Distributed Collaboration:
Engineering Practice Requirements

M. Deacon
13586440

Thesis presented in fulfilment of the requirements for the degree of Master of Civil Engineering at the University of Stellenbosch

Study Leader: Dr G.C. van Rooyen

March 2007
Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and has not previously in its entirety or in part been submitted at any university for a degree.

Ek, die ondergetekende verklaar hiermee dat die werk gedoen in hierdie tesis my eie oorspronklike werk is wat nog nie voorheen gedeeltelik of volledig by enige universiteit vir ’n graad aangebied is nie.

Signature: Date:
Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own work original work and has not previously in its entirety or in part been submitted at any university for a degree.

Ek, die ondergetekende verklaar hiermee dat die werk gedoen in hierdie tesis my eie oorspronklike werk is wat nog nie voorheen gedeeltelik of volledig by enige universiteit vir n graad aangebied is nie.

Signature: ___________________________________________ Date: ______________________________
Acknowledgements

Dr G.C. Van Rooyen, my study leader, for the insight and guidance.

A.H. Olivier, for his assistance and ideas in the software development.

Anton Eygelaar and Eliz-Mari Lourens, for their ideas and working with them over the past two years.

My parents, Waldo and Francis Deacon, for the support they have given me through the years and for giving me the opportunity to further my studies.

Janie Pretorius, Georg Beckert and Andrew Cerff, for their continual encouragement, support and dreams.

Jesus Christ, for giving me life and purpose in everything.
Synopsis

An extended project was undertaken to develop structural design software (called the integrated software) that supports network collaboration. Three projects preceded this thesis study in which the development of the integrated software was initiated. In these projects three software architectures were developed for a finite element model, a structural steel member design model and a structural steel connection design model. These projects cover the analysis and design aspects of the integrated software.

This thesis study addresses the communication aspects of the integrated software. The communication aspects include communication between the various modules of the integrated software as well communication between people and between people and the software. No graphical user interface for the creation of finite element models was developed in the preceding projects, which was done in this thesis. The models developed in the preceding projects must be able to communicate with one another in order for the software to operate as a whole. Some of the communication links required in the integrated software are established in this thesis study. The communication of the integrated software is not to be confined to a local workstation. Therefore a software architecture is built into the integrated software in order to support network communication, thereby making network-based collaborative design a real possibility.

The integrated software that is being developed is specifically for use by structural engineers. Therefore the engineers’ opinion of such design software that supports network collaboration is invaluable. In the last part of the thesis practicing engineers the views of are reported on topics of how collaborative designs could be done in practice and how it could be supported by design software. The results of the interviews are then summarized and an assessment is made of the engineers’ requirements for software that supports network collaboration. Finally recommendations are made for the future development of the integrated software.
Opsomming

‘n Oorhoofse projek was onderneem om strukturele ontwerpsagteware (genoem die geïntegreerde sagteware) te ontwikkel wat netwerk gebaseerde samewerking ondersteun. Die ontwikkeling van die geïntegreerde sagteware het begin met drie projekte, wat afgehandel is voor die begin van die tesis. In die projekte is was sagteware argitekte ontwikkel vir ‘n eindige element model, ‘n strukturele element ontwerpsmodel en ‘n strukturele verbindingsontwerpsmodel. Die projekte spreek die analise en ontwerpsaspekte aan van die geïntegreerde sagteware.

Die tesis spreek die kommunikasie aspekte van die geïntegreerde sagteware aan. Dit sluit in die kommunikasie tussen die verskeie modules van die geïntegreerde sagteware asook kommunikasie tussen mense, en tussen mense en die sagteware. Geen grafiese gebruikerskoppelvlak vir die skepping van eindige element modelle was ontwikkel in die voorafgaande projekte nie. Dit gebruikerskoppelvlak is ontwikkel in die tesis. Die modelle wat ontwikkel is in die voorafgaande projekte moet met mekaar kan kommunikeer om te werk as ‘n geheel. Van die kommunikasieskakels wat benodig is in die geïntegreerde sagteware is bewerkstellig in die tesis. Kommunikasie van die geïntegreerde sagteware is nie beperk tot ‘n lokale werkstasie nie. Dus is daar ‘n sagteware argitektuur ingebou in die geïntegreerde sagteware om kommunikasie oor ‘n netwerk te ondersteun. Sodoende word die moontlikheid van netwerk gebasseerde samewerkendeontwerp ‘n werkljidheid.

