have a major bearing on the overall cost and quality of the completed project with knock-on effects on downstream issues spanning all project stages (Ruikar, 2005).

Design changes are usually made in isolation from each other in an over-the-wall manner (Anumba et al., 2000). The "over the wall" approach to building design promotes the linear flow of information from one discipline to another during the design and construction process. It is problematic as it allows information wastage, loss and repetition, and long lead times – changes to the design are made and passed on to the next professional for their updates. This approach leads to fragmentation of design and construction information since data generated in one stage is rarely made available for reuse further along in the process. The lack of collaboration and coordination between disciplines leads to misunderstandings in design intent and rational leading to unwarranted design changes and an increase in design cost and time.

The increasing complexity of modern products and the globalization of product development further necessitate a distributed and collaborative design environment where different computer programs and distributed specialists in similar or different disciplines need to be collaboratively involved in a common design activity (Sun and Aouad, 1999). Therefore, the integration and communication of engineering information are two of the most critical technical factors in ensuring successful collaboration.

Early employment of computing in construction provided support for activities where information was created. Examples of these are the use of CAD-systems for drawing production and spreadsheets for cost calculations. However, CAD tools are oriented towards specification of design solutions in terms of geometric and material properties, while the design intent, functional or performance requirements remain un-captured (Laitinen, 1998). The rapid development of information technologies has greatly impacted the product development process in all industries. During the last few years, new emerging information technologies have increasingly been used to facilitate information management and data transfer during the design process. Good examples of such technologies are computer networking, document management systems, the Internet, database technology and interoperability standards (Laitinen, 1998). The potential of these technologies for data sharing has not been fully explored and exploited in the construction industry.

1.2. Research Motivation

The benefits and the needs of following concurrent and collaborative design practices within a building design environment are nowadays widely recognised (Anumba et al.,

2002). Project teams are encouraged to work together more closely and to exchange project information in a more structured way. Collaborative engineering attempts to advance the design activities by maximising concurrency and collaboration in practice (Evbuomwan and Anumba, 1995). The need for improved collaborative working in the construction industry has been highlighted in a number of government-initiated reports and publications such as Constructing the Team (Latham, 1994), The Technology Foresight Report on Construction (OST, 1995) , the DoE/BT Report (Construction IT—Bridging the Gap) (1995) , Rethinking Construction (Egan, 1998), and the NIST report (Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry) (2004).

Besides the reports from the research centres which highlight the benefits from following concurrent and collaborative design practices, the vendors of building design software (e.g. MircoStation, XSteel, StruCad) started to envisage the need of their applications to provide adequate level of collaboration.

1.3. Engineering Collaborative Design

To motivate and guide this research work, the limitations of current information systems and the requirements of engineering collaborative design need to be identified. Engineering collaborative design is a very wide term that is closely related to several technical and social fields such as engineering design, computer science, human computer interaction, decision-making, behavior, and social sciences. In this research, the term “engineering collaborative design” refers to a design activity where distributed participants in a design

team jointly perform engineering design tasks towards a common goal with the help of computer based tools and computer network such as CAD systems and the Internet.

It is actually a complex task to integrate and communicate engineering information between distributed engineering information systems due to some characteristics, besides information heterogeneity and distribution, that are peculiar to the engineering information and engineering collaborative design environment. Some of these characteristics are (Haoxue, 2004):

- Efficiently managing and manipulating enormous amounts of information having interrelated representations, and long-duration data transactions during the project design.
- Distributed participants need to collaborate during the design process. Some examples of necessary collaborative activities are: real-time interaction between participants, timely exchange of relevant information between different software, capturing changes made by other participants, and access control to shared resources.
- Maintaining consistency between distributed data sources is also a very important issue. Failure on data consistency control may lead to invalid data or even result in a completely incorrect design.

The development of CAD systems has revolved from drawing CADs, wireframe CADs, surface CADs, and solid CADs, i.e. from 2D drafting to 3D solid modeling (Lacourse,

1995). Since the early 1980s, CAD technology has become more economically attractive and evolved into a fully developed tool that is now widely used in industry. Current CAD systems are much more than powerful tools drawing simple geometric shapes by electronic means. They allow engineers to realize and maintain the design intent from concept through manufacturing by means of design, analysis, test, and simulation in a virtual environment. However, the original design philosophy of a CAD system was to support individual work in resolving increased product complexity. Despite much progress, modern CAD systems still have inherent limitations regarding collaborative design. Some typical limitations are (Haoxue, 2004):

- CAD systems are stand-alone and single-user applications. Single-user access to data and associated information (e.g. check-in and check-out mechanisms) means that users have insufficient interaction and collaboration in terms of online CAD model and participants have to coordinate their design activities by other means, e.g. telephone, email, or video conferencing, rather than in their current CAD environment.
- CAD systems use file-based storage and data exchange. This causes some shortcomings in terms of concurrent access to CAD data structure. Besides, if product data is distributed amongst several files, accessing and combining the required part of the file is not an easy task.

Many businesses use Product Data Management (PDM) systems to manage their product information. PDM systems provide mechanisms to manage documentary product data and

relationships between them so that it is easier to access, archive, refer to and reference. They can also manage workflow and work history by keeping track of necessary data and supporting documents and by using the mechanisms that provide revision, approval and release management. However, current PDM systems have limited capabilities to solve many problems raised in engineering collaborative design in terms of integration and communication of engineering data such as CAD models (William Xu and Liu, 2003).

Bearing in mind the above observations, it is concluded that successful engineering collaborative design systems should fulfill the following high level requirements:

- Provide participants with a shared workspace fully capable of performing design tasks at different levels (Saad and Maher, 1995).
- Provide an effective storage mechanism that can represent and manipulate CAD model data having complex representation.
- Provide functionalities enabling geographically distributed designers to communicate and interact with each other in synchronous or asynchronous ways. In other words, designers should be able to propose, explore and test alternative ideas during the design phase (Johannsen et al., 1996;Saad and Maher, 1995). This requirement can be further decomposed into several sub-requirements. These are
    - Coordinate design activities by providing effective transaction management and consistency control between distributed systems.
    - Enable some degree of reversion and history management.
    - Enable some degree of access control to design information.

- Enable participants to be aware of the other peers' design stages via visualisation tools such as 3D CAD model viewer.

1.4. Research Problem

The observed limitations of current information systems and the requirements of engineering collaborative design have led to the following formulated research question of this thesis:

How can modern information technologies be employed in distributed applications to better support multi-disciplinary collaborative building design.

The broad research question can be further decomposed into a few major sub-questions, namely:

- How should huge amount of complex engineering information that are distributed and continuously evolving be efficiently managed and manipulated?
- What is the suitable product model structure that can meet not only the engineering data representation but also support the management of the collaboration process?
- How is collaboration work between distributed participants assisted?

1.5. Research Aim and Objectives

The overall aim of this research is to investigate the feasibility and the new opportunities of utilising distributed information technology alongside a suitable product model to support multi-disciplinary collaborative design.

In order to achieve the aim, the following objectives are set:

1. Review typical design process in the construction industry.

2. Assess the state of the art of collaborative design and the technologies used to support their development.
3. Identify the main functional requirements and specifications of a multi-disciplinary collaborative building design system.
4. Identify the suitable technologies for creating a virtual collaborative building design system.
5. Devise a design collaboration system to support data sharing and collaboration in real time.
6. Evaluate the effectiveness of the proposed system to support better collaboration during the design process.

1.6. Adopted Research Methodologies

Most sciences have their own specific methodologies. The methodology used in a research in general varies from quantitative to qualitative or a combination of both of them. Quantitative methodologies involve developing statistically valid samples and searching for the conclusive proof of a hypothesis, whereas qualitative research involves the use of qualitative data, such as interviews, documents, and participant observation data, to understand and explain social phenomena (Lee et al., 1997).

Qualitative research is best used for depth, rather than breadth, of information, while quantitative surveys are an outstanding medium for gathering a breadth of information regarding ”How many?” or ”How much?”, qualitative research is the best research method

for discovering underlying motivations, feelings, values, attitudes, and perceptions (Garson, 2002).

Quantitative research is the systematic scientific investigation of quantitative properties and phenomena and their relationships. Quantitative research is widely used in both the natural and social sciences, including physics, biology, psychology, sociology, geology, education, and journalism. The objective of quantitative research is to develop and employ mathematical models, theories and hypotheses pertaining to natural phenomena. The process of measurement is central to quantitative research because it provides the fundamental connection between empirical observation and mathematical expression of quantitative relationships (Burns, 2000).

Besides the quantitative and qualitative methods, other methodologies can be used alongside to formulate the overall methodology of a research. As this research work involves using information technologies, the methodologies used in software engineering were also looked at.

A software development process is a structure imposed on the development of a software product. There are several models that describe the activities that take place during the process. Examples are waterfall model, spiral model, model driven development, and rapid application development (Cadle and Yeates, 2004). The software engineering process in general is composed of the following activities:

- Requirements analysis
- Specification
- Software architecture
- Implementation
- Testing
- Documentation
- Training and support
- Maintenance

In the context of this research project, the used methodologies are discussed against each individual objective. The objectives can be categorised in three groups: Preliminary investigation, Development, and Evaluation. The preliminary investigation and evaluation stages are carried out using qualitative methods and the development stage is carried out using software engineering methodology.

1.6.1. Objective 1: Review typical design process in the building industry.

The first research objective focused on exploring current design practice in the building industry. The interest is to gain an understanding of the building design process from the point of view of the building design disciplines. To meet the goals of this objective, a comprehensive review of the relevant literature with information drawn from various resources including research and industry publications, the Internet, and conferences was carried out. It would have been desirable to carry out direct observations, but this was not possible under the time constraints of the research.

1.6.2. Objective 2: Assess state of the art of collaborative design and the technologies used to support their development.

As Objective One aimed to gain a general understanding of the design process as a whole, this objective aimed at assessing the state of the art of collaborative engineering design. The main focus of this review was the technologies and the techniques used to build engineering collaborative design systems .This objective was done by reviewing some similar projects and researches in the area of civil engineering and engineering design in general.

1.6.3. Objective 3: Identify the main functional requirements and specifications of a multi-disciplinary collaborative design system

The main specifications and requirements of a collaborative building design system were drawn from the preliminary investigation gained from the first and second objective. Although this objective would be best done through interviews and direct observation, it is thought that the amount of related publications were enough to highlight and draw the main functional requirements.

1.6.4. Objective 4: Identify the suitable technologies for creating a virtual collaborative building design system.

One of the research interests is to adopt the latest information technologies used to develop distributed systems. To meet the goals of this objective, a comprehensive review of the relevant literature were carried out. The potential focus was their availability, maturity, and scalability. The technologies that are believed to suit the research most were then chosen.

1.6.5. Objective 5: Devise a design collaboration medium, which will allow users to share and collaborate in real time.

This objective was achieved using a software engineering approach. It splits into two stages: system design and system implementation. The first stage involved the development of the system design models. Information drawn together from the main functional requirements and specifications was analysed using qualitative methods to develop a theoretical framework for these models.

The second stage involved the development of a prototype application. The development of the application was an iterative process based on the rapid application development (RAD) methodology.

RAD is an iterative build-execute-modify loop which continues until the user is satisfied with the demonstration of the prototype. The prototype is then used to build the final version of the software through the use of the architecture included in the prototype, as well as the validated set of requirements constructed during the prototyping process (Maner, 1997).

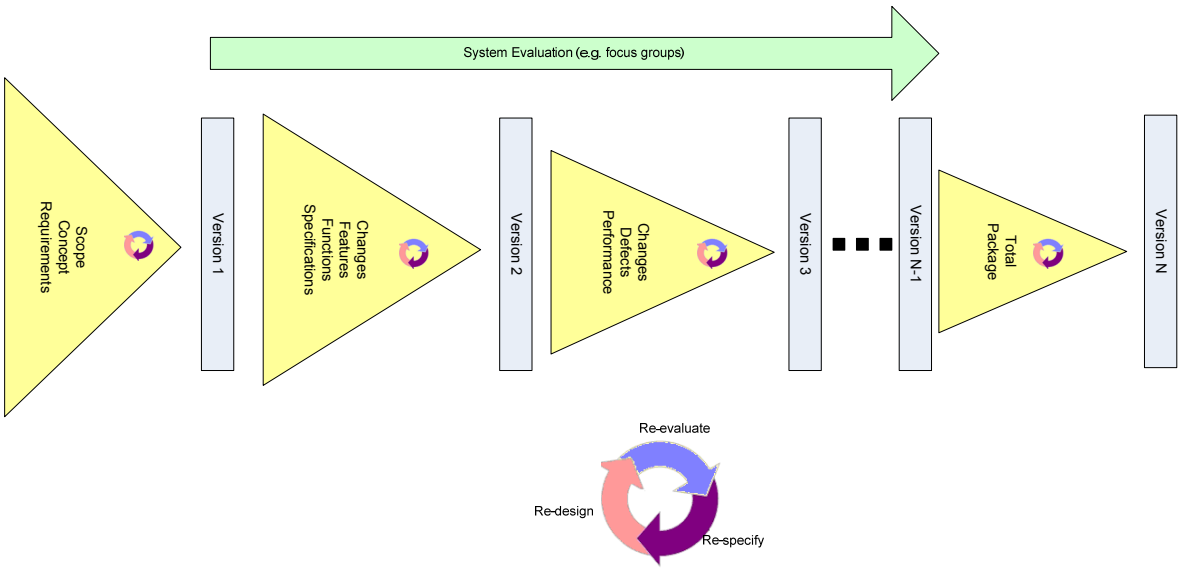


Figure 1-1: Rapid Application Development using Iterative Prototyping (Maner, 1997)

The benefits of the RAD include:

- Increase speed of development through methods including rapid prototyping, virtualization of system related routines.
- The use of computer-aided software engineering tools.
- Allows for an early end-user involvement.

On the other hand, it may reduce scalability and features when a RAD-developed application starts as a prototype and evolves into a finished application. Reduced features occur due to the time frame when features are pushed to later versions in order to finish a release in a short amount of time (Maner, 1997).

1.6.6. Objective 6: Evaluate the effectiveness of the proposed system to support better collaboration during the design process.

The prototype application was evaluated using a number of methods including self-evaluation by the researcher during the development phase and then through the end-user evaluation of the final prototype application. The evaluation goal was to evaluate the prototype system from the industrialist's perspective, with respect to satisfying the system requirements in supporting collaborative design and get some feedback and ideas for further research.

The system was continually tested by the researcher after every milestone in the development process. The tests included system performance, data structure, data processing, and data transaction management. Upon the completion of the prototype application a group of researchers and practising engineers were selected to evaluate the system from the end-user point of view and whether the system meets the main functionalities of a collaborative design system.

Details of the work undertaken for evaluation are included in Section 7.5 and details of the evaluation findings are included in Section 7.6.

1.7. Research Scope

The scope of this research is quite wide, and so it was necessary to reduce the domain of the research in terms of:

- The scope within the design process;

- The scope of the types of information considered;
- The level of detail;
- The scope of implementation

1.7.1. Process Scope

A building project undertakes the following major stages:

- The feasibility study stage;
- The conceptual and detailed design stage;
- The construction stage;
- The facilities management stage.

This research is principally concerned with the process that spans from the end of the feasibility stage to the beginning of the detailed design stage. The design of a building is considered from the client's initial brief up to and including the completed building design.

The beginning of the feasibility stage is not easily formalised, and requires the consideration of a diverse range of factors that cannot easily be structured in a formalised way. The research is limited in being unable to fully consider the detailed design stage, due to the complexity of the system that would require more time and manpower. The research, however, makes every effort to ensure that the proposed system is scalable in such a way that the detailed design stage could be incorporated at a later stage.

1.7.2. Information Scope

The type of building under consideration is limited to multi-storey steel framed structures, though it is anticipated that the general strategies proposed will have further reaching

application across building design as a whole. The research is limited to this type of building in order to provide a focus for the research, and due to the high prevalence and complexity of this type of buildings.

1.7.3. Model Detail

The product model developed in this work is not intended to provide a fully functioning implementation, but instead to provide a framework for the development of a complete system. In order to evaluate the proposed effectiveness of the system, the model places the emphasis on breadth of information considered rather than the depth of detail in any one aspect.

1.7.4. Implementation Scope

The implementation and development of prototype software is that by definition it is an incomplete software system that cannot fully implement the entirety of the product and process model. Instead, it is meant as a proof of concept rather than a complete system. However, the implementation takes into account that the system can be extended to a more comprehensive level.

1.8. Structure of The Thesis

This thesis is composed of eight chapters including the introduction chapter. Below is a brief description of the chapters in order:

- Chapter 1: **Introduction**. This chapter includes the research background and justification, the research question, the research aim and objectives, and the research methodology.

- Chapter 2: **Process of Building Design**. This chapter is an overview of the current work practices in the design process of multi-storey steel framed structures. The chapter highlights some of the bottlenecks and inefficiencies in the current design process.
- Chapter 3: **State of the Art of Collaborative Design**. This chapter presents the current advances in both concurrent and collaborative design systems. It focuses on the information technologies and techniques used in developing collaborative engineering systems. It also presents the current state of using collaborative design systems in the building industry.
- Chapter 4: **Requirements Analysis for a Collaborative Building Design System**. This chapter discusses the key requirements for a collaborative building design environment that are fundamental to allow real-time collaboration work. It also presents various available techniques for implementing them.
- Chapter 5: **A Proposed Collaborative Building Design System.** This chapter describes the design and the implementation of a prototype collaborative design system that meets the requirements defined in Chapter 4.
- Chapter 6: **A Case Study Using the Proposed System**. This chapter demonstrates the application of the developed system discussed in Chapter 5. A case study is used to help presenting the system usefulness. The conceptual and preliminary design process of a typical four storey office building is described in the case study.
- Chapter 7: **Evaluation**. This chapter presents the research findings including the evaluation results of the prototype application. The chapter initially discusses the

objectives of the evaluation process and explains the evaluation methodology, including the development of the evaluation questionnaire. It finally discusses the results of the evaluations.

- Chapter 8: **Conclusion**. This chapter highlights the main research findings and focuses on the future application of these findings. The chapter also states the final conclusions and the main contributions of this research.

1.9. Summary

This chapter laid the foundations for the research work. The following points were presented:

- The research background and justification.
- The formulation of the research problem.
- The research aim and objectives.
- The adopted methodologies.
- The research scope and domain.
- The thesis structure.

Chapter 2.

Process of Building Design

2.1. Introduction

This chapter outlines the current design process. This involves determining the chain of activities undertaken from the project inception and initiation of architect's drawings to the point where these are handed over for construction on-site. It also highlights some general problems in the current design process.

Construction is a complex industry comprising of a wide array of disciplines involved at different stages of the design process. A standard construction project includes several disciplines working together towards a common goal. It is a team effort, which involves several, inter-organisational activities, dialogues and information exchanges. The complexity of the design process may seem to result from the complexity of the building itself but it is rather more due to the complexity of design interrelation between the various disciplines involved in the design process. It is the multi-faceted nature of the construction industry that makes the building design process difficult to manage (Ruikar, 2005). The nature of the construction industry is such that fresh teams are formed for virtually every project and these teams often disperse once the project is completed. All these factors collectively contribute towards the complexity and fragmentation of the design process. Regardless of these factors, construction projects share some common features. According to Allinson (1997) construction projects are one-off, unique, finite, goal-oriented ventures that are undertaken in real-time. The project itself goes through several stages namely:

- The feasibility study stage.
- The conceptual and detailed design stage.
- The construction stage.
- The facilities management stage.

Although the construction process has been often described to be linear in structure, it is in fact a combination of iterative and linear processes, hence highly complex. It is therefore very important to plan and manage the entire process efficiently and effectively.

2.2. The Current Design Process

Building design practice varies considerably across the industry in terms of organisational structure and, due to the particular requirements of each project, the flow of information from concept to realisation. Therefore, it is more appropriate not to base the research on a single idealised design process, but instead to consider the more typical activities that are carried out during building design. The design process is generally well-documented and described in many previous researches. Ruikar (2005) and Smith (2005) have documented the current design process for multi-storey steel framed structures and highlighted the inefficiencies within the process. Since this research scope is also considering multi-storey steel structures, the process activities are summarised below.

In a typical design project work starts with the development of the building concept. This usually involves the architect and the client. The architect prepares the client brief and specifies the total space arrangements that conform to the requirements. The circulation

areas, number and size of cores are detailed by the architect. Conceptual floor layout drawings are prepared and then passed on to the project engineer. These drawings are usually 2D plans and elevations that may be prepared by hand or with 2D CAD software (e.g.AutoCAD). The project engineer prepares several general arrangements that specify the positions of columns, primary beams, and secondary beams in plan. Cost models for different structural grid arrangements are typically calculated using rough cost estimating tools (i.e. spreadsheets). The favourite arrangements are passed back and forth between the project engineer and the architect until a general arrangement is reached that compromises between structural efficiency and building functionality and aesthetics. After the general arrangements have been finalised the building cores or bracing systems are specified to provide lateral stability to the structure. This completes the conceptual design of the building.

After the completion of conceptual design, work progresses to the detailed design stage. The process starts with the development of the architectural working drawings, which are completed using CAD software. Once these are finalised they are passed on to the design engineer. The design engineer starts by formulating the overall framing arrangement on the architectural flooring plans and the structural layout plans are finalised. The appropriate loads are calculated, structural members selected and connections are designed. Analysis and design software, such as QSE and STAAD, is used to carry out this step. Once design is completed, a set of structural calculations, usually in paper form, passed from the design engineer to the project engineer to produce the final General Arrangement (GA's)

drawings. The project engineer then submitted the drawings to the CAD detailer for developing a 3D model of the building using special software (e.g. Xsteel or StruCAD). The main purpose of this 3D model is for visualisation and client benefit and for clash detection checks, to ensure that structural design models and the architectural models match. The current advanced steel detailing software can also be used to electronically issue various drawings such as fabrication and purchasing drawings. After several iterations in which the design is refined and drawings finalised the completed drawings are submitted for costing to the quantity surveyors (QS) and to the fabricator for fabrication. The production department then handles the various aspects of fabrication and transportation to site while the purchasing department handles the procurement of materials for the job. Once the materials transported to site then the erection team will erect the building under the supervision of the project manager. Once it is complete then it is handed over to the client and the project is closed.

The design process has been presented in a relatively linear fashion, though in practice, design conflicts arising at any stage may result in the design being sent back to a prior design stage, and resolved or improved through iteration. Figure 2-1 provides the typical sequence of activity during the design process.

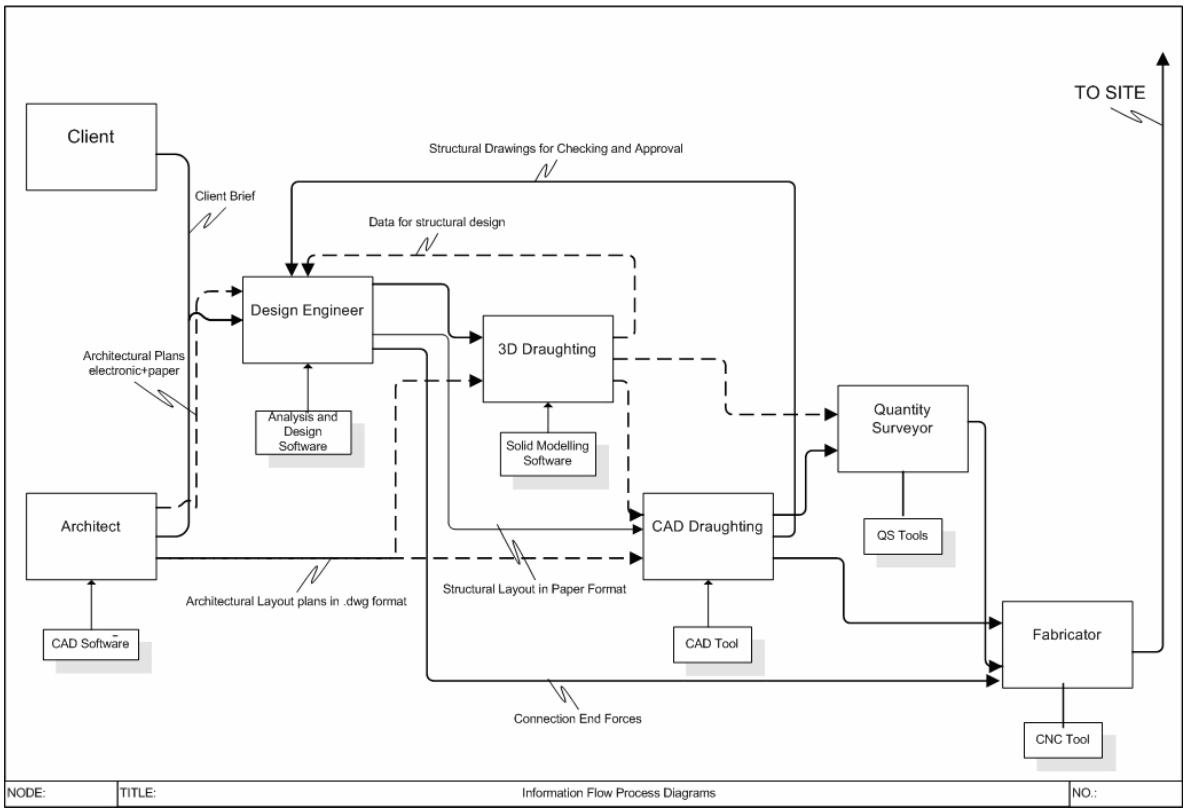


Figure 2-1: Information Flow in the Design Process (Ruikar, 2005)

2.3. Information Technology in the Design Process

The design process has a heavy reliance on IT systems for automation. The amount of data transferred during the design stage is considerable. The different software packages used at each stage of the design process do not necessarily allow for the seamless transfer of data between processes and disciplines. For example, in the design process illustrated in Figure 2-1, the design engineer usually uses an in-house design spreadsheets and analysis software for the structural design process and the 3D detailer uses for the 3D modelling process specialised solid modelling software. The design layout finalised by the design engineer is sent to the 3D detailer in the form of design spreadsheets. This format of information is not compatible with the solid modelling software requiring manual re-entry of building

information to the 3D solid modelling software by the 3D draughting engineer. The 3D draughting process may adjust the structural design of the building to synchronise it with the architectural layout. These alterations in design information are communicated back to the design engineer for re-evaluation. This is done by directly communicating with the design engineer by phone or by sending a memo to the design engineer outlining the appropriate changes. The design engineer interprets this information and makes the appropriate change to the building design for reassessment. This generates an iterative process that involves a high level of re-working of data. Such data flow leads to a lack of integration between different stages in the design processes and hinders collaborative design. This type of information flow creates a fragmented design process that only allows rigid or mostly paper-based information to be transferred across boundaries (Ruikar, 2005).

2.4. Drawbacks in the Current Design Process

The design process is a complex activity and there is a considerable amount of data transfer between each of the design stages. The design team also uses homogeneous software packages as an integral part of the process. The current implementation of software within the process works in a largely fragmented fashion. Each software satisfies the needs of the individual processes and thus limiting real integration within the entire design process. This fragmented nature of the design process is a major barrier to the efficiency and integrity of the process as a whole. The efficiency of the current building design is prevented from reaching its true potential, by the lack of integration within the design process. This is largely due the limitations inherent in the use of non-integrated design systems (Ruikar, 2005). Some general problems associated with the current design process can be as follows.

2.4.1. Poor Communication

Construction is a multi-organization and interactive process. Therefore, successful completion of a project depends on the accuracy, effectiveness and timing of communication and exchange of information and data between the supply chains. Unfortunately, the inefficiency of the existing method of communication has become a barrier to several innovative construction processes developed for the industry over the past four decades.

The transfer of design data between the various partners is currently predominantly based on 2D drawings, which predominantly are exchanged in paper format even though they are increasingly produced using CAD. The exchange of paper documents can be quite slow. Particularly in the early design stages, the other designers participating in the design process (structural, building services, and other specialist consultants) have to wait for the architect's designs and changes before they can proceed with their own work. Munday and Karlen (2004) confirm that managers involved in the construction activities waste almost half of their working time on tasks devoted exclusively to information transmission in terms of processing and management.

2.4.2. Lack of Interoperability in Current Information Exchange

A prerequisite for integrating design and construction management is a conflict-free exchange of design data between the partners and from system-to-system. It is very unlikely that the design team use the same IT applications or have the same requirements for the content of the data to be processed. Unless these applications

belong to an integrated suite of software tools, these applications have little to do with each other – they are "unaware" of each other, often describe essentially the same data in different ways, and do not exchange or share data. This is resulting in an unnecessary generation of duplicate data, and is causing a lot of unnecessary errors and omission, cost and delays. Some statistics (Janis, 2000) show that 50-85% of construction problems are caused by missing or bad information, 10-30% of time spent by facility engineers is searching for information, on average, each cost estimating item is calculated seven times and 20% of design and construction costs are due to waste. The National Institute of Standards and Technology (NIST) report (Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry) in August 2004 mentioned that 40% of engineering time is spent locating and validating data, poor communication between systems waste 30% of project costs and reduction up to 50% in delivery time is possible through improved communication using enabling technologies.

2.4.3. Lack of Decision Support

The client's needs and demands are currently difficult to be specified in sufficient detail neither are they presented in the form of measurable attributes. The designers try to understand the client requirements and form the design solutions based on their own judgement. Figure 2-2 illustrates the discrepancy between the client's initial expectations and the obtained result.

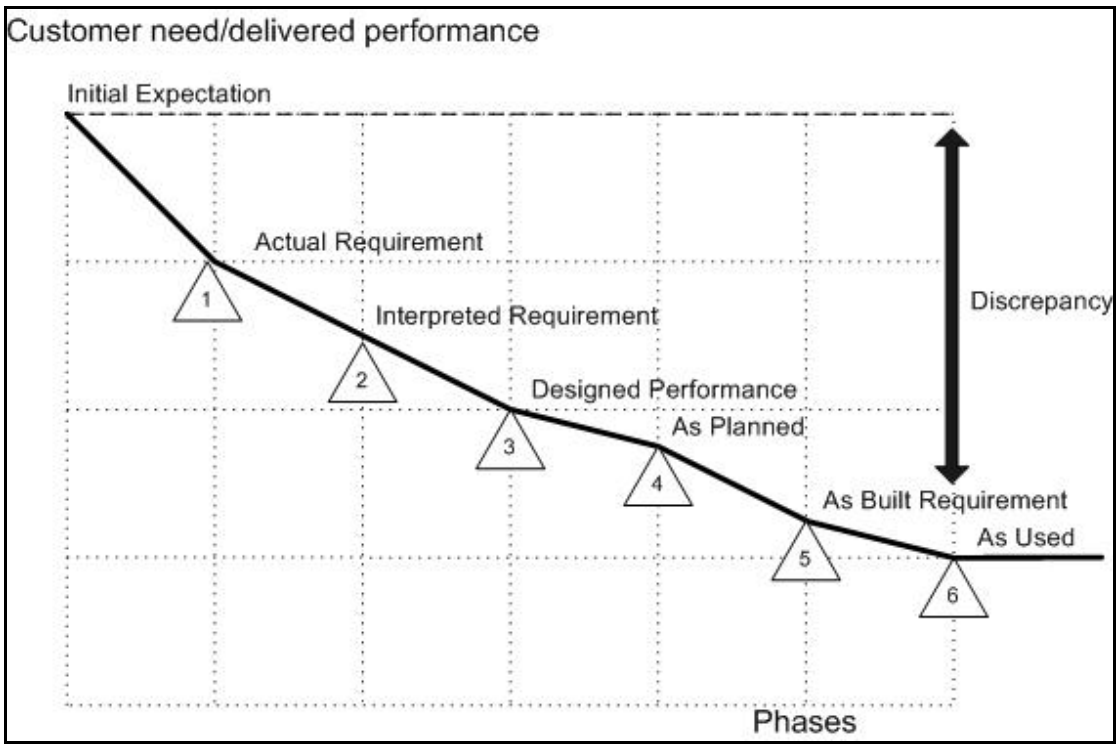


Figure 2-2: Differences between initial client expectations and final results (Laitinen, 1998)

There are various reasons that contribute to these discrepancies. The customers usually find it difficult to express their expectations and they mostly do it verbally and the designers may understand these requirements in a different way than the client. Moving downstream in the design process, the expectations may be increasingly misinterpreted. The contractor on site, for example, may take into account constructability and execution of work in different way to the original intentions.

Most building design decisions are made without testing their effect first. Whatever testing takes place in the design and construction phases is limited to only a few aspects of performance and it is often very costly in both time and money.

It is obvious that tests and comprehensive verification of the product performance are very difficult to attain when each product is essentially a very costly "one of a kind," and when it takes a long and laborious multidisciplinary effort to design and build it. Some industries test their products thoroughly by physical prototypes, virtual prototypes or both ways. It is clear that the building industry will be able to test its product (i.e. buildings) in a comprehensive manner only virtually. It will have to first build virtual buildings, test them against the requirements and make the necessary design decisions and physically construct them only after that.

2.4.4. Lack of Consistent Data Models

There is wide range of software used during the design process. The structural design and analysis, architectural design and fabrication design may be completed using different software packages. These software applications may store data in incompatible data formats. The use of different data formats within the design process makes it difficult to a have a single model representation for the structure that describes the building as a whole. Currently, design information is represented with the help of 2D CAD drawings. These drawings are used to represent the architectural spaces, structural elements and building service layouts. Also as described by Smith et al. (2003), there is a general preference to use large paper diagrams, or CAD printouts during the building design process to match design compatibility and exchange information. However, these AutoCAD or 2D drawing representations of the building can have the following major drawbacks:

- They provide static information that is mainly geometric in nature.

- It can be difficult to translate into anything intelligent such as reusable data that can be used by the different parts of the design process and transferred over the existing data boundaries that have been described earlier.
- The generic 2D drawing representations do not hold any type of process information e.g. structural analysis.

There have been many researches to develop standard data formats that describe the whole lifecycle of the building such as IFC (IFC, 2001). The main objective of these attempts is to provide a neutral interface between homogenous software used by design team.

Although such standard data formats would help reduce data transfer time and minimise human errors that occur during the re-input process, their usage in current practice is still limited. That is because most available standards are either uncompleted or incomprehensive and even with the cases when an application can import a whole model from a neutral data file, the quality of the imported models are not as good as if the model is built from scratch since it often requires lots of processing before it is fully satisfactory and that is not due to the standard limitation rather the importing applications.

2.5. Summary

This chapter has described the typical working methods for the design of steel structures. It did so by mapping the typical design process and by highlighting the current inefficiencies within this process. The existing IT tools and software used in the design process are mainly limited to straightforward automation of the design activities aiding humans in

performing them. However, the recent advances in information technology (e.g. networking and communications technology) have yet to offer more innovative ways and increase possibilities for re-engineering and improving the design process.

Chapter 3.

State of the Art in Collaborative Design

3.1. Introduction

The global marketing process in the construction industry is currently continuing its progress through support of advanced Information Technology (IT). The manufacturing industry is already steps ahead of the construction industry in using state of the art computer based technologies to enable their marketing abilities (Egan, 1998). Many engineering products are produced in companies located in different geographical locations around the world. Therefore, the industry is involved in "an attempt to optimise the design of a project and its construction process to achieve reduced process time, and improved quality and cost by the integration of design, production activities, and by maximising concurrency and collaboration in working practices" (Anumba et al., 2002). Building design often requires collaborative working between members of a construction project team. In many cases, due to the geographical distribution of participants, the need for effective information and communication technologies become necessary. An important step to an effective collaborative working process is a common working environment to facilitate the integration and interaction between various disciplines (Ugwu et al., 2001).

The application of information technology in the construction industry is a relatively young field of research. Since the 1970's the technology was used in two different ways for automating activities aiding humans in performing them. It was first applied for straightforward automation. Only after a number of years have enterprises learnt about the

opportunities offered by IT and started to use IT in more innovative ways. The recent developments in networking and communications technology and the miniaturisation of the hardware have also started to offer increasingly possibilities for re-engineering (Bjork, 1999).

CAD (computer-aided design), for example, was first used as an electronic tool for drawing lines and soon after it started to proliferate, since then CAD technology has become more matured and evolved into a fully developed tool capable of compressing product design and manufacturing cycles, reducing design and production costs, and improving quality (Geng, 2004). Modern CAD systems are much more than powerful tools drawing lines by electronic means. They allow engineers to realize and maintain the design intent from concept through manufacturing by means of design, analysis, test, and product simulation in a virtual environment (Bjork, 1999). However, the original design philosophy of a CAD system was to support individual work in resolving increased product complexity. Despite much progress, current CAD systems still have inherent limitations concerning collaborative design. One major limitation is that most CAD systems are stand-alone and single-user applications. Designers only use a couple of hours per day doing very concentrated drawing production work, the rest of the time is occupied with information retrieval, communication with co-workers, etc (Bjork, 1999).

The situation is changing dramatically nowadays. Developments in LAN and WAN networks, the Internet, mobile phones, video-conferencing and other technologies have

extended IT support to a much more comprehensive coverage of the communication and information retrieval activities (Laitinen, 1998). Recently, many research projects have been carried out to provide collaborative and distributed solutions from the perspectives of CAD, PDM, workflow management, engineering and geometric data streaming communication, distributed system infrastructure, etc. (Li and QIU, 2006)

Collaborative applications can generally be design or management centric. CAD and PDM are regarded as two primary pillars to support product design. CAD is a design-centric tool to provide a platform for embodying geometric and engineering design models and drawings, and PDM is a process-centric tool to streamline the information communication and coordination among design departments through storing, managing and exchanging design models and processing information. Since collaboration is nowadays becoming increasingly frequent in global manufacturing industrial landscape, the current common direction of CAD and PDM is to be integrated together to establish an entire product lifecycle management (PLM) (Li and QIU, 2006) (Figure 3-1).

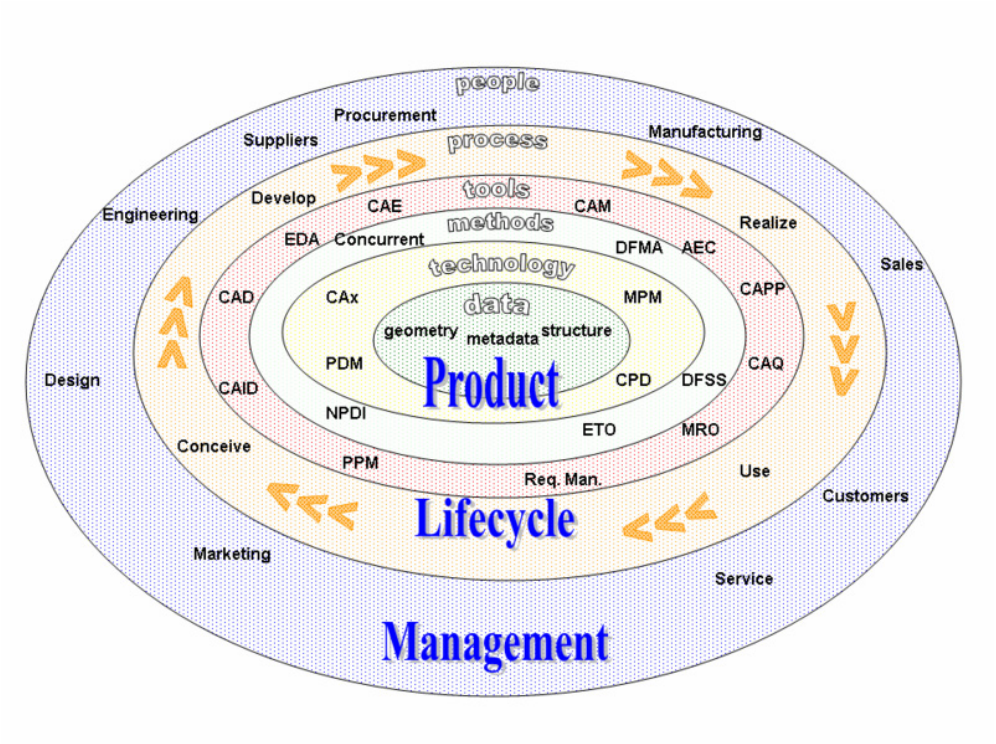


Figure 3-1: PLM concept outline (Wikipedia, 2007)

A collaborative design system development requires two kinds of capabilities: distribution and collaboration. Distribution focuses on building up a backbone infrastructure to connect dispersed design systems whereas collaboration associates and coordinates individual systems to fulfil a common design target systematically. Even though distribution and collaboration having different focuses, the two are closely inter-related and complementary (Li and QIU, 2006).

Since the design team is highly encouraged to work closely in during the project time especially in the early stage of the project, the distribution infrastructure of a collaborative design system attempts to provide concurrency. Concurrent Engineering (CE) refers to a design process where all life cycle stages of a product are considered simultaneously from

the conceptual stage through to the detailed design stage. Concurrent engineering design systems usually support collaborative design or aim to.

This chapter outlines the state of the art of collaborative design through the following points:

- An introduction of the concurrent design concept.
- Introduce the current information technologies that enable concurrency.
- An introduction of the current state of concurrent and collaborative applications in construction industry.

3.2. Definitions of Concurrent Engineering

The concept of Concurrent Engineering (CE) was initially proposed as a means to reduce product development time (Barkan, 1988). Since then, the term CE has been described in different ways with various implications. The following are some common definitions:

- Probably the most common definition of the term CE is that of Winner (1988) . According to Winner (1988), CE is “a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. This approach is intended to cause the developers, from the outset, to consider all elements of product life-cycle from conception through disposal, including quality, cost, schedule, and user requirements”.
- NASA Systems Engineering Handbook SP6105 describes concurrent engineering as "the simultaneous consideration of product and process downstream requirements by multidisciplinary teams."

- The Computer-aided Acquisition and Logistics Support (CALS) office (1999) sees CE as "cost-effective robust design (conception through disposal) and simultaneous design of all downstream processes during upstream phases".