Die geïntegreerde sagteware word spesifiek ontwerp vir die gebruik van strukturele ingenieurs. Dus is die opinie van die ingenieurs omtrent sagteware wat netwerk gebasseerde samewerking ondersteun besonders belangrik. In die laaste deel van die tesis word die oogpunte van praktiserende ingenieurs, oor onderwerpe soos hoe samewerking in die praktyk gedoen word en hoe dit ondersteun kan word deur sagteware, gerapporteer. Die resultate van die onderhoude is saamgevat en ‘n “assessment” word gemaak van wat ingenieurs se vereistes is van van sagteware wat netwerk gebasseerde samewerking ondersteun. Aan die einde word voorstelle gemaak vir die verdere ontwikkeling van die ontwerpsagteware.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECLARATION</td>
<td>i</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>ii</td>
</tr>
<tr>
<td>SYNOPSIS</td>
<td>iii</td>
</tr>
<tr>
<td>OPSOMMING</td>
<td>iv</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xi</td>
</tr>
<tr>
<td>GLOSSARY</td>
<td>xii</td>
</tr>
</tbody>
</table>

## 1 INTRODUCTION

1.1 Background Information | 1
1.2 Scope and Limitations | 2
1.3 Work Packages | 5
1.4 Plan of Development | 5

## 2 PRECEDING PROJECTS

2.1 The Finite Element Model | 6
2.2 The Steel Member Model | 8
2.3 The Steel Connection Model | 15
2.4 Final Remarks | 17

## 3 THE FINITE ELEMENT MODEL CREATOR

3.1 GUI Architecture | 19
  3.1.1 FEMFrame | 19
  3.1.2 FEMPanel | 20

## 4 LINKING THE FE MODEL AND THE STEEL MEMBER MODEL

4.1 Connecting the Steel Member Model to the FE Model | 24
  4.1.1 Persistent Identifiers | 24
  4.1.2 Edit GUI of the Steel Member Model | 25
4.1.3 The Unit Model 27
4.2 Restructuring of the Software 29

5 DISTRIBUTED COLLABORATION 30
5.1 Remote Method Invocation 30
5.2 Communication Architecture 31
5.2.1 The Server 32
5.2.2 The Client 34

6 SYSTEMATIC WALKTHROUGH OF INTEGRATED SOFTWARE 36

7 INTERVIEWS: COLLABORATION TYPES 43
7.1 Sequential Collaboration 43
7.1.1 In-Phase Sequential Collaboration 44
7.1.2 Between-Phase Sequential Collaboration 44
7.1.3 Interview Findings 44
7.2 Parallel Collaboration 46
7.2.1 Parallel Collaboration in the Development of Finite Element Models 47
7.2.2 Parallel Collaboration in Detail Design of Structures. 47
7.2.3 Parallel Collaboration in the Design of Independent Sections. 48
7.2.4 Interview Findings 48
7.3 Interactive Collaboration 53
7.3.1 Interview Findings 53
7.4 Feasibility of Collaborative Design 55
7.5 Future Development 56

8 INTERVIEWS: GENERAL ASPECTS 60
8.1 File Management 60
8.2 Networks 61
8.3 Data Recording 62
8.4 History 63
8.5 Recommendations for Design Software 65
8.5.1 FE modelling 65
8.5.2 Detail Design 65
8.5.3 Links Between Disciplines 66
8.5.4 Design Software Extras 67
8.6 Future Implementation 68

9 CONCLUSIONS 70
9.1 Software Development 70
9.2 Interview Findings 71

References 73

Addendum A: Survey I
Addendum B: Interview Data VIII
Addendum C: Source Code IX
# List of Figures