3.3. Concurrent Engineering vs. Sequential Engineering

Traditional engineering design, also known as sequential engineering, is often characterized by poor cross-functional communication. It is done as a sequence of stages and each stage of the development process is carried out separately and the next stage cannot start until the previous stage is completed. It has been described as a brittle and inflexible design method (Ye and Churchill, 2003). It makes the design expensive and time-consuming and it also reduces product quality and limits the number design of alternatives that can be examined (Ye and Churchill, 2003). Therefore, many industries are now adopting CE approach to replace the traditional design approach to a simultaneous design with parallel, less interrelated processes. Different from traditional linear patterns of working, CE improves communication between all personnel involved in the process of managing the entire lifecycle of a product (Backhouse and Brookes, 1996). The essence of concurrent product and process development is an integrated and collaborative process, where different parties cooperate to specify and design products through coordination, communication, and negotiation (Ganesan, 1997).

3.4. Benefits of Concurrent Engineering

There are several benefits that concurrent engineering can bring. Concurrent engineering can benefit companies of any size, large or small. While there are several obstacles to

initially implementing concurrent engineering, these obstacles are minimal when compared to the long term benefits that concurrent engineering offers. Although it is difficult to quantify many of these benefits by using numbers, some studies on many different implementations of CE have consistently indicated the substantial advantages of CE (Cleland and Ireland, 2004):

- Shrinking time-to-market (by 50%).
- Reducing product life cycle cost (by 40%).
- Reducing engineering changes and rework (by 50%).
- Improving productivity and design quality.
- Active customer involvement.

3.5. Basic Principles of Concurrent Engineering

The foundation of CE is laid on eight basic principles: feeling of ownership, common-understanding, early problem discovery, teamwork affinity, work environment structuring, early decision making, constancy of purpose, and knowledge leveraging (Dhillon, 2002).

The feeling of ownership means the team members will work effectively to produce a good product if they are provided with the authority to shape its design as considered appropriate. Mutual understanding means the team members will work better when they aware of what other members are doing. Early problem discovery the problems found in the early design stage are easier to solve than the ones found at a later stage. Teamwork affinity means teams will have a better affinity when there is trust among teams. Work structuring means that from the practical aspect, unlike parallel computers, the human minds cannot

work simultaneously on multiple tasks. However, the human mind is quite good at systematically structuring the work or structuring the work environment so that an individual task can be executed independently by either a human or machine. Early decision making means the window of opportunity to affect a given design is much wider during the early stages of design than later ones. Constancy of purpose means everyone involved contributes his or her best when working toward a common and consistent goal. knowledge leveraging means as the domain of product design is often quite large, inter-linking decision support tools with spurts of human knowledge base is the most viable approach to find solutions to complex problems (Dhillon, 2002).

3.6. Information Technology in CE

Information technology (IT) can be defined as the branch of technology devoted to the study and application of data and the processing thereof. IT can also be thought of as applied computer systems, including both hardware and software, usually in the context of a business or other enterprise, and often including networking and telecommunications (Allen and Morton, 1994).

With the rapid development of IT, its use is becoming more critical to many organizations as a strategic tool to gain competitive advantage in a global market, with benefits of improving productivity and performance, enabling new ways of managing and organizing, and develop new businesses.

Fast advances in computing such as distributed and communication technologies are creating a new approach for product design and manufacturing. It provides the enabling foundation for CE. It provides soft prototyping, visualization, product data management, multimedia, and electronic data exchange, etc. Prasad (1996) chose technology and tools as two of seven aspects that influence the domain of CE.

Saad and Maher (1995) proposed that CE tools must fulfil four high-level requirements. These are:

- Information sharing in which the representation of the design objects are shared using a language that can be understood by all the participants;
- Communication media in which the participants in the collaborative design can communicate their intentions, planning and actions;
- Process management where the participants can determine the stage of the process and what is to be done next;
- Exploration space, in which alternatives can be proposed, tested and changed.

Similar to that, Johannsen et al. (1996) argue that CE requires support in four areas: communication support, cooperation support, coordination support, and IT architectures.

3.6.1. Basic Communication Media in CE

- Electronic mail – Electronic mail (E-mail) provides a facility for people to communicate, and collaborate to some degree with practically anyone, anywhere, and at

any time. It provides a cheap, fast and effective tool for asynchronous communication and exchange data (Mills, 1998).

- Voice mail – Voice mail is another asynchronous communication and collaborative facility within all kinds of organizations as other stored media, such as E-mail (Mills, 1998). Even though it is always considered a secondary means of communication to actual telephone conversation, voice mail may still a need element for integrated CE system.
- Desktop Videoconference – Desktop Videoconference offers a synchronous communication method among people to facilitate collaboration on many complex projects (Anumba et al., 1997). It also allows geographically distributed groups to transfer, co-view, discuss, co-edit data files, and supports application sharing, file transfer, and textual communication which can be saved as a record of the conference (Anumba et al., 1997).
- Virtual Meeting room – Virtual meeting room (VMP) represents an extension of the concept of desktop videoconferencing. In a virtual meeting room, team members are able to interact intuitively in three-dimension space and feel as though they were all in the same room (Anumba et al., 1997). They can collaboratively share files with participants, working directly on a file together, and easily send files back and forth during the meeting. VMP is also can be an important tool for collaboration and communication between geographically distributed organizations.
- Mobile Communication System – Mobile Communication System is a system that services voice and data over a wireless communication network (Anumba et al., 1997).

MCS may be ideal to improve communications between the design office and the construction site (Thorpe et al., 1995).

3.6.2. Information Technology for Developing CE Applications

Unlike most desktop or stand-alone applications, where data is locally available and accessed by an individual user, contemporary enterprise applications often involve collaborative work and frequently require access to information distributed across remote locations. In this net-centric vision, so many technologies have emerged to support distributed applications development. These technologies differ in their capabilities and purposes.

Research centres have experimented with different kinds of distributed technologies based on the research interests. Below is a review of the most used technologies.

3.6.2.1. Multi-Agent Technology

Multi-agent systems are an emerging sub-field of artificial intelligence that is concerned with a society of agents interacting in order to solve a common problem. Multi-agent systems are a relatively new field of research. They have only been studied since about 1980, and the field has only gained widespread recognition since about the 1990s (Oliveira et al., 1999;Wooldridge, 2002). Various definitions have been proposed for the term multi-agent system (MAS). Durfee et al.(1989) defined a MAS as a loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver. More recently, Ferber (1999) defined a MAS as a system composed of a population of autonomous agents, which interact with

each other to reach common objectives, while simultaneously each agent pursues individual objectives. Oliveira et al. (1999) defined a MAS as a collection of, possibly heterogeneous, computational entities, having their own problem-solving capabilities and which are able to interact in order to reach an overall goal.

The agent concept is the heart of the technology. Unfortunately, there is no universally accepted definition of the term agent, and indeed there is much ongoing debate and controversy on this subject (Wooldridge, 2002).

- "Most often, when people use the term 'agent' they refer to an entity that functions continuously and autonomously in an environment in which other processes take place and other agents exist."(Shoham, 1993).
- "An agent is an entity that senses its environment and acts upon it." (Russell, 1997).
- "Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program, with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires." (The IBM Agent)(Gilbert and Janca, 1997).
- "Intelligent agents continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions." (Hayes-Roth, 1995) (Figure 3-2).

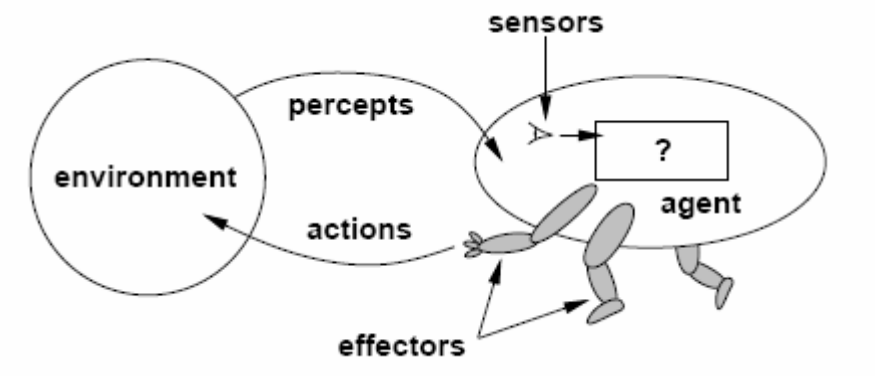


Figure 3-2: Symbolic representation of a software agent

Regardless of the various definitions, the literature identified the key properties that characterise an intelligent agent (Ferber, 1999;Wooldridge, 2002):

- Autonomy: agents operate without the direct intervention of humans or others, but have some kind of control over their actions and internal state using a set of tendencies. Tendencies are individual goals to be achieved by the agent.
- Social ability: agents cooperate, negotiate, and communicate with other agents.
- Reactivity: agents perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives.
- Pro-activity: agents do not simply act in response to their environment; they are able to exhibit goal-directed behaviour by taking the initiative. Agents are capable of handling complex and high-level tasks. The decision as to how such a task is best split up into smaller sub-tasks, and in which order, and way, the sub-tasks are best performed, should be made by the agent itself.
- Temporal continuity: agents are continuously running processes.

- Mobility: an agent has the ability to transport itself from one computer to another, retaining its current state.
- Learning: agents are able to learn and adapt themselves to fit their environment.

Communication between agents is the backbone of any multi-agent system. To support communication processes among agents it is necessary to develop agent exchange information protocols. There are many communication protocols that have been developed to facilitate communication. They are beyond the research interest to be all reviewed. KQML (Finin et al., 1994), as an example, is a known supporting information exchange protocol for agent-based systems. KQML (Knowledge Query and Manipulation Language) is a language and protocol for communication among software agents and knowledge-based systems. It was developed in the early 1990s as part of the ARPA Knowledge Sharing Effort, which was aimed at developing techniques for building large-scale knowledge bases which are shareable and reusable (Shen et al., 2001).

Multi-agent technology has been already applied to engineering design. Research centres have already applied the technology to concurrent engineering, collaborative engineering design and, planning and control systems and other similar domains (Shen et al., 2001). Agent technology is reported to be well-suited for use in applications that involve distributed computation or communication between components (Parunak, 1998). It is widely recognised as a promising paradigm for the future engineering design systems (Shen et al., 2001). Below are some researches that have applied agent technology to collaborative and concurrent engineering design.

- PACT (Cutkosky et al., 1993), Palo Alto Collaborative Testbed, was a leading research for joint experimentation in computer-aided concurrent engineering being pursued by research groups at Stanford University, Lockheed, Hewlett-Packard, and Enterprise Integration Technologies. Its objective was to explore a new methodology for cooperatively solving engineering problems based on knowledge sharing. Its application domain was electro-mechanical design and robotics. The PACT architecture is based on interacting agents (programs that encapsulate engineering tools). The agent interaction in turn relies on shared concepts and terminology for communicating knowledge across disciplines, an Interlingua for transferring knowledge among agents, and a communication and control language that enables agents to request information and services.
- DIDE (Shen and Barthes, 1996), Distributed Intelligent Design Environment, is a multi-agent system interested in large engineering projects such as an automobile, a harbor, or an aircraft, at the University of Technology of Compiégne. The objective was to verify whether it is possible to build a real open system, that is, systems for which users can freely add or remove agents without having to halt or reinitialize the system. In DIDE, agents are autonomous cognitive with deductive, storage and communication capabilities. An agent is composed of: (1) network interface, (2) communication interface, (3) symbolic models of other agents, (4) a model of its own expertise (internal knowledge), (5) a model of task to be performed (local knowledge).

- CAIRO (Pena-Mora et al., 2000), Collaborative Agent Interaction and synchROnization system , is a distributed conferencing architecture for managing designers and engineers in a distributed design meeting. The CAIRO system allows designers and engineers to work together in virtual teams by supporting multi-media interactions over computer networks.
- ADLIB (Anumba et al., 2003) is an Agent-Based Support System for the Collaborative Design of Light Industrial Buildings. It was developed by a research group at Loughborough University. The main objective was to develop a multi-agent system framework for the representation of activities and processes involved in collaborative design of light industrial buildings. The design process that ADLIB agents tried to automate is the interaction and negotiation between specialist design team members. Within the final ADLIB design system the agent owner provides the system with a simple specification of the design, such as external dimensions, maximum cost, target cost, plus any constraints. The automated system then functions much like a human-based collaborative design process. By a process of negotiation, the agents will converge on a design solution that satisfies all of them. All agents try to meet minimum design standards as defined within their internal knowledge bases.
- CLOVER (Zhao et al., 2001), an agent-based approach to systems interoperability in cooperative design systems, it was a cooperative research between institute of manufacturing systems at Beijing University of Aeronautics and Astronautics, China and National Research Council Canada, Canada. It proposed that agent

technology can be used to improve the interoperability among applications, and more importantly, it can support higher level dynamic and autonomous cooperation among applications. The agents in CLOVER are fundamentally classified into four types: process management agents (PMA), design task agents (DTA), tool agents (TA) and product data agents (PDA). Process management agents (PMA) are responsible for managing the design process based on representation of a general design process model. The design process includes requirement analysis, conceptual design, detailed design, and process planning. Design task agents play the design role using their resources (tools) following related rules. Tool agents are responsible to find, provide and manage general tools/applications design task. Product agents are responsible for saving and managing product data. A design task agent is a type of the general agent while both tool agents and product data agents are mainly wrappers to facilitate communications between other agents and the application tools. CLOVER, in its prototype environment, adopted KQML as the agent communication language and XML (eXtensible Markup Language) as formats for the content of communication.

3.6.2.2. Communication Technology

Various kinds of technologies to implement the infrastructure of distributed applications have emerged over time. Below is a summary of most used communication technologies to support distributed application development. The summary will only provide high level definitions of the technologies. This is to avoid any technical issues which are irrelevant to the research focus.

- CORBA. The Common Object Request Broker Architecture is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to interoperate (CORBA, 1997).
- Java /RMI. Java Remote Method Invocation (Java RMI) enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines (Java RMI, 1994).
- COM+ was first introduced the technology as Microsoft Transaction Server on Windows NT Service Pack 4 to provide developers with support for distributed transactions. With Windows 2000, significant extension to the technology was incorporated to the operating system and it was renamed COM+ (COM+, 1993).
- DCOM. Distributed Component Object Model (DCOM) is a Microsoft proprietary technology for software components distributed across several networked computers to communicate with each other. DCOM has been deprecated in favour of Microsoft .NET (DCOM, 1996).
- Microsoft .NET. .NET Framework is the heart of the .NET technology. .NET is a new computing platform that simplifies application development in the highly distributed environments (.NET Framework, 2000). .NET provides very rich access to technologies that enable developers to build distributed applications. These include ASP.NET, .NET Remoting, Network Communication, Data Access, and XML (.NET Framework, 2000). .NET is an integrate platform that helps developers to build complex systems rapidly

and let them focus on the system design rather than its implementation. The main criticism of the technology is that it is tied too closely to the Microsoft Windows operating system.

The aforementioned technologies are used to implement the communication infrastructures of distributed applications. These applications can use various data communication technologies to exchange information. XML (XML, 1998) is an international standard for electronic document exchange, optimized for the Internet. It is short form of Extensible Markup Language. XML is one of the latest technologies used for data exchange over networks. It is a way of exchanging information over the Internet with a tremendous significance for e-business, since it promises to enable different applications to describe the components of business processes and exchange data intelligently. What also makes XML special is that it is text-based, independent of the platform, support hierarchy structure and good at providing a format for messaging and protocols. XML is regarded as the next-generation Internet mark-up language, and has become more popular as a standard exchange format of design and manufacturing data in collaborative environments.

3.6.2.3. Information Modelling Technology

In all stages of the product life cycle, the need for intensive data exchange and information sharing are very important. This can be achieved by data exchange standards providing basis for flexible exchange of product model data during the product period. There are many integration standards that already in use or in the development stage. Below is a summary to the most relevant integration standards.

- STEP (ISO 10303-11, 1994) , Standard for Exchange of Product Model data, is a series of standards under development within the International Standardization Organization (IOS). Launched in 1983, ISO/STEP is a very ambitious long-term project to develop information sharing standards that span all sectors of engineering based on product modelling concepts.
- CIS (CIMsteel, 2003), The CIMsteel Project deployed the CIS as a set of computing specifications that would allow software vendors to make their engineering applications mutually compatible. These standards enabled vendors to develop and implement translator for export and import of engineering data related to the design, manufacture and erection of steel framed structures. The initial implementation facilitated the electronic transfer of information in the form of data exchange files, which were passed between CIS-compatible applications and thus, between the project participants.
- IFC (IFC, 2001), The International Alliance for Interoperability (IAI) began with a mission to define, promote and publish a specification for sharing data throughout the project life cycle, across disciplines and technical applications within the domains of Architectural, Engineering, Construction and Facilities Management. Its member organizations intended to specify how the 'things' that could occur in a building (such as doors, windows, walls) should be represented electronically. Each specification (called a 'class') is used to describe a range of things that have common characteristics. The classes defined by IAI are terms 'Industry Foundation Classes' or IFC.

IFC is likely to be the future integration standard for Architecture/Engineering/Construction (AEC) domain. IFC is already adopted in academic research and many software vendors

have implemented it in their products. It is an open standard with an international interest. Research efforts are devoted for the standard. Some are working to expand it for more comprehensive cover while others are trying to apply it in various design applications. Below is a summary of the use of IFC in some research projects for collaborative design in construction.

- The WISPER (Faraj et al., 2000) project is well known as an early endeavour to establish an IFC-based model server environment to support design collaboration. Even though WISPER was based on very early versions of the IFC schema, it demonstrated the potential of using a shared building model.
- IFC Model Server (IMSVR) framework (IMS, 2002) was developed to store IFC object model within a central database and run on the internet. The project developers believed that if model server functionality is provided on the internet, IFC compatible applications can communicate with each other via the internet and utilize functions implemented in the model server such as partial model import / export. This would improve the communication between IFC compatible applications and limit the need for file base exchanges.

3.6.2.4. Visualisation Technology

Visualisation in collaborative engineering design systems is usually an important feature. Graphical interfaces can vary in their sophistication. They may be used for mark-up design models or in more intuitive way for modify the models. The selection of the suitable technology depends on the application type (i.e. API or Web) and its function.

In order to deliver and manipulate interactive 3D objects effectively on the Web, some concise formats, such as VRML (Virtual Reality Modelling Language), X3D (eXtensible 3D), W3D (Web 3D) and MPEG-4, have been launched. VRML is fundamental for these standards to represent geometric elements and scenes. X3D is the successor to VRML and features extensions to VRML with the ability to encode the scene using an XML syntax. The above formats are for generic usage but not suitable for representing complex design models since they lack feature and assembly structures to organise design information (Li and QIU, 2006). There are some initiatives and effort to establish a common open format for sharing visualization data between disparate engineering applications such as OpenHSF (OpenHSF, 2004).

For the applications that are not web-centric, the technologies used to support visualisation are different. There are two fundamental standards for graphics API, namely, OpenGL and Direct3D. OpenGL (Open Graphics Library) (OpenGL ARB, 2004) is a standard specification developed by Silicon Graphics. OpenGL is widely used in CAD, virtual reality, scientific visualization, information visualization, flight simulation and video game development. Direct3D (DirectX, 1994), which is only available for Microsoft's various Windows operating systems, is also used to render three dimensional graphics in applications where performance is important, such as games. There are other platforms that support 3D application programming interface. These platforms are usually run on top either OpenGL or Direct3D to simplify the implementation. Java3D is an example of such platforms that runs on top of either OpenGL or Direct3D. Compared to other solutions,

Java 3D is not only a wrapper around these graphics APIs, but also an interface that encapsulates graphics programming using a real, object-oriented concept.

3D visualisation technologies have been used successfully throughout the construction industry, ranging from improving construction education (Messner and Horman, 2003), to being used at a larger scale for landscape and town planning (Makanae, 2003), as a decision support system for construction planning (Dawood et al., 2000), and as model viewers (Fu et al., 2006) . These systems have generally made use of simple 3D models such as VRML that limit the user to a relatively static representation of the design information. Some experimentation has been made in adopting computer game engines for use within building design IT systems in order to make use of the graphical flexibility inherent in such engines in a dynamic way (Shiratuddin et al., 2000) . However, such solutions have generally been found to be unsuited for use in building design, as they are generally unable to support the building design tools and processes that are required in complete building design software. This limits the use of computer game engines to displaying relatively static 3D representations of a building design, in a similar manner to VRML.

3.6.2.5. Programming Languages

Programming history witnessed more than 2500 languages (Scott, 2006). The ones that are most used in the mainstream software application development nowadays are those that support the concept of Object-Oriented Programming (OOP). Object-oriented programming is a programming paradigm that uses "objects" to design applications and computer programs. It utilizes several techniques from previously established paradigms, including

inheritance, modularity, polymorphism, and encapsulation. Today, many popular programming languages (such as Java, JavaScript, C#, Visual FoxPro, VB.Net, C++, Python, Perl, PHP, Ruby and Objective-C) support OOP concepts with each has its own strengths and weaknesses.

The choice of the programming language for coding a system may be determined by many factors such as the targeted operating systems or the adopted technologies in the system developments. For example Java and C++ are candidates for applications targeting multiple operating systems or when CORBA is the implementation technology of the communication infrastructure among distributed resources. C# or any other .NET language can be an option if the .NET technology is adopted to implement the communication infrastructure of the system.

3.6.2.6. Modelling Tools

During developing a concurrent design system, a modelling language or modelling languages can be used to express information, knowledge, and processes in a structure that is defined by a consistent set of rules. A modelling language can be graphical or textual. Graphical modelling languages use diagram techniques with named symbols to represent concepts and lines that connect the symbols and to represent relationships and various other graphical annotation to represent constraints. Textual modelling languages typically use keywords accompanied by parameters to make computer-interpretable expressions. Below is a summary of some common used modelling languages in support of developing concurrent systems.

- UML (UML, 1997)– The Unified Modelling Language is a standard language for specifying, visualizing, constructing, and documenting software systems.
- IDEF0 (IDEF0, 1993) – IDEF0 is a method designed to model the decisions, actions, and activities of an organization or a system. It was originally developed during the 1970s as part of the U.S. air force program for Integrated Computer Aided Manufacturing (ICAM) and formalized by publication of the IDEF manual in the early 1980s.
- EXPRESS (National Institute of Standards and Technology (NIST), 2000)– EXPRESS is a data modelling language that combines ideas from the entity-attribute-relationship family of modelling languages with object modelling ideas of the late 1980s. It became an international standard (ISO 10303-11) in 1994 for use in engineering data exchange.

3.6.3. Implementation Approaches of CE Applications

3.6.3.1. System Architectures

According to LI and QIU (2005) basic architectures of collaborative concurrent systems can be classified into three types:

- Communication server and modelling client (thin server and strong client).
- Modelling server and visualised-based manipulation client (strong server and thin client).
- Application or service sharing (peer-to-peer).

In the scenario of communication server and modelling client, clients are equipped with whole CAD functions and some communication tools. A server is an information exchanger to broadcast design models or commands generated by a client to other clients during a collaborative design process. An example of such system is CollabCAD (CollabCAD, 1999).

In the scenario of modelling server and visualisation-based manipulation client, a server can establish more effective collaborative functions to realise controllable remote services. The data structures in clients are light-weighed to support visualisation and manipulation functions (such as selection, transformation, changing visualisation properties of displayed parts, etc.). The main modelling activities are carried out in a common workspace on the server side. A thin/strong representation proposed in client/server respectively can enhance the performance of the whole system effectively. Developed systems include Alibre Design (Alibre, 2005), and OneSpace.net (OneSpace.net, 2005).

Different from the client/server architecture, peer-to-peer architecture is a group of computers which can connect with equivalent responsibilities to pool their resources and decentralise the management. Under the peer-to-peer architecture, some computers and executive applications are transformed into shared resources that are accessible from each of the computers on the peer-to-peer architecture.

Considering the characteristics of collaborative design systems, the above three architectures show potentials in different aspects. The implementation of the first

architecture is quite straightforward compared with the other two architectures. Through equipped with a communication tool, standalone systems can be conveniently re-developed as design clients and linked together by a server with the functionalities of information exchange and coordination. This architecture can effectively meet the requirement of real-time interactive operations on design models since most of the geometric computing for modelling and modifications are carried out on the client side. At the same time, it can support a heterogeneous collaborative environment with different CAD system clients, in which a neutral information exchange format can be designed for communication in the environment (Li and QIU, 2006). This architecture is being commonly used in large software company (e.g. XSteel and StruCad for steel detailing) to support a multi-users feature in their applications because of the relatively short time required for implementation and the amount of changes on the original application structure. However, the adaptability of the architecture is not easily maintained and such architecture is difficult to be migrated to a Web application. The second architecture is getting popular since it brings a new kind of business model—ASP (application service providers). With an ASP-enabled collaborative system, small and medium-sized enterprises (SMEs) or even individual designers with specific domain knowledge can hire on-line high-end collaborative design systems, so that they are able to participate and co-operate in the design process with large design companies. Through this manner, renting on-line high-end design systems running on some high-end CAD systems such as CATIA (CATIA, 1998) and Unigraphics (Unigraphics, 1991) have now become affordable for SMEs, and not just for large companies (Li and QIU, 2006). The scalability of the system can be enhanced since it is

convenient to add or remove seats. Problems of this architecture include the increased implementation difficulty and the sluggish communication speed caused by the vast amount of information exchange across networks. In the third architecture, services of a peer with design functions can be manipulated by another peer. The architecture is high-performance for point-to-point communication, and provides a more flexible manner to integrate individual systems in the co-design environment. However, it is not suitable for a large group of users to work together due to the restrictions of the peer-to-peer computing capability. Another obstacle is that the co-ordination among the de-centralised systems is more complicated comparing to the first two architectures.

3.6.3.2. System Communication Mechanisms

Communication in a collaborative system can be synchronous or asynchronous. Synchronous and asynchronous are two major modes to support collaborative activities. Synchronous or concurrent communication enables users to work together in 'same time-different place', therefore, to meet some efficiently collaborative requirements. Asynchronous communication can enable collaboration to happen in 'different time-different place'.

In a synchronous paradigm, each user is able to participate in design collaboration simultaneously with modelling and modification capabilities. During iterative design sessions, changes imposed by a user can be communicated with other project participants and merged with their concurrent design models. Suitable coordination and synchronisation mechanisms are crucial to schedule a design activity in parallel and ensure no conflict

arises during this real-time and iterative process. A real-time data sharing in a collaborative concurrent design system is difficult to be realised due to the contradiction of the large-volumetric design models and the limited bandwidth of the current network (i.e. Internet) (Li and QIU, 2006).

In an asynchronous paradigm, an activity is centrally managed and coordinated in an assembly level. Assembly constraints are encapsulated as interfaces to support different designers to cooperate, and to ensure that sub-assemblies and components allocated to individuals are compatible with each other. Although real-time sharing is not achieved, an optimised representation strategy for assemblies to simplify data to avoid the sluggish transmission can be still met. Meanwhile, a propagation mechanism for changes happened in a sub-assembly or component to the entire assembly structure is imperative to maintain the assembly consistency (Li and QIU, 2006).

3.6.3.3. System Integration Mechanisms

An effective mechanism is very important to seamlessly integrate dispersed functional modules from the upstream design and the downstream manufacturing. According to Boddy et al (2007) integration can be developed with respect to two axes: Semantic focus or Application domain focus. Semantic axis spans the whole spectrum of past, existing, and future applications with underlying semantics ranging from data structures conveyed through data models to rich-semantic representations through ontology. Application domain axis represents the focus of research effort on a continuum from application and

data centric, to process and people centric (Boddy et al., 2007). Based on the two axes, Boddy et al (2007) draws four areas of research in system integration. Theses are:

1. Integration at data–application level.
2. Integration at data–process level.
3. Integration at application–semantic level.
4. Integration at the process–semantic level.

According to Boddy et al (2007) integration at data–application and data–process level are the predominating research effort areas.

Data-centric integration relies on providing a neutral information exchange format. Amongst the first efforts at integration were those born of the increasing use of CAD in design offices since the mid eighties. The necessity of transferring CAD data from one system to another resulted in de facto standards that persist to this day, such as the Drawing (or Data) Exchange Format (DXF) and the Initial Graphics Exchange Specification (IGES). Among recent coordinated standards popular in engineering domains is STEP which is used in some research to fulfil system integration (Boddy et al., 2007). XML, which is regarded as the next-generation Internet mark-up, provides ways to describe and store complex data structures suitable for exchanges over the Internet. It has become more popular as the exchange format for design and manufacturing data in a collaborative environment (Li and QIU, 2006).

Process-centric integration relies on project processes as the natural level at which people interact with their work in an organisational or project context. There is already a trend towards business process integration in the wider commercial context, with many vendors offering 'solutions' in the domain (Boddy et al., 2007). Several significant service-centric mechanisms for collaborative system integration have been reported, including interface-wrapping services, MAS services and Web services. In order to address the inherent complexity of structures and interactions among the functional modules in an integrated system and provide interpretability between the different systems, a common 'component interface' mechanism has been developed, through which various application components can be made to interact with each other (Li and QIU, 2006). The field of 4D-CAD has attempted to integrate project scheduling with the building data model, primarily to better understand and coordinate the construction process. Other moves to go beyond 4D CAD into nD, with tools for multi-dimensional interrogation and analysis of the building model have been proposed, such as the 3D to nD platform. Also in this area many research begin to move toward distributed architectures offering services and service integration.

Many researches have worked on providing integration on both data-application and data-process at the same time. Han et al (1999) developed a distributed service architecture that enables the delivery of building design services over the Internet based on a common IFC server. As examples of building design services, the prototype implements a project manager service with a companion CAD package, a disabled building code analysis service, and a service that generates and displays an accessible path for a wheelchair for a given building design using motion planning and animation techniques. The SABLE

project (SABLE, 2002) has also made progress in this respect having discipline specific interfaces to server based IFC building models. These interfaces include client briefing/space planning, architecture, HVAC design, cost/quantity takeoff and scheduling.

3.6.3.4. Access Control

Collaborative design environments, in general, and integrated design environments, in particular, impose special requirements for access control (Bakis et al., 2007). These special requirements have led to the development of a number of role-based and task-based access control models. With role-based access control, there is not any particular owner of a resource; the access permissions are associated with roles and users are assigned to appropriate roles. With task-based access control, the environment controls whether a user is allowed to perform his/her task and the access permissions are valid only for the duration of that task.

The research into the implementation of role and task-based access control in distributed data sharing design environments has been more substantial than that of design transaction management (Bakis et al., 2007).

The PerDiS project is an example to provide role/task based access control that can be implemented in distributed data sharing design environments (Coulouris et al., 1998). The current research implementation of access control, as in the PerDiS project, is limited to clusters of objects and so there is a potential need for fine grained-control of access to individual objects and their attributes. Besides, the PerDiS project has a limited set of

access modes (Read or ReadWrite) and that is constrained by the architecture of its platform (Coulouris et al., 1998).

3.6.3.5. Version Management

Version management is the control of multiple revisions of the same unit of information. Version management functionality may be required in integrated design environments for various purposes. This includes the need to maintain the design consistency, each time one of the versions is modified, all interrelated versions of other parts and aspects of the design must be accordingly updated. This is would be very important in an asynchronous communication environment. Also since the design process is an iterative activity, the development of each part may pass through a number of versions and the ability to return to any earlier state of the design may be necessary.

Based on Bakis (Bakis et al., 2007), the PerDiS platform (Coulouris et al., 1998) is the only attempt to provide design version management functionality as part of its transactional system. However, the support for change propagation is lacking.

Nevertheless, the interest into versioning management control in this research is limited to the need to track the changes history of the design data. This is because in a concurrent collaborative design system with a centralized shared model there will be always one version of the model shared by all designers at one time and so versioning for merging purpose is not crucial.

3.7. Collaborative Engineering in Construction

In the application of information technology for collaborative engineering in the construction industry valuable steps have already been achieved. The collaborative design applications can generally be divided into three levels, namely, document-based exchange level, common format-based exchange level, and common model-based collaboration level.

Electronic document-based exchange level is a low form of collaboration based on electronic document exchange. This kind of system only stores, manages, and organises the documents of a project. The system has no knowledge of the detailed contents of the documents. The main aim of such systems is to link different forms of information together so that a project team or organization can easily access and control the information (Sun and Aouad, 1999).

Common format-based exchange level is about data exchange within or between project team members. In this approach, each profession in the project has its own internal data format, however, all of these data format subscribe to a neutral one (Sun and Aouad, 1999). A sending application translates data from its internal format and encodes it into an established neutral format. This file is then transferred to the receiving application where the data is translated into the internal format of the receiving system. DXF is a well-known file format for exchange graphical data between CAD systems. Recently, the STEP physical file format has emerged as the neutral format for exchanging full product data in engineering domains. In AEC domain, IFC is a good example of product model to provide

interoperability between homogenous software. Common format-based approach for collaboration is still restricted to exchange level.

Common Model-based collaboration level is a high level of collaboration based on an integrated consistent model that can be accessed by the dispersed participants in the design process. The practicality of this kind of collaboration depends on how far it considers the entire product lifecycle. Many researchers see the need for more research on distributed collaborative environments that integrate the design process.

Sun and Aouad (1999) compared the previous types of collaboration in many aspects. They concluded that the integration of document contents is a very shallow form of collaboration and it is a comparatively simple form of integration that is already in wide use and commercial systems are available. They also mentioned that common format-based data exchange has been the main focus of many recent product data technology research initiatives. However, So far, there is still no concerted research on some of the key components of concurrent engineering systems. Therefore, system of this kind remains a distant prospect and it will be soon feasible to develop an integrated concurrent engineering system which can support project information sharing between different organisations located remotely especially with the rapid development of distributed technologies.

In the same context, Plume and Mitchell (2007) reported from their experience in running a multidisciplinary design studio using a shared building model hosted on an IFC server that

a few key issues need to be addressed in the development of collaborative design systems. The first key issue is the importance of creating a building model that is suitable to support collaborative design and the need of the existing models to be re-oriented towards the used in multi-disciplinary applications. The second issue is the issue of model management to ensure concurrent access with maintaining the semantic integrity of the model (i.e. versioning control). Finally, attaching ‘‘intentions’’ to elements in the project model as in a cooperative design environment, there is a need to find a way to convey the intent behind the decisions that have been taken.

Moreover, Alshawi and Faraj (2002) described the current practice of design and construction in industry as a stage controlled process in which the design lifecycle of a project is divided into many isolated stages. The drawbacks in the current design process can be overcome by providing better communication structure for information flow within the design process. Research by Anumba et al. (2000) also suggested that development of integrated solutions for the design life cycle is required to overcome some of the problems in the current design process. These integrated solutions can be provided in many ways including:

- Integrating different software applications within the design process to provide a broader solution.
- Developing neutral file formats for all information used in the building design process.
- Developing a central information repository/database that can be accessed by the different participants in the design process.

- Developing a structured means of communicating between the different players in the design process.
- Developing sophisticated software within the process that helps to make informed design decisions and provides assistance to the user through a more holistic design view.

3.8. Summary

This chapter has summarised state of the art of collaborative design. It first gave an overview of collaborative design concept and then introduced the technologies that are being used in the development of concurrent and collaborative design systems. Finally it outlined the current state of collaborative design applications in construction industry.

Chapter 4.

Requirements Analysis for a Collaborative Building Design System

The aim of a virtual collaborative design system is to provide designers with a set of mechanisms and tools for accessing and sharing information in coordinated and organised manner. Central to the design of such a collaborative environment is extracting the system high level requirements based on the needs of the collaborative design process.

From the preliminary investigation stage of this research, presented in the previous chapters, the key requirements for a collaborative building design environment are identified. Some of these requirements are fundamental for the system to allow real-time collaboration work. These requirements are discussed next.

4.1. Concurrency

The communication strategy in a system can be either asynchronous or synchronous. In an asynchronous paradigm, an activity is locally managed and coordinated in an assembly level. The asynchronous manner helps avoid the sluggish transmission (Li and QIU, 2006). However, this strategy requires data merging techniques to ensure data consistency and it is more suitable for applications that do not require concurrency.

In a real-time collaborative environment, concurrent access to shared resources is a key feature to supporting collaboration. However, providing concurrency to shared resources in

building design is a complex task as that may lead to conflicts and data inconsistency. Different strategies can be implemented to ensure the consistency of the shared data.

Data consistency can be maintained by using locking technique (also know as pessimistic method). It is the simplest method for preventing concurrent access problems. It locks data so that only one user at a time has write access to the central repository data. Once one user checks out, others can read that data, but no one else is allowed to change that data until that user checks in the updated version (or cancels the checkout) (Paulraj, 2003). Although this method ensures data consistency, it may lead to delay in achieving tasks especially if data is left exclusively locked for too long.

Version merging method, on the other hand, can be used to provide concurrent access to data (also know as optimistic method). It allows multiple users to edit the same data at the same time and the system provides facilities to merge changes into the central repository (Paulraj, 2003). This method is suitable when the concurrent changes are not highly interrelated which is not the case for building design data.

It is recognized that using pessimistic locks reduces the concurrent access to the data. However since time is usually not a driver and designers in building design have sufficient time to document their input and because the engineering data is greatly interrelated, the author believes that the pessimistic method is the most suitable for a collaborative building design system to provide concurrency and ensure data consistency. The inefficiency of the

pessimistic method can be alleviated by adopting the following guideline during the system development:

- Minimise communication with the central repository (e.g. validate data locally).
- Minimise and possibly compress information transferred over the network.
- Automate the process of user check in/out (i.e. locking/unlocking) so users will feel as if they have concurrent access to the data.

4.2. A Product Model

The data complexity and the amount exchanged among all participants of a project are huge so that providing a sophisticated product model is essential to manage such interrelated data. However, a product model in a shared workspace needs not just to cover the data representation of the building but it also needs to include control of the identification and ownership of the model entities. This is thought to be a vital requirement in a shared workspace environment (Shen and Dewan, 1992).

The need for an integrated product model to support collaboration is highlighted by many researches (Plume and Mitchell, 2007;Sun and Aouad, 1999). In the development of a collaborative design system the use of existing neutral data formats should be highly encouraged, so that information can be portable and easily transferred between software.

Therefore, one strategy for the provision of suitable product modelling support is to adopt existing models (e.g. IFC, CIS). However, the use of CIS should be discounted despite its comprehensive support for steel structures and detailing of the building, and its structural

analysis model, it does not represent the architectural aspects of the building design, and the non-steel elements of the building can not easily be modelled. The use of IFC should also be discounted, even though it is a more comprehensive model than CIS, because of the following aspects of The IFC model classes (Smith, 2005):

- The model is very generic, and does not constrain the design process. The research is concerned with tying the building design process closely to the information being used, and so this would not be appropriate.
- The model is not designed to support concurrency. In fact, the model developers' main goal is to provide a neutral data format to exchange data between different software and it was not developed to be used as an internal model.
- The incompleteness of the model. Although the model is relatively comprehensive, the required domain is not yet fully covered (i.e. steel building domain).

It is therefore thought that the existing product models in the building industry are not suitable to be used directly in a collaborative design system since they are mainly developed for supporting only the representation of engineering data. However since the potential product model for a real-time collaborative design system still needs to represent the engineering data, the model does not have to be built from scratch and instead a suitable model can be extended to support collaboration.

4.3. Access Control

Collaborative computing environments and integrated design environments impose special requirements for access control. Access control can be implemented using various models

such as role-base and task-based access control models (Bakis et al., 2007). However these models have some limitations that, in the author opinion, make them inadequate for implementing the access rights mechanism for a collaborative design system.

With role-based access control, there is not any particular owner of a resource; the access rights are with associated roles and users are assigned to appropriate roles. This limits users from preserving their own data for those who are associated with the same role. One solution for this problem is to create more roles to allow more data preservation and ownership control. However, this may lead to too many roles and increase the complexity of the role management. With task-based access control the environment controls whether a user is allowed to perform his/her task and the access rights are valid only for the duration of that task. The problem with the task-based model is that users have full access to resources during their tasks. This again limits the users by their inability to preserve their data from future changes.

An efficient access control model needs to meet the following requirements:

- Implementation of default access rights to the data based on user roles.
- Allowing users to preserve ownership of their own input.
- Implementing various access right types (Coulouris et al., 1998).
- Integrating the access rights with the product model so that access to product model elements can be granted by type-level, element-level and attribute-level (Coulouris et al., 1998).

4.4. Communication Tools

Communication is the backbone of any collaborative application. It is essential for activity coordination so providing proper tools to facilitate all types of communication is an essence of such systems. The following is a list of the main facilities that may be needed to ensure effective communication.

- Asynchronous communication :
    - Email.
    - Notification facility.
    - Discussion boards.
- Synchronous communication :
    - White board.
    - Instance message exchange.
    - Visualised graphical discussion.
    - Video conferencing.

4.5. 3D Intuitive Interface

A 3D intuitive interface is an important component for any modern design system (Bouchlaghem et al., 2005). The virtual graphical representation of the building has many benefits. It is in many cases more illustrative than words. A sophisticated 3D interface would satisfy the following requirements.

- Visualize the design model.
- Model rendering.
- 3D display and projection views.

- Access and modify the model.
- Interaction with and navigation of the model.
- Simulate the model behaviours.

The selection of the technology for implementing the graphical interface of an application can be highly influenced by its type. VRML or X3D, for example, are candidates for web applications and OpenGL or DirectX for desktop applications.

4.6. Performance

The data transferred in a collaborative design system is expected to be huge and complex so the performance of the system is an important factor when developing the system. Various techniques may be adopted to improve the transfer time of the data over a network.

One factor that highly influences the performance of a collaborative design system is its architecture. In the previous chapter, collaborative system architectures are classified into three main types (Li et al., 2005):

- Communication server and modelling client (thin server and strong client): In the scenario of communication server and modelling client, clients are equipped with the whole system functions and some communication tools. A server is an information exchanger to broadcast design models or commands generated by a client to other clients during a collaborative design process. The main advantage of this type of architecture is that it minimises the communication between the clients and the server. This helps in increasing the speed of data processing (e.g. data validation). On the other

hand, having the whole system installed on the client machine reduces the portability of the system.

- Modelling server and visualisation-based manipulation client (strong server and thin client): In the scenario of modelling server and visualisation-based manipulation client, a server plays more effective collaborative functions to realise controllable remote services and the client structure is light-weighed to support visualisation and manipulation functions. The performance can be sluggish especially if the model data is frequently updated and the data amount transferred across networks is huge. On the other hand it is more portable than thin server and strong client architecture.
- Application or service sharing (peer-to-peer). Different from the client/server architecture. Peer-to-per architecture consists of a group of computers that are inter-connected and have equivalent responsibilities to pool their resources and decentralise the management.

For a real-time collaborative engineering design system, the author believes that the best architecture to adopt is a thin server and strong client. Although a thin client / strong server architecture has many advantages (e.g. portability and scalability) and it is very popular in many applications such as e-commerce and mark-up tools, it is thought to cause a sluggish performance for a real-time collaborative design system. The peer-to-peer architecture is thought not to be appropriate for a real-time collaborative building design system since it does not assist system integration and make data ownership and access right control more difficult than the other two architectures.

4.7. Version Control

A version control mechanism is increasingly recognized as being necessary for the organization of multi-user projects (Plume and Mitchell, 2007). It is most commonly used in engineering applications to manage multiple visions of the same project design during its development. The simplest way to provide a version control mechanism is to create backups of the data every certain milestone. However, this method has the following downsides:

- The number of the backup versions of the model in a building project would be very large since many design scenarios may be examined.
- The size of the backup versions of the model would be very huge even when compressed.
- The need for an automated process to maintain the created visions.
- Restoration process of a previous version in a distributed design system can be a cumbersome operation.

In this work, the author believes that it is more feasible to maintain multiple versions of the commands (the actions) carried out during the project rather than the final result of the commands execution. This would have the following advantages:

- The command data size is much smaller than its execution results. This improves the system performance since only the command data is transferred over the network and its execution is done on the received machine.

- There would not be a need to keep multiple copies of the model since the commands can be stored as a part of the model data.
- The commands can be stored in a way that allows the system to move backward and forward in the model history.
- The restoration process of a previous version is much quicker since the command data file can be quickly downloaded on each user machine and then executed. This would be much faster than transferring the whole model data over the network.

4.8. Design Automation

The provision of sufficient design task automation would save the engineers conducting low-level designs and reduce human error. The designers in this case provide the design intents data (i.e. the high level design requirements) and low-level details will be then generated using intelligent algorithms.

The automation requirement in a collaborative design system is more crucial for decision making than collaboration. The level of design automation in an engineering design system can vary from simple algorithms to complex intelligent ones.

4.9. Document Management

It is well known that the document number in an ordinary project would reach of few thousands thus it is essential to have a facility to manage this huge amount of documents in during design process. The following are the main features that the system needs to mange documents:

- Automatic data backup and recovery.
- Reporting mechanisms.
- Support for document versioning.
- Use the check-in/check-out feature to block others users on the network from trying to edit a document that is locked or there is a user is currently working on it.

Since this research is more design centric and many commercial products that support document management are already available (Sun and Aouad, 1999), the requirement of document management has not been addressed.

4.10. Summary

This chapter has summarised the key requirements for a collaborative building design environment. The implementation techniques of these requirements are also highlighted. Some of the identified requirements are considered important for a real-time collaborative design system such as concurrency, access right control, and a collaboration-aware product model.

Chapter 5.

A Proposed Collaborative Building Design System

The previous chapters described the investigation stage of this research work. This chapter describes the development process of a prototype application of a proposed system to support collaborative design workspace and satisfy the main requirements defined in the previous chapter. The vision of this system is to provide a collaboration environment through a shared workspace where designers will be able to work collaboratively with adequate level of concurrency on a shared model and encompass most necessary communication and data exchange electronically.

The development process of the prototype follows software methodology for developing an IT system. A development process or life cycle of a software system is a structure imposed on the development of the product. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process. Software engineering processes are composed of many steps, notably the following:

- Requirements Analysis.
- System architecture.
- Implementation and coding.
- Testing and evaluating.

There are other common activities during the life cycle of a software product such as documentation, training and support and maintenance. However, these steps are less important than the rest for a software prototype as it is usually used as a proof of concept.

The previous chapters can be considered as the requirements analysis stage of the system. This chapter describes the system architecture and its implementation. Demonstrating and evaluating the system is covered in Chapter 6 and 7.

5.1. System Specifications and Requirements

Chapter 4 has identified the main requirements of a collaborative building design system. It also discussed the various methods and techniques used in their implementation and highlighted the suitable ones to be used from the author point of view.

Below is the summary of the system requirements:

- Concurrency: Providing adequate concurrent access to data in a collaborative building design system is necessary. However, uncontrolled concurrent access to the model data may lead to inconsistency thus automate locking/unlocking method will be used to provide concurrency.
- Product Model: a product model will be developed to provide two features. Data representation and collaboration support.
- Access Control: multi-level access to the product model will be incorporated with the product model and the collaboration process to support data preservation and ownership.

- Communication: the system will support asynchronous or synchronous communication method among the design team.
- 3D Interface: A sophisticated 3D interface is to support model navigation and manipulation.
- Version Control: The system will maintain the history of the model and provide the facility to review the history of the model elements and to restore the model to any point in its life.
- Design Automation: Various level of automation can be implemented to support the design process. In the developed prototype, the design automation is limited to simple algorithms since this is not the focus of the research.
- Document Management: the document management requirement will not be addressed in the prototype implementation since the research is more design centric and there will not be a research value from addressing it as many commercial products that support document management are available.

5.2. System Architecture and Design

The system architecture and design is the second stage in the development process after the requirements analysis stage.

The overall chosen architecture of the proposed system is client/server architecture with centralised shared resources. This type of architecture is very common for multi-user applications. The main advantage of client/server architecture is providing a secure central data repository. This would help reducing data fragmentation and increase data integrity.

The system structure can be presented in terms of physical or virtual structure. In terms of the physical structure, the system is a central server, maintaining the design models, and many workstations (i.e. clients) where designers can access the model data via a network, while the virtual structure of the system is a three-tier model: design tier, communication tier and data tier. Figure 5-1 shows a macro view of the system architecture.

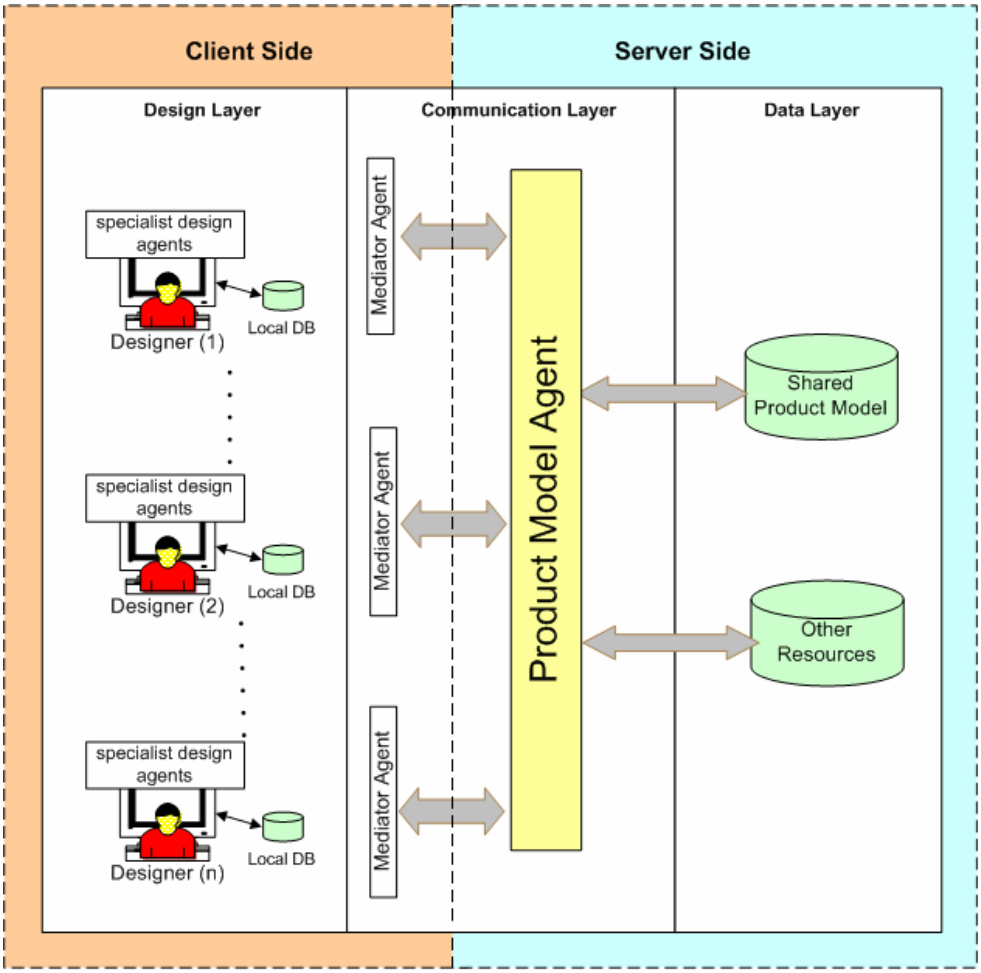


Figure 5-1: System architecture

The design tier includes the design system which composes specialist software design agents. These agents assist the designer during the design process and they are located on the client side.

The communication tier includes a Mediator Agent and the Product Model Agent. The Mediator Agent is located on the client side and the Product Model Agent is located on the server side. Data exchange is achieved through the communication between those two agents. The Mediator Agent is responsible for submitting the changes carried out by the designer to the Product Model Agent and sensing any changes upon the shared model carried out by the other designers. In other words, The Mediator Agent is responsible for synchronising the local data on the client side with the shared data on the server.

The data tier consists of a shared product model and all other relevant resources. It is located on the server side and maintained by the Product Model Agent. However, a copy of the product model database will exist on each client side. The reason for that is to reduce the number of calls between the client and the server. This would improve the system performance substantially. Although having a copy of the shared model on each client machine may seem to cause inconsistency and data fragmentation. This is not the case as the local copy of the shared model on each client machine is a mirror of the shared product model and not part of it and the copy is maintained and updated all the time to match the shared model.

The detailed description of software system architecture is usually difficult to be described in its entirety. It is thought that the detailed system architecture is best described through various models. In this work the models that describe the system architecture and design are: The Agents Model, The Collaboration Model, The Communication Model, The Product Model, The Action Model, and The VR Model. These models aimed to give a better understanding of the system. These models were to document the decisions made about the system design, specify the system structure or its behaviour, and serve as guidance during its implementation and construction.

5.2.1. The Agents Model

This model describes the agents used in the system and their main roles. The agents in the system can be seen as complex software entities taking actions to satisfy internal goals based upon their perceived environment (Wooldridge, 2002). The Agents Model consists of a collection of software agents that in effect controls the processes of the system including engineering design and communication processes.

All Agents in this collection are developed by inheriting from a single software agent called the Generic Agent. The Generic Agent is specifically developed for the prototype system described in this research work. Any of the higher level agents are developed by inheriting from the Generic Agent and then adding the required specialised features.

5.2.1.1. The Generic Agent

The Generic Agent provides the typical attributes and functionalities normally provided in software agents (Shen et al., 2001). The main aim of developing the Generic Agent is to

simplify the implementation of the system agents. This also means that any typical feature required to apply to all agents can be applied once into the Generic Agent. Figure 5-2 shows the internal architecture of the Generic Agent.

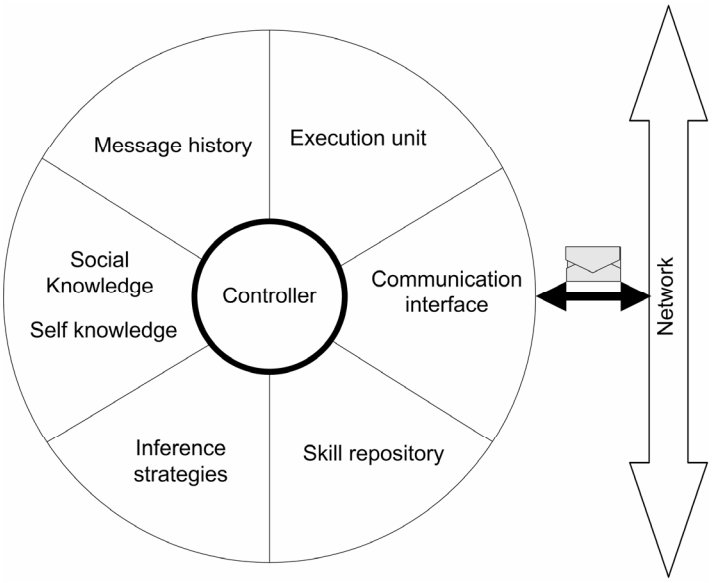


Figure 5-2: Internal agent architecture

The Generic Agent architecture consists of the following components:

- Communication interface: responsible for sensing the environment for any incoming messages. The incoming message is processed though two steps: (i) storing it in the message repository and (ii) decoding it and passing it to the control unit for further processing. The communication interface is also responsible for processing the out-going messages by encoding them and mailing then in accordance with the exchange protocol.
- Execution unit: once the agent has recognized that a significant event has occurred, either by sensing the environment or being notified by another agent, and need to take

some action(s), the execution module will execute the proper selective actions which are implemented in the skill repository.

- Skill repository: represents all actions the agent is able to execute. The appropriate skill to execute is called by the execution unit.
- Message history: store all incoming and outgoing messages. This history may be used by the agent when making decisions.
- Social knowledge: allows the agent to interact with other agents. It consists of the communication channels, the location and the skills of other agents.
- Self knowledge: contains the agent's knowledge about itself such as its name and location.
- Controller: it determines the agent behaviour. It is responsible for manipulating the decoded messages and communicating them, if needed, with the users. It is also responsible for next action(s) and determining what to communicate.
- Inference strategies: The agent controller may consult the inference engine to make decisions according to its knowledge.

5.2.1.2. Agent Society

The agent society in the system composes of five types of agents. These are: the Product Model Agent, the Mediator Agent, the Interface Agent, the Structural Analysis Agent, and the Design Agent. As previously stated, all agents are specialised agents that are derived from the Generic Agent.

The Product Model Agent is located on the server and it works on managing the shared models. This agent ensures that no more than one user (i.e. designer) can access the model at any one time. It communicates the model messages with the Mediator Agents.

The Mediator Agent is located on the client side and it facilitates the communication between the client machine and the Product Model Agent. It is responsible for maintaining the local copy of the shared model on the client side and sensing for any changes on the shared model and updates the local model accordingly.

The Interface Agent is located on the client side and its main role is to generate the proper graphical representations of the product model. It allows the user to manipulate the model directly though the graphical interface and interprets that to appropriate messages and then passes them to the relevant agents.

The Design Agent is located on the client side and it is responsible for generating the low-level design details based on the design intent data. For example, the number and spacing of a group of secondary beams, or their dimensions would be auto-generated based on a particular floor system specified by the designer(s). It is also responsible of checking the design after the structural analysis is done.

The Structural Analysis Agent is located on the client side. Its main role is to generate the structural arrangement of the product model and conduct the structural analysis and check the stability of the structure or a part of it based on predefined structural assemblies.

Table 5-1 summarizes the characteristics of each agent in the system, in particular their competencies. In Table 5-1, human actors (i.e. designers) are considered as a type of agents (i.e. human agents).

Table 5-1: Agent society

Product Model Agent	**Location**	Server
	Role	Manage the shared product model.
	Tasks	• Lock the shard model when necessary. • Identify client computers. • Maintain the model data. • Communicate the model with client computers. • Update the model.
	Type	Software
	Acquaintances	Mediator Agents
Mediator Agent	**Location**	Client
	Role	Facilitate the communication between the client and the Product Model Agent.
	Tasks	• Communicate the local model database changes with the Product Model Agent. • Notify the designer about the statue of the shard model.
	Type	Software
	Acquaintances	Product Model Agent, Interface Agent
Structural Analysis Agent	**Location**	Client
	Role	Structural analysis**.**
	Tasks	• Conduct the structural analysis of the structure or part of it. • Check the stability of the structure.
	Type	Software
	Acquaintances	Design Agent, Interface Agent

Design Agent	**Location**	Client
	Role	A/E Design automation**.**
	Tasks	• Check the design against the design code • Suggest an initial design. • Suggest changes when the initial structure is inadequate
	Type	Software
	Acquaintances	Structural Analysis Agent, Interface Agent
Interface Agent	**Location**	Client
	Role	Visualize the model
	Tasks	• Interface the local design agents to the designer. • Generate the graphical representation of the model.
	Type	Software
	Acquaintances	Structural Analysis Agent, Design Agent , Mediator Agent , Designer
Designer	**Location**	Client
	Role	Project owner/ Architect/Structural Engineer/Service Engineer
	Tasks	• Define the project brief. • Define architectural layout. • Define structural frames. • Define services layout. • Modify/Confirm the local design agents' decisions.
	Type	Human
	Acquaintances	Interface Agent
Project Manager	**Location**	Server
	Role	Project administrator.
	Tasks	• Specify the initial design parameters. • Define project constraints. • Resolving conflicts. • Specify the project preferences. • Define the project team members.
	Type	Human
	Acquaintances	Interface Agent

5.2.2. The Communication Model

This model describes the communication process among the system agents and justifies the adopted approach of communication. The communication among the system agents is carried out by exchanging messages. The messages encapsulate the data needed by the agents to perform the relevant activities.

Working with distributed applications involves handling network calls within the limitation of a given network speed. This is an important issue if to avoid networking becoming the bottleneck of the design process. The adopted approach was to get more done on the client side with fewer calls across the network. Reducing the number of calls is critical when creating high-performance distributed applications.

The performance objective is to be achieved by:

- Placing a copy of the shared product model on each client side so inquiry and manipulate the model data is taking place locally.
- Validating the input data locally before it is submitted to the server.
- Packing (i.e. compress) each exchanged message to reduce the size of the data transferred over the network.

One challenge is managing the shared model integrity by ensuring that no two users can attempt to update the same information simultaneously, leading to corruption of the data being stored. The system provides Lock and UnLock method to ensure that only a single

user can update the shared model at any given time. The system automatically locks the model database prior to modifying any data stored in the database, and unlocks it once the modification is complete. The user, however, can optionally own the model lock and prevent all other users from modifying the model until it is released again. It is recognised that such model locking might hinder collaboration. However, it is provided so that a large one-off modification or radical changes can be made efficiently.

Figure 5-3 shows the sequence of calls occurs in a typical communication scenario between a mediator agent and the Product Model agent. It can be seen that most actions are carried out on the client side with fewer calls with the server. This ensures a better performance.

The communication calls between the client side and the server side can be grouped into three steps: preparation, action execution, and finishing. In the preparation step, the local model is synchronised with the shared model and the shared model is then secured. In the action execution step, the actual action is executed on both local and shared model. In the finishing step, the shared model is released.

In the preparation stage, when the user requests an action (e.g. add column), the interface agent asks the mediator agent to begin model editing. The mediator agent then tries to secure the shared model. Once the model is locked, the mediator agent starts synchronising the local copy of the model with the shared one. Once this done, the process control is returned back to the interface agent which validates the input against the updated model.

In the action execution stage, the mediator agent applies the action first locally then submits it to the Product Model Agent which applies the same action on the shared model. The Product Model Agent will not validate the data before applying the action, as it should already be validated on the client machine. Once this is done the process control is back to the mediator agent and then to the interface agent which updates the 3D views if it is required.

In the finishing stage, the interface agent asks the mediator agent to end editing the model. The mediator agent, in turn, asks the Product Model Agent to release the shared model so other users can now apply their changes.

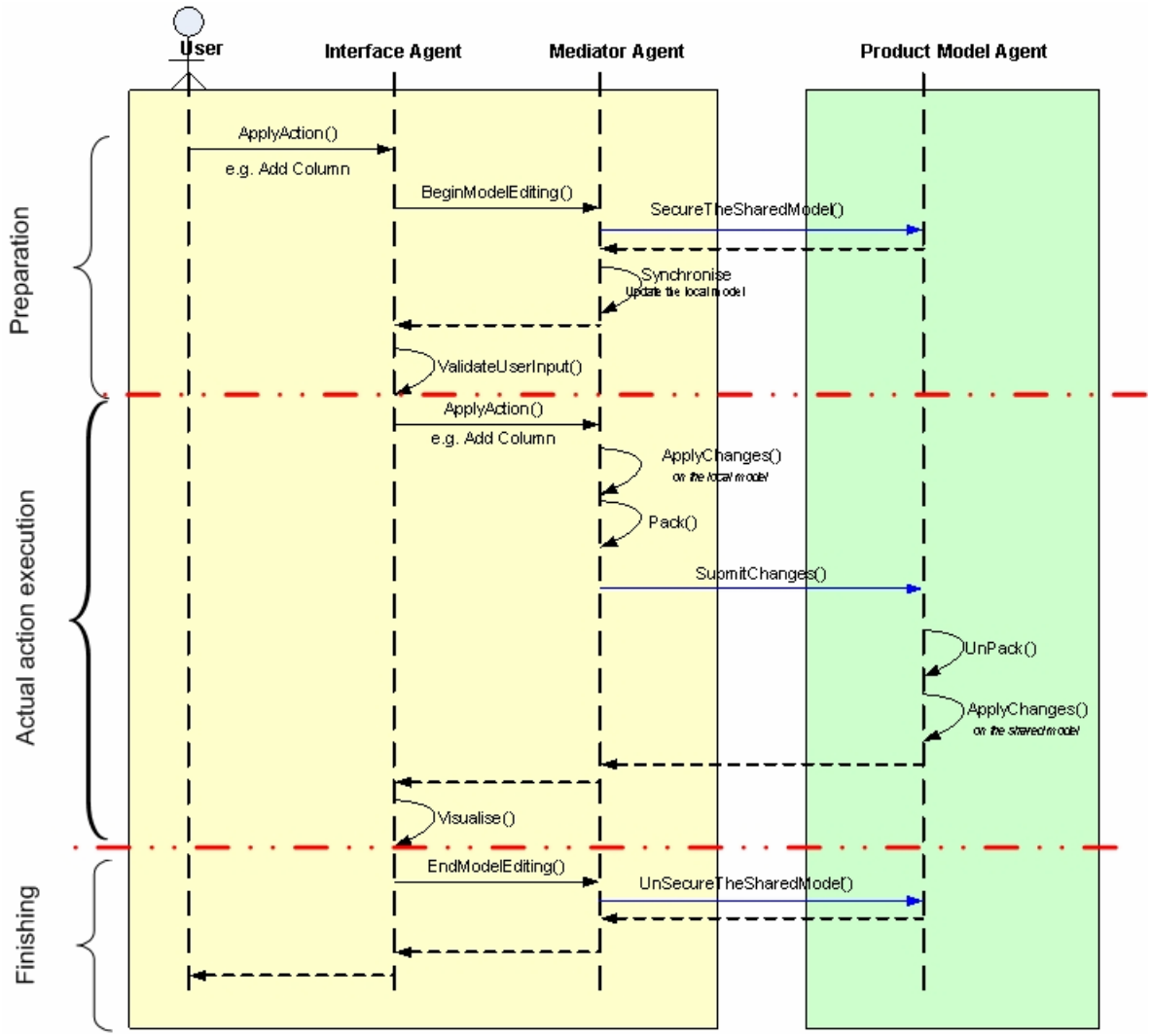


Figure 5-3: Communication model (UML Sequence Diagram)

The communication between the mediator agents and the Product Model Agent is happening all the time. The mediator agents are sensing the changes on the shared model. If they find new changes, they bring them to the client sides and if necessary pass them to the interface agents to update the visualisation of the model.

5.2.3. The Collaboration Model

The Collaboration Model does not suggest how the design team members should collaborate or deal with design conflicts. However, the design team will be able to place their status of acceptance on the design. They will have three options: complete agreement, no agreement and no comment. Under 'no agreement' the design members can communicate electronically in asynchronous or synchronous manner through the system tools to sort out the disagreements and reach consensus. It is possible that after several rounds of iteration, the 'no agreement' situation could remain unchanged. Under such circumstances, it may be necessary to prioritise specific design requirements at different levels and communicate synchronously (e.g. e-chat, face-to-face).

Figure 5-4 represents collaboration in an asynchronous manner. Although the designers can work concurrently on the shared model (i.e. synchronous communication), the design decisions are disjointed and the design members input their designs separately. Time is usually not a driver and each group has sufficient time to document their input in the improvement proposal. However, since each member's appearance is only weakly connected to the other design participants a longer time period is normally required for obtaining a consensus. In this instance the asynchronous approach is less effective in resolving the 'no agreement' case.

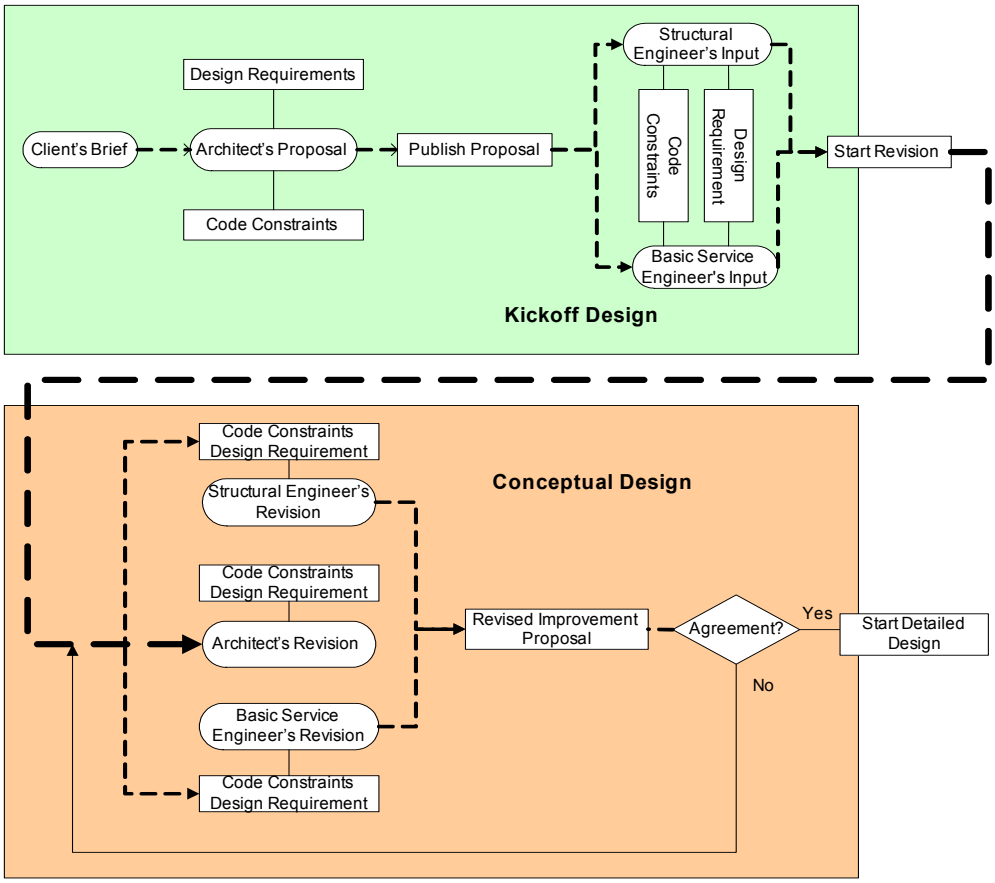


Figure 5-4: Collaboration model

5.2.4. The Product Model

The need for an integrated product model to support collaboration is highlighted by many researchers (Plume and Mitchell, 2007;Sun and Aouad, 1999). In the development of this system the use of existing neutral data formats was highly encouraged, so that information can be portable and easily transferred between software.

Therefore, preliminary strategies for the development of the proposed system considered the adoption of existing product models (e.g. IFC, CIS) to be used as part of the system shared product model.

The CIS/2 was initially considered, due to its comprehensive support for steel structures and detailing of the building, and its structural analysis model. However, it did not represent the architectural aspects of the building design, and the non-steel elements of the building could not be easily modelled. The IFC was then considered as it provides a more comprehensive model. However, experimenting with the model identified the following problems when working with IFC classes:

- The model is very generic, and does not constrain the design process. The research is concerned with tying the building design process closely to the information being used, and so this would not be appropriate.
- The model was not designed to support concurrency. In fact, the model developers' main goal is to provide a neutral data format to exchange data between different software and it was not developed to be used as an internal model.
- The incompleteness of the model. Although the model is relatively comprehensive, but the required domain is not yet fully covered (i.e. steel building domain).

It was therefore decided that the best strategy, to allow the greatest freedom within the research, would be to develop a suitable product model that is not based on either of these models but makes use of some of the IFC and CIS schemas, and structuring the model in a way to support collaboration.

The developed model builds on earlier research (Smith, 2005). The adopted model has been chosen as the starting point for developing the collaboration-aware product model. Figure

5-5 shows the product model structure. It comprises three tiers: Design Intent, Manufacturing Model, and Analysis Model.

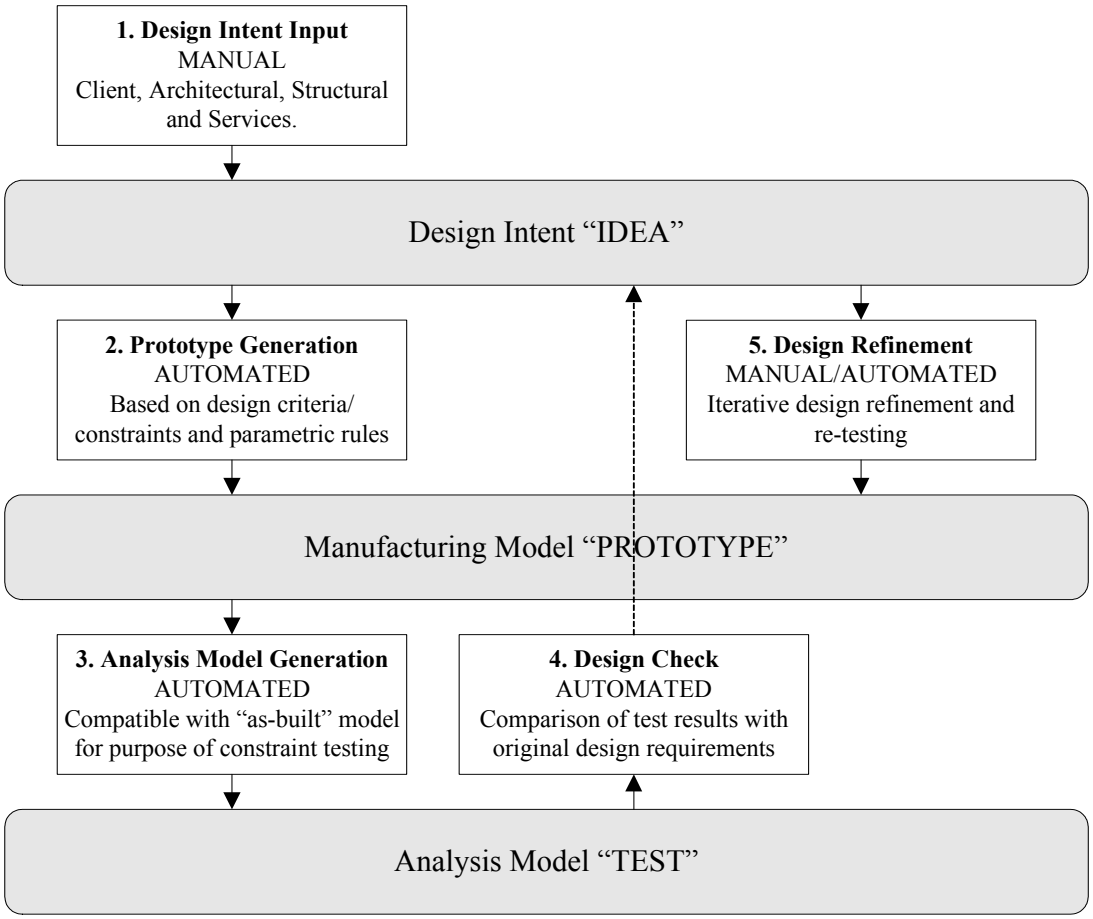


Figure 5-5: The product model tiers (Smith, 2005)

The design intent is input, formalised and stored in the first of the product model tiers (Design Intent Model) constituting the "IDEA" (tier 1). The IDEA is composed of the decisions and choices made by each of the building designers. As such, this tier contains the most valuable information in the building's design, and the other two tiers are ultimately a logical development of the ideas expressed within.

A manufacturing model constituting the "PROTOTYPE" (tier 2) is then generated from the "IDEA" using the second process group: Prototype Generation. The PROTOTYPE is the outcome of the total design process and principally includes the physical product such as the steel frame components, floor system and cladding system.

Having generated a workable PROTOTYPE, various processes are applied to test the conformance of the building to the set constraints and the general engineering principles. This is done by generating transient "TEST" analysis models (tier 3). And then by carrying out suitable checks on these models in order to report on the conformity with the set IDEA.

The adopted model has been extended to provide two important features needed for product model to provide collaboration. These are multi-access and revision control mechanisms which are described in much detail later on.

Access control (read, write, delete privileges) is an essential part of a collaborative environment to protect data from unauthorised users. It is important to determine what the data access control requirements are and relate these requirements to the product model structure. Users in the collaborative process must be identified by properly established access control mechanisms before access is granted or authorisation is issued. The primary objective of access control in collaborative environments is the preservation and protection of information. The access control mechanism in the developed model allows multi-level access. Three concepts are included to manage the access rights to the model data. These are Actors, Roles, and Permissions. An Actor in the model represents a single designer or a

group of designers. Actors will have default granted general permissions based on their default assigned roles. The permission concept in the model is the key element in the access control mechanism. They are generically four types: read only, allow delete, allow create, and allow change ownership. Permissions in the model can be classified as general and specific. The general permissions define the designer access right to all elements of a particular type while the specific permissions define the designer access right to a specific design element.

Revision control refers to the ability to return to any earlier state of the design, for cases in which an engineering dead-end was reached in the development of the design or for tracking of changes made by the users. The product model structure provides the ability to track and filter the changes made upon the model. Each element in the product model holds its history changes since its creation. It contains the changes of the element data, the changing time, and who did the changes. This revision control mechanism of the product model allows the system to retrieve any earlier state of the model which can be very useful in various situations such as file corruption or design conflict state.

5.2.5. The Action Model

The agents in the system communicate through messages and respond by taking the appropriate actions. The actions in the system can be divided into two groups. The first group contains those that will change the product model (e.g. Add column). The second group contains those that are irrelevant to the product model (e.g. User login). An action, in the system, refers to the execution of a set of commands.

The Action Model simply stores all actions that affect the product model. The Mediator Agent and the Product Model Agent can then inspect the latest changes and determine what action(s) to execute on both the client and server side to ensure integrity. This is done in accordance with the procedure described above under the Communication Model.

The actions in the Action Model will be used by both the Mediator Agents and the Product Model Agent. The Mediator Agent will apply the actions on the local copy of the model whilst the Product Model Agent will apply them on the shared model. The actions only contain the commands that change the product model; the validation of the input that will affect the product model is done before submitting the data to the Product Model Agent as it is seen on the Communication Model Diagram (Figure 5-3). The Product Model Agent will not attempt to validate the user input data prior the actions execution as they would be already validated on the client side.

Isolate and separate the execution commands from the agent structure and the product model structure and form them in a set of actions has the following advantages:

- The actions will be used by both the Mediator Agents and the Product Model Agent, so actions will be written once.
- Future modification on the actions will not affect the agent structure or the product model structure.
- The actions data can be stored in a separate file or storage which is much smaller than the product model data and it can be used for various purposes such as restore the model to any pervious state by re-executing the actions in sequence.

Figure 5-6 shows the sequence of the interaction between the internal components of an agent until an action is executed. The execution of the action itself is an execution of a list of statements (i.e. commands). If the action is a model action, its statements will modify the product model data.

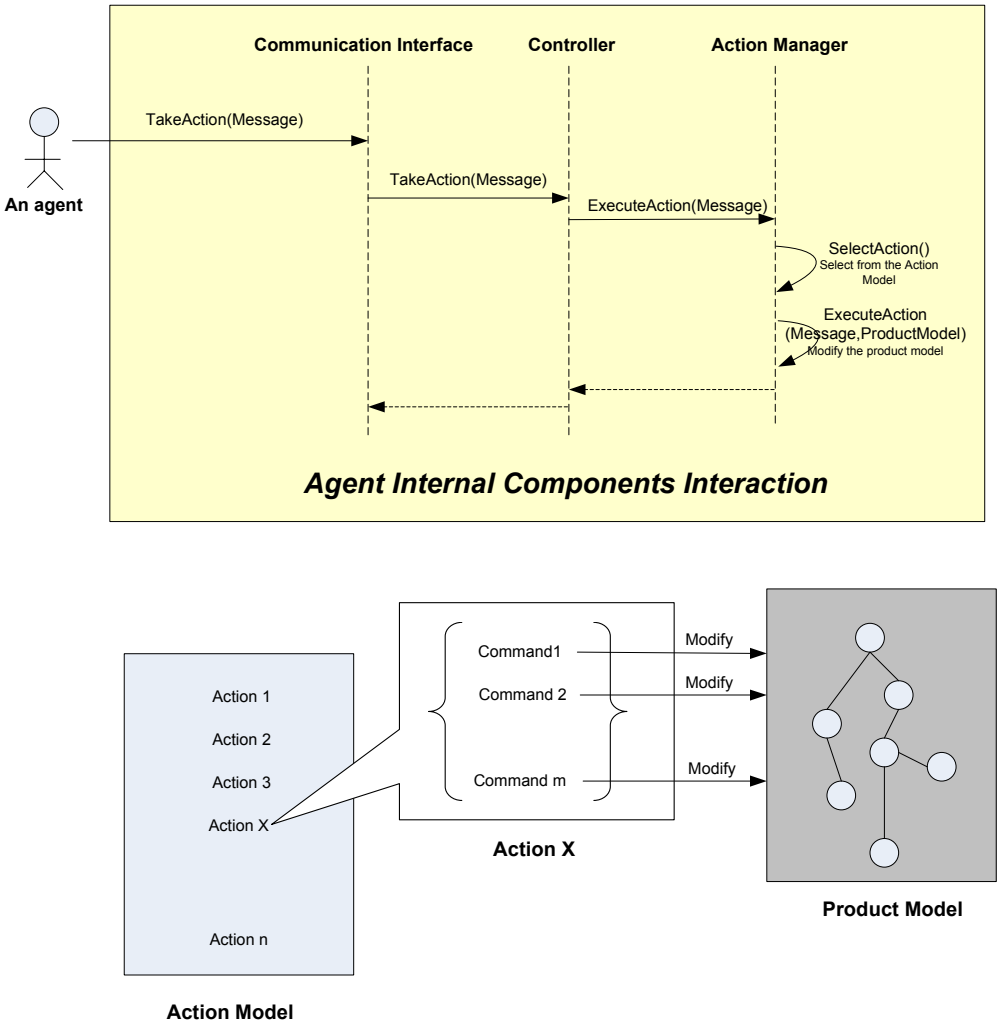


Figure 5-6: Internal process of an action execution

5.2.6. The 3D VR Model

The graphical representation of the design data (i.e. the product model) is central for engineering design systems. It is necessary for the 3D model to be well-linked to the

product model so that the designers are able to handle the product model through the interaction with the graphical model. Figure 5-7 shows the links between the two models.

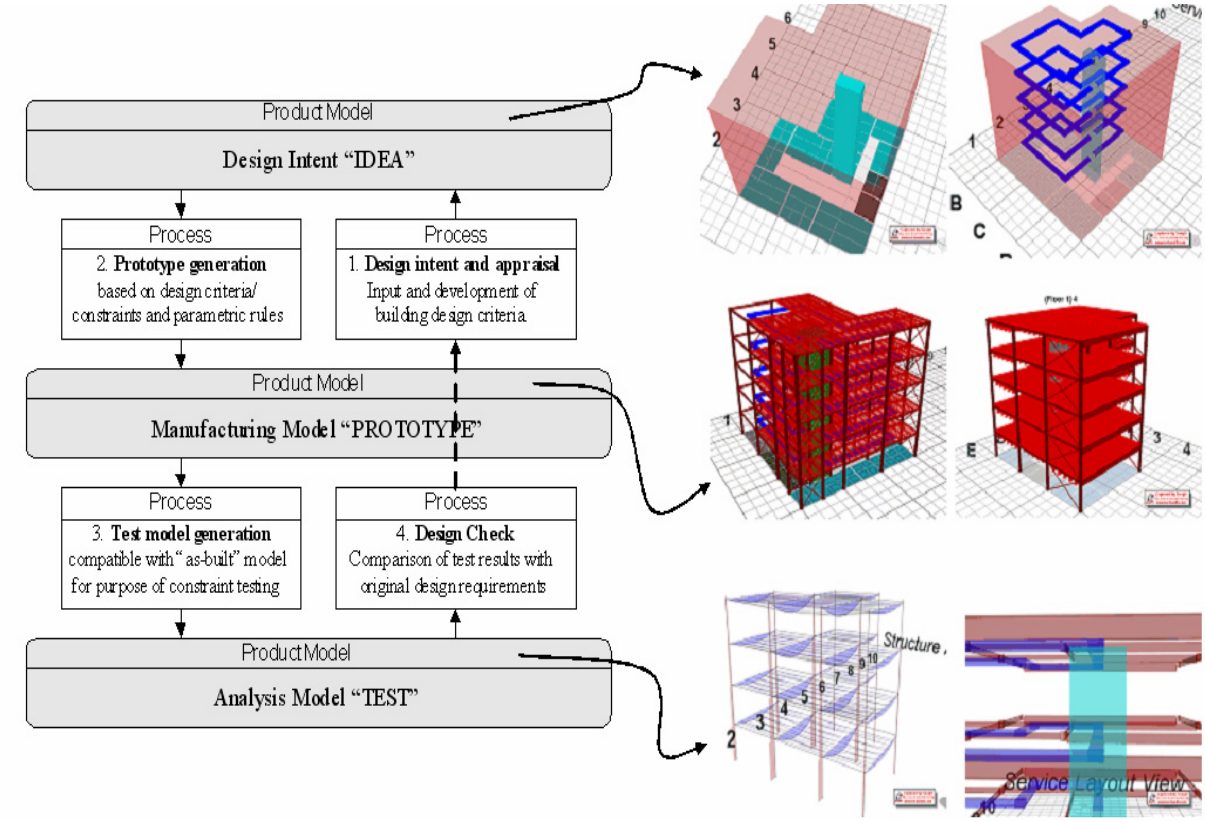


Figure 5-7: The product model tiers and the VR representations

The VR Model consists of the graphical representation of the product model element's types. It is considered that the graphical representation model should be able to show the whole model details and to be easily navigated and manipulated.