Figure 1.1: Flow of Information during the Design Process ............................................................. 3  
Figure 2.1: Bending Moments of an Analysed of a Portal Frame ...................................................... 7  
Figure 2.2: Principal Stress Distribution of a Slab using Plate Elements ........................................ 8  
Figure 2.3: Displacement of a Column Analysed with 3D Brick Elements ..................................... 8  
Figure 2.4: Defining of a Structural Member ................................................................................. 9  
Figure 2.5: Lateral Restraints along the Length of a Structural Member ....................................... 11  
Figure 2.6: Steel Member Model – Defining Structural Members ................................................. 11  
Figure 2.7: Selection of Profiles for Structural Members .............................................................. 12  
Figure 2.8: Calculation Results of Adequate Profiles .................................................................... 13  
Figure 2.9: Calculation Results for a Profile that Failed ............................................................... 14  
Figure 2.10: Application of the Design of Structural Steel Connections ....................................... 17  
Figure 3.1: FEM Creator – Creating Finite Elements ..................................................................... 18  
Figure 3.2: FEM Creator – Creating Supports ............................................................................. 19  
Figure 3.3: Fem Creator – Creating Loads .................................................................................... 19  
Figure 3.4: Architecture of Mouse-Listeners in the FEMPanel ....................................................... 21  
Figure 3.5: Node Property Popup Frame .................................................................................... 22  
Figure 3.6: Architecture of Popups in the FEMPanel .................................................................... 23  
Figure 4.1: Event-Listener Architecture used in the DrawPanel .................................................... 25  
Figure 4.2: Code for Tool the Implements Interface DrawPanel.Listener ...................................... 26  
Figure 4.3: Structural Layout of the Unit Models .......................................................................... 27  
Figure 4.4: Layout of the Unit Model .......................................................................................... 28  
Figure 4.5: Overview of the Software Structure .......................................................................... 29  
Figure 5.1: RMI Concept .......................................................................................................... 31  
Figure 5.2: Communication Architecture for Collaborative Design .......................................... 32  
Figure 5.3: Remote Open Dialog .................................................................................................. 33  
Figure 5.4: Server Application ................................................................................................... 33  
Figure 5.5: Client Application .................................................................................................... 34
DISTRIBUTED COLLABORATION:
ENGINEERING PRACTICE REQUIREMENTS

Figure 6.1: Server Setup Definition .................................................................36
Figure 6.2: Blank Server Application ...............................................................36
Figure 6.3: Client Setup Definition .................................................................37
Figure 6.4: Blank Client Application ...............................................................37
Figure 6.5: Client Registered on Server Application .................................38
Figure 6.6: Development of FE Models in the Client Application ..........38
Figure 6.7: Default Opening of the Remote Browser .................................39
Figure 6.8: Creating a Directory on the Server ...........................................39
Figure 6.9: FE Model File is Stored in the Created Directory ...............40
Figure 6.10: Steel Member Model Loaded in the Client Application ....40
Figure 6.11: Selecting the FE Model for the New Steel Member Model ...41
Figure 6.12: Structural Steel Member Design Being Done in the Client Application ....41
Figure 6.13: Files Created Stored on the Server .......................................42
Figure 6.14: Activity of Clients Shown on the Server ...............................42
Figure 7.1: Sequential Collaboration within a Phase ...............................44
Figure 7.2: Sequential Collaboration between Phases ...............................44
Figure 7.3: Side-view a Multi-storey Building’s FE Model to be Developed. ....47
Figure 7.4: Side view a Multi-storey Building’s Detail Design to be done. ....47
Figure 7.5: Sections of an Airport Subdivided for Design .......................48
Figure 7.6: A Local Floor Being Designed of a Tall Building ..................50
Figure 7.7: Building with Sections Below and Above Ground Level ..........50
Figure 7.8: Parallel Collaboration in Stadiums ..........................................51
Figure 7.9: Interactive Collaboration Model ..............................................53
Figure 7.10: Usage of Interactive Collaboration .........................................54
Figure 7.11: The Impact of Software Supporting Design Collaboration from Remote Locations ....55
Figure 7.12: Priority of Future Software Development According to Engineers ....56
Figure 7.13: Sequential and Interactive Collaboration ..............................57
Figure 7.14: FE Nodes Moved of a Structural Member (a) in a straight line and (b) in a bend ....58
Figure 8.1 History Record of the Development of a Building ..................63
Figure 8.2: Editing Models Simultaneously and Managing Conflicts. ................................................64

Figure 8.3: (a) Continuous Column (b) Discontinuous Column .......... Error! Bookmark not defined.
List of Tables

Table 2.1: Structural Member Types.................................................................9
Table 2.2: End Condition of Structural Members ..............................................10
Table 2.3: Steel Connections ............................................................................15
GLOSSARY

Data Model: Any software model of which only the data is recorded when it is stored. No record of the development process is recorded.

Design model: A model in which detail design aspects of structural design is done. In this case either a steel member of steel connection model.

Extended project: The overall project in which the network-based collaborative design software is being developed and of which the preceding projects and this thesis forms a part of.

Graphical user interface: An interface that assists engineers to do structural design work by means of graphics that are displayed on a computer screen.

Integrated Database: A database that contains all the design information for the various models used in the design of structures.

Integrated Software: All the software included to create a design software that supports network collaboration i.e. all the software discussed in this thesis study.