5.3. System Implementation

The system design described earlier through the following models: the Agent Model, the Communication Model, the Collaboration Model, the Product Model, the Action Model,

and the VR Model aimed at simplifying the system architecture description. The next step in the system development process is the implementation. The implementation of the system is divided into three stages (Figure 5-8). These stages are:

1. Implementing a generic agent to encapsulate all common functions expressed by all agents.
2. Implementing an agent framework. This is the system specialist agents and the communication infrastructure to facilitate the interaction among the agents.
3. Implementing the rest of the application components (i.e. the Product Model, the Action Model, and the VR Model).

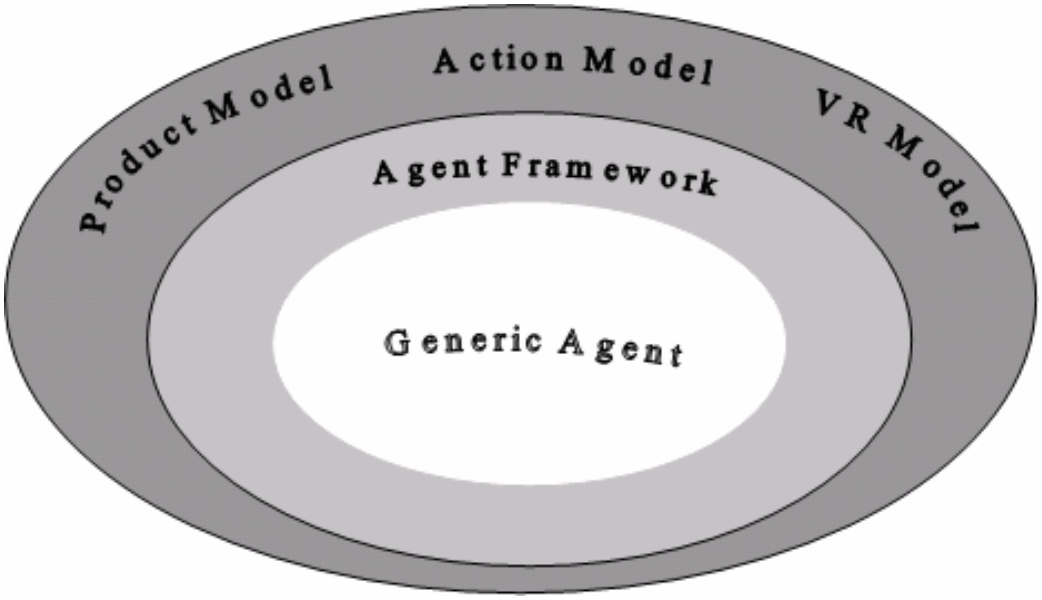


Figure 5-8: The system main components

5.3.1. The Adopted Technologies

The selection of the appropriate development tools, frameworks, or the programming languages before implementing an agent-based system is very important as for any IT system. There are many available products that can be used to develop such systems. These products can be divided into two categories. These are traditional object-oriented programming languages and frameworks or tools that have already been developed to easy the implementation of the system (Shen et al., 2001).

With traditional object-oriented programming languages, such as C++ and Java, the agents are usually modelled as comprehensive objects, with communication among agents being by message passing. This approach is suitable to implement a proof-of-concept prototype or when flexibility in constructing the system agents is needed (Shen et al., 2001). On the other hand, it is time consuming approach as the main agent framework need to be first built and a high programming experience is required.

Frameworks that have already been developed to easy the implementation of multi-agent systems (e.g. AgentBuilder and Zeus) usually offer less development time and they require less programming experience. However, it is difficult to find the appropriate package and they have their own limitations which will impose undesirable constraints on the system (Shen et al., 2001).

Although the object-oriented approach is time consuming and requires a very good programming experience, a decision was taken to use this approach for the following reasons:

- It offers more flexibility in constructing the system entities.
- The aim of the proposed system is to provide a proof of concept and it suits such applications.
- Adopting existing agent tools would impose undesirable constraints on the system such as integrating the 3D interface with the framework and reduce flexibility in constructing the system entities.
- The incompleteness of such tools and the lack of sufficient documentation.
- Most of the tools built to develop multi-agent systems are not comprehensive.
- The communication among agents in the proposed system is mainly straightforward. This limits the need for an already developed agent implementation package that is usually built to support the implementation of complex communication and negotiation approaches among agents.

5.3.1.1. .NET Technology

Programming history witnessed hundreds of programming languages. Each has been developed with its relative strengths and unique features. In this research, the desire was to investigate some of modern technologies that were developed with a potential to support the implementation of distributed systems.

The selection of the implementation technology was to use Microsoft .NET Framework. Microsoft .NET Framework is probably the most sophisticated modern technology that emerged in the last few years. The framework is a new computing platform that simplifies

application development in the highly distributed environments (.NET Framework, 2000). It is also provides an integrated development environment that has made it possible to build an efficient, reliable, and easy to program agent framework for the development of enterprise agent-based systems (Grosso et al., 2003).

.NET Framework offers the following advantages (.NET Framework, 2000):

- Provide a consistent object-oriented programming environment.
- Provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- Make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- Provide access to technologies that enable developers to build distributed systems for all various applications using a range of technologies such as ASP.NET, Windows Forms, .NET Remoting, Network Communication and XML.
- Build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code such as XML.
- Include many programming languages (Visual Basic .NET, Visual C#, and Visual C++) with the ability to write the programme code by more than of language which provides the programmer with more flexibility to choose what suit them most.
- Provide comprehensive documentation of the framework and huge resources and demonstration applications.

The programming languages that can be used with the .NET technology are C# (pronounced C sharp), Visual Basic, and C++. C# is the latest language in the C family of languages. C# is a modern, simple, object-oriented language that inherits its features from C++, and Java languages. C# is the main language for .Net framework and it is designed to combine the raw power of C++ with the high productivity of Visual Basic.

In this work, the programming languages that have been used for developing the proposed system are C# and C++.C# is used to develop the application interface and the agent framework while C++ is used alongside OpenGL to develop the 3D interface.

Although the system is implemented using .NET which imposes some limitations such as restricting the use of the system to Window Operating Systems, the system design is abstract and it is not influenced by the information technology used. In other words, any other implementation tool that can achieve the system design requirements is a candidate for the system implementation.

5.3.2. The Generic Agent Implementation

The main purpose of the implementation of the Generic Agent was to wrap all the common functions among the system agents. This will make any future modification on the common structure of the agents easier. The system agents are then extended to implement the required features. The profile of an agent was implemented in the ***CGenericAgent*** class. Figure 5-9 shows the static diagram of the Generic Agent.

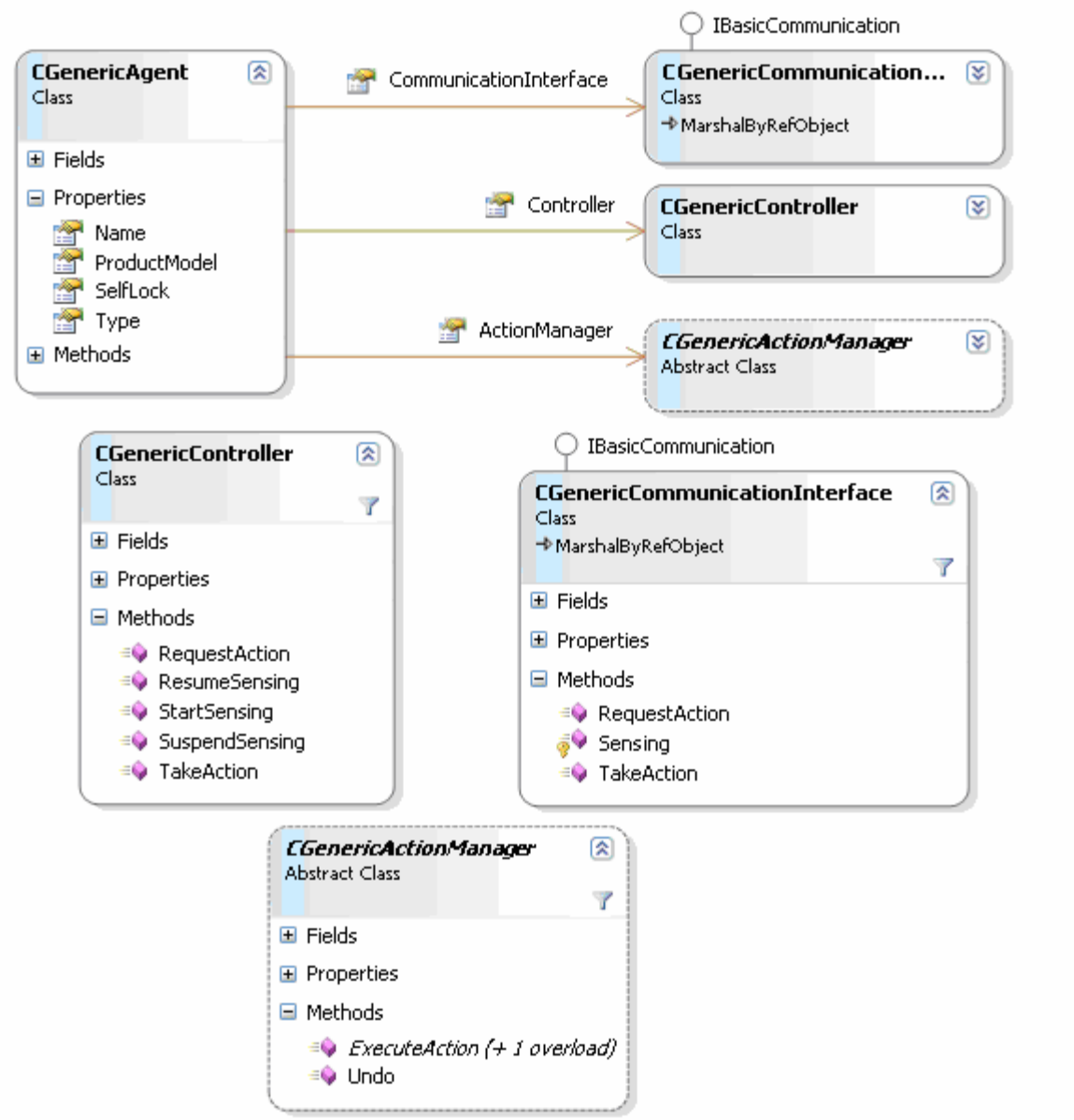


Figure 5-9: Static diagram of the Generic Agent structure

The main components of *CGenericAgent* class are: *CGenericCommincationInterface*, *CGenericActionManager*, and *CGenericController*.

CGenericCommincationInterface class allows the agent to communicate with the other agents in the systems. It has three main functions (methods), namely, *RequestAction*, *Sensing*, and *TakeAction*. *RequestAction* method is called by the agent controller when the agent cannot perform an action by itself and needs help from others. *Sensing* method allows the agent to continuously keep sensing the environment for any changes. *TakeAction* method is used by the agent acquaintances to request an action from the agent. This method then forwards the request to the controller to decide what to do next. *CGenericController* has two main methods: *RequestAction* and *TakeAction*. *RequestAction* method is called when the controller agent needs help from other. It forwards the request to the communication interface to communicate with the appropriate agent. *TakeAction* method allows the agent to take the appropriate action. It decides either to forward the request to the communication interface by calling *RequestAction* method or to forward it to the action manager *CGenericActionManager*. *CGenericActionManager* class has one main method; which is *ExecuteAction*. *ExecuteAction* is called when the agent controller asks the action manager to perform a certain action. Figure 5-10 summarises how the agent components interact.

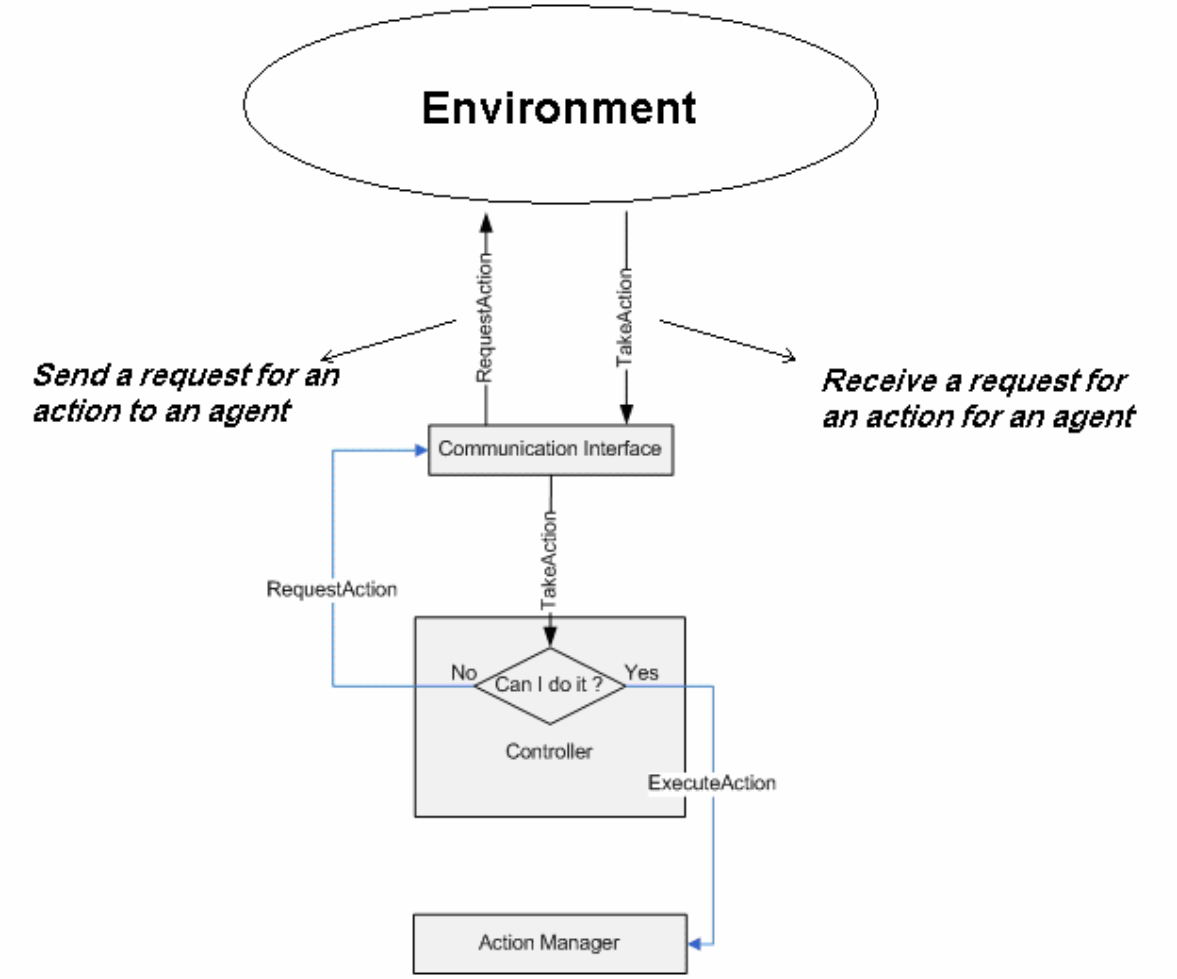


Figure 5-10: The interaction among the agent components

5.3.3. The Agent Framework Implementation

The Generic Agent is developed to wrap the common functions of the system agents. All the system agents are then derived from the Generic Agents. The agents communicate through their communication interfaces (i.e. *CGenericCommunicationInterface class*). Each agent in the system stores the addresses of all agents that it may need to communicate with. The communication channels with other agent represent a part of the social knowledge of the agent about the others. *TakeAction* method of *CGenericCommunicationInterface* class

of the *Generic Agent* is the method that is used by other agents to communicate any message with the agent (i.e. the communication channel).

5.3.3.1. The System Agents

The current implementation of the system includes five specialist agents (Figure 5-11). These are the Model Agent, the Interface Agent, the Structural Analysis Agent, the Design Agent, and the Mediator Agent.

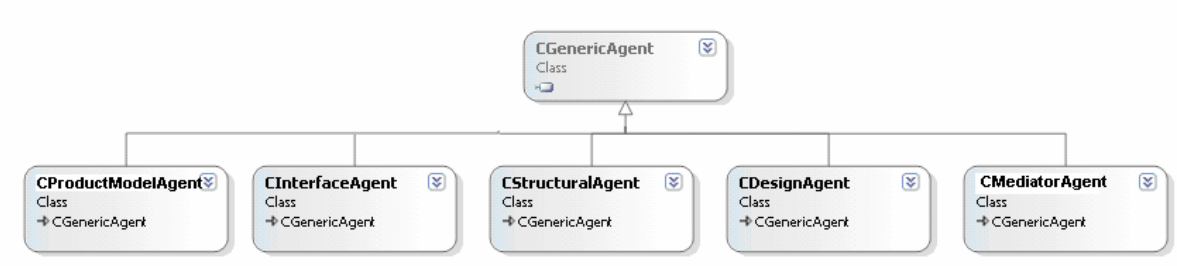


Figure 5-11: System agents

Below is the implementation description of each agent.

5.3.3.1.1. The Product Model Agent:

The agent responsible of maintaining the shared product model and it is located on the server. The profile of this agent is implemented in *CProductModelAgent* class (Figure 5-12).

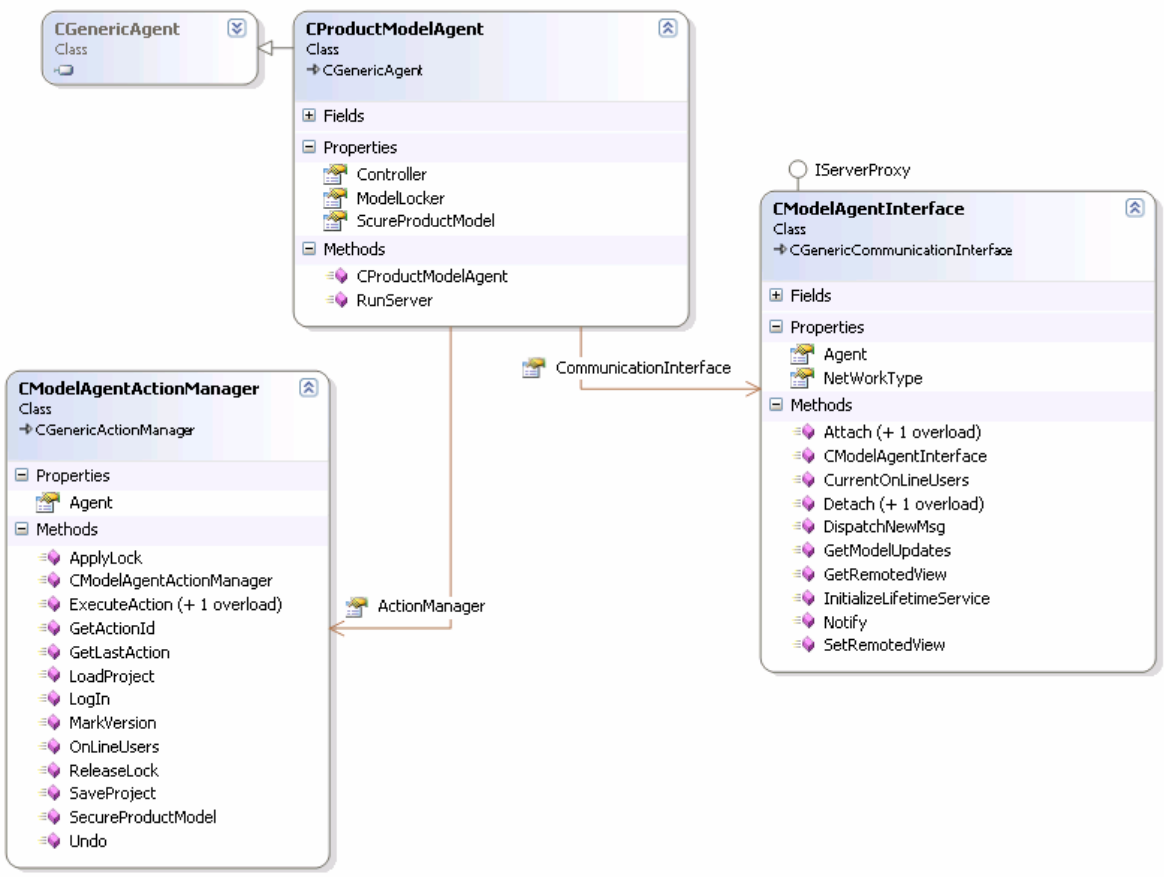


Figure 5-12: Static diagram of the Product Model Agent structure

The main functions of the Product Model Agent are:

- Update the shared product model on the server. When a user applies an action on the local product model, the Mediator Agent communicates with the Product Model Agent to update the shared model accordingly. The action manager of the Product Model Agent (i.e. *CProductModelAgentActionManager* class) will then update the shared product model by executing the proper action using *ExecuteAction* method.
- Facilitate communication among the users. When a user log into the system, the Product Model Agent run *Attach* method to add that user to the available users and

when a user leave the system, the agent run *Detach* method to remove the user from the available user list. An online user can see all other online users and communicate with then instantly. *Notify* method of the Product Model Agent routes any message from an online user to the others. This allows users to interact in synchronous manner. There is a similar functionality implemented that allows a user to remote the graphical interface to the others to enhance the communication. *SetRemoteView* and *GetRemoteView* are the methods to facilitate the graphical views exchanges.

- Locking and unlocking the product model when necessary. If a user has to do a one-off larger modification on the model, he can lock the model till he finishes the modification. *ApplyLock* and *ReleaseLock* are the methods to perform the locking/unlocking mechanism. When the model is lock to a user, the other users can only view the shared model but they can not modify it till it is released.

5.3.3.1.2. The Mediator Agent:

Its main role is to maintain the local copy of the product model and ensure its consistency with the shared model. This agent is located on the client side. The profile of the agent is implemented in *CMediatorAgent* class (Figure 5-13).

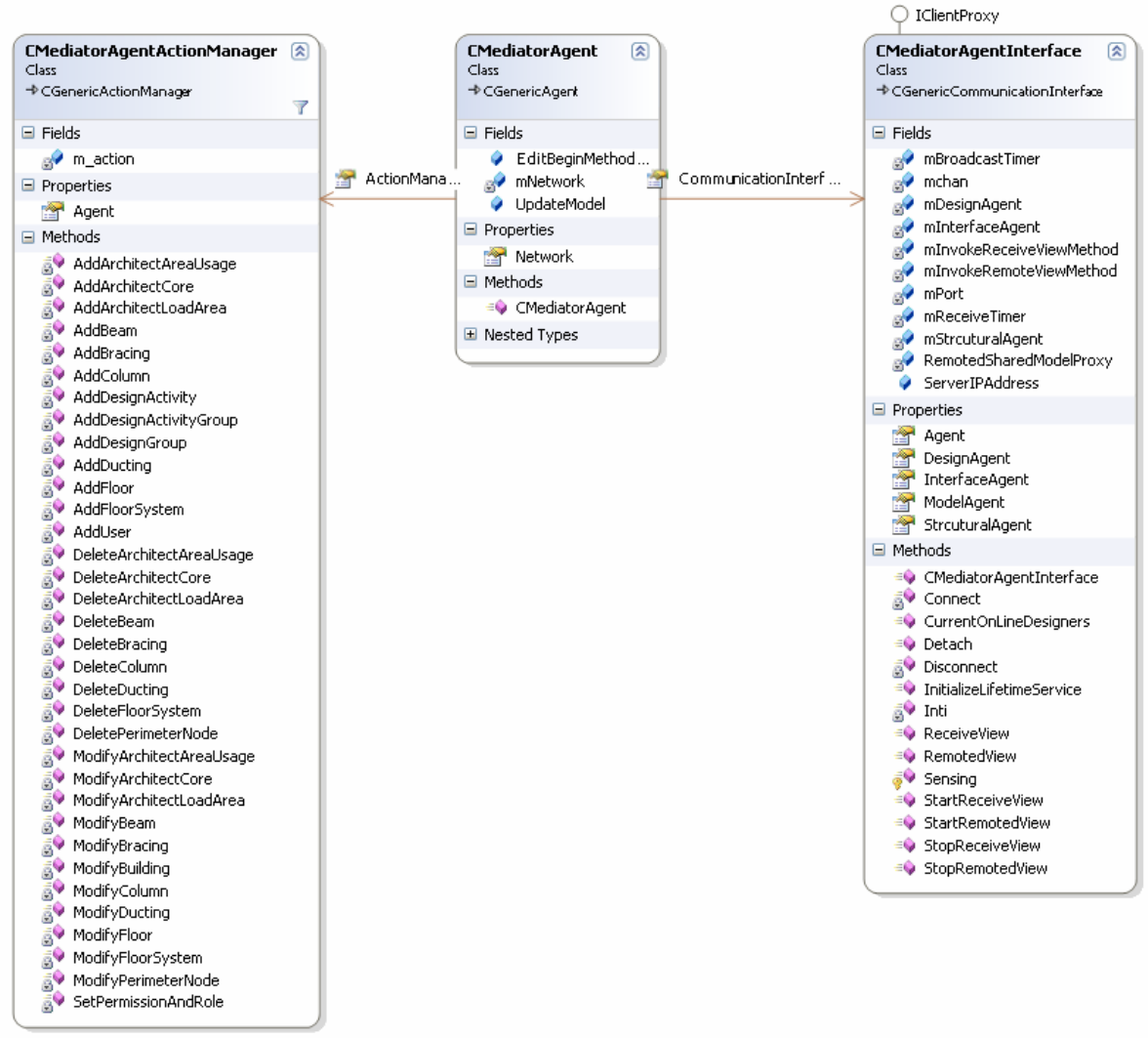


Figure 5-13: Static diagram of the Mediator Agent structure

The main functions of the Mediator Agent are:

- Maintain the local copy of the product model. When a user changes the product model (i.e. add, delete, modify), the changes are passed to the agent to update the local model. The suitable action then is selected by the agent controller and executed by the agent action manager. Figure 5-13 shows *CMediatorAgentActionManager* profile and a

subset of the actions that can be executed by it. *AddColumn* action, for example, is called and executed when a new columns is added to the system.

- Sensing the changes on the shared product model and update the local copy accordingly. *Sensing* method of the agent communication interface, implemented in *CMedatiatorAgentCommunicationInterface*, is live method and it keeps comparing the local model with the shared model all the time. If it finds any differences between the two copies, it then requests the changes from the server. The required changes on the local copy are then passed to the action manager to run the suitable action(s). For example, if a column is added to the shared product model by some other user, the agent's sensing method will recognise that the shared product model is changed and the local copy is out of date so it brings the necessary changes and pass them to the agent controller which in turn will pass them to the action manager which calls and executes an *AddColumn* action as if the command is initiated by the user on the same machine.

5.3.3.1.3. The Interface Agent:

Its main role is to allow the user to interact with the system and visualise the product model and manipulate it intuitively. The agent is located on the client side. The profile of this agent is implemented in *CInterfaceAgent* class (Figure 5-14).

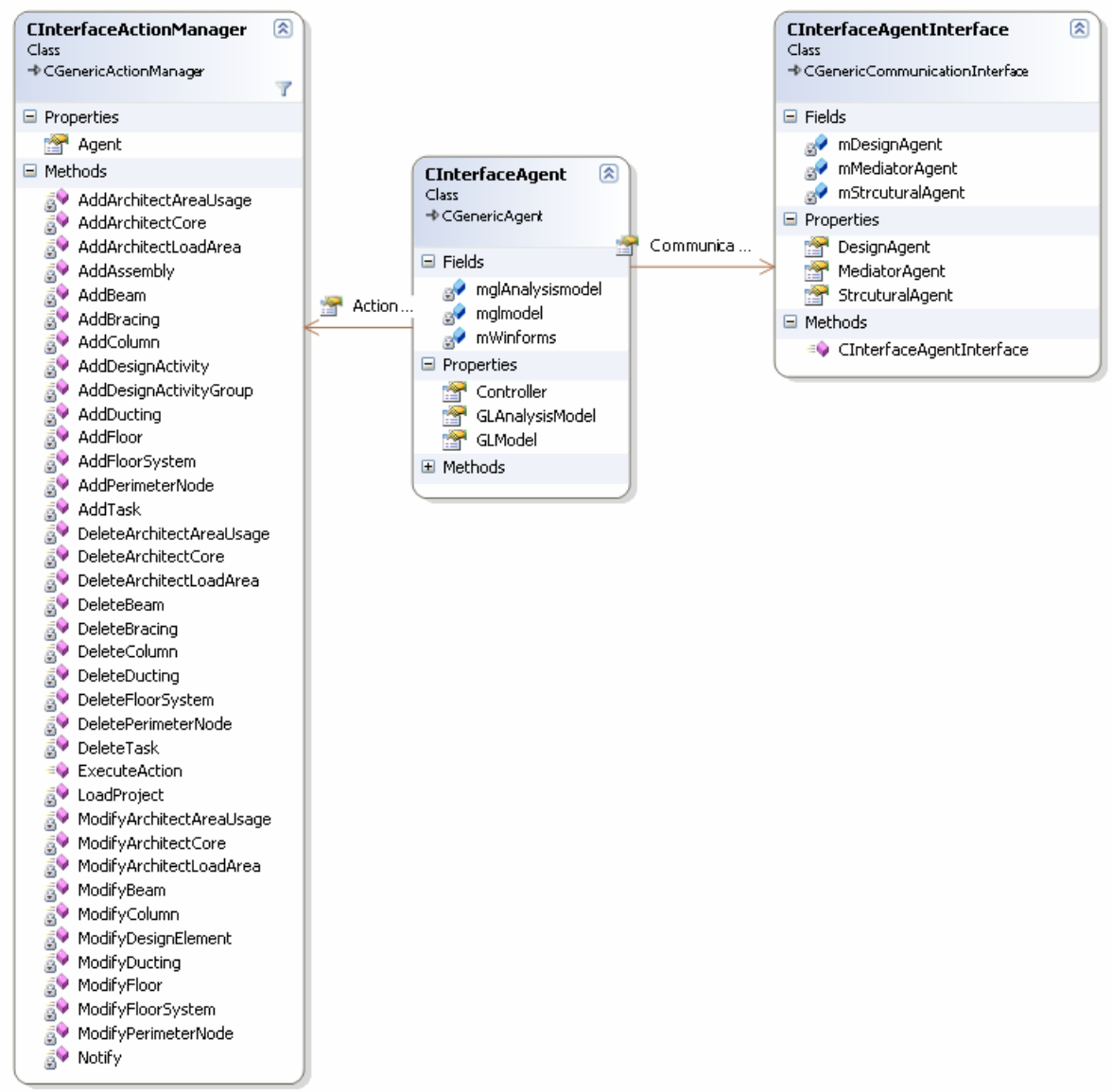


Figure 5-14: Static diagram of the Interface Agent structure

The main functions of the Interface Agent are:

- Communicate the user input with the appropriate agent. The user manipulation of the graphical representation of the product model is translated to the suitable message and passed to the appropriate agent. For example, if the user deletes a column, the agent will

communicate the command with the Mediator Agent which in turn executes an *DeleteColumn* action.

- Visualise the product model. If the local product model is changed and the changes affect the graphical representation of the product model, the agent will update the 3D view accordingly.

5.3.3.1.4. The Design Agent:

Its main role is to suggest an initial design of the structure members and use the structural analysis results to design/check the structure members. The agent is located on the client side. The profile of this agent is implemented in *CDesignAgent* class (Figure 5-15).

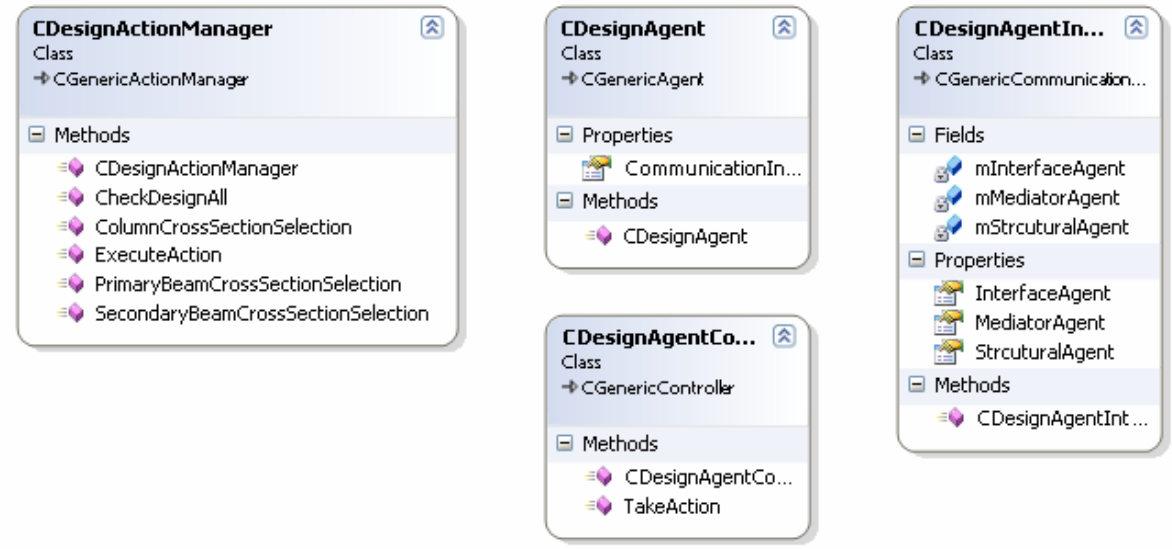


Figure 5-15: Static diagram of the Design Agent structure

The main functions of the Design Agent are:

- Suggest an initial cross section for any new added member. The current implementation provides an initial suggestion of the cross section for columns, primary beams, and secondary beams. The agent current decision capability to choose a cross section is limited to simple rules and for the aforementioned member types. However, It is thought that the level of implementation was enough as a proof of concept and further complex algorithms can be inserted to enhance the agent capability in straightforward way.
- Design or check the structural members based on the result of the structural analysis. The agent checks the design strength of the structural members based on the forces obtained from the analysis. The agent also can re-design the cross section of a member if it finds that it is not sufficient provided the cross section is set to AutoDesign mode by the user.

5.3.3.1.5. The Structural Analysis Agent:

Its main role is to conduct the structural analysis of the structure and check its stability. The agent is located on the client side. The profile of this agent is implemented in *CStrucutralAgent* class (Figure 5-16).

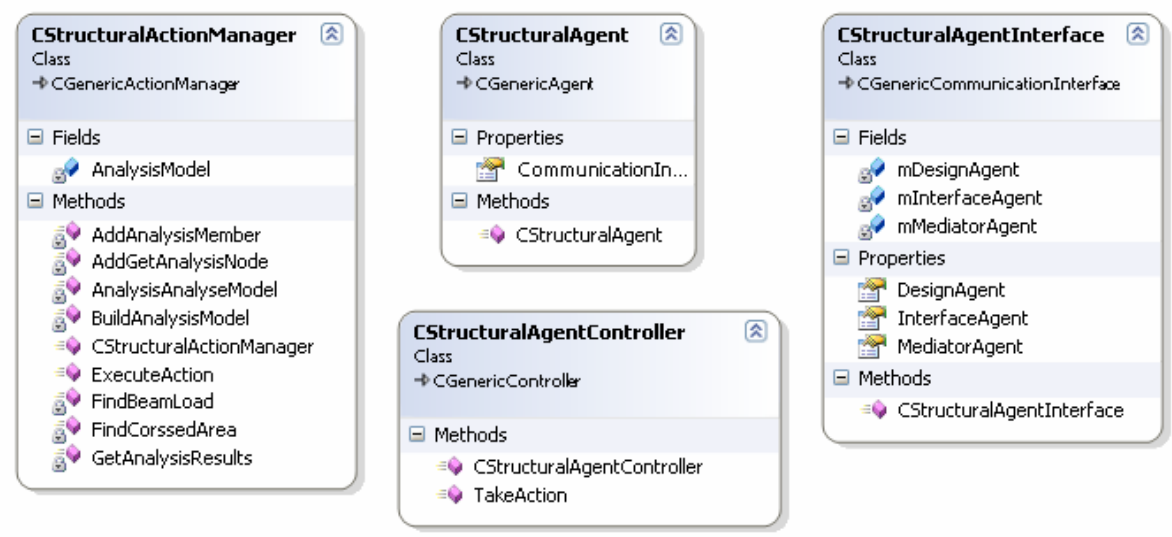


Figure 5-16: Static diagram of the Structural Agent structure

The main functions of the Structural Agent are:

- Generate the structural analysis model of the actual product model. The agent reads the actual product model and generates the suitable structural analysis members, connections, and loading.
- Conduct the structural analysis of the generated structural model and obtain the forces that are used in the design. An already developed 3D structural analysis engine based on the stiffness matrix method is used by the agent to conduct the structural analysis.

5.3.3.2. Inter-Agent Communication

The communication among the agents is carried out by means of exchanging messages by invoking specific methods. The communication interface unit of an agent is composed of several methods for treating all incoming and outgoing messages.The communication language in the system is not based on any known standards such as KQML(Finin et al., 1994). However the developed method was inspired from reviewing such standards.

There are two concepts that are used to facilitate the communication among the system agents. These are: Message and Ontology. Massages exchanged in the system encapsulate the information needed by an agent to execute an action while ontology is used to select the suitable actions to execute. The exchanged messages are expressed in XML format and the ontology is defined as a simple set of vocabularies that describe the associated actions.

```
<Message>
  <Ontology>LogIn</Ontology>
  <Id>d6a39dce-0c24-4592-5cd6</Id>
  <Sender>Admin</Sender>
  <Receiver>ModelAgent</Receiver>
  <DateTime>14/02/2007 13:18:38</DateTime>
  <Content>
    <Username>Admin</Username>
    <Password> 123 </Password>
  </Content >
</Message>
```

Figure 5-17: A message example

```
public  class Ontology
{
    public  const string AddFloor ="AddFloor";
    public  const string ModifyFloor = "ModifyFloor";
    public  const string AddColumn="AddColumn";
    public  const string DeleteColumn="DeleteColumn";
    public  const string ModifyColumn="ModifyColumn";
    public  const string AddBeam="AddBeam";
    public  const string ModifyBeam="ModifyBeam";
    public  const string DeleteBeam="DeleteBeam";
    public const string AddTask = "AddTask";
    public const string AddUser = "AddUser";
    public  const string LogIn="LogIn";
    public  const string SecureModel="SecureModel";
    public  const string ApplyLock="ApplyLock";
    public  const string ReleaseLock="ReleaseLock";
}
```

Figure 5-18: Subset of the ontology vocabularies

All messages in the system are inherited from a generic message class that implements the common features of all messages. The generic message is implemented in *CGenericMessage* class. Figure 5-19 shows the class structure and some sub-type classes of it. When an agent requires an action by another agent, it creates a new instance of the suitable message type and set its attributes and ontology and then passes it to the communication interface which sends the message to the concerned agent.

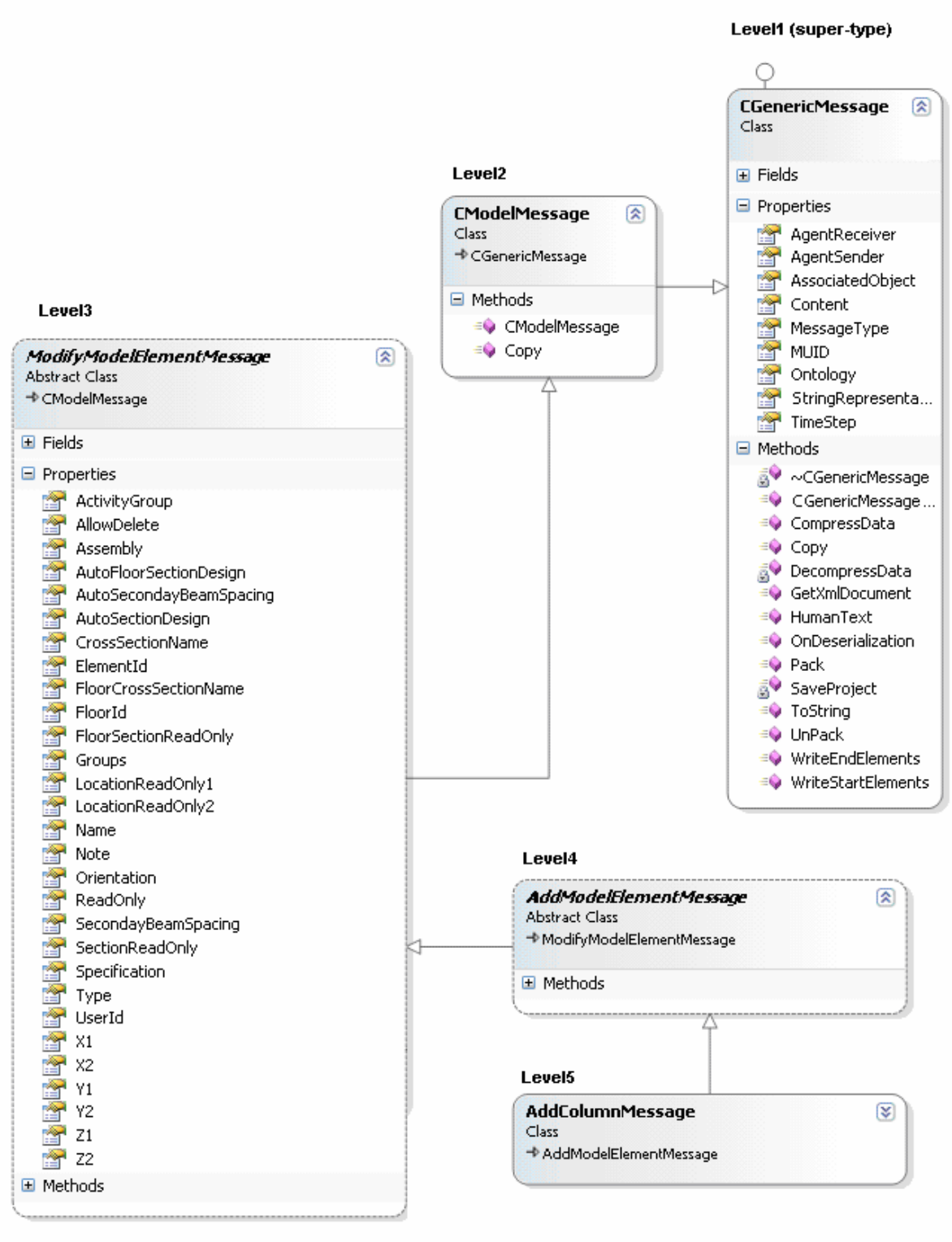


Figure 5-19: *CGenricMessage* and some derivatives

Table 5-2 shows the description of some of the *CGenricMessage* class's properties.

Table 5-2: *CGenericMessage* main properties

Property	**Data Type**	**Description**
MUID	string	Specifies the unique identifier given to the message.
AgentSender	string	Identifies the sender agent by its name.
AgentReceiver	string	Identifies the receiver agent by its name.
Ontology	string	Identifies the message ontology name
MessageType	Enum	{ProductModelRelated, NonProductModelRelated}

In order to achieve effective and robust communication, three important things are taken:

- Adopting XML (eXtensible Mark-up Language) as the method for structuring the messages. XML is chosen for its flexibility and simplicity to structure and represent complex information and the readiness to syntactically and semantically verify an XML document. In addition, XML is a standard language which started to show its performance to build communication languages.
- Packing each message, using *Pack* method of *CGenericMessage* message class, before transferring from the sender to the receiver which improves the system performance. Although this increases the CPU usage on the sender machine but it reduces the transferring time which is a critical factor in developing a distributed application.

- Validating the user input locally on the client machine. This will help reducing the communication with the server if the input data is invalid. The data will be submitted only when it is valid and the Product Model Agent will update the shared model without any further checks on the incoming data.

5.3.4. The Product Model Implementation

During the early stages of this research, an attempt was made to make use of the IFC product model directly by converting it into a C# model. However, the existing systems were unable to convert the IFC schemas into C#, as they could not cope with modelling 'external mappings' in a robust manner. An in-house made tool was therefore developed called IFCSchemaGen that takes EXPRESS schemas as its input, and creates the appropriate C# files to model the EXPRESS entities as C# classes.

Although the resulting product model was found to be very huge and complex, the conversion was successful and the system was able to manipulate IFC2x2 schemas. However, the resulted classes were so huge and generic. For example, in a usual IT system, a 'length' dimension would generally be stored as a single or double variable. Within the IFC model, however, this would be represented using at least three separate objects that structure this information in a highly rigorous, and yet very inefficient manner. In addition, the current IFC model does not support concurrent capability.

The previous reasons were the primary motivations for not using the IFC or CIS/2 models as the internal information representation within the software. This does not reduce the

value of the IFC model, but it means that they are not the best option for internal data representation. In fact, the primary reason of developing IFC or CIS/2 model was to ensure interoperability between incompatible software not to be used as an internal data format.

It was decided, therefore, to develop a product model that suits the requirements. The developed model schemas were mapped to C# classes thereafter. The main focus of the product model implementation, besides storing the data, is to meet the following objectives:

- Allow user to control their inputs by applying different constraint types.
- Trace the history changes of the model elements.
- Preserve a unique identification and ownership of each element.

The implemented model has a clear hierarchical structure. All classes in the model have the prefix "CDesign" to indicate that they are part of the product model. The top root element, called CDesignPointer, is a non-physical element that represents the abstract super-type of all sub-types in the model. Figure 5-20 shows *CDesignPointer* structure. The class has one main data member called *Id*. *Id* is a string variable that represents a unique identification for each instance of the class or any derived class.

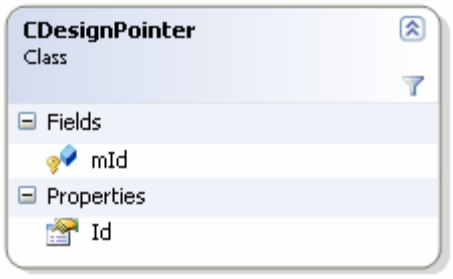


Figure 5-20: *CDesignPointer* Class

The next top root class which is derived from *CDesignPointer* is *CDesignRoot*. This class has one main data member called *Name. Name* is a string data type that may represent the name of the element.

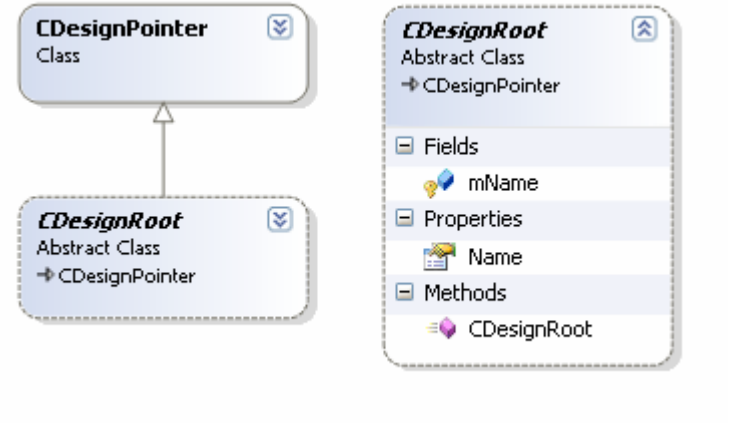


Figure 5-21: *CDesignRoot* class

Moving down in the product model, its structure can be divided into two categories. The first category is related to data representation of the engineering design and the second to the management of the collaboration process.

5.3.4.1. Product Model Data Representation

The classes in the first category are headed by the 'super-type' class: *CDesignProduct*. It represents the root of all physical elements such beams and columns. Figure 5-22 shows the main classes in this category. Comprehensive details of this part of the product model can be found in reference (Smith, 2005).

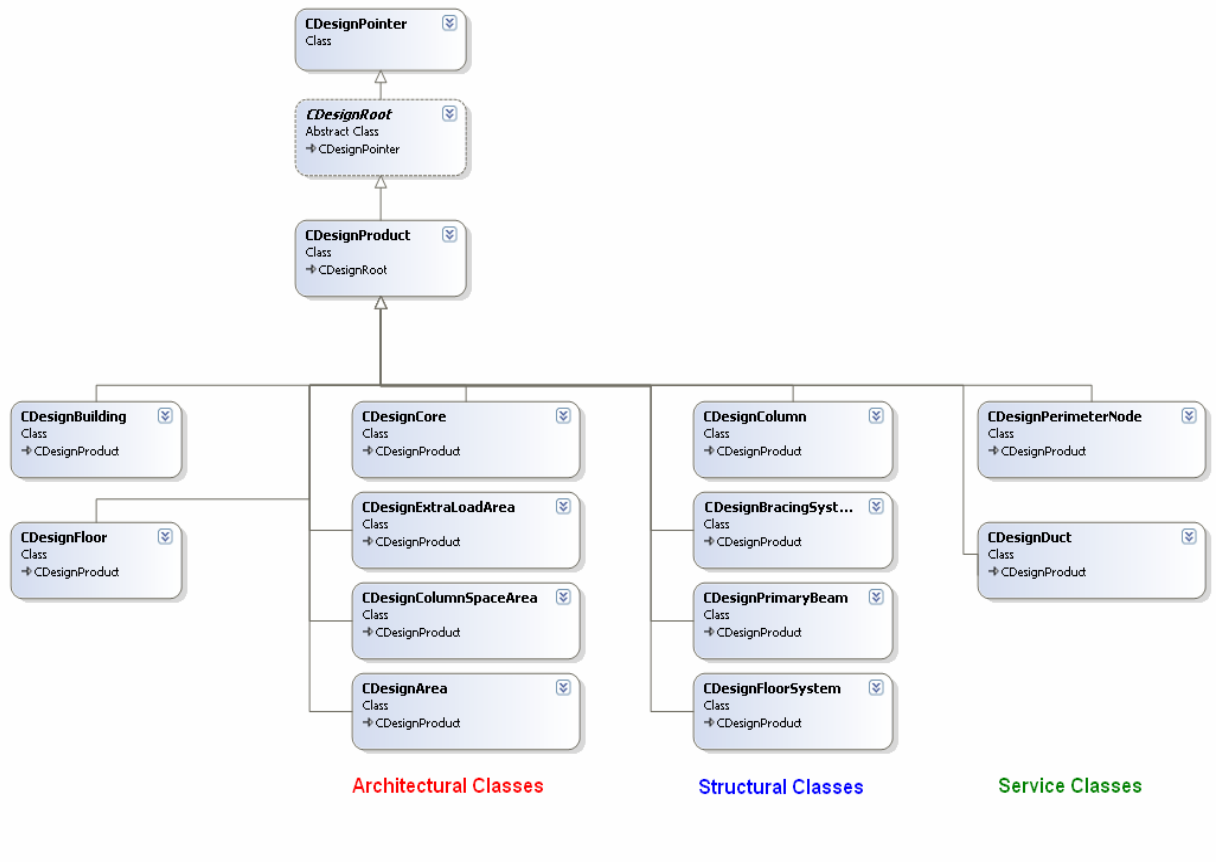


Figure 5-22: Main classes of the original product model

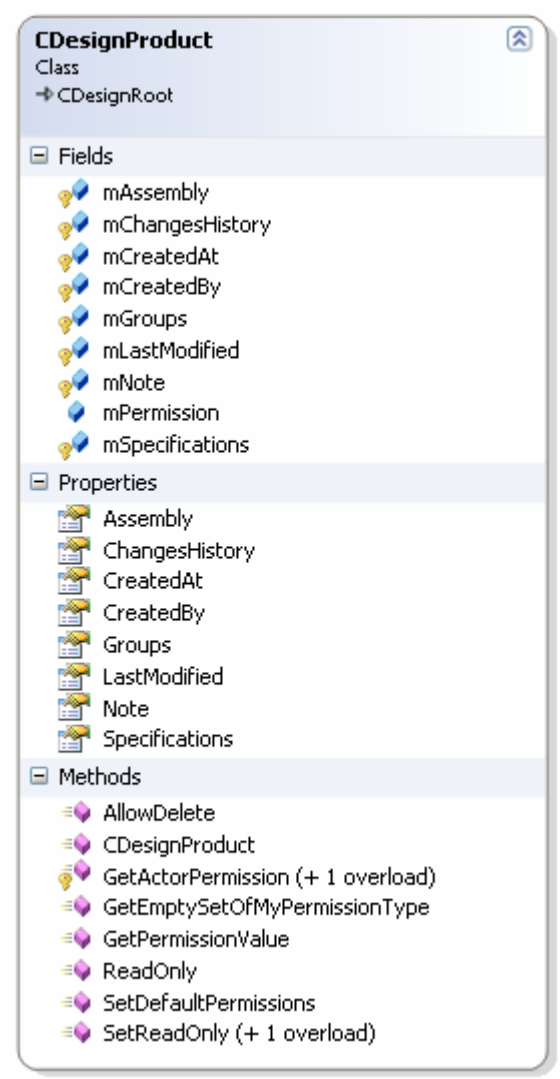


Figure 5-23: *CDesignProduct* attributes

The description of some data members of *CDesignProduct* class are shown in Table 5-3.

Table 5-3: *CDesignProduct* data members' description

Property Name	**Type**	**Description**
CreatedBy	CDesignActor	A pointer to the element owner
CreatedAt	DateTime	Time of element creation
HistoryChanges	List	All changes occurred upon the element
Note	String	User defined note
Specification	String	User defined specification

In addition to the engineering design data, the classes model construction scheduling management. This was done for two purposes: 1) to allow for the structural engineering checking of the parts of the structure as building progresses and 2) to simulate the construction process, albeit with coarser details than traditionally available in the so called 4D models (Dawood et al., 2003). This information can also be used to inspect the structural engineering relationships between connected elements. The planning information is implemented in three classes: *CDesignActivity* and *CDesignActivityGroup* and *CDesignAssembly*. *CDesignActivity* class represents a single activity. It is described by a start time, a finish time, and a status. The status can be one of the following: pre-active, active, post-active. When the date is before the start date of the activity the status is pre-active. When the date is between the start date and the finish date the status is active. Finally, when the date is after the finish date of the activity the status is post-active. *CDesignActivityGroup* class is a collection of *CDesignActivity*. It is also described by a

start time, a finish time, and a status with some differences from *CDesignActivity* as to the meaning of the attributes. These are calculated automatically based on the included activities. The start time is the start time of the earliest activity in the collection and the finish time is the finish time of the last activity in the collection. The status is pre-active if all the activities in the collection are pre-active and active if an activity in the collection is active and post-active if all activities are post-active. *CDesignAssembly* represents a group of design elements which can be checked separately against the structural constraints and requirements. Figure 5-24 shows how the planning classes are related to the 3D design elements.

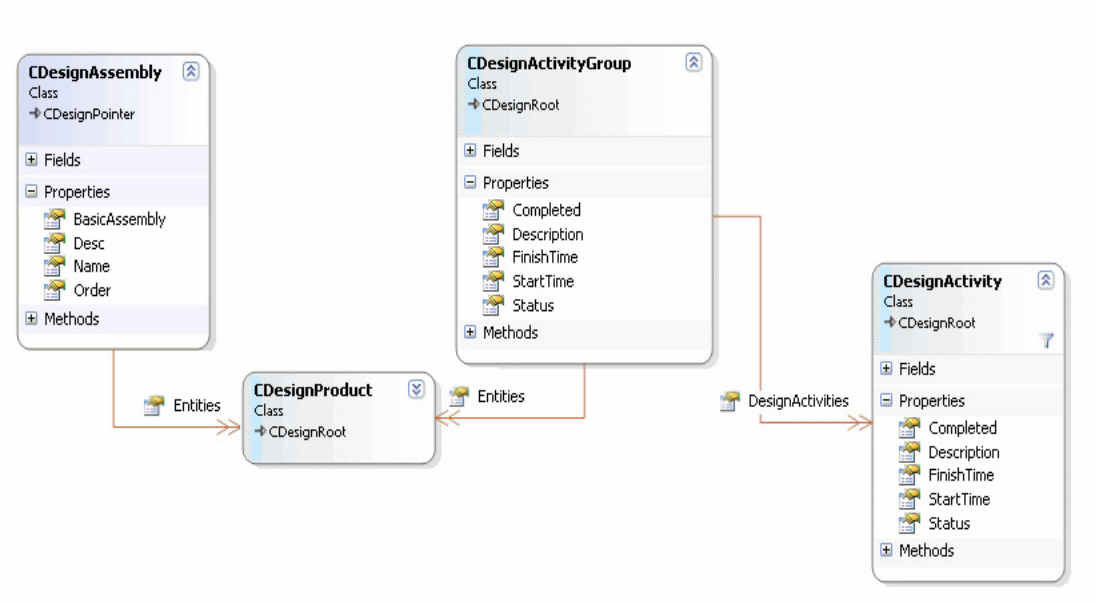


Figure 5-24: *CDesignActivity* and *CDesignActivityGroup* diagram

5.3.4.2. Product Model Data Management

The second category of the Product Model deals with the management of the collaborative process. This part of the model together with some of the Agents Model processes controls the shared access to the product model and manages user access and model versioning. Access and revision control mechanisms as represented in the model are described next.

- **Access control mechanism**

The access control mechanism includes three concepts to manage the access rights to the model data. These are Actors, Permissions, and Roles.

Access control is an essential part of any collaborative environment to protect data from unauthorised users. It is important to determine the requirements for data access control and relate them to the product model structure. Users within the collaborative process must be identified by properly established access control mechanisms before access is granted or authorisation is issued. The primary objectives of access control in collaborative environments are the preservation and protection of information and the management of roles and responsibilities. This is so that each collaborator will be clear about what part of the model or the design he is responsible for and how is this related to others. Access rights can therefore vary from read only to part or the whole model, to full rights to part or the whole model and a mixture of the two.

Below are the descriptions of the main concepts of the access control mechanism (i.e. actor, permission, and role).

- *CDesignActor* class represents an actor in the system. An Actor represents a single designer or a group of designers. Actors will have default granted general permissions based on their assigned roles. *CDesignActor* has two sub-type classes (Figure 5-25). These are *CDesignUser* and *CDesignGroup*. *CDesignUser* represents a single designer while *CDesignGroup* represents a set of designers. Permissions can be granted individually or by groups.

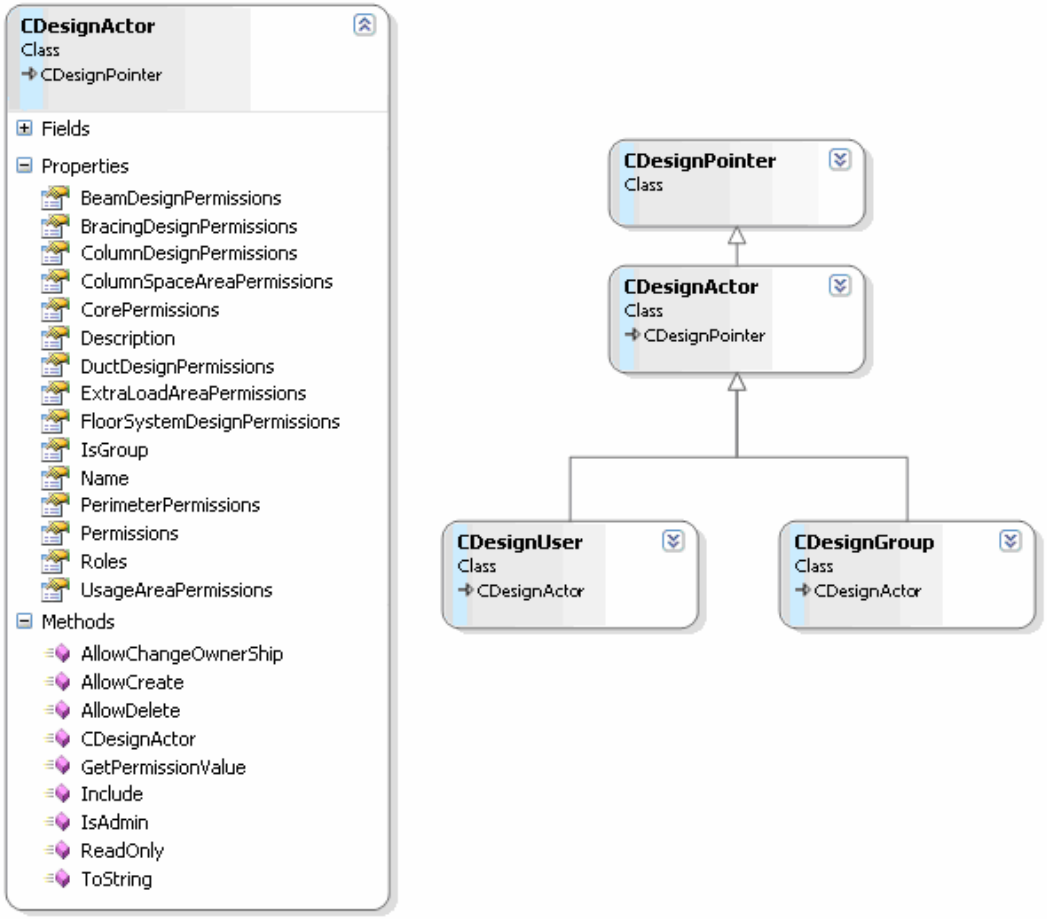


Figure 5-25: *CDesignActor* Class

- *CDesignBasicPermission* class represents the super-type of all permissions kinds. The permission concept is the key control concept regulating access rights of actors to any part of the product model. Permissions in the system can be seen as the link between the engineering data representation and the information required to collaboration management. Figure 5-26 shows the schematic view of the relationships between the model elements, the actors, and the permissions.

Figure 5-26: Relationship between *CDesignActor*, *CDesignPermission* and *CDesignProduct*

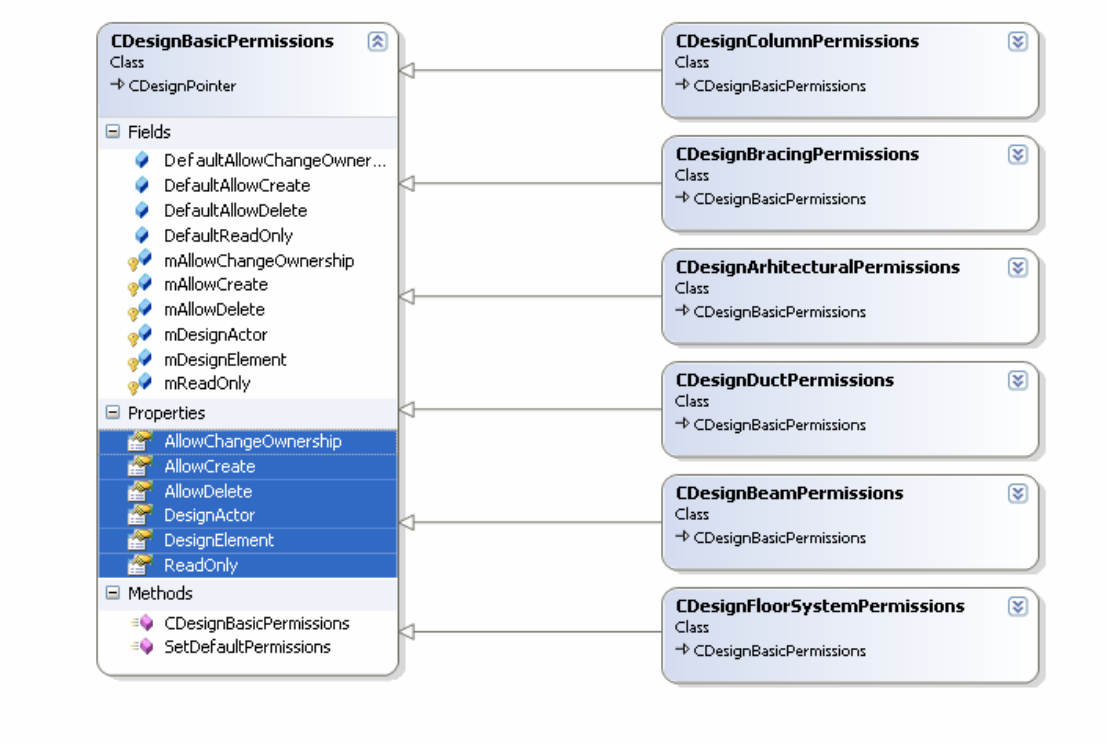


Figure 5-27: Examples of sub-type classes of *CDesignBasicPermissions* class

There are generically four types of permissions: allow modify, allow delete, allow create, and allow change ownership. The permissions can be classified on two levels; general or specific. The access rights can be granted by type-level, element-level and attribute-level too. For example, if the architect is to define the column positions, he can add the columns to the building and apply read-only permission on them but at the same time leave the cross sections free to be edit by the structural engineer. The general permissions define the actor right to access all elements of a particular type (i.e. type-level). The general permissions to the product model elements can be seen as role-base permission. It is the default level used in the system to grand access to the product model. The general permissions are defined by the system administrator.

When a new element is added to the model, the system will automatically create a set of permissions equal to the number of the actors with default access based on each actor role.

Permissions are then linked to both the actors and the new added element. Similarly, if a new actor is added, the system will automatically create a set of permissions equal to the number of the elements with a default access based on the new actor role. On the other hand, the specific permissions are element-based permissions (i.e. element-level). The specific permissions define the actor access right to a specific product model element. The permissions are not controlled by the system administrator; they are defined by the owners of the resources. The owners can adjust the default general permissions of the other actors to access their resources. The default owner of an element is its creator.

For the sake of access right implementation, a special One-to-One and One-to-Many relationships mechanism has been implemented to link the product model elements, the permissions, and the actors.

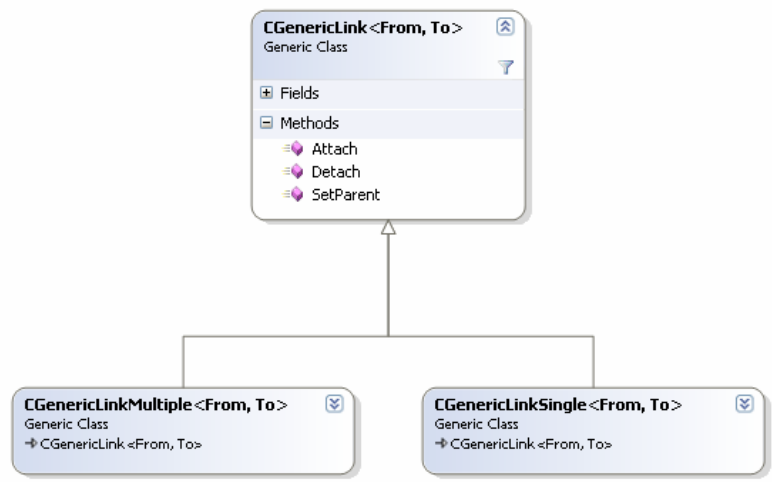


Figure 5-28 : One-to-One and One-to-Many relationship classes

Figure 5-29 shows the permission class and its link to the actors and the product model. Each permission object created will be linked to one element and one actor using one-to-one relationship. Figure 5-30 shows a product model element and its link to permissions. Each element object created will be linked to a list of permissions equal to the actor number using a one-to-many relationship. Similar to element object, each actor object created will be linked to a list of permissions equal to the element number using a one-to-many relationship (Figure 5-31).

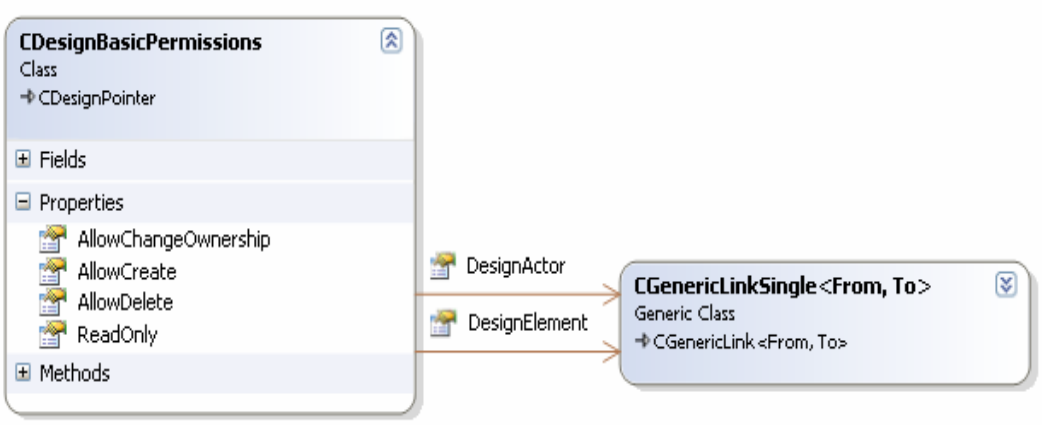


Figure 5-29 : *CDesignBasicPermissions* class relationship with actors and product model elements

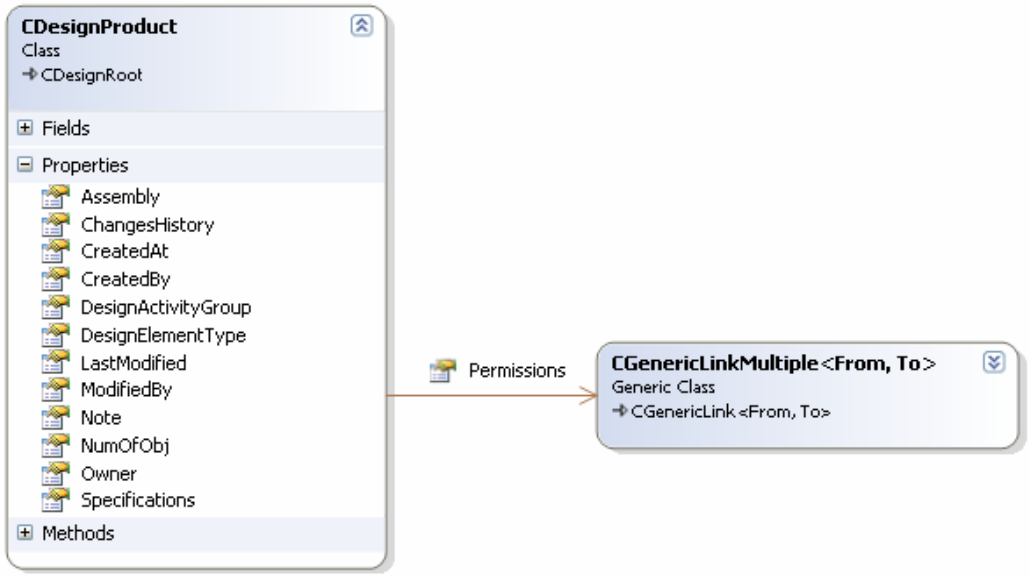
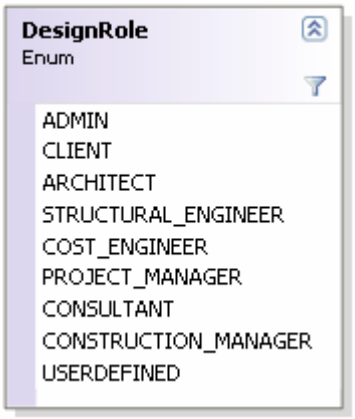


Figure 5-30: *CDesignProduct* class relationship with permissions

Figure 5-31: *CDesignActor* class relationship with permissions

- The Role concept is used to allow the system to grant default permissions to the designer(s) to access the product model elements based on their roles. However, these permissions can be altered afterward. For example, by default project managers will have full access rights and clients will have read only rights.



- **Revision control mechanism**

The revision control mechanism provides two revision features. These are:

  - Review the changes history of any design element. The product model structure provides the ability to track and filter the changes made upon the model. Each element in the product model holds its own changes history since its creation. Each record in the changes history contains the changes on the element data, the changing time, and who did the changes. The changes history attribute of the *CDesignProduct* is a list type that stores all changes occurred to each design element since its creation. The data stored in the list are the messages that are used in the actions upon the design element. To ease the process of reviewing the changes, *CGenericMessage* class (Figure 5-32) has a special method (*HumanText*) used to convert XML message to human readable representation.

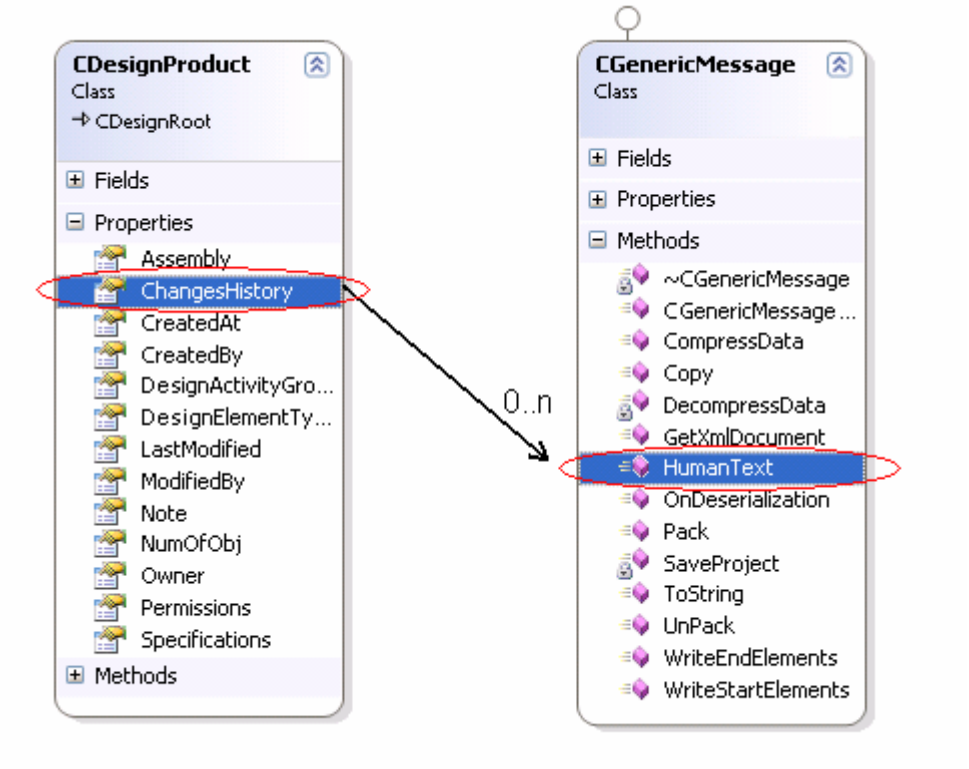


Figure 5-32: Changes history of *CDesignProduct*

- The second feature of the revision mechanism is the ability to return to any earlier state of the design, for cases in which engineering dead-end was reached in the development of the design. The implementation of this mechanism allows the system to move backward and then forward in the history of the product model. Figure 5-33 shows the classes of the product model that are used to store the actions used in the restoration process. When the product model is marked with a version number at each significant milestone in the design process, each unmarked action from the last version will be marked by the new version number and stored in a new *CDesignVersionNode* class instance.

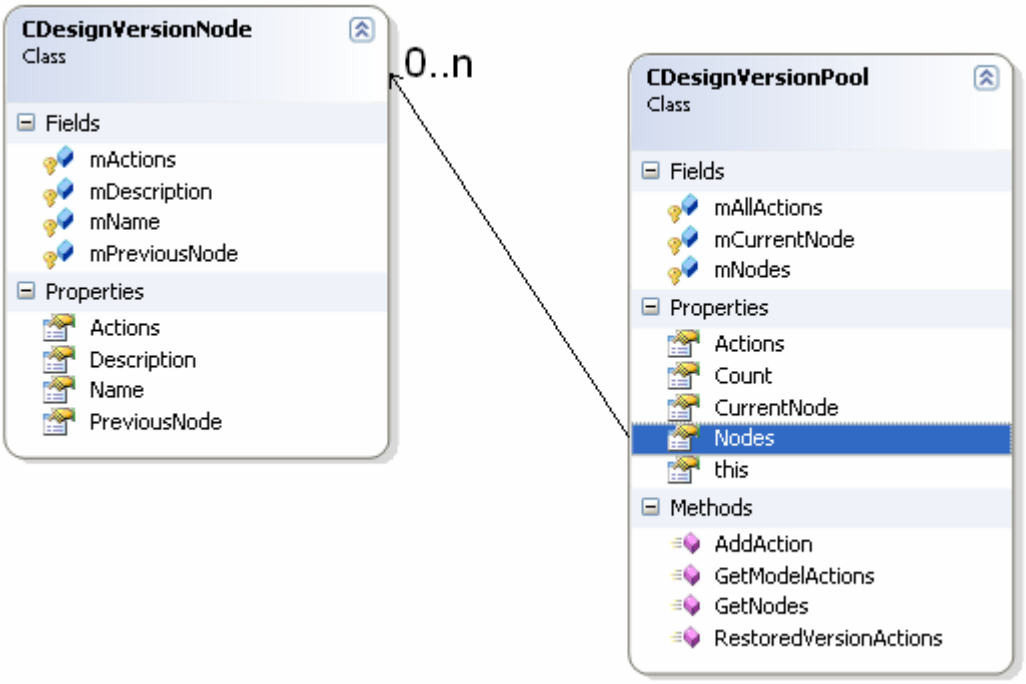


Figure 5-33: *CDesignVersionNode* and *CDesignVersionPool*

When the system carries out the restoration process, it will re-execute the actions in sequence up to the desired version. However, this process will not delete any previous stored actions that occurred after the restoration point. This allows the system to move backward and then forward in the history of the product model. As an example, Figure 5-34 shows an illustration of a tree form of stored versions. Each node in this tree is a version that holds a sub set of the actions occurred during the product model life. If the system is to restore the model up to the Node5, it will re-execute the actions in nodes 1, 2, and 5 only.

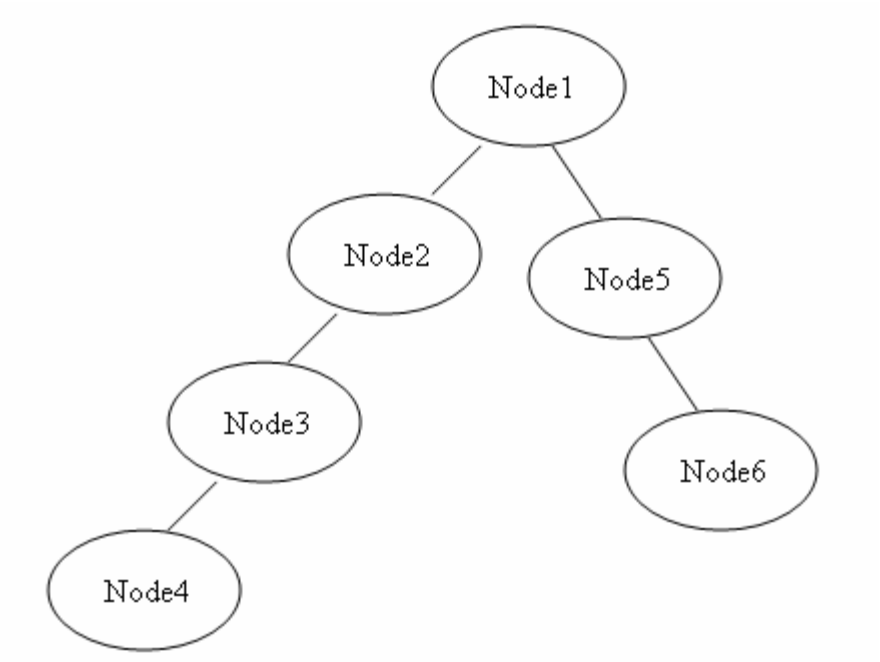


Figure 5-34: Version node tree

The actions data can be stored in a separate file that is much smaller than the actual product model database which makes it more portable. The system then can easily generate a product model from an actions file by executing the stored actions in sequence.

5.3.5. The Action Model Implementation

The Action Model consists of all actions that can be applied to the model. Figure 5-35 shows the high level of the Action Model classes. All Actions are derived from a generic action class called *CGenericAction*. The product model actions are generically divided into three types: Add, modify, delete. Figure 5-35 shows the hierarchy structure of the Action Model classes and Table 5-4: shows the main attributes of the *CGenericAction* class.

.

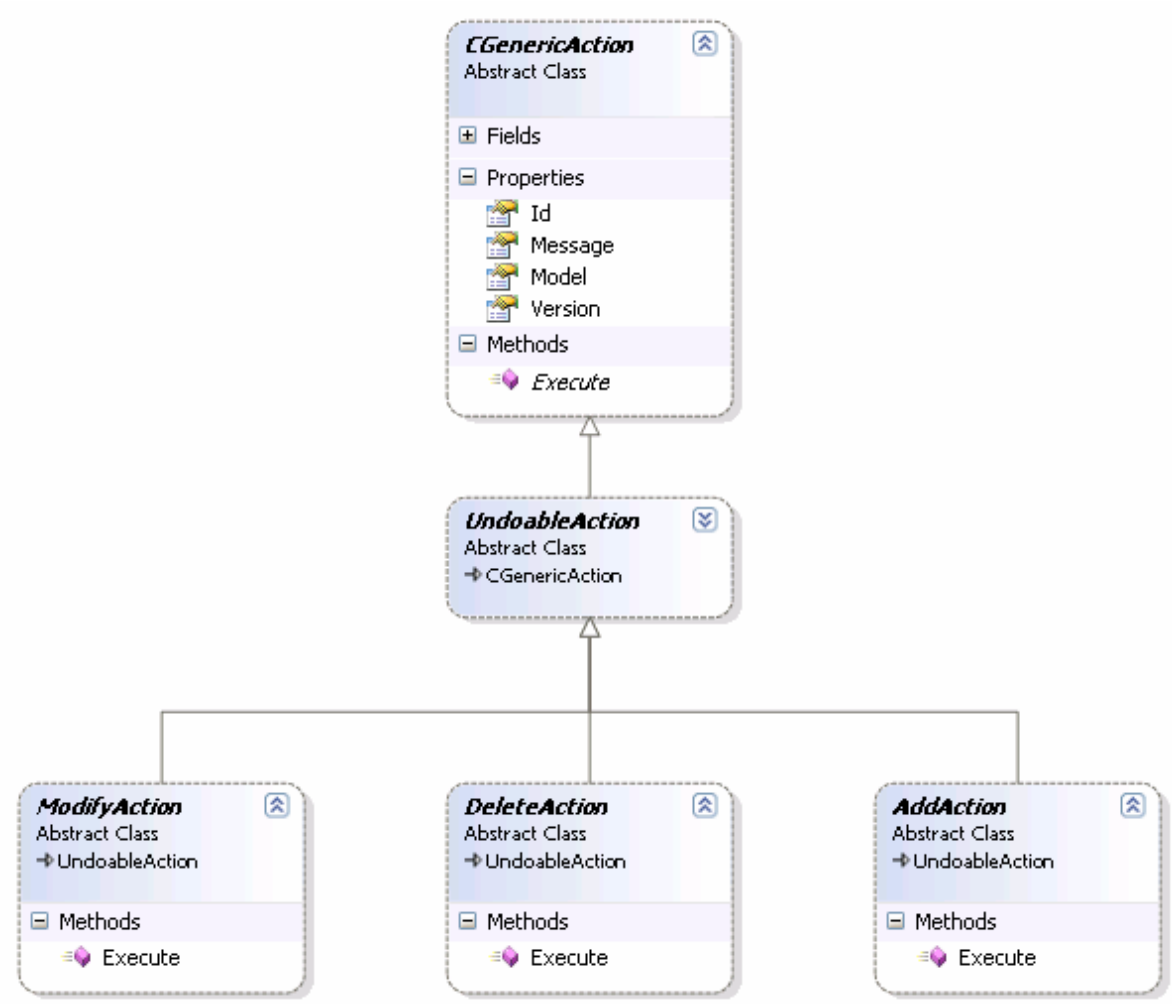


Figure 5-35: Hierarchy structure of the Action Model classes

Table 5-4: *CGenericAction* attributes

Name	**Type**	**Description**
Execute	Method	An abstract method that will be overridden in the derived classes.
Model	Data Member	The product model that is under changing.
Message	Data Member	The received message that holds the data needed to execute the action.
Version	Data Member	The model version that the action is belong to.