Interactive Collaboration: is where several users work together on the development and editing of a single model. When a change is made by one user, the change is immediately seen by all users working on the model.

Merging requirements: Requirements that have to be met when two separate models are merged together in parallel collaboration.

Network-based collaborative design software: A software that supports collaboration over a computer network for the design of structures.

Parallel Collaboration: Parallel collaboration is where two or more portions of design work are done at the same time by different people, where the various portions of work add to the overall design of a structure.

Parent Model: A model that provides information for a subsequent model.

Preceding projects: Projects that were completed before the commencement of this project and which forms a part of the software development project.

Sequential Collaboration: Two or more people work together on consecutive design tasks that are related. One or more tasks have to be completed before one or more tasks can commence.

Series Model: Any software model of which all the changes are recorded when it is stored. The model records the data contained in the model and its history of development.

Steel connection model: A model in which the steel connections between structural steel members are designed.

Steel member model: A model in which the structural steel members are designed.

Structural Member: A member of a structure that influences the structural behaviour of a structure.

Subsequent Model: A model that is dependent on a parent model for information.
1 Introduction

1.1 Background Information

Over the last two to three decades many design software packages have been developed to assist engineers in their design of structures. They have evolved over the years and have become very powerful design tools. Originally they were created to address very specific design requirements. With time they were further developed to address more complex and a wider range of problems.

When these packages were originally developed all the knowledge known today was not available. What is possible with computers have changed dramatically and there are certain functionalities sought after in software today that was not important when these packages were created. Consequently the architectures of these packages were not structured to meet these requirements and therefore they cannot be easily converted to fulfil these new requirements. An example of such a software package is Abaqus which is written in FORTRAN. FORTRAN is a good programming language when it comes to number crunching but for networking it falls short. Object oriented languages are much better suited for these purposes.

In procedural programming languages like FORTRAN and C a programmer represents a formula by a series of lines of code. A value would be entered into the formula and a single value result would be returned. A value in these languages has no meaning outside of the context in which they are used. In object oriented languages it is possible to represent entities as they exist in reality through the creation of objects. An object can represent a person, a company, i.e. any entity of the real world. These objects can then perform functions in computer code, where these functions are similar to what the actual entities do in real life. Networking with languages like FORTRAN is very difficult as the programmer will have to govern the entire process and check that the values are correct. Working with objects is much easier as they can more easily be structured to transfer information and perform the appropriate calculations at the right time.

Network-based collaborative engineering is currently a topic widely researched around the world. The German Research Foundation (DFG) is sponsoring research into this field, which is one of their priority projects. At the University of Stellenbosch, the Informatics Department in Structural Engineering also does research in this field, in co-operation with German Universities. In 2000 a new project was launched to develop structural design software that supports network collaboration. The project was subdivided into smaller projects and will therefore be referred to as the extended project. One of the priorities in the development of this software is to determine the needs of practicing engineers and to develop the software accordingly.
1.2 Scope and Limitations

This thesis addresses issues concerning the information flows that take place in the process of collaboratively designing structures using appropriate software. Information flow involves data being transmitted between various modules of the software, the input and output of data between the software and engineers and communication between people. Since design engineers are the target market for such software, a large part of this thesis focuses on perceptions and requirements of practicing engineers regarding network supported collaboration.

The design process itself has to be studied in order for the relevant information flows to become evident. Currently only the design of steel structures has been included in the extended project, therefore the design process will be viewed in light of the design of steel structures. In the first step a finite element (FE) model is created by an engineer. An analysis is then executed on this model and results or output are produced. The FE model now contains input from the engineer and results of the analysis. Next the structural members are designed using information provided by the FE model. The engineer provides the input required for defining structural members; i.e. where they are located in the structure and their properties. Once these are defined the software does the necessary calculations to check if the structural members are adequately designed. Then the connections that hold the structural members together are designed. In order to design a connection the structural members that are to be connected needs to be known as well as the forces that are acting within the connection. The information required to design a connection is obtained from the FE model and the steel member model. Once this information is obtained the engineer specifies the properties of the connections and the design calculations are then executed.

Before this thesis commenced three projects were done in which the analysis and design components of the extended project were addressed. These projects will be referred to as the preceding projects. In these projects the software architectures for the FE model, the Steel Member Model and the Steel Connection Model were developed. These projects ran independently from one another; however attempts were made to structure the model architectures in such a way that they would support interaction between one another. The steel member model is responsible for the design of structural steel members as is the steel connection model responsible for the design of structural steel connections. The steel member model and the steel connection model may be referred to as design models.