As can be seen from Figure 5-35, *CGenericAction* class has three main attributes (Message, Version, and Model) and an abstract method (*Execute*). The message attribute contains the information needed to execute the action. The version attribute holds a version number to mark the action by a version number that can be used later in the restoration process of the model to a previous state. The model attribute represents the product model instance that will be affected when executing the action. *Execute* method of the class is abstract. It will need to be implemented for each derived action. *Execute* methods of the actions that affect the product model only contain the commands that update the product model (i.e. add, modify, and delete) but not those to validate the input as it should be validated beforehand. The actions in the Action Model are executed by either the mediator agents on the client sides or the Product Model Agent on the server. If the action is executed on the client side, the data would be validated before the action is executed while if it is executed on the server by the Product Model Agent, the data does not need to be validated as they would be

already validated on the client side. This helps to minimise the communication between the clients and the server which reflects on the system performance.

There are currently about seventy actions implemented to handle the product model. Below is a list of some of the implemented actions in the Action Model.

Table 5-5: Subset of model actions

Action name	**Description**
`AddFloorAction`	Add a new floor to the building
`ModifyFloorAction`	Modify a floor data
`DeleteFloorAction`	Delete a floor
`AddColumnAction`	Add a column to a floor
`ModifyColumnAction`	Modify a column data
`DeleteColumnAction`	Delete a column from a floor

5.3.6. The 3D VR Model

The 3D graphical interface of the system is implemented using C++ and OpenGL. The interface implementation represents the graphical representation of the product model geometry. The implementation of the graphical model includes two libraries: OpenGlDll and OpenGlCtl.

OpenGlDll is the low level implementation of the graphical model. It is defines the basic geometric shapes such as points, lines, surfaces, polylines, etc. It is implemented using C++ and OpenGl. The library has two basic classes: *CGeometry* and *CGLObject*. *CGeometry* is

the root class of all geometry shapes and includes the data needed to define the shape (e.g. X,Y,Z of a point). The *CGLObject* is the root class of all graphical shapes that includes all data needed to render the geometry on the screen (e.g. colour, transparency)

Below are the super types of the graphical representation classes.

```
class  CGeometry
{
public:
      CGeometry();
      virtual ~CGeometry();
      virtual void Translate(const COneAxis&, const double& amt);
      virtual void Translate(double dx, double dy, double dz) = 0;
      virtual void Translate(const CVector3D&) = 0;
      virtual void Translate(const CPoint3D&, const CPoint3D&) = 0;
      virtual void Rotate(const COneAxis&, double) = 0;
      virtual void Scale(const CPoint3D&, double) = 0;
      virtual void Mirror(const CPoint3D&) = 0;
      virtual void Mirror(const COneAxis&) = 0;
      virtual void Mirror(const CPlane&) = 0;
private:
       static int refCount;
protected:
      GeometryType geomType;

};
```

```
class CGLObject
{

public:
      CGLObject();
      virtual ~CGLObject();
      virtual void DefineDisplay() = 0;
      virtual void Display(const GLDisplayMode& = GLWIREFRAME) = 0;
      virtual void Hilight(const GLDisplayMode&) = 0;
      unsigned long int GetObjID() const { return glObjID; }
      void SetObjID(long Id)  {  glObjID = Id; }
      virtual CGeometry* Geometry() const { return 0; }
      void SetVisible(bool pVisible){     Visible     =pVisible;}
      bool GetVisible(){return Visible ;}
protected:
      void ApplyMaterial(const GLMaterial&);
private:
      void AddMaterial(float*);
      public:
            virtual void SetColour(float colour[3])   ;
            virtual void SetColour(CColour* pcolour)   ;
```

```
            virtual void GetColour(float pColour[3]);
            virtual void SetColour(float r, float g, float b);
            virtual void SetColour(float r, float g, float b, float a);
            virtual void SetBlended(bool bBlended);
            virtual void ZeroGLList(){GLList=0;}
            virtual void ApplyColour();

    protected :
            CColour*  m_Colour;
            bool Visible;
};
```

OpenGlCtl is the higher level implementation of the graphical model. It defines the graphical representations of the product model elements such as beams, columns. It is implemented in C++.Net. The implementation using C++.Net was necessary to make use the OpenGlDll in the system.

The representations of the product model elements were all inherited from a top class called *MCGLObject*. Below is a short description of the class and an example of a derived class (*MCGLColumn*).

```
public class MCGLObject
{
 public :
 MCGLObject(CGLObject* pCtrl);
 void SetObjID(long timestep);
 CGeometry* GetCGeometry();
 void  SetColour(Color^ color, double Alpha)
 void ApplyColor(Color^ color, double Alpha)
 void  Highlight(bool value)
 void  SetBlended(bool bBlended)
 void  SetAlphaRatio(double ratio)
 double  GetAlpha()
 void ZeroGLList()
 CGLObject* GetCObject()
 property  bool Visible
 protected:
 CGLObject* m_pCtrl;
 bool m_visible;
```

```
 Color^ m_Color;
 double m_Alpha;
 public:
 long  lId;
};



public class MCGLColumn : MCGLObject
{
      private :
            double x,y;
      public:


      MCGLColumn(String^ Id,double x1,double y1,double z1,double z2,
            double w, double d, double ft, double wt, double rotation) ;

      bool virtual CheckBoundary(double px1, double py1, double pz1,
            double px2, double py2, double pz2) override ;
 };
```

The graphical entities are linked to their product model data through unique IDs. This allows identifying the relevant model element when the user interacts with the graphical interface.

5.3.7. System Support Utilities

Complementary tools were developed to enhance the collaboration among the design team. The current implementation of the system tools are meant to demonstrate the usefulness and the need for such tools in a collaborative system rather than completeness. The implemented tools are:

- **System Restore Tool**

There are occasions where it would be necessary to revert the model back to an earlier stage of design. Examples of this include cases of unsolved conflicts or the need to follow what-if scenarios and experimentation or simply to back up the model. For that purpose, a restore

tool was developed. It allows for the restoration of the design model data to a prior state for whatever reason. The system administrator can specify the restoration time of the product model or choose from a predefined restore points. The System Restore Tool will automatically rebuild the product model up the restored point.

- **Design History Viewer**

The product model may be changed by any of the design members, and each design element may be reached by more than one designer. So from design point of view, it will be helpful to check the history of changes occurred upon any element. Moreover, it is sometimes important to make inquiry to determine the responsibility for any changes. The product model was implemented in away that each design element stores all the changes it has since its creation.

The Design History Viewer was implemented to display the history changes of the design elements. It allows designers to inquiry about the changes by element name, or type, or date interval. Figure 5-36 shows the history of a column. The figure shows three actions in the column changes history (a creation action and two modification actions).

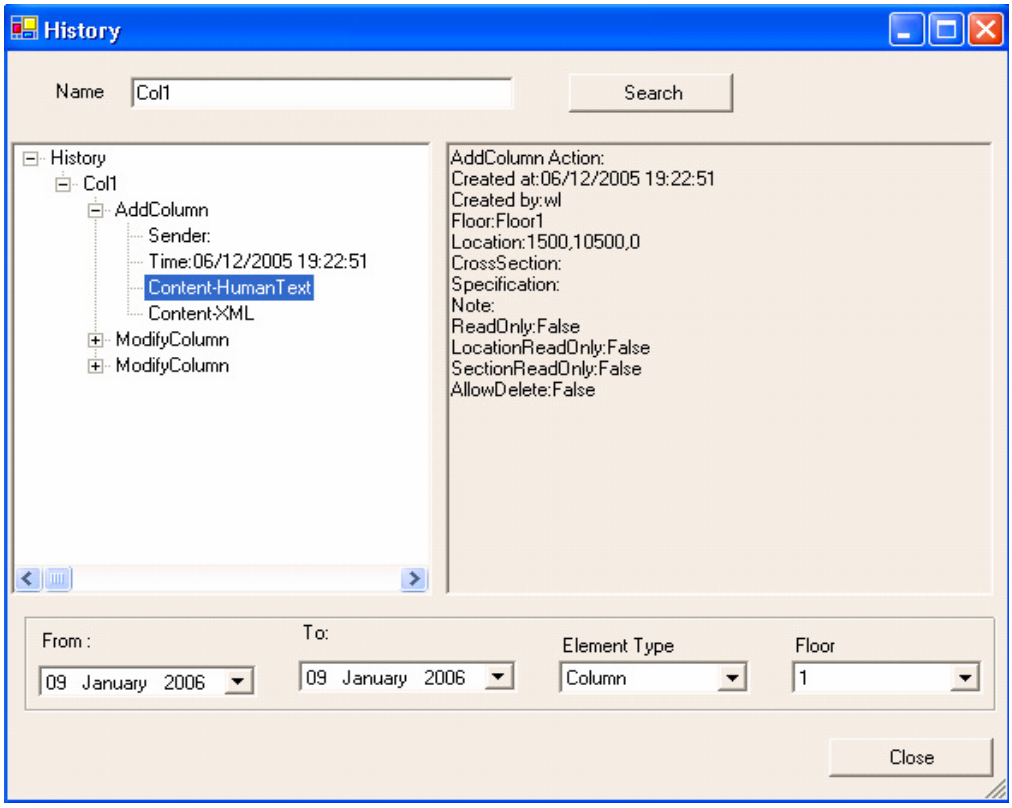


Figure 5-36: Changes history of a column

Because the messages exchanged in the system are formatted using XML, which is difficult for human to read, the messages are mapped automatically to human readable text before they are displayed.

- **Communicator**

During the design process, the design team may need to communicate synchronously. The system offers a virtual meeting facility. It allows a real-time interaction among those who are on-line. Designers can directly contact each other to ask questions or discuss various topics.

Besides the ability to exchange instance text messages, the designers are able to remote their design view screens which make the discussion more feasible.

Figure 5-37 shows a scenario of text message exchanging between two designers with a remoted view.

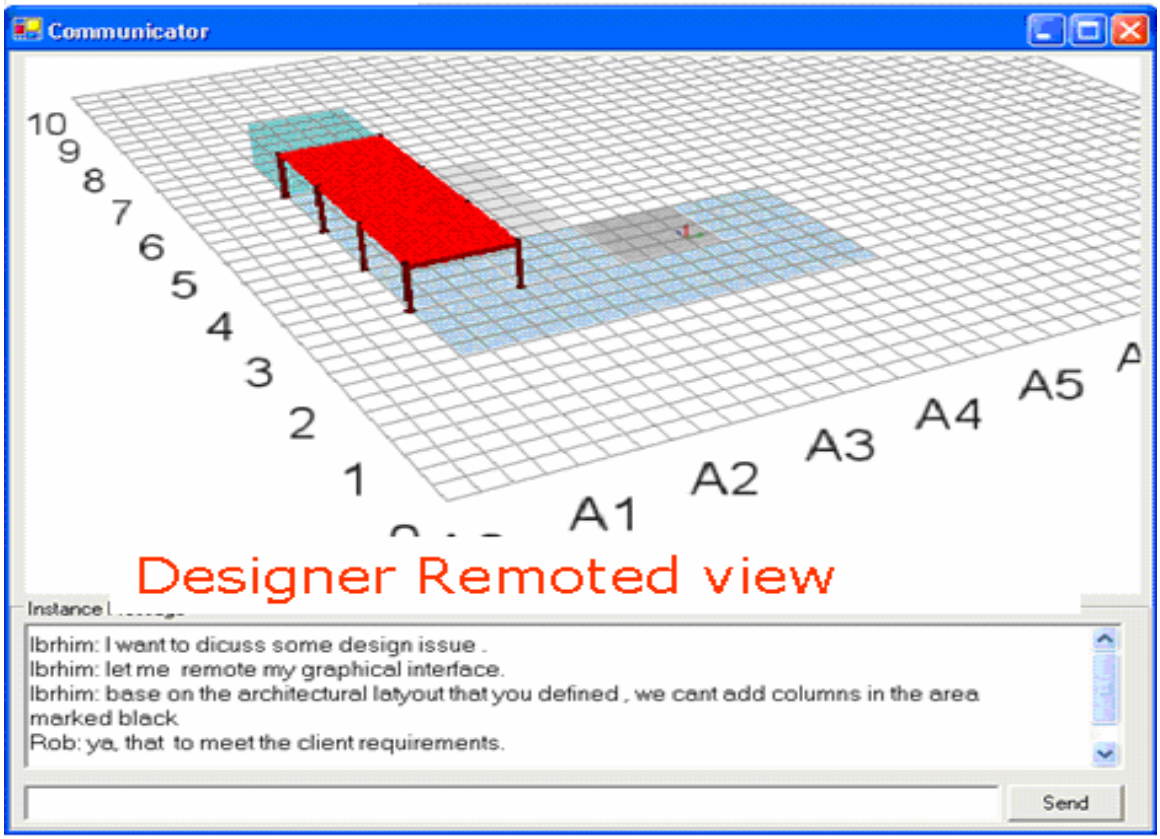


Figure 5-37: Virtual communication

- **Task Management Tool**

This tool allows designers to communicate asynchronously. It is similar to an email system. It offers the following benefits:

- Assign a task for another designer.
- Automatically notify the designer about any new received tasks.

- Automatically identify the definer of the tasks when the tasks are completed.

5.4. Summary

This chapter described the development of a proposed collaborative design environment for multi-steel storey structures. The chapter included:

- The system high level requirements
- The system design and architecture
- The system implementation.
- The system support tools.

Chapter 6.

A Case Study Using the Proposed System

6.1. Introduction

This chapter demonstrates the proposed software system. A case study is used to help presenting the usefulness of the system. The demonstration will illustrate the conceptual and preliminary design processes of a typical four storey office building.

A research goal of the case study was to investigate the effects of the integrated collaborative design environment on the design process. This includes the following points:

- The ability of geographically dispersed team members from different disciplines to work together.
- The ability of the design team to expose their presence to the others.
- The ability of the design team member to preserve their design input.
- The ability of the design team to experiment with many design scenarios.

The case study is based on a multi-storey building; its plan illustrated in Figure 6-1. The building is a regular multi-storey office building based on a regular 1500 mm grid with three cores. The building consists of 3 by 7 bays with standard bay size of 7.5 m by 7.5 m. The selected case study is relatively simple in terms of the building shape. This was necessary because of the incompleteness of the implemented prototype in terms of the front end-user interface and the product model. The system currently allows only for the input of

regular building shapes. However, the ability of the system to support collaboration can still be illustrated.

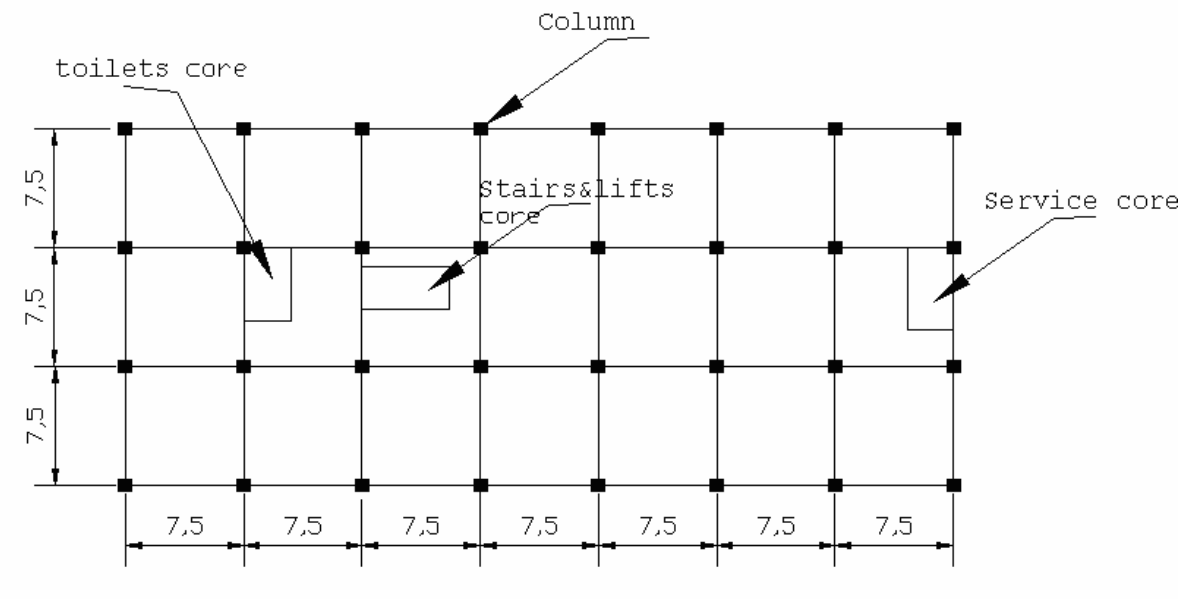


Figure 6-1: Architectural floor-plan

Although the design process in this case study conducted concurrently, it is described using different project views in a sequential way. The views are:

- The client's view.
- The architect's view.
- The designer's view.
- The service engineer's view.

In this case study, the design team consists of five persons. Those are the project manager, the client, the architect, the designer, and the service engineer. The project manager is assumed to be the administrator of the project.

6.1.1. Project Setup

Before the design process start, the project manager creates a new project (i.e. the shared model on the server) and defines the design team members. The project manager acts in this case study as the administrator of the project. He creates the new project file on the server and adds in the actors with their roles (Figure 6-2). The actors are granted default permissions based on their respective roles defined by the project manger. The project manager then emails the design team with their initial usernames and passwords to access the model on the server through the system on their machines.

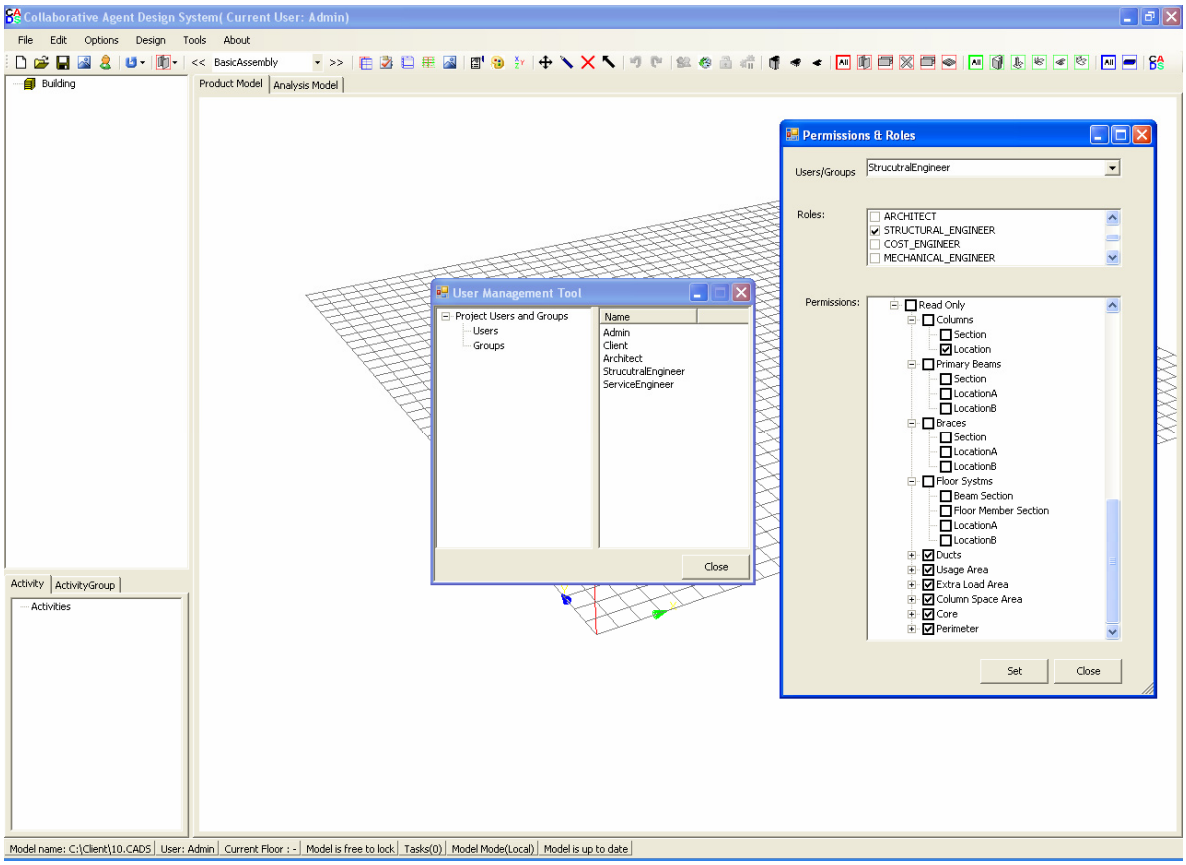


Figure 6-2: Project setup

As an example in this case study the designer is given a role of a structural engineer. Table 6-1 shows the default general permissions of the structural engineering role.

Table 6-1: The default permissions of the designers

Element\Permission	Allow Change Ownership	Allow Delete	Allow Create	Read Only	
Column	No	No	No	Location	True
				Section	False
Beam	No	Yes	Yes	Locations	True
				Section	True
Brace	No	Yes	Yes	Locations	True
				Section	True
FloorSystem	No	Yes	Yes	Locations	True
				Sections	True
Duct	No	No	No	Locations	False
				Section	False
UsageArea	No	No	No	False	
LoadArea	No	No	No	False	
ColumnSpaceArea	No	No	No	False	
Core	No	No	No	False	
Perimeter	No	No	No	False	

6.1.2. The Client View

The client, in this case study, starts the design process by defining the building's overall requirements. This includes the building's target cost that may be subdivided between the structure, services, flooring and cladding (Figure 6-3). The client is also responsible for setting the total floor space required within the building as a whole. At all times during the building's design, the costs and areas are automatically calculated and compared with the client's requirements. All building design participants will be notified in case the actual figures fail to meet the targets.

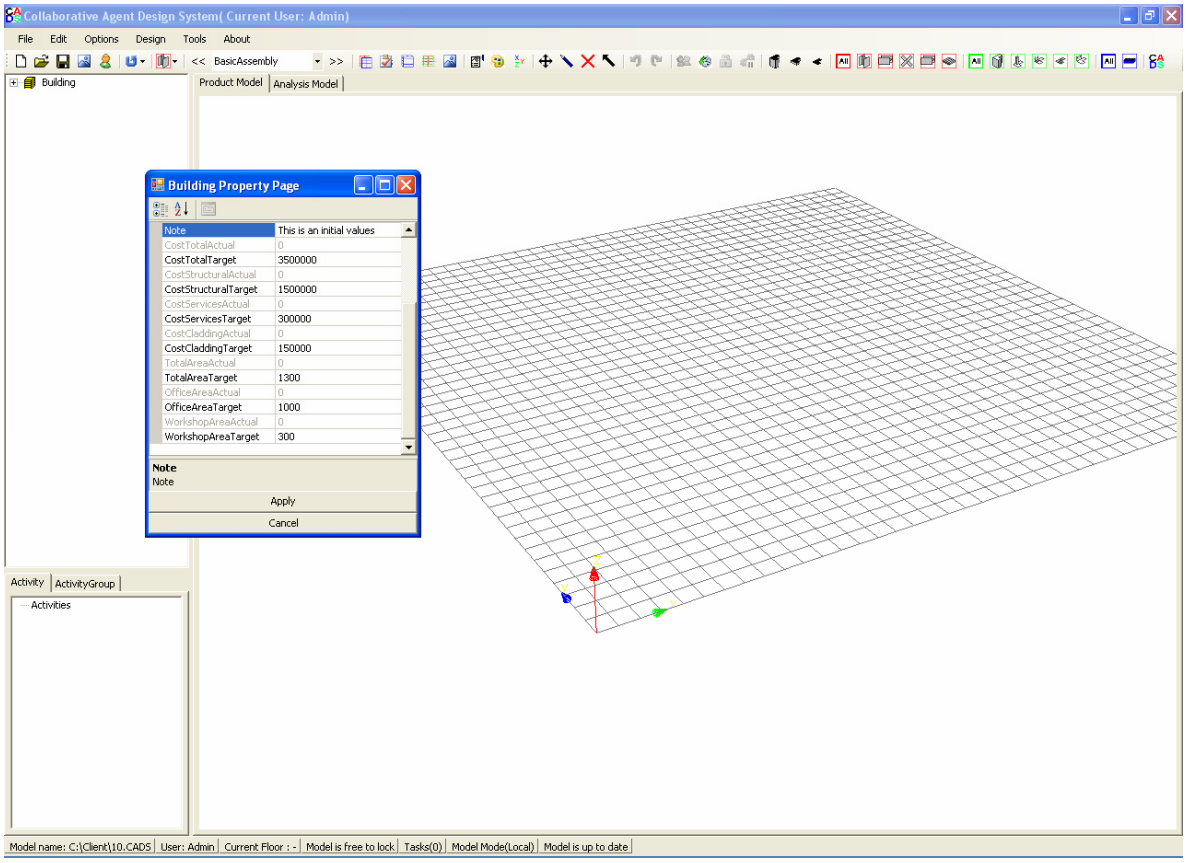


Figure 6-3: Building management dialog box

Also in this case study the client with the project manger will define the overall activities in the project. Figure 6-4 shows the overall activities grouped under an activity group called Schedule. All the design team can see the activities tree and be aware of the project progress.

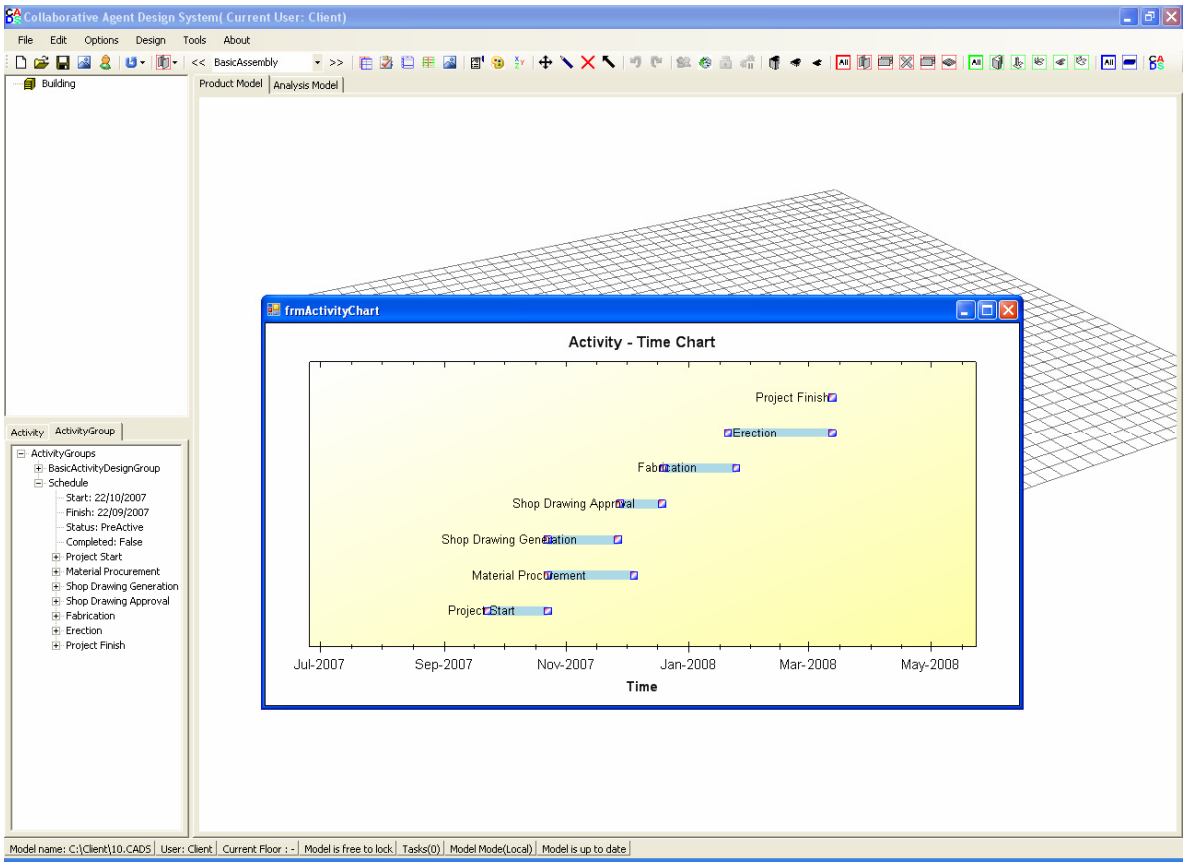


Figure 6-4: Project overall Schedule

After the client defines the general requirements, he sets the architect a design task to put the architectural floor plan through Task Management Tool of the system and informs the project manger that the design process started. The project manager, in turn, communicates with the rest of the design team that they can start monitoring the project progress for any concerns or to put their initial design based on the architect work.

6.1.3. The Architect View

Once the architect receives the notification from the client or the project manager, the architect then specifies the layout of the building's perimeter, the number of floors, and the internal height requirements for each floor using Floor Management Tool (Figure 6-5). The

height of any given storey of the building can be given in terms of the non-structural floor depth, clear depth and the ceiling depth, which includes the services and structural depths. This defines a set of requirements for the structural and services design to conform to, and sufficient information to specify a general model of the building in terms of size and spacing.

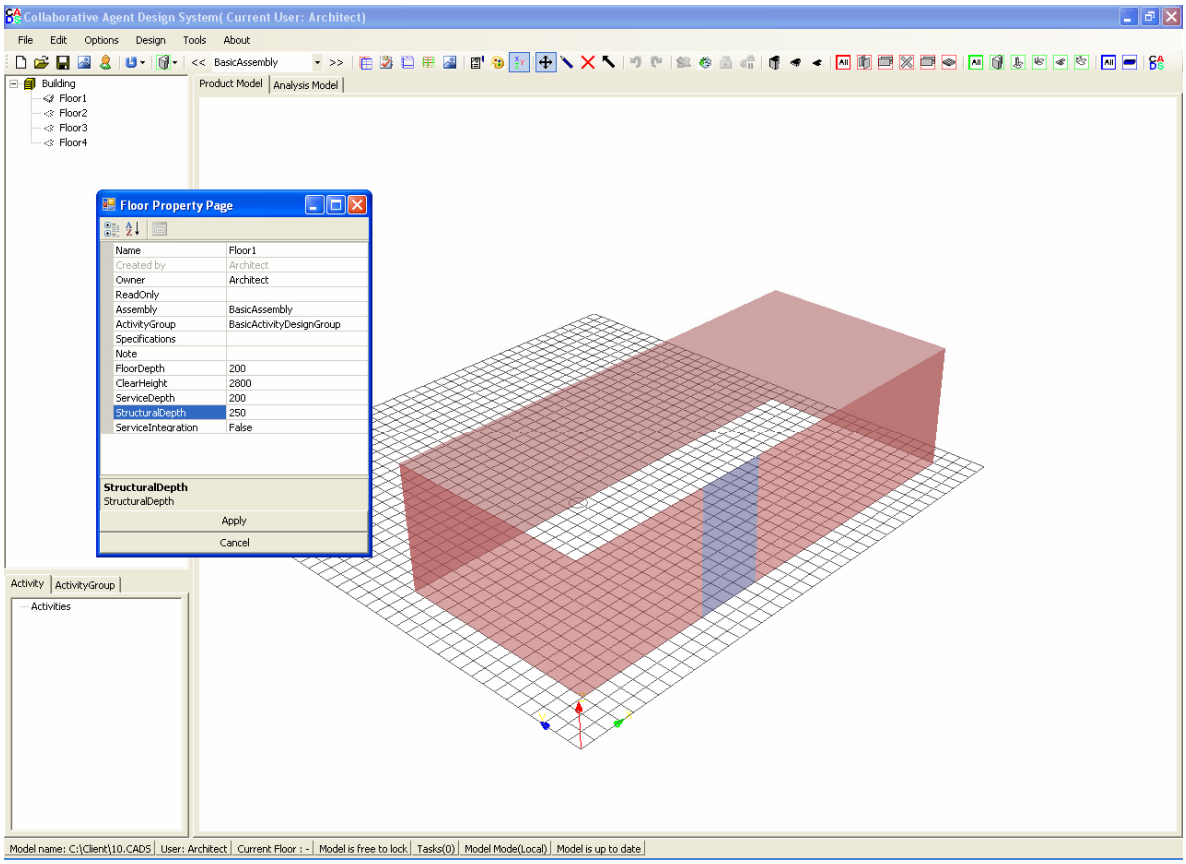


Figure 6-5: Definition of building boundary

The architect is also responsible for specifying the area designations that dictate how the floor area will be utilised and the cladding types assigned to each part of the building's perimeter. Different colours are used to show the different types of area, such as 'office' and 'corridor' (Figure 6-6). The cladding input provides the data to calculate the claddings cost, which will then be compared with the clients brief. It also provides an early indication

as to the external look of the building, which may be useful for assessment of early prototypes by the client. In terms of the building's structure, the cladding information also provides data regarding its loading requirements (i.e. self-weight of the cladding) that will then be considered into the structural analysis.

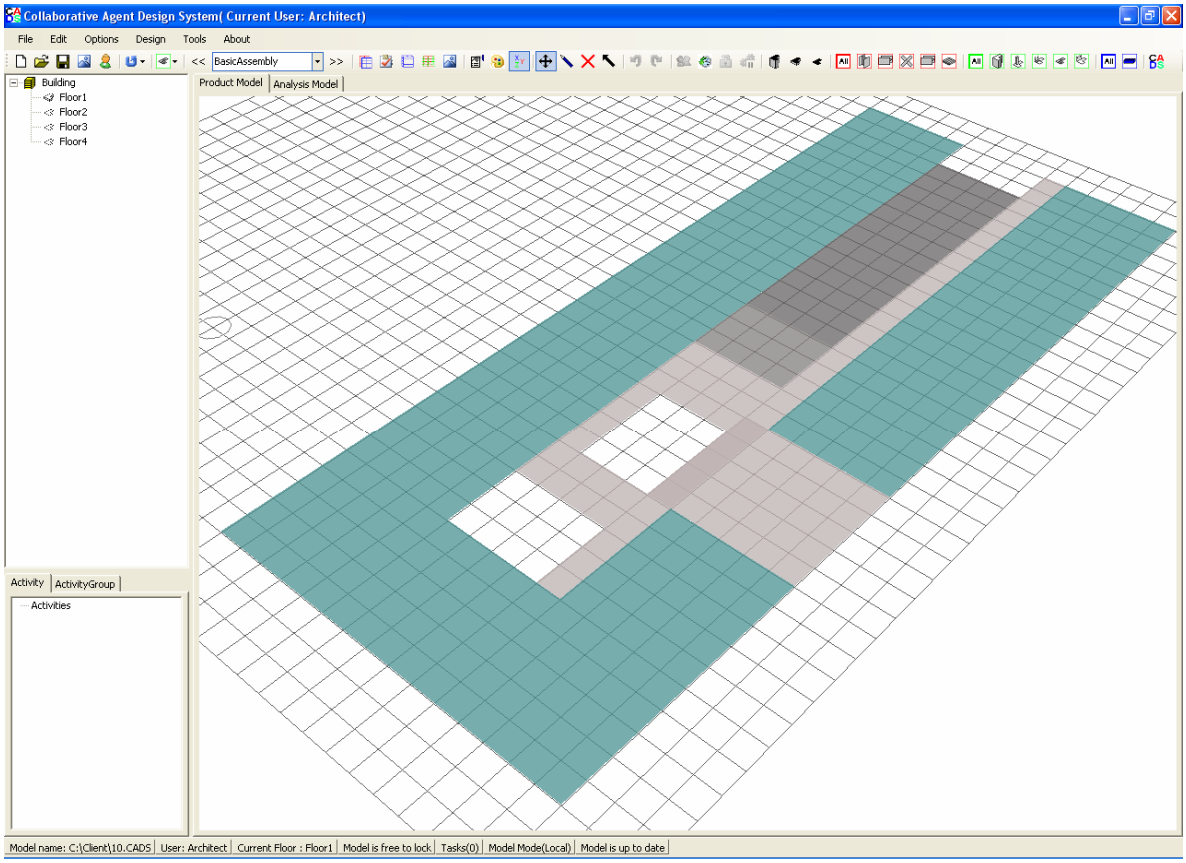


Figure 6-6: Definition of floor plan and usage

The architect may also provide information regarding extra loading areas within the building, such as around building cores so that the appropriate loads will be added to the building's structural analysis model. He is also responsible for specifying cores in terms of their location and designated purpose (Figure 6-7). This information will be important for use by the structural engineer and the services engineer.

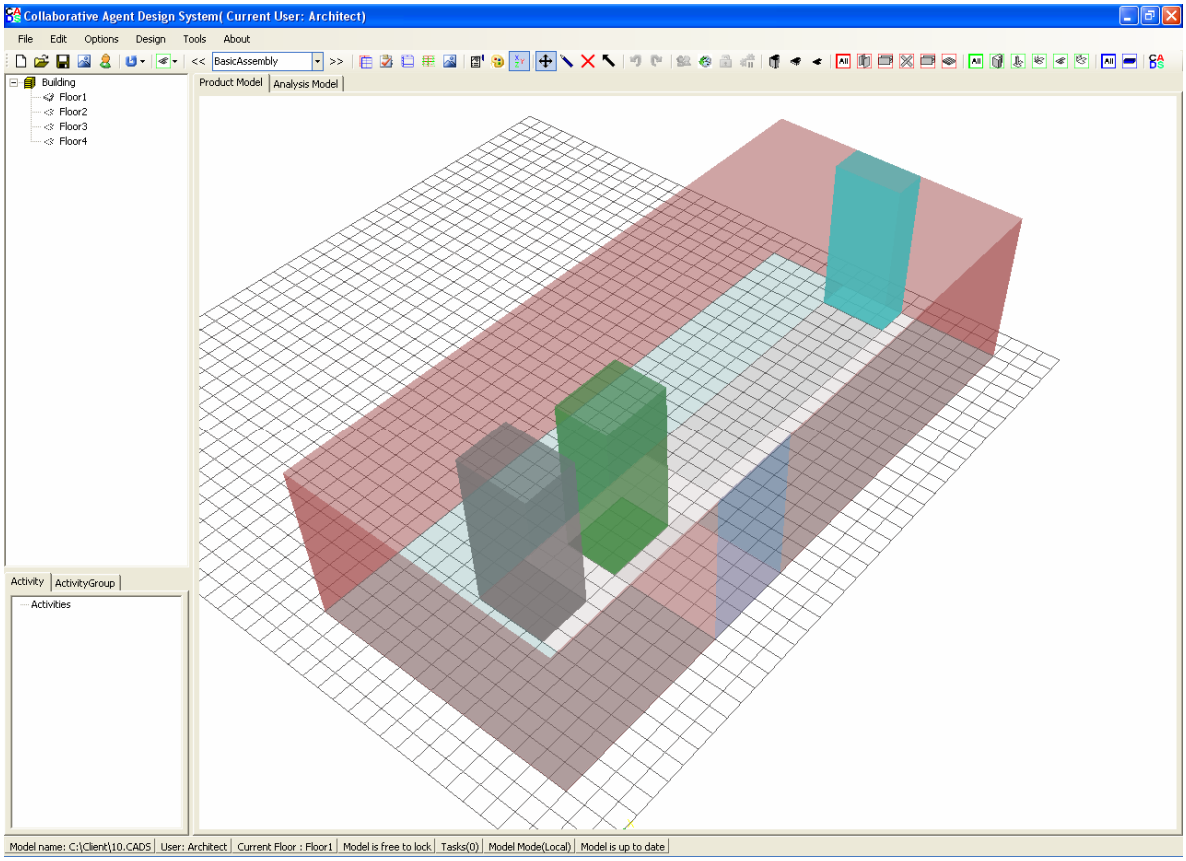


Figure 6-7: Definition of building cores

The architect may also designate column spacing areas (i.e. column free areas) within the building, in order to restrict the structural engineer from positioning columns inappropriately; or positioning the columns himself and constrain them to be repositioned and optionally free their cross sections to be changed by the structural engineer. Although, in this case study, the arrangement of the column positions is heavily influenced by the shape of the building, it is assumed that the column positions are constrained by the architect to ensure they will not be changed. The process of defining the column grid within the system is a semi-automated process. The column height is not required to be specified, as it can be automatically calculated based from the predefined architects

building information. It is also not necessary to specify the exact dimensions of the columns at this stage, as they may be calculated automatically from parametric rules (using rule of thumb as specified in the manual for the design of steel structures). These can then be finalised based on the columns structural response (acquired at the analysis stage).

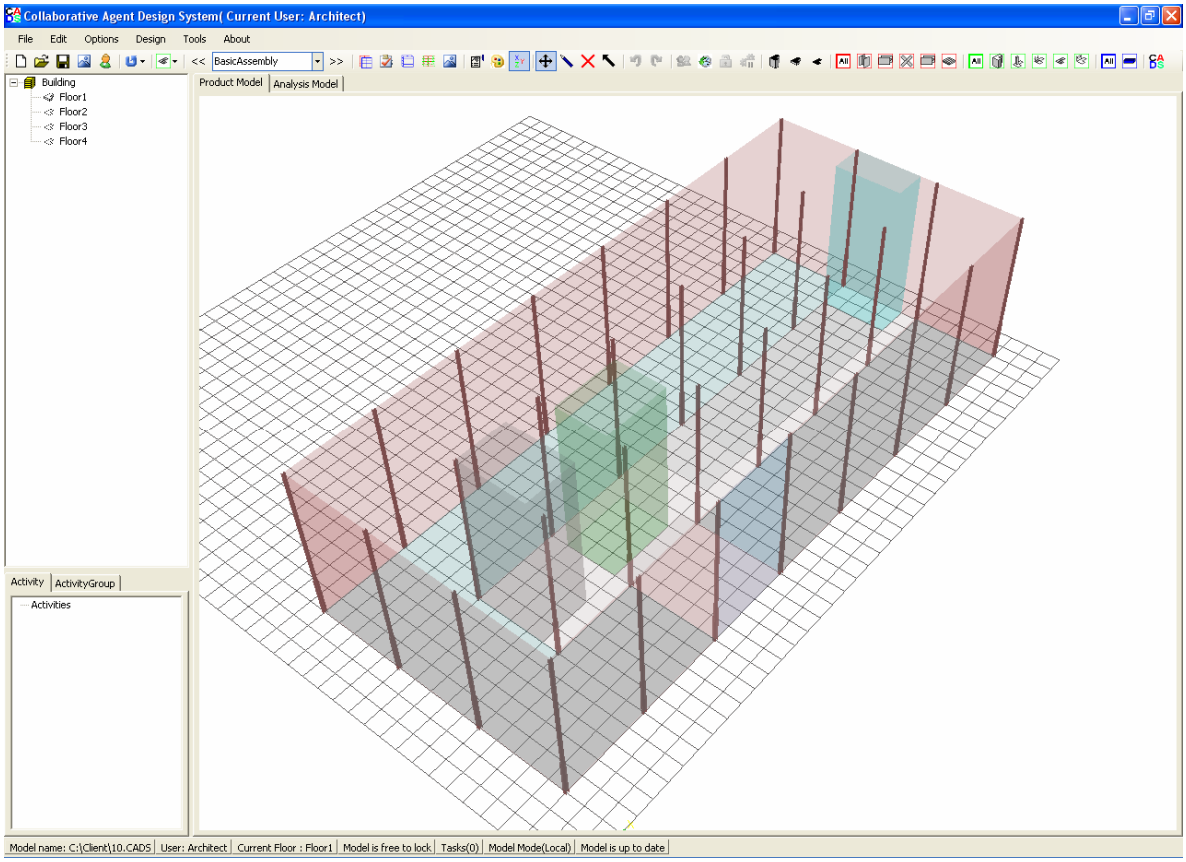


Figure 6-8: Positioning of column grids to suit architectural constraints

Once the architect completed the floor plan, he can arrange an online meeting with the client to discuss the plan. This can be done through the system communication tool. He can also invite other design team members to have their views on the initial floor plan. The architect's design input has a far-reaching effect on the building's overall design. It must meet the client's requirements in terms of the spacing requirements, while setting forth the requirements of the structural building services engineers.

6.1.4. The Designer View

After the design team approves the architectural floor plan put by the architect. The designer's first task is to specify the column positions within the building if not specified by the architect. In this case study, the column grid is initially defined by the architect and will be modified by the designer. The architect limited the designer ability to modify the column positions and left the cross sections to be decided by the designer based the structural design requirements.

Once the column positions are finalised, the designer carries on defining the rest of the structural part of the building. He adds the primary beams by selecting each pair of columns in turn from within the 3D view. As with the columns, the primary beam dimensions can either specified by the designer or allow the Design Agent to automate this process. Figure 6-9 shows both the architectural input and the structural frame of the building. The architectural input is shown with transparency.

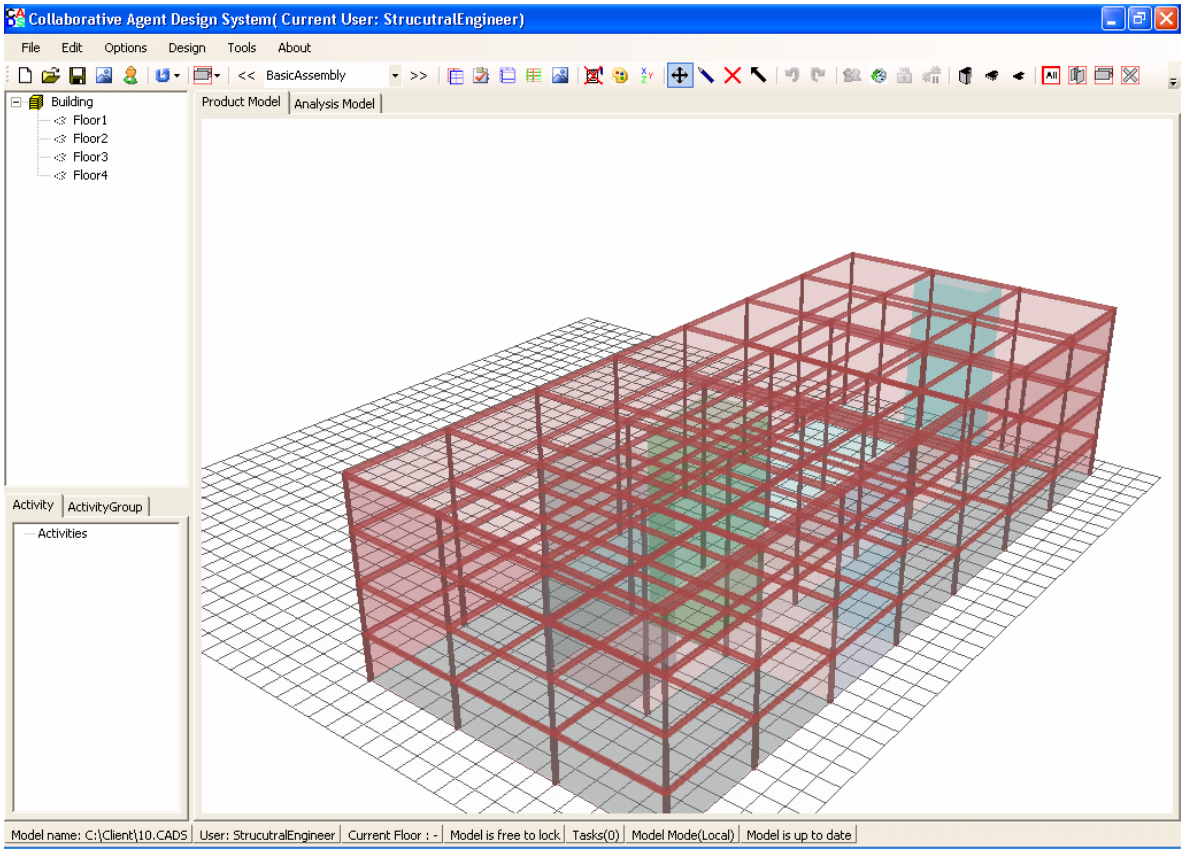


Figure 6-9: Proposed structural framing system

The floor systems are then added to the building. For each rectangular area bounded by four columns, the diagonally opposing columns are selected, and a steel decking arrangement is selected. As with the previous structural design stages, any or all aspects of the structural floor may be specified by the designer including the secondary beam dimensions, spacing, orientation, and the floor type and floor dimensions. The detailing for the remaining design aspects, if any, will then be calculated automatically.

The designer next defines the bracing system. He can add the bracing systems to the model by selecting the two columns that the bracing system will span. The bracing systems (for

lateral stability) are usually provided within the building core. The cores may contain the staircase, services, and toilets and are lined with block work. In most cases, the core acts as a shear wall and provides stability to the building. If extra bracings are required, then their location needs to be finalised in consultation with the architect. The designer can either work alone and consulate the architect after the bracings are defined or work concurrently with architect in an online session to save the time for later modifications and redesign. As with the rest of the structure, the designer can specify the size of the braces or allow the Design Agent to automate this process.

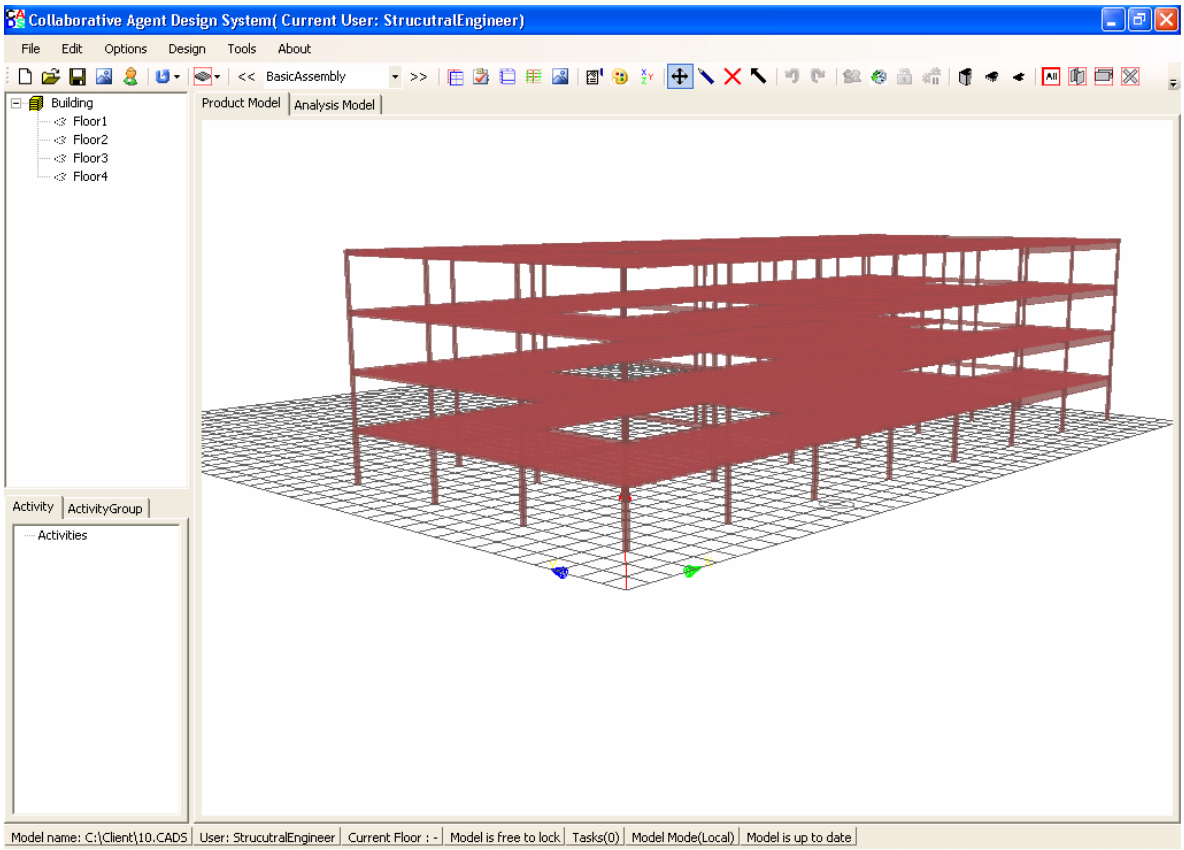


Figure 6-10: Complete structural system prototype

With the information provided through these processes, there is sufficient data to produce a 3D structural analysis model of the building (Figure 6-11). This is generated automatically

by the Structural Analysis Agent to accurately model the members and connections with the appropriate use of analysis members and dummy members to model connection offsets, hence maintaining compatibility between the as-built structure and the analysis model.

Loading information is also automatically generated from the data previously input, in terms of factored dead and live loads. The analysis is carried out to produce the structural analysis response model. This information is used later to perform a design check on the structural members, which may then be used by the designer or automated processes to iteratively improve the building's design.

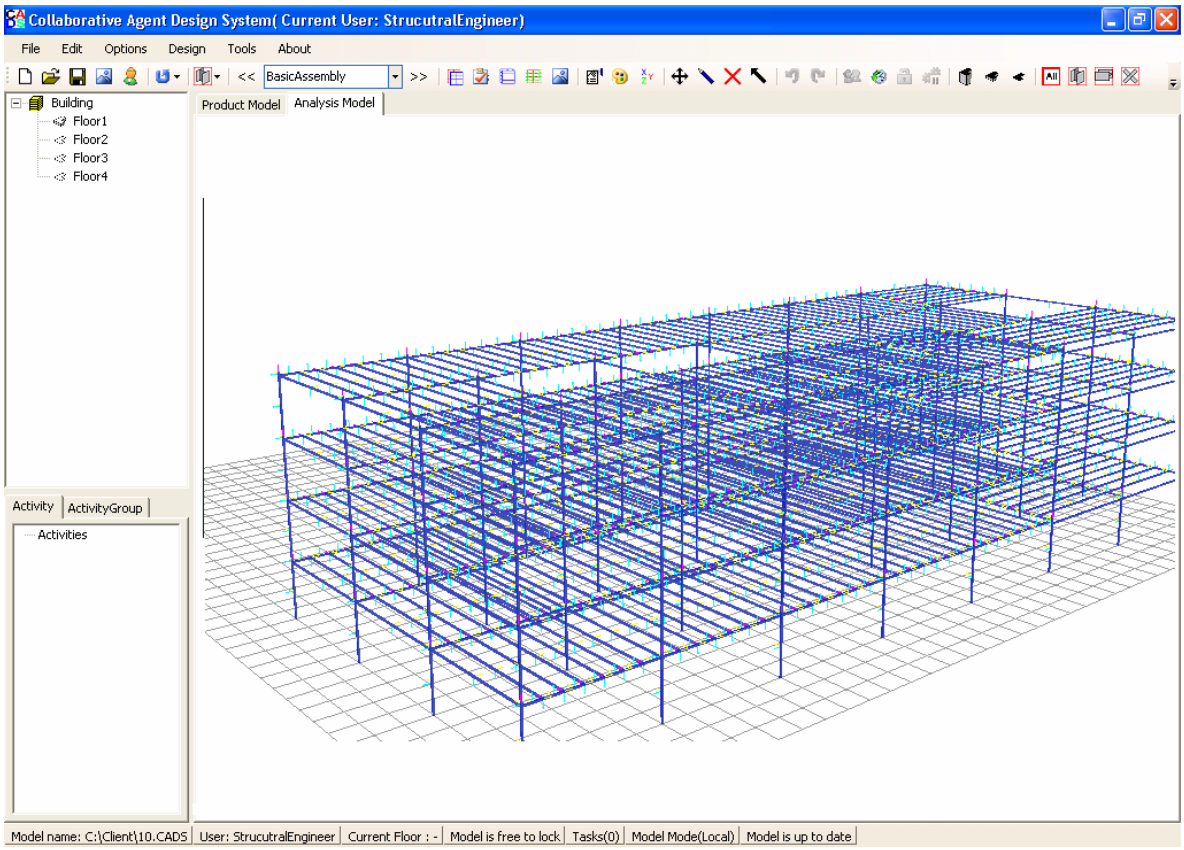


Figure 6-11: Structural analysis model

Once the structural model is generated the designer can perform a design check. Once the structural elements design is check by the Design Agent the designer can view the check results and adjust the design if necessary. Figure 6-12 shows the design results and an edit column by the design to modify the column properties.

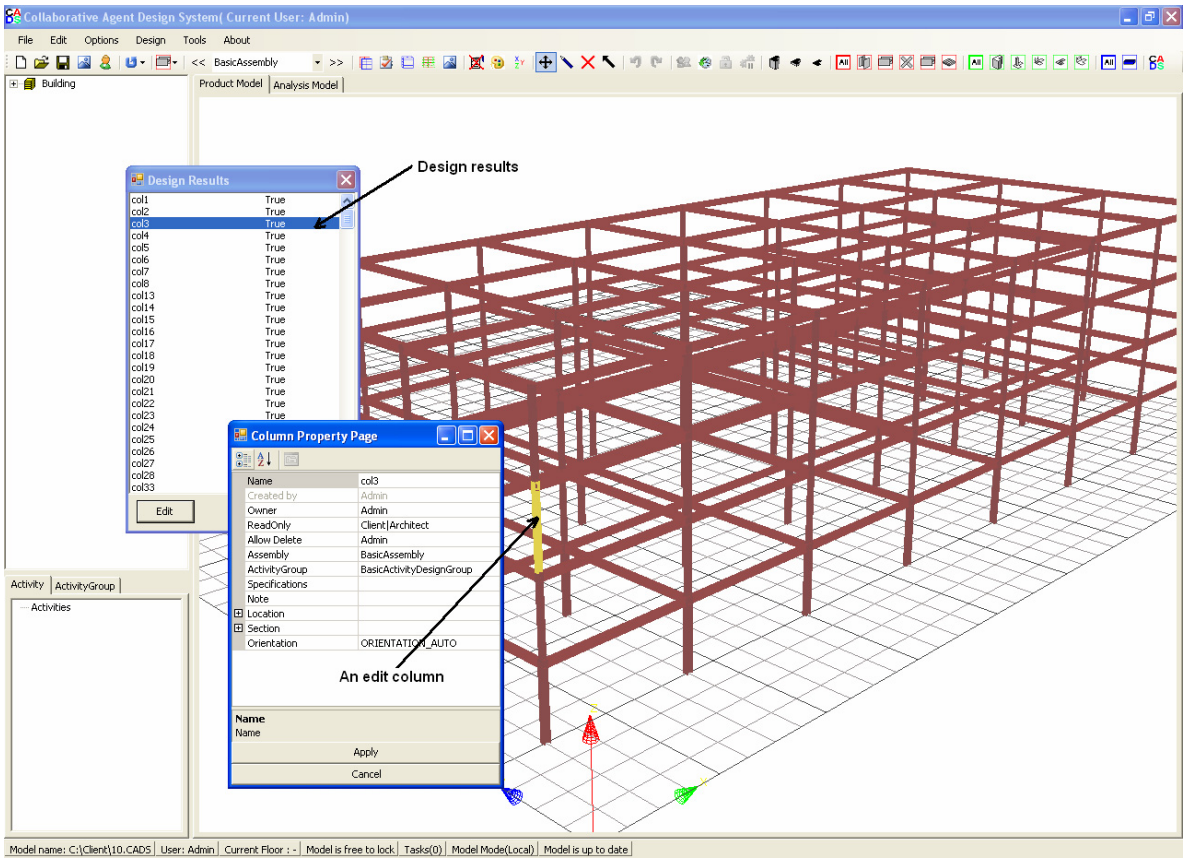


Figure 6-12: Design check results

The designer can also check the stability of a part of the structural model by grouping elements in an assembly based on their erection sequence. In this case study, the structure was divided into 6 sequences (Table 6-2). Figure 6-13 shows the structure erection for the first five sequences and highlights Sequence 4.

Table 6-2: Structure sequences

Sequence	Description
1	Columns of the first two floors of the first three bays from the left.
2	Columns of the first two floors of the last for bays.
3	Beams of the first two floors of the first three bays from the left.
4	Beams of the first two floors of the last for bays.
5	Columns and beams of the last two floors of the first three bays from the left.
6	Columns and beams of the last two floors of the last four bays.

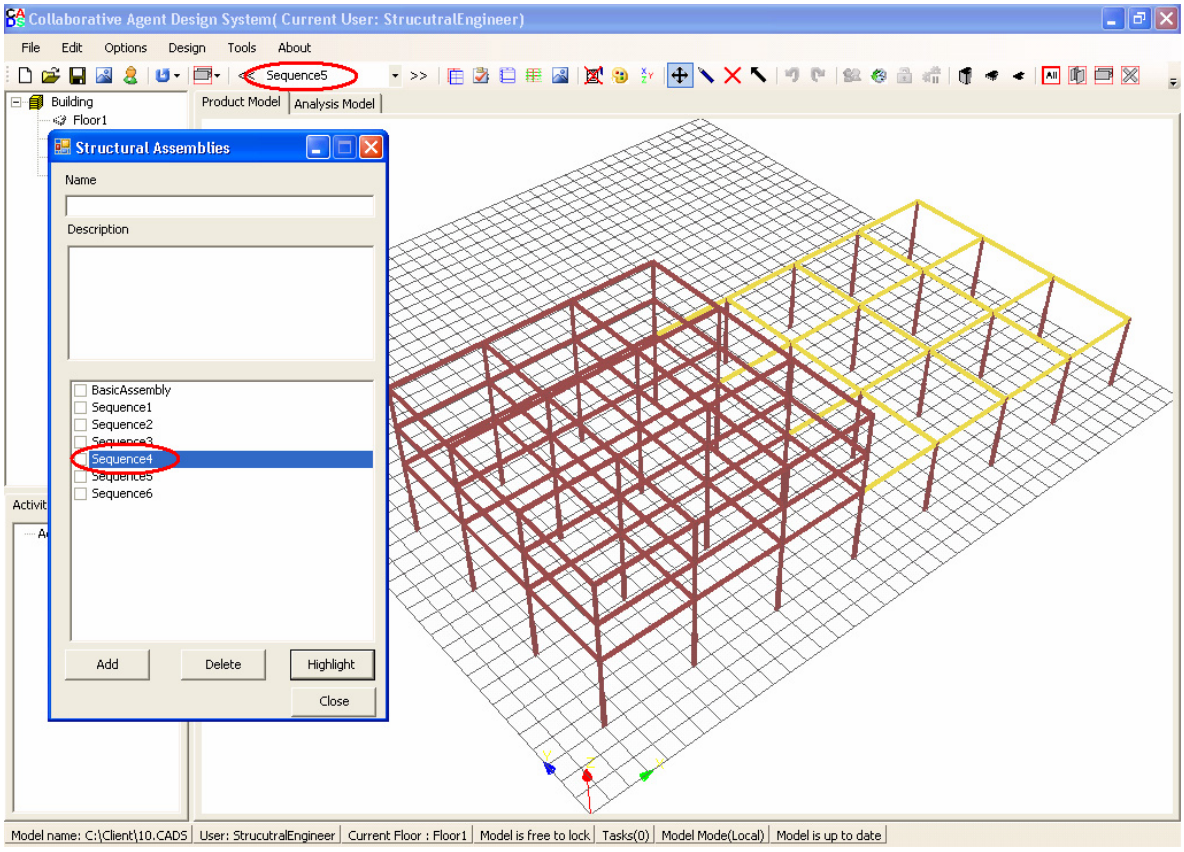


Figure 6-13: Structural assemblies

The designer at any stage can limit the access right of the other team member to the structural design. This may be very important especially to ensure that the design compatible with the structural analysis results.

6.1.5. Building Services View

The services engineer is responsible of specifying the location of the service entry points to the building, the location of service cores and the routing of the main services at each floor. The dimensions of the actual service ducts are based on the project requirements. The service ducts may either be positioned within the structural layer or beneath the structural layer. In this case study they are positioned beneath the beams. Figure 6-14 shows the service ducts and the structural frame. The structural frame is shown with transparency.

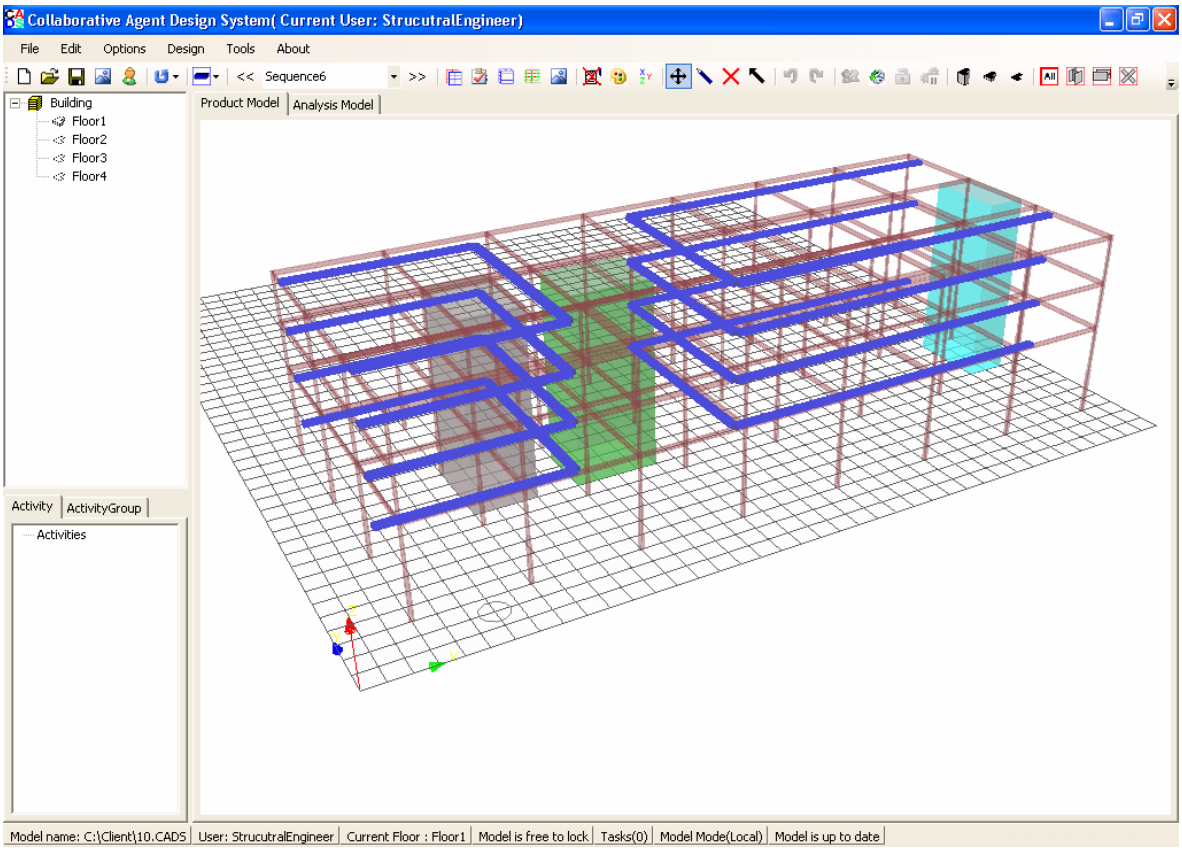


Figure 6-14: Adding services duct connected to services core

The service engineer may conduct an online session with both the architect and the designer to discuss the service arrangements. This allows the architect and the designer to raise any concerns regarding the services instantly. The other option is that the service engineer finish placing the service ducts and notify the architect and the designer to review them. In this case study, we assume that the service engineer allocated design task to the architect and the designer to review his input. To make sure that there is no one will modify his/her input, the service engineer applies read-only permission to his/her input.

6.1.6. Design Testing Stage

At this point, a basic building design has been achieved. The design can now be modified to test other design options. An extra floor may be added to the building by modifying the number of floors in the floor management dialog. Without any work on the part of the user, the columns, primary beams, and floor systems are re-generated in-line with this new requirement. The increase to the building cost can be immediately appreciated, and the column dimensions have been adjusted to take into account the new loading.

For any redesign scenario the project manager can mark the current accepted design using Revision Tool. Revision Tool allows the design team to view all marked versions of the model and move in the history of the building backward and forward to any marked version.

6.2. Summary

This chapter demonstrated the application of the proposed system though a case study of a multi-storey steel framed structure design. It shows a typical design scenario involving the architect, the client, the designer, the project manager, and the service engineer. Although, the design process may seem sequential, the design team can work concurrently all along the design process. They can view and inspect the model at any time. They also can exchange their views of the design and be notified by the actions of the others.

The information describing the building together with all relevant input is stored in the shared product model. However, any designer can work on a local model so partial building designs may still be carried by concentrating the work on a single discipline. For example, a structural engineer can input a steel frame and carry out analysis and design without the availability of architectural constraints. .

Chapter 7.

Evaluation

7.1. Introduction

This chapter describes the methodology used to evaluate the developed prototype application and the outcome of the evaluation. The chapter initially discusses the objectives of the evaluation process and explains the methodology adopted, including the development of a questionnaire. It finally discusses the results of the evaluation.

The overall aim of this research is to investigate the feasibility and the potential advantages of using distributed information technology alongside a suitable product model to support multi-disciplinary collaborative design.

The motivation for this evaluation consists of four primary issues:

- Validation of the application prototype with real data against the main planned functionalities and system requirements.
- The suitability of using the prototype system within the design process and whether it offers a step change from current work practice.
- Ascertain the areas that need further research.
- Assess the readiness and suitability of the modern distributed information technologies for developing real-time collaborative design systems.

The first three issues will be answered by the end-users (i.e. the engineers) based on their own experience in tackling engineering design problems. End-users are experts in their domain and constitute a valuable source of information. They tend to have good insight of the task at hand and are well aware of the typical problems that are faced. The forth issue will be concluded based on the feedback given on first three issues.

7.2. Evaluation Methodology

The system was developed to provide the designers with better design collaboration support. The purpose of the evaluation is to assess the extent to which it satisfies the defined system requirements in supporting collaborative design from the end user point of view.

There are many methodologies to evaluate a software system such as Factor-Criteria-Metric (FCM), Quality Function Deployment (QFD), and Goal-Question-Metric (GQM) (Shepperd, 1995). Several different approaches can be adopted to assess the performance of the implemented prototype system and evaluate its validity and suitability for the collaborative design process. In this research, GQM have been selected because of the following reasons (Chang, 2001):

- It is a well-established, flexible and effectively applicable method to perform measurement of software process.
- It is a goal-oriented through top-bottom definition of metrics via questions.

The idea behind GQM is to use a systematic mechanism for defining, measuring, and evaluating software engineering processes (Basili, 1992). The literature typically describes GQM in terms of a six-step process where the first three steps are about using business goals to drive the identification of the right metrics and the last three steps are about gathering the measurement data and making effective use of the measurement results to drive decision making and improvements. Basili described his six-step GQM process as follows:

1. Develop a set of corporate, division and project business goals and associated measurement goals for productivity and quality.
2. Generate questions (based on models) that define those goals as completely as possible in a quantifiable way.
3. Specify the measures needed to be collected to answer those questions and track process and product conformance to the goals.
4. Develop mechanisms for data collection.
5. Collect, validate and analyse the data in real time to provide feedback to projects for corrective action.
6. Analyse the data in a post-mortem fashion to assess conformance to the goals and to make recommendations for future improvements.

Goals of the organisation and its projects are defined and refined into a set of questions characterising the objects of measurement (e.g. products, processes, or resources). Each question tries to characterise the object of measurement with respect to a selected quality issue and to determine its quality from the selected viewpoint. Questions in turn are refined into metrics that can be used for measuring the issues involved (whether they be objective or subjective to the viewpoint taken) in a quantitative manner.

In this research, the evaluation of statements such as "Designers can collaborate and share design activities" or " the system can provide effective collaboration" is difficult in absolute terms. The best strategy of the evaluation should be to identify metrics for the validity of such statements and then to compare these metrics for the developed prototype versus “traditional tools”. Traditional tools are tools that reflect the current state of practice of design in industry. A set of variables can be introduced into the comparisons to identify specific circumstances under which the system being evaluated leads to more effective design. However, this work has concentrated on developing a proof of concept and has not been able to carry out such in-depth evaluation. This would be recommended for future research for this work. Besides, applying GQM paradigm in its entirety to the prototype evaluation would not be feasible. Most of the metrics would be very subjective and would reflect only the views of two different perspectives, namely those of a developer and the end user. There is no "organisation" that could answer the questions and give a broader view to the applicability of the system. However, the GQM approach can still be used to define the evaluation goals and refine them into generic questions about the feasibility of

various features. These questions can then be refined to be presented during data collection steps. So at this stage of evaluating the system, the main problem is not to find reliable and accurate metrics for measuring the software, but to ask the right questions to get as detailed comments and feedback as possible, and to discover problems not anticipated by the researcher. A goal will be set to define the main focus of the evaluation, and a relevant set of questions will be used for data collection. The feedback combined with the developer's subjective evaluation of the system can then be used as a basis of further development work.

7.3. The Goal

The GQM methodology is based upon the assumption that to gain a practical measure one must first understand and specify the goals of the software being measured, and the goals of the measuring process. More specifically, it is important to specify what is being evaluated, what task it should fulfil and from what perspective to view the measurements. Once this framework has been established, it is possible to direct investigation and measurement towards the data that defines the goals operationally.

The overall goal of our evaluation is stated as:

"To evaluate the prototype system from the industrialist's perspective, with respect to satisfying the system requirements in supporting collaborative design".

7.4. The Questions

Having stated the goal, the process is continued by generating a broad set of questions that can provide some indication of the individual issues encapsulated by the main goal. Table

7-1 shows a set of questions to be answered by the evaluators. The questions were presented to the user through a questionnaire. The actual questionnaire is provided in Appendix A.

Table 7-1: Questions

Goal	
Analyse	the prototype Application
In order to	evaluate its support for collaborative design
with respect to	satisfying the system requirements
from perspective of	end user/designer
in context of	the applicability of the developed system to support collaboration
Questions	
Q1. Does the system have a positive impact on the design process in general?	
Q2. Does the system follow a logical design process?	
Q3. Can the system improve the design process by allowing the design team to work concurrently in the early design stage?	
Q4. Can the features available in the system assist in the sharing of design ideas?	
Q5. Can the system help in reducing the re-design possibility?	
Q6. Can the system help in reducing the design conflicts?	
Q7. Can the system help in reducing the design time?	
Q8. What are your general impression on the on the security features (access right restriction)?	
Q9. Are the default access rights suitable for the assumed design roles?	
Q10. Does data access control provide the appropriate restrictions to data in data ownership, division of responsibility and security?	

Q11. Does the revision tool assist the design process by recording milestones and recover from a design dead end?
Q12. How do you rate the usefulness of the communication tools of the system?
Q13. How do you rate the system capability, comparing with the traditional design tool in terms of improve decision making, reduce delivery time and improve communication?

7.5. Data Collection Approach

The approach used for data collection was through presenting the working prototype software to academics and industry practitioners, and to gauge their response through a questionnaire. This would allow the research to be both peer and industry reviewed. The principal drawback of this methodology, besides its subjectivity, was that the direct presentation of the work could easily introduce bias into the results. In order to minimise these effects, the research was presented as objectively as possible.

Five industry practitioners and seven academics were included in the survey. The participants had experience in building design and the use of information and communication technologies. It was considered that the evaluation group was sufficiently representative to provide a fair assessment of the system.

In order to ensure that the evaluators fully understood the concepts involved in the use of the prototype, the following procedure was followed:

- The evaluators were first given a presentation. This presentation introduced the characteristics and features of the system. The presentation also covered the broader theoretical issues behind the system development.

- This presentation was followed by a demonstration of the prototype application. The demonstration illustrated the main features of the system and a typical case study example for the design of a multi-storey steel framed building (similar to that shown in Chapter 6). The demonstration highlighted the main features of the system and how it can be operated during a collaborative design process.
- A group of the evaluators were engaged in a live collaborative design sessions using a case study over few days. During that time the researcher worked closely with the evaluators to answer their questions about the system use. During the case study, the evaluators played different roles in the design process represented all major players in such design in practice (i.e. Architect, Client, Service Engineer, Structural Engineer)
- The evaluators were then given the questionnaire provided in Appendix A.
- The questionnaires were collected ready for assessment.

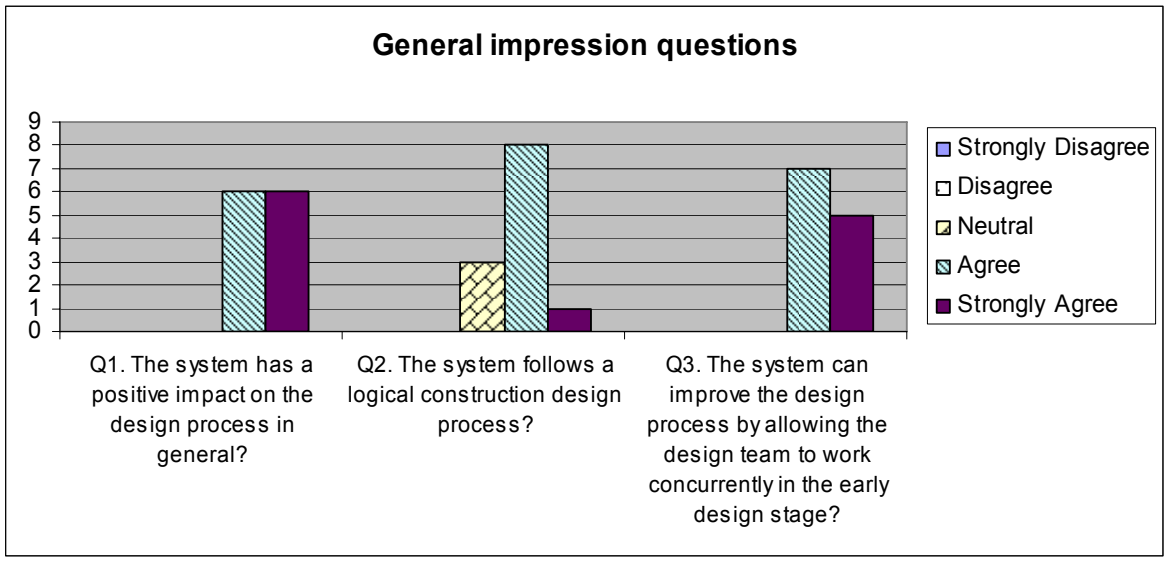
7.6. Evaluation Findings

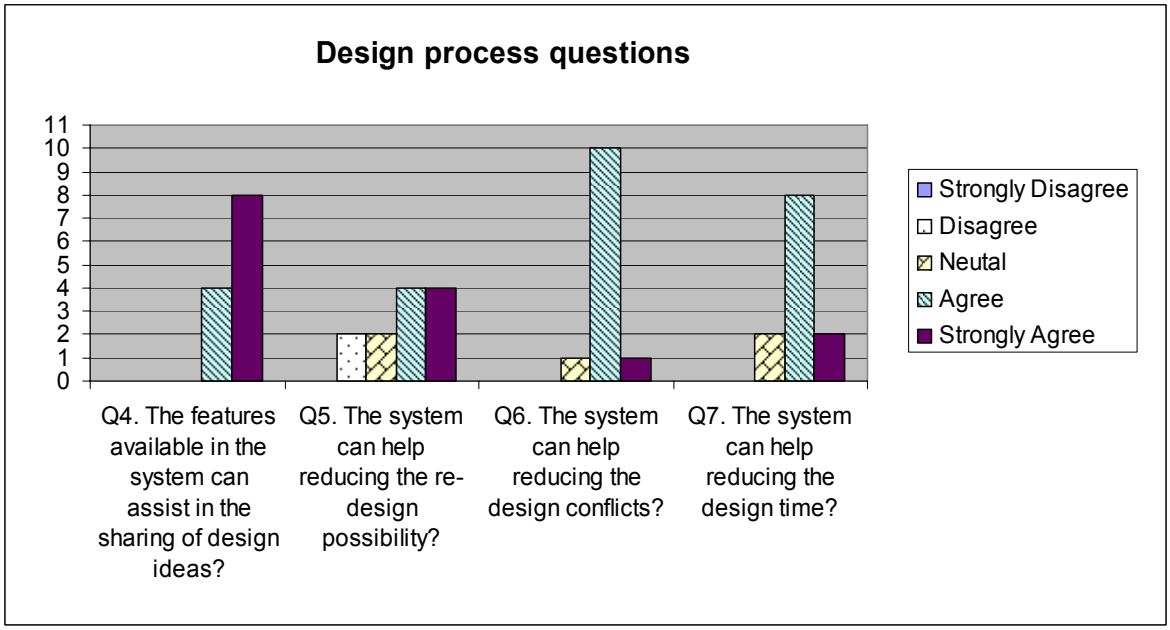
The results of the evaluation in the areas of collaboration support, process understanding, and process improvement can be classified in the following six groups:

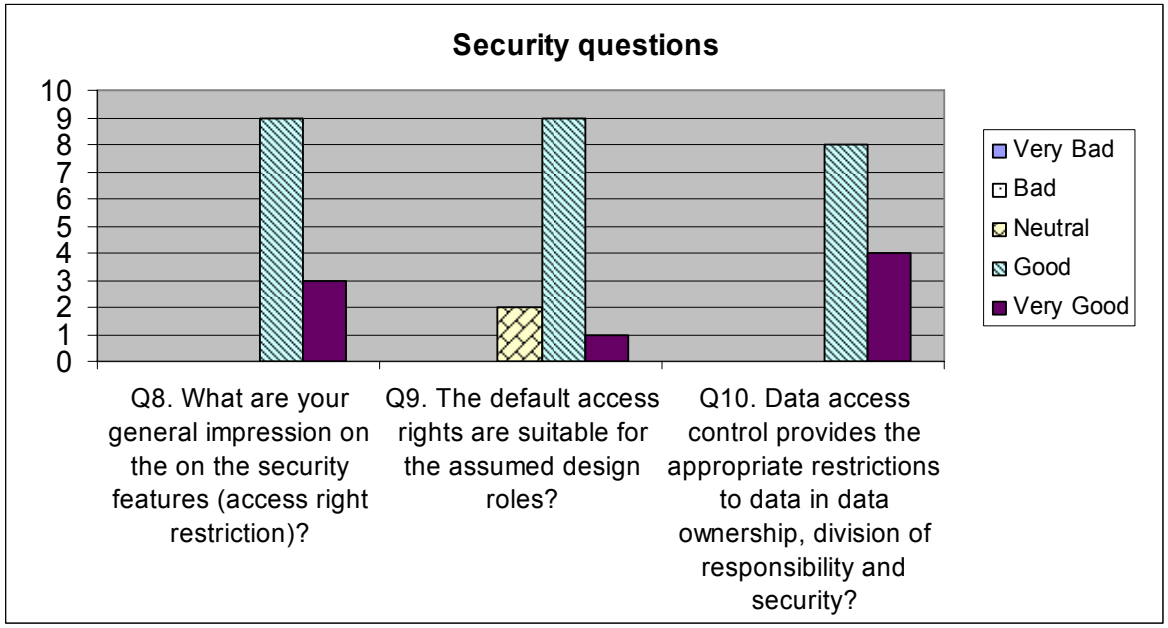
1. General impression: there was general agreement that the system would have a positive impact on the design process and the ability for the design team to work concurrently, especially in the early stage of the design.
2. Design process: Most of the evaluators agreed that the system assists the design team in sharing of design ideas and it can help reducing the design time.