Figure 1.1 shows a graph of the models that forms a part of the integrated software. The integrated software is the software that is being developed in the extended project and this software supports network-based collaborative design. In the integrated software there are information flows that need to take place in order to do design work. These information flows will be called links and they can be categorized as follows: definition of input into the design software by a user, execution of analysis and design calculations on inputs, transferring information between models, and producing calculations and
results output to the user. All these links have to be established in order to have a fully functional design software package.

![Flow of Information during the Design Process](image)

**Figure 1.1: Flow of Information during the Design Process**

The first part of the thesis is concerned with establishing the communication links between the various components of the integrated software and between people. All the links of importance in the integrated software are indicated by arrows. The red arrows are links that have been established, the dotted red arrow indicated a link that has been established but is out of date and lastly the grey arrows are for links which have not yet been established.

The first link that had to be addressed was allowing people to define FE models. It was not that the creation of these models was impossible; it was merely that there was no graphical user interface (GUI) that supported the engineer in this task. Unless the engineer was able to define the model in Java code he would not have been able to use the software. Therefore a basic GUI for this was developed for this purpose.

The next step was to update the link between the FE model and the steel member model. The steel member model is dependent on information provided by the FE model and these two models were developed concurrently. Whenever the FE model changed the link to the steel member model had to be updated. Consequently the steel member model had to be updated regularly due to changes in the FE model. This continuous updating of the link was counter productive for the development of the steel member model. A decision was made to freeze a FE model that would not change over time and develop the steel member model from this frozen FE model. At a later stage the link between the FE
model and the steel member model would have to be updated, which was to be done next in this thesis.

The last link within the software that still had to be established was where information was needed in the steel connection model from the steel member model and the FE model. If all these links were established the design software would be able to address every aspect regarding the design of steel structures. Unfortunately the links to the steel connection model were not established. Too much time had been taken up in updating the link between the steel member model and the FE model. It was decided to concentrate on other issues, since the links to the steel connection model was not essential to continue with the rest of the thesis.

Up to this point communication links between the integrated software’s models on a single workstation were addressed. Next the communication had to be extended over a network. The link between the FE model and the steel member model was about transferring information from the FE model to the steel member model. This information transfer had to be done over a network. Therefore a software architecture was developed to support network communication within the integrated software. This network communication support makes network collaboration a reality. One user could work on a FE model in one place while another could develop the steel member model in another place. In the future when the links to the steel connection model are established this functionality could be extended to communicating to the steel connection model over a network.

The software development that was done in this thesis study was not aimed at producing commercial design software. The software development had a two-fold purpose. The first purpose of the software development was to serve as a learning experience for the author. The aim was that he would learn about the information flows that take place in design processes, what challenges software developers are confronted with and to find solutions some of these problems. The second goal was to develop pilot software that could support the basic features needed to do network-based collaborative design. This pilot software was then to be used to demonstrate to engineers what is possible in network-based collaborative design.

The final part of the thesis was to determine practicing structural engineers’ requirements of software that support network-based collaborative design. A demonstration of the pilot design software was shown to the engineers and they were then interviewed on topics relating to collaborative design supported by software. Questions were asked about how collaborative design can be done in practice and how design software could be used for this. One of the aims of the survey, for example was to determine whether or not engineers believed that there is a need for software that support network-based collaborative design at all, Finally, an assessment was to be made as to what features engineers are looking for integrated software and to make proposals for further development of the extended project.
1.3 Work Packages

The work packages required to address the issues described above are the following:

- Develop a GUI that can create FE models.
- Establish a proper communication link between the existing FE model and steel member model.
- Organize the various model architectures into a single software architecture.
- Develop a software architecture that enables network-based collaborative design.
- Determine and evaluate the requirements of practicing engineers regarding software that supports distributed collaboration.
- Consider possibilities for future development.
- Make recommendations for future development.

1.4 Plan of Development

The thesis starts with a discussion of the projects that preceded this thesis study. These projects provided the context in which the thesis started and also affected the scope of the work that had to be done. The three projects include the development of software architectures for a FE model, structural steel member design model and a structural steel connection design model. These three projects and this thesis forms part of an overall project which is the development of a network-based collaborative design software.

The description of the preceding projects is followed by three chapters discussing the software development that was done as a part of this thesis study. This software development addressed aspects that were not covered in the preceding projects. These include the development of a graphical user interface for the creation of 2D wire frame FE models, establishing the link between the design and analysis models, and developing a software architecture that allows for network communication within the design software. Each of these is discussed in a separate chapter and in the order as they are mentioned here.