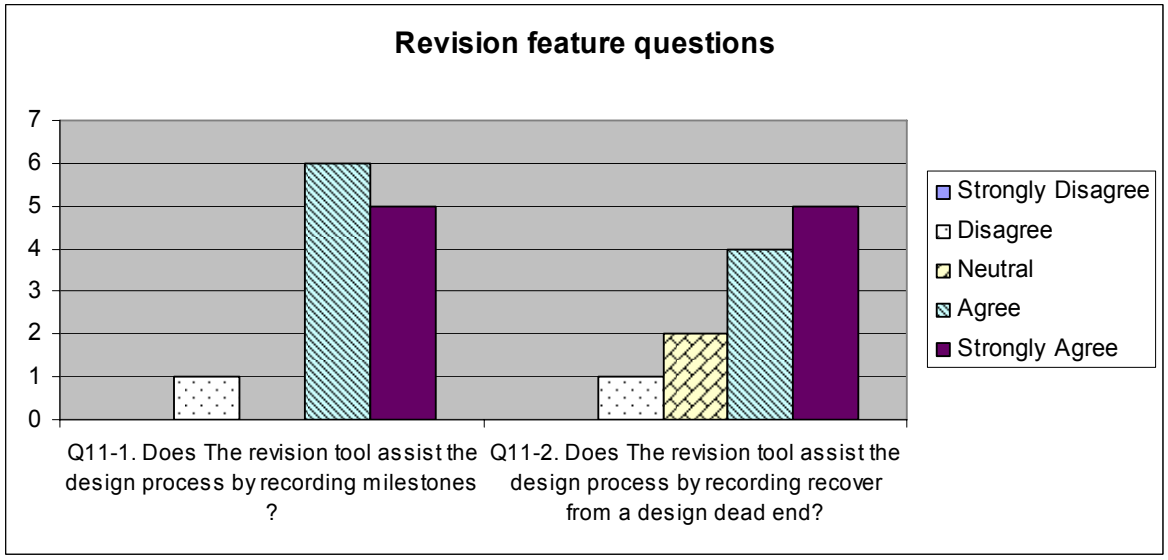
3. Security feature: All evaluators ranked the ability for user to protect their own data and to grant different level of access to the product model either good or very good.
4. Revision feature: All evaluators ranked the ability to restore the product model to a previous version either good or very good. Some evaluators suggested more enhancements on this feature.
5. Communication tools: although the implementation of the communication tools were not as difficult as the system main framework but all evaluators have acknowledged the idea of having comprehensive communication tools built within the design system as very essential.
6. The system versa traditional design tools: The system was ranked good or very good based on the evaluators' feeling of the system compared with the traditional tools in terms of improve decision making during the design process, reduce delivery time and improve communication quality.

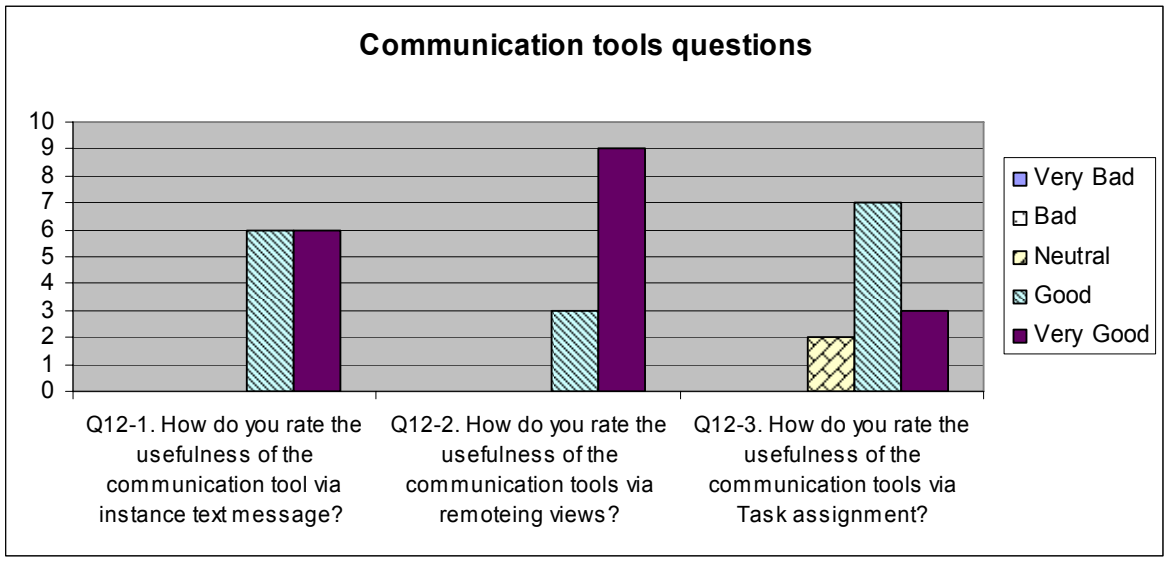
The results of the evaluation of the prototype application are also presented graphically below.

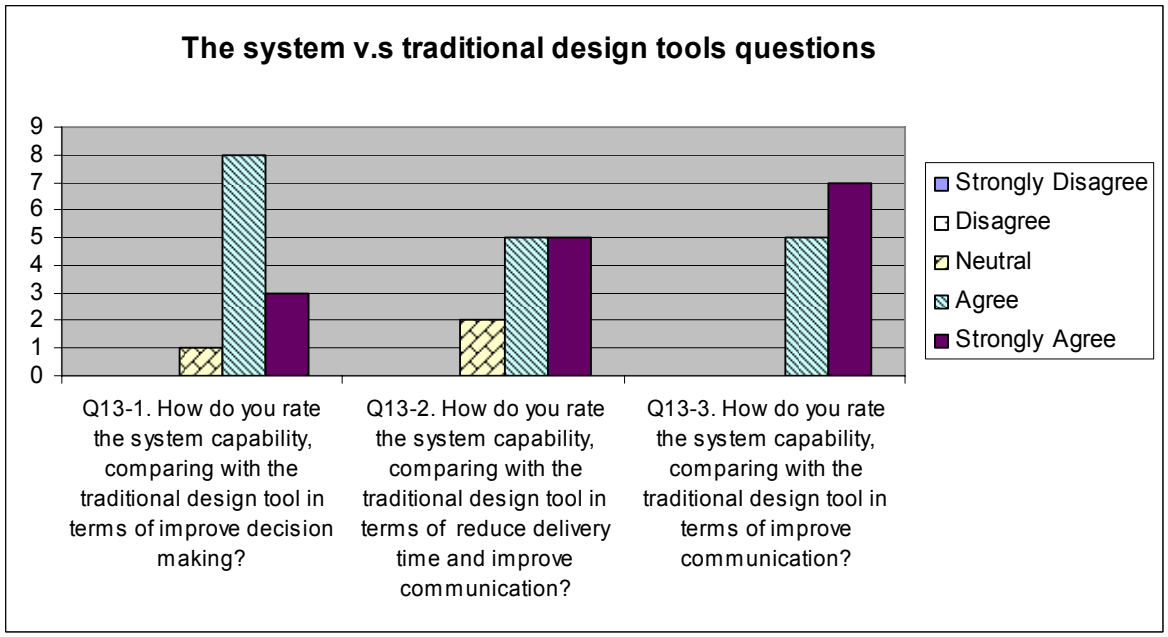

General impression questions
9
8
7
6
5
4
3
2
1
0
Q1. The system has a positive impact on the design process in general?
Q2. The system follows a logical construction design process?
Q3. The system can improve the design process by allowing the design team to work concurrently in the early design stage?
Strongly Disagree
Disagree
Neutral
Agree
Strongly Agree


Design process questions
11
10
9
8
7
6
5
4
3
2
1
0
Q4. The features available in the system can assist in the sharing of design ideas?
Q5. The system can help reducing the re-design possibility?
Q6. The system can help reducing the design conflicts?
Q7. The system can help reducing the design time?
Strongly Disagree
Disagree
Neutal
Agree
Strongly Agree


Security questions
10
9
8
7
6
5
4
3
2
1
0
Very Bad
Bad
Neutral
Good
Very Good
Q8. What are your general impression on the on the security features (access right restriction)?
Q9. The default access rights are suitable for the assumed design roles?
Q10. Data access control provides the appropriate restrictions to data in data ownership, division of responsibility and security?


Revision feature questions
7
6
5
4
3
2
1
0
Strongly Disagree
Disagree
Neutral
Agree
Strongly Agree
Q11-1. Does The revision tool assist the design process by recording milestones ?
Q11-2. Does The revision tool assist the design process by recording recover from a design dead end?





7.7. Evaluators' Recommendations

The evaluators were also asked to suggest future improvements to the system. The recommendations were quite pertinent to further enhancement of the design environment and the design process and can be included in the future versions of this prototype. Two

main directions for the future are envisioned by the evaluators. The first direction is extending the engineering design capabilities and the other direction is enhancing the collaboration capabilities.

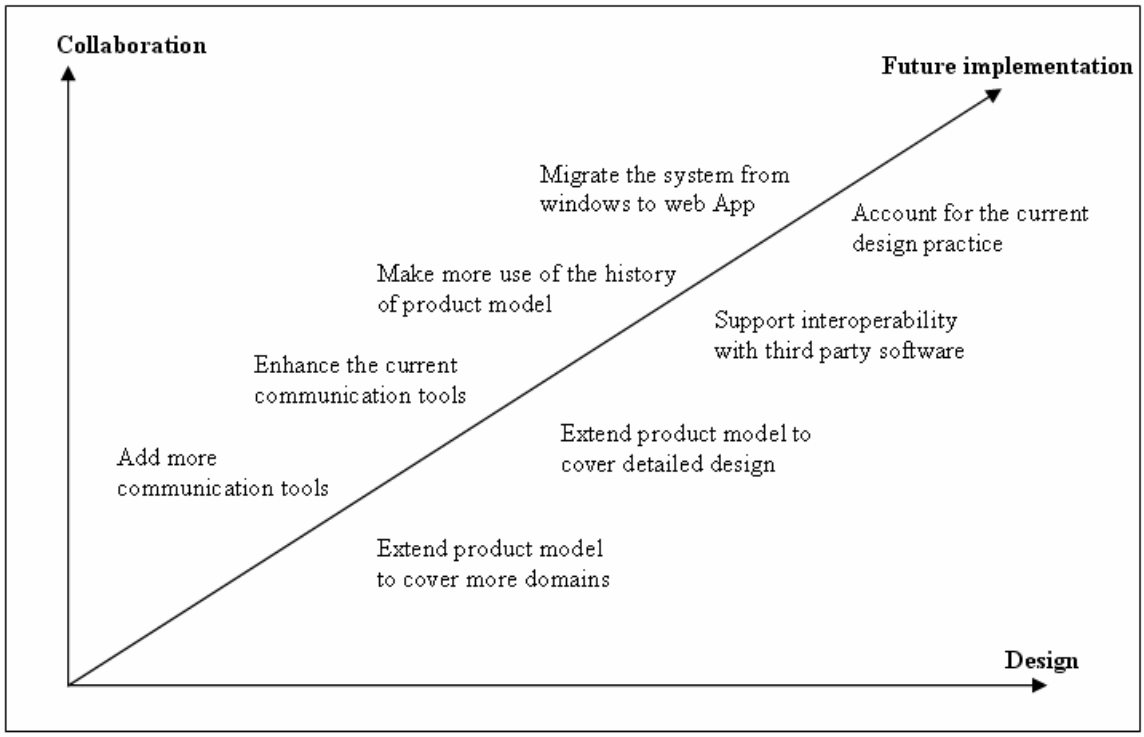


Figure 7-1: Evaluators' recommendations

On the design axis, as it was expected, most evaluators commented on the completion of the product model. Most evaluators thought that for the system to be of practical use, it needs to cover most AEC domains and the product model needs to cover the detailed design stage. Some evaluators also mentioned that every design office has its particular design software applications and the system should be able to interface with these applications through common neutral data formats. Some evaluators from the industry

think that any new building design system suggesting re-engineering the design process should be flexible and adaptable to account for the current design practice.

On the collaboration axis some evaluators suggested adding more communication tools to the system such as voice communication, a bulletin board system and a shared sketching board. Another feature that could be included in the communication tool was the possibility of keeping an archive of the real–time communication (i.e. the instance text messages exchanged) that takes place during the design process similar to the messages exchanged using the Task Management Tool. Some evaluators said that this information could be very viable for accountability issues. Also it was suggested that it would be beneficial to allow the system to carry out some automated communication on behalf of its user. In other words, to enable the system to automatically notify the respective users who would be affected by a change on the product model instead of the user manually examine the change to determine the prospective affected users.

7.8. Perceived Barriers to the Adoption of the New System

There was a general feeling among the evaluators that the introduction of any technology or change needs to take into account cultural and process barriers. The cultural aspects account for people within construction industry. It takes into account the attitudes and feelings of people towards change brought by new technology adoption. The evaluators feel that, for the business to successfully adopt a new technology, the people who are the ultimate users of that technology need to have the appropriate skills, the interest and the positive attitudes toward the change. In terms of the process barriers, the evaluators believe

that companies in the construction industry have their own design practice and in order to gain the maximum benefits of the new technology (e.g. reduced response time and improved integration of activities) they need to investigate the feasibility of the change in their existing processes and ensure the processes ability to adopt change.

In the researcher judgement, the barriers are related to human factors only and thus to solve them we should answer the following question : do researches in the use of modern technologies in the construction industry has always to account for the current design practice to gain the industry acceptance even though they can offer more innovative ways of doing things and they can overcome the shortcomings in the current practice ; or do people in the industry need to realize the potential benefits of the new technologies and be willing to change their attitude and process. The researcher believes that both research centres and the industry should have some degree of flexibility and a willingness to change. Because, if the industry keeps resisting any change either for fear of risking the business or just for fear of change, it will lag far behind other industries. On the other hand, if research centres do not consider the concerns of the industry, they may face plenty of rejection.

7.9. Summary

This chapter described the evaluation of the developed prototype system. It initially discussed the objectives of the evaluation process and explained the methodology adopted for the evaluation including the development of the evaluation questionnaire. The chapter then discussed the results of the evaluation process and the recommendations made by the evaluators.

Chapter 8.

Conclusion and Future Work

This chapter sums up the main elements of this research. The following points are covered in this chapter:

- The research work summary,
- The research contribution,
- The research findings summary,
- Outlines further research directions that may be undertaken.
- Lists the papers published from this work,

8.1. Research Work Summary

Collaborative design is a task distributed across multiple functional perspectives addressing interrelated issues of a single product. Many current researches and reports highlighted the limitations of the current design practice and underlined the importance of effective collaborative design environments. A proper collaborative building design can result in improved on-time delivery of products, improved quality, reduced costs, and increased productivity.

The overall aim of this research was to investigate the feasibility and the potential advantages of using distributed information technology alongside a suitable product model to support multi-disciplinary collaborative design. To carry out this research the author established and met the following objectives:

1. Review typical design process in the construction industry. Chapter Two outlined the design process and the typical activities during the process. This objective met by reviewing of the relevant literature with information drawn from various resources including research and industry publications, the Internet, and conferences.
2. Assess state of the art of collaborative design and the technologies used to support their development. The main focus of this objective was to identify the common used technologies and the techniques to build engineering collaborative design systems. Chapter Three presented the reviewed technologies and the techniques and the current methods of collaboration in building industry.
3. Identify the main functional requirements and specifications of a multi-disciplinary collaborative building design system. Chapter Four and based on the preliminary investigation stage of this research presented these requirements. it also discussed the implementation methods and techniques that the author believes are suitable for a real-time collaborative building design system implementation.
4. Devise a design collaboration system to support data sharing and collaboration in real time. Chapter Five described the development process of a prototype application of a proposed system to support collaborative design workspace and satisfy the main identified requirements. The vision for this system was to provide a collaboration environment through a shared workspace where designers will be able to work collaboratively with adequate level of concurrency on a shared model and encompass most necessary communication and data exchange electronically.

5. Evaluate the effectiveness of the proposed system to support better collaboration during the design process. Chapter Six demonstrated the proposed software system. A case study is used to help presenting the usefulness of the system. Chapter Seven presented the evaluation process of the system through a selected panel of practicing engineers and academic researchers and explained the methodology adopted, including the development of a questionnaire. It finally discussed the results of the evaluation.

8.2. Research Contribution

This work employs modern technology and distributed computing in the area of collaborative building design. The focus is to investigate the feasibility and the potential advantages of using distributed information technology to facilitate distributed collaborative design. Through the development of the prototype system, it is shown that the current technology with a proper system design can provide the functionalities that are required for a collaborative building design system.

The approach in this work proposed a framework and a product model to extend the functionalities of the stand-alone and single-user design systems to facilitate synchronous collaborative design where distributed designers can work concurrently on a shared model and achieve most communication and data exchange electrically.

The proposed framework has the following characteristics:

- Provides a data transaction management approach that ensures adequate concurrent access to the shared model and data consistency between the databases. This is achieved by an efficient and automatic locking/unlocking method.
- Provides efficient information transaction. To improve the system performance and to avoid networking becoming the bottleneck of the design process, the framework implemented various techniques. These are minimising the communication between the clients and the server, minimising the transferred data size, and compressing the data before transferring it.
- Employs software agents to automatically and effectively access and operate on the information to be exchanged among the collaborators.

The proposed product model in this work extends an already developed model to support three important features needed for a collaborative building design system. These are:

- Access right control. Three concepts, namely actor, role, and permission, are integrated with the adopted model to manage the access right to the model data. These concepts are used to define various types and levels of access rights that a designer can be granted. There are generically four types of permissions: allow modify, allow delete, allow create, and allow change ownership. The access rights can be granted by type-level, element-level and attribute-level too. The permissions can be classified on two levels; general or specific. The general permissions define the default actor right to access all elements of a particular type (i.e. type-level). On

the other hand, the specific permissions are element-based permissions (i.e. element-level) and it defines the actor access right to a specific product model element.

- Version management control. This allows the system to track and filter changes made to the model and restore the design to any earlier state. It also allows propagating the model elements history. This is achieved by storing the actions carried out on the product model in a tree of nodes. The product model can then be restored to a previous state by re-executing the actions from the root node to any desired branch of the tree.
- Element changes history: This allows the users to track the changes made upon a product model element. This is achieved by storing the information (i.e. XML messages) used to modify the element data in a list and equipping the system with the capability to covert the stored messages to human readable text.

The proposed framework and the product model in this work have shown their ability to extend the functionalities of traditional design tools to implement a collaborative and concurrent building design environment. The work presented in this research can be the basis for further research, as discussed next, and it can also provide the basis of a new commercial application that is built on the philosophy of collaboration support.

8.3. Research Findings

The research findings can be divided into the system implementation findings drawn from the author gained experience through the development of the prototype application and the evaluation findings drawn from the evaluators of the prototype system. The evaluation findings associated with the evaluator's recommendations were presented in the previous chapter. This section lists some of the findings that the author believes could be useful for a similar prototype implementation.

1. Agent technology is proved to be very useful and effective in encapsulating the system interactive components. The author believes that it is a suitable technology to facilitate communication between homogenous software by warping legacy applications. However, using agents in the same process on the same machine, if not justified, may increase the complexity of the system implementation and affect the system performance because of the unnecessary communication.
2. The generic agent framework has eased the implementation of all system agents.
3. Communication between agents in general has clear overhead on the system performance so various techniques are required to overcome the problem.
4. Using the Action Model as a separate component is believed to be very useful. It eased the system implementation and its maintenance. In other words, separating the data layer (i.e. the Product Model) and the business layer (i.e. the Action Model) has brought the following advantages:

- The code for processing the product model is written once for both client and server side. Any future requirements on the product model processing (e.g. change the design code) can be done in isolation from the product model and can be written once.
- The product model can be restored to any point of its history by re-executing the appropriate actions in sequence based on the model history changes.
- The model file size can be highly minimised since it will be enough to store the messages (i.e. the actions input data) and the model can then be rebuilt by re-executing the appropriate actions with the input from the changes history file. This can be an efficient way of moving the whole model, if required, over the network rather than transferring the whole model data.

8.4. Future Work

Whilst the evaluation findings are encouraging with regards to the practical effectiveness of collaborative design systems and the use of modern distributed application technologies in developing and implementing such systems, there are still some issues that remain to be addressed by further research. Some of which are identified below.

- **Detailed product model**

The product model presented in this research is principally aimed at the conceptual stage of design, though the strategy as a whole is designed to be extendible into the detailed design stage. Future research work on the product model can be carried out in the following two areas:

1. The prototype and analysis tiers could be improved by adopting the IFCs. However, the design intent information requires more research to be carried out in order to improve the level of details to which the building design may be loosely specified at the design intent level.
2. The design intent layer should consider the practicality in identifying the information needs to be captured. This may mean more research that involves real observation or surveys of the current practices of building design. The scalability of the research in this respect has not been tested, and improving the detail of the design intent model may result in a fundamentally more complex model, than the one outlined in this research.

- **Larger domain (Globalization)**

This research has had to limit the domain of the physical building and analysis methods that were considered, in order to allow the completion of the research in a reasonable period and because it is meant as a proof of concept. It is perceived that the benefit of the proposed system would be most apparent with the consideration of more disciplines involved in the design process.

- **Greater use of artificial intelligence**

The design intent information is effectively constrained by the ability of the software agents to convert the design intent model into the prototype model. The current capabilities of the design agents are relatively limited to some simple rules. More sophisticated AI techniques could be implemented. For example, AI techniques capable of positioning building columns and developing the main building structure or other design aspects can be added to the design agent skills. However, the designers should be able to delegate as much or as

little control over the design to the design agents as they require. The current implementation of the generic agent framework is potentially fixable to extend the agents capabilities and can be easily implemented as Plug and Play.

- **Performance**

The communication in the system involves exchanging complex and huge volume of data. The speed of transferring the information from on side to another is a bottleneck on the system performance. The current implementation maximizes the system performance by minimizing the calls between the clients and the servers and pack each massage before submit it. However, more techniques may be implemented to improve the system performance.

- **Improve portability**

The current system is implemented as a client-server system. Client-server systems can be implemented in two different architectures. The first is a thick client with a thin server or a thin client with a thick server. Each of this has its own advantages and disadvantages. The proposed system was implemented as a thick client with a thin server. This type architecture makes the system less portable than the other one. Investigating the possibility to map the system to the other architecture is worthwhile. The common implementation of the other architecture is the web application where the web browser is used as the front user interface and the communication with the web server is carried out over the Internet.

- **Communication protocols**

There are many advances in applying information technology to improve interoperability between homogenous software by providing neutral data formats (e.g. IFC). This can improve the communication between the design team. However, these formats cannot

effectively support concurrency. The author believes that more research in communication protocols for the construction industry can help different design systems communicate in real time. Such communication protocols would have many advantages. The software venders would not have to rewrite their applications and it would be enough to warp up their legacy application with an extra layer (e.g. agency layer) and these layers then communicate using the prospective neutral communication protocol. This mean shorter delivery time and minimum cost.

8.5. Dissemination

The research work carried out as part of this thesis has been disseminated in conference proceedings.

- FAHDAH, I. and TIZANI, W., 2007. A product model for collaborative building design. In: The Ninth International Conference on the Application of Artificial Intelligence to Civil, Structural and Environmental Engineering. pp. paper 1, ISBN 978-1-905088-20-1
- FAHDAH, I. AND TIZANI, W., 2006. Virtual Collaborative Building Design Environment Using Software Agents. In: ISSA, R., ed. 6th International Conference on Construction Applications of Virtual Reality, Orlando, Florida, 3-4 August 2006. Rinker School of Building Construction, University of Florida, pp. On CD
- FAHDAH, I. and TIZANI, W., 2005. Specifications and Design for a Multi-Agent Collaborative Structural Design System. In: B.H.V. TOPPING, ed. the Eighth International Conference on the Application of Artificial Intelligence to Civil,

Structural and Environmental Engineering, Rome, Italy, September. Civil-Comp Press, Stirling, Scotland, pp. Paper 4

8.6. Summary

The researcher believes that the utilisation of the modern information and communication technologies that are currently available can induce positive change in the design process toward a more collaborative and concurrent design. However, the challenge for the construction industry is to realise the power of the new technologies to fashion new patterns of information and communication systems that would greatly impact the product development process. But until this happen advances in the industry are unlikely to be easy and the use of the technologies will be mostly limited to simple and straightforward automation and operations support.

REFERENCES

.NET Framework. .NET Framework Developer's Guide. 2000. Available from http://msdn2.microsoft.com/en-us/library/aa720433(VS.71).aspx [Accessed February 2007].

Alibre. Alibre Design. 2005. Available from www.alibre.com [Accessed February 2007].

Allen, T. and Morton, M. S. 1994, Information technology and the corporation of the 1990s Oxford Univ. Press,New York,

Allinson, K. 1997, Getting there by design Architectural Press, UK

Alshawi, M. and Faraj, I. 2002, Integrated Construction Environments: Technology and Implementation, Construction Innovation, vol. 2, no. 1, pp. 35-51.

Anumba, C. J., Bouchlaghem, N. M., Whyte, J., and Duke, A. 2000, Perspectives On An integrated Construction Project Model, International Journal of Cooperative Information Systems, World Scientific Publishing Company, vol. 9, no. 9, pp. 283-313.

Anumba, C. J., Duke, A., and Baron, G. 1997, Information and Communications Technologies to facilitate concurrent engineering in construction, BT Technology Journal, vol. 15, no. 3, pp. 199-207.

Anumba, C. J., Ren, Z., Thorpe, A., Ugwu, O. O., and Newnham, L. 2003, Negotiation within a multi-agent system for the collaborative design of light industrial buildings, Advances in Engineering Software, vol. 34, no. 7, pp. 389-401.

Anumba, C. J., Ugwu, O. O., Newnham, L., and Thorpe, A. 2002, Collaborative design of structures using intelligent agents, Automation in Construction, vol. 11, no. 1, pp. 89-103.

Backhouse, C. J. and Brookes, N. J. 1996, Concurrent Engineering: What's Working where Gower Publishing, Ltd.,ISBN 0566076667.

Bakis, N., Aouad, G., and Kagioglou, M. 2007, Towards distributed product data sharing environments -- Progress so far and future challenges, Automation in Construction, vol. 16, no. 5, pp. 586-595.

Barkan, P. 1988, Simultaneous Engineering, Design News, vol. 44, p. A30.

Bjork, B. C. 1999, Information technology in construction: domain definition and research issues, International Journal of Computer Integrated Design And Construction, SETO, London, vol. Volume 1, no. Issue 1, pp. 1-16.

Boddy, S., Rezgui, Y., Cooper, G., and Wetherill, M. 2007, Computer integrated construction: A review and proposals for future direction, Advances in Engineering Software, vol. 38, no. 10, pp. 677-687.

Bouchlaghem, D., Shang, H., Whyte, J., and Ganah, A. 2005, Visualisation in architecture, engineering and construction (AEC), Automation in Construction, vol. 14, no. 3, pp. 287-295.

Burns, R. B. 2000, Introduction to research methods, 4 edn, Addison Wesley Longman Australia Pty Limited,

Cadle, J. and Yeates, D. 2004, Project Management for Information Systems Pearson Education,ISBN 0273685805.

CALS. First principles of concurrent engineering: A competitive strategy for electronic product development. 1999.

CATIA. 1998. Available from http://www.appliedgroup.co.uk [Accessed February 2007].

Chang, S. K. 2001, Handbook of Software Engineering & Knowledge Engineering World Scientific,ISBN 981024973X.

CIMsteel. CIMsteel Integration Standards Release 2: Second Edition. 2003. Available from : www.cis2.org [Accessed February 2007].

Cleland, D. I. and Ireland, L. R. 2004, Project Manager's Portable Handbook McGraw-Hill Professional,ISBN 0071437746.

CollabCAD. 1999. Available from www.collabcad.com [Accessed February 2007].

COM+. Microsoft Component Object Model. 1993. Available from http://msdn2.microsoft.com/en-us/library/ms685978(VS.85).aspx [Accessed February 2007].

CORBA. Common Object Request Broker Architecture. 1997. Available from http://www.omg.org/gettingstarted/corbafaq.htm [Accessed February 2007].

Coulouris, G., Dollimore, J., & Roberts, M. Role and Task-based Access Control in the PerDiS Groupware Platform. 1998. Department of Computer Science and Queen Mary and Westfield CollegeUniversity of London.

Cutkosky, M. R., Engelmore, R. S., Fikes, R. E., Genesereth, M. R., Gruber, T. R., Marh, W., Tenenbaum, J. M., & Weber, J. C. PACT:An Experiment in Integrating Concurrent Engineering Systems. 1993.

Dawood, N., Hobbs, B., Akinsola, A., Mallasi, Z. 2000, The Virtual Construction Site (VIRCON) - A Decision Support System for Construction Planning, CONVR 2000, University of Teesside, Middlesbrough, UK.

Dawood, N., Sriprasert, E., Mallasi, Z., and Hobbs, B. 2003, Development of an integrated information resource base for 4D/VR construction processes simulation, Automation in Construction, vol. 12, no. 2, pp. 123-131.

DCOM. Distributed Component Object Model. 1996. Available from http://msdn2.microsoft.com/en-us/library/ms809340.aspx [Accessed February 2007].

Dhillon, B. S. 2002, Engineering and Technology Management Tools and Applications Artech House,

DirectX. Microsoft DirectX. 1994. Available from http://www.microsoft.com/windows/directx/default.mspx [Accessed February 2007].

Durfee, E. H., Lesser, V. R., and Corkill, D. D. 1989, Trends in Cooperative Distributed Problem Solving, IEEE Transactions on Knowledge and Data Engineering, vol. 1, no. 1, pp. 63-83.

Egan, S. J. 1998, Rethinking Construction, Report of the Construction Task Force on the Scope for Improving the Quality and Efficiency of the UK Construction Industry, Department of Environment, Transport and the Regions (DETR), London, UK.

Evbuomwan, N. F. O. and Anumba, C. J. 1995, Concurrent life cycle design and construction, in: Developments in Computer-Aided Design and Modelling for Civil Engineers (B.H.V. Topping), Civil-Comp Press, Edinburgh, pp. 93-102.

Faraj, I., Alshawi, M., Aouad, G., Child, T., and Underwood, J. 2000, An industry foundation classes Web-based collaborative construction computer environment: WISPER, Automation in Construction, vol. 10, no. 1, pp. 79-99.

Ferber, J. 1999, Multi-agent systems: An introduction to Distributed Artificial Intelligence Addison-Wesley., London

Finin, T., Fritzson, R., McKay, D., McEntire, R. 1994, KQML as an agent communication language. In: Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94), p. 456.

Fu, C., Aouad, G., Lee, A., Mashall-Ponting, A., and Wu, S. 2006, IFC model viewer to support nD model application, Automation in Construction, vol. 15, no. 2, pp. 178-185.

Ganesan, S. 1997, Advances in Concurrent Engineering CRC Press,ISBN 1566766044.

Garson, D. 2002, Guide to Writing Empirical Papers, Theses, and Dissertations NY: Marcel Dekker, Inc.,

Geng, H. 2004, Manufacturing Engineering HandbookISBN 0071398252.

Gilbert, D. & Janca, P. 1997, White Paper on Intelligent Agents. Technical report,IBM Research.

Grosso, A., Gozzi, A., Coccoli, M., & Boccalatte, A. An Agent Programming Framework Based on the C# Language and the CLI. 2003. First International Workshop on C# and .NET Technologies on Algorithms, Computer Graphics, Visualization, Distributed and WEB Computing. University of West Bohemia, Plzen, Czech Republic. Copyright UNION Agency - Science Press ISBN 80-903100-3-6.

Han, S. C., Kunz, C. J., and Law, H. K. 1999, Building Design Services in a Distributed Architecture, Journal of Computing in Civil Engineering, vol. 13, no. 1, pp. 12-22.

Haoxue, M. 2004, Integration and Communication of Engineering Information in Collaborative Design, PhD, Luleå University of Technology.

Hayes-Roth, B. An Architecture for Adaptive Intelligent Systems. Artificial Intelligence: Special Issue on Agents and Interactivity 72, 329-365. 1995.

Hiremath, H. R. and Skibniewski, M. J. 2004, Object-oriented modeling of construction processes by unified modeling language, Automation in Construction, vol. 13, no. 4, pp. 447-468.

IDEF0. Integration Definition for Function Modeling. 1993. Available from www.idef.com [Accessed February 2007] .

IFC. IAI International web site. 2001. Available from www.iai-international.org [Accessed February 2007] .

IMS. IFC Model Server. 2002. Available from http://cic.vtt.fi/projects/ifcsvr [Accessed February 2007] .

ISO 10303-11. Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual. 1994.

Janis, k. 2000, Using IT for competitive advantage,CIIA 2000 5th Annual Conference.

Java RMI. Java Remote Method Invocation. 1994. Available from http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp [Accessed February 2007].

Johannsen, A., Haake, J., & Streitz, N. Telecollaboration in Virtual Organisations : The Role of Ubiquitous Meeting Systems. 1996. GMD Arbeitsbericht No. 974.

KeyNote 2006, Construction Industry Market Review, UK.http://www.keynote.co.uk [Accessed February 2007].

Lacourse, D. E. 1995, Handbook of Solid Modeling McGraw-Hill,ISBN 0070357889.

Laitinen, J. 1998, Model Based Construction Process Management, KTH doctoral thesis.

Latham, M. Constructing the Team, Final Report of the Government/Industry review of procurement and contractual arrangements in the UK Construction Industry, HMSO, London. 1994.

Lee, A. S., Liebenau, J., and DeGross, J. I. 1997, Information Systems and Qualitative Research Springer,ISBN 0412823608.

Li, W. D., Lu, W. F., Fuh, J. Y. H., and Wong, Y. S. 2005, Collaborative computer-aided design--research and development status, Computer-Aided Design, vol. 37, no. 9, pp. 931-940.

Li, W. D. and QIU, Z. M. 2006, State-of-the-art technologies and methodologies for collaborative product development systems, International Journal of Production Research, vol. 44, no. 13, pp. 2525-2559.

Makanae, K. 2003, Development of the VR-CAD System for Landscape and Town Planning, CONVR 2003, Virginia Tech.

Maner, W. Rapid Application Development using Iterative. 1997. Available from: - http://csweb.cs.bgsu.edu/maner/domains/RAD.gif [Accessed February 2007].

Messner, J. and Horman, M. 2003, Using Advanced Visualisation Tools to Improve Construction Education, ConVR 2003, Virginia Tech.

Mills, A. Collaborative Engineering and the Internet, Society of Manufacturing Engineers, Dearborn, MI, USA. 1998.

National Institute of Standards and Technology (NIST). EXPRESS Information Modelling Language. 2000. Available from: http://www.eurpc2.demon.co.uk/part1.htm [Accessed February 2007].

NIST. Cost Analysis of Inadequate Interoperability in the U.S Capital Facilities Industry. 2004. available at http://www.bfrl.nist.gov/oae/publications/gcrs/04867.pdf. [Accessed 20th February 2007] .

Oliveira, E., Fischer, K., and Stepankova, O. 1999, Multi-agent systems: which research for which applications, Robotics and Autonomous Systems, vol. 27, no. 1-2, pp. 91-106.

OneSpace.net. 2005. Available from www.OneSpace.net [Accessed February 2007].

OpenGL ARB 2004, The OpenGL Reference Manual – The Bluebook, 1.4 edn, Addison-Wesley Professional,

OpenHSF. 2004. Available from www.hoops3d.com/ [Accessed February 2007].

OST 1995, Technology Foresight; Progress through Partnership; Construction, Office of Science and Technology, London.

Parunak 1998, What can Agents do in Industry, and Why?An Overview of Industrially-Oriented R&D at CEC.

Paulraj, P. 2003, Database Design and Development: An Essential Guide for IT Professionals John Wiley & Sons,ISBN 0471218774.

Pena-Mora, F., Hussein, K., Vadhavkar, S., and Benjamin, K. 2000, CAIRO: a concurrent engineering meeting environment for virtual design teams, Artificial Intelligence in Engineering, vol. 14, no. 3, pp. 203-219.

Plume, J. and Mitchell, J. 2007, Collaborative design using a shared IFC building model--Learning from experience, Automation in Construction, vol. 16, no. 1, pp. 28-36.

Prasad, B. 1996, Concurrent Engineering Fundamentals Upper Saddle River, New Jersey: PTR Prentice Hall,

Ruikar, D. 2005, Design Process Improvement Using A Model Based Approach, PhD, The University of Nottingham.

Russell, S. J. Rationality and Intelligence. 94, 57-77. 1997. Artificial Intelligence.

Saad, M. and Maher, M. 1995, Exploring the possibilities for computer support for collaborative designing, In: M. Tan and R. Teh, Editors, The Global Design Studio, Centre for Advanced Studies in Architecture, University of Singapore , pp. 727-738.

SABLE. SABLE Project. 2002. Available from http://www.blis-project.org/~sable [Accessed February 2007].

Scott, M. L. 2006, Programming Language Pragmatics Morgan Kaufmann,ISBN 0126339511.

Shen, H. and Dewan, P. 1992, Access control for collaborative environments, Proceedings of the 1992 ACM conference on Computer-supported cooperative work, Toronto, Ontario, Canada , pp. 51-58.

Shen, W., Norrie, D. H., and Barthes, J. P. A. 2001, Multi-agent systems for concurrent intelligent design and manufacturing Taylor & Francis, LondonISBN 978-0748408825.

Shen, W. and Barthes, J. P. 1996, An Experimental MultiAgent Environment for Engineering Design, International of Cooperative Information Systems, vol. 5, no. 2/3, pp. 131-151.

Shepperd, M. 1995, Foundations of Software Measurement Prentice Hall,ISBN 0133361993.

Shiratuddin, M., Yaakub, A., Che Mohamed Arif 2000, Utilising First Person Shooter 3D Game Engine in Developing Real World Walkthrough Virtual Reality Application: A Research Finding, University of Teesside, Middlesbrough, UK, CONVR 2000.

Shoham, Y. 1993, Agent-oriented programming. Artificial Intelligence, 60(1), p.51-92, 1993.

Smith, R. A. 2005, Design Centric Information And Process Modelling For Integrated Building Design, PhD,The University of Nottingham.

Smith, R. A., Tizani, W., Ruikar, D. 2003, Modelling design constraints for an automated design process, Egmond aan Zee, ed., Proceedings of the Seventh International Conference on The Application of Artificial Intelligence to Civil and Structural Engineering, The Netherlands.

Sun, M. and Aouad, G. 1999, Control Mechanism for Information Sharing in an integrated Construction Environment, CEC99, Espoo, Finland, pp. 121-30.

Thorpe, A., Baldwin, A. N., Carter, C., Leevers, D., Madigan, D. 1995, Multimedia Communications in Construction, Proceedings of the Institution of Civil Engineers, pp. 12-16.

Ugwu, O. O., Anumba, C. J., and Thorpe, A. 2001, Ontology development for agent-based collaborative design, Engineering Construction and Architectural Management, vol. 8, no. 3, pp. 211-224.

UML. OMG Unified Modeling Language Specification. 1997. Available from www.omg.org [Accessed February 2007].

Unigraphics. 1991. Available from http://www.ugs.com/index.shtml [Accessed February 2007].

W.D.LI & Z.M.QIU. State-of-the-art technologies and methodologies for collaborative product development systems. 2005.

Wikipedia 2007, Wikipedia, Available from www.wikipedia.com [Accessed February 2007] .

William Xu, X. and Liu, T. 2003, A web-enabled PDM system in a collaborative design environment, Robotics and Computer-Integrated Manufacturing, vol. 19, no. 4, pp. 315-328.

Winner, R. I. e. a. 1988, The Role of Concurrent Engineering in Weapons Systems Acquisition, Institute for Defense Analyses(IDA), R-338.

Wooldridge, M. 2002, An introduction to multi-agent systems. John Wiley & Sons, Chichester, England

XML. World Wide Web Consortium. 1998. Available from www.w3.org [Accessed February 2007] .

Ye, Y. and Churchill, E. F. 2003, Agent Supported Cooperative Work Springer,ISBN 1402074042.

Zhao, G., Deng, J., and Shen, W. 2001, CLOVER: an agent-based approach to systems interoperability in cooperative design systems, Computers in Industry, vol. 45, no. 3, pp. 261-276.

APPENDIX A

EVALUATION QUESTIONNAIRE

Name (optional):

Role:

Design experience in years:

Email:

Note: if you prefer to complete the questionnaire some other time and use the computer to fill it in, please specify your email address so we email it to you.

General impression

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Q1. The system has a positive impact on the design process in general?	θ	θ	θ	θ	θ
Q2. The system follows a logical design process?	θ	θ	θ	θ	θ
Q3. The system can improve the design process by allowing the design team to work concurrently in the early design stage?	θ	θ	θ	θ	θ

Design process

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Q4. The features available in the system can assist in the sharing of design ideas?	θ	θ	θ	θ	θ
Q5. The system can help reducing the re-design possibility?	θ	θ	θ	θ	θ
Q6. The system can help reducing the design conflicts?	θ	θ	θ	θ	θ
Q7. The system can help reducing the design time?	θ	θ	θ	θ	θ

Security feature

	Very Bad	Bad	Neutral	Good	Very Good
Q8. What is your general impression on the security features (access right restriction)?	θ	θ	θ	θ	θ
Q9. The default access rights are suitable for the assumed design roles?	θ	θ	θ	θ	θ

Q10. Data access control provides the appropriate restrictions to data in?

	Very Bad	Bad	Neutral	Good	Very Good
Data ownership	θ	θ	θ	θ	θ
Division of responsibility	θ	θ	θ	θ	θ
Security	θ	θ	θ	θ	θ

Revision feature

Q11. The revision tool assists the design process by?

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Recording milestones	θ	θ	θ	θ	θ
Recover from a design dead end	θ	θ	θ	θ	θ

Communication tools

Q12. How do you rate the usefulness of the following communication tools of the system?

	Very Bad	Bad	Neutral	Good	Very Good
Instance text message	θ	θ	θ	θ	θ
Remoting views	θ	θ	θ	θ	θ
Task assignment tool	θ	θ	θ	θ	θ

The system v.s traditional design tools

Q13. How do you rate the system capability, comparing with the traditional design tool, to achieve the following goals?

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

Reduce data duplication	θ	θ	θ	θ	θ
Reduce error	θ	θ	θ	θ	θ
Improve decision making	θ	θ	θ	θ	θ
Reduce delivery time	θ	θ	θ	θ	θ
Reduce cost	θ	θ	θ	θ	θ
Improve communication	θ	θ	θ	θ	θ

System improvements

Q14. What other communication tool do you think is needed?

Q15. Is there any additional security features you would like to see in the system?

Q16. Which parts or features of the system you found particularly useful?

Q17. Which features of the system you did not find useful?

Q18. What are the barriers to the adoption of such system for real design scenario?

Q19. What are the additional features or requirements of a real-time virtual collaborative design environment that you would like to add to the system?

Q20. Any additional comments:

Thank you