The chapters on software development, is followed by a chapter that demonstrates the functionality of the integrated software. The reader is taken through a systematic walkthrough, with screenshots and explanations, of the integrated software application.

Following the chapter on the walkthrough of the integrated software are two chapters reporting the findings of interviews that were done with practicing engineers. The first chapter addresses topics relating to how collaborative design could be done in practice through supporting software. The second chapter addresses some general aspects that are to be included in the design software.

In the last chapter concluding remarks are made on the software that have been developed. Recommendations are also made for future development of the integrated software.
2 Preceding Projects

Three projects were completed before this thesis study commenced and they provided the context for the work reported here. In these projects the software architectures for the analysis and design models of steel structures were developed namely, a FE model, a steel member model and a steel connection model. In this chapter these models are discussed and they are of importance since the work done in this thesis is based on these models.

2.1 The Finite Element Model

The work done in the development of the FE model can be seen in Reference [6]. The FE model has the same basic functionality that other FE packages have today with respect to analysis. The objective of this project was to develop a FE model in an object oriented language that could do analysis of structures with some post processing abilities. The objective was not to develop a GUI that would make it user friendly for developing FE models, however a GUI that is capable of displaying a structure and its analysed results was developed. A GUI that support users in creating FE models would be developed at a later stage.

There are several components that make up a FE model namely nodes, elements, supports, constraints, materials, cross-sections and loads. All of these, except constraints and cross-sections, are always present in an analysed FE model. If any of these are absent the model will either be unstable or the results would be meaningless. The most important of these are the elements used to make up a FE model and the loads that are applied to a structure. They are discussed below:

At present five finite elements types are supported. They are:

**Truss Element**
A 2D element with degrees of freedom only along the element itself (axial).

**Frame Element**
A 2D or 3D element with translational and rotational degrees of freedom. The element has 6 or 12 degrees of freedom for 2D and 3D respectively.

**Constant Strain Triangle**
A triangular shaped membrane element that has constant strain over its area.

**Triangular Thin Plate Element**
A thin triangular shaped plate element with 9 degrees of freedom in the local element system (2 in plane rotations and 1 transverse displacement at each node.)

**Brick Element**
A 3D element that comprises 8 nodes, each with 3 translational degrees of freedom.

Three types of loads are supported in the FE model. These are:

**Node Load**

A node load can be specified anywhere within the structure where there is a node.

**Point Load**

A point load is specified anywhere on an element between two nodes with a percentage defined offset from the element's start node.

**Linear Line Load**

A linear line load is specified on an element with a starting value and an end value. The load is distributed linearly between the two points.

When loads are added to a model they are added to a specific load case. A load case represents a certain loading condition on a structure; for instance all the forces resulting on a structure due to wind would be added to a wind load case. Load cases can also act in combinations upon a structure i.e. the wind load and own weight of a building. Therefore load combinations can be specified to determine the impact that a combination of loads have on a structure. The aim of these combinations is to determine the worst loading scenario in a structure and to design the members and connections of a structure accordingly.

The rest of this section contains screenshots of the FE model software. Several FE models were created and analysed. The results of the analyses are shown and indicate what can be done in the software.

In Figure 2.1 a portal frame is shown that consists of frame elements with node loads that have been applied to the structure. Post-processing was done on the analysis results and the bending moments throughout the structure were calculated. These bending moments are also shown where the red indicates a sagging moment and the blue a hogging moment.
An arbitrary shaped slab is built up from triangular thin plate elements. A transverse load was applied throughout the structure, with supports being placed at various points (shown in pink). The principal stress distribution is shown where blue indicates compression and red tensile stress.

![Image of principal stress distribution](image)

**Figure 2.2:** Principal Stress Distribution of a Slab using Plate Elements

A column and its supported foundation is modelled in 3D using 8-node brick elements. The column is subject to both horizontal vertical and loading. In Figure 2.3 the displacement of the column can be seen.

![Image of column displacement](image)

**Figure 2.3:** Displacement of a Column Analysed with 3D Brick Elements.

### 2.2 The Steel Member Model

The detail design of structural steel members is done in this model, and the original development is recorded in Reference [3]. To design a steel member its geometry, support conditions and the force distribution within that member needs to be known. The force distribution and certain aspects of the geometry is contained within the FE model. Therefore a FE model is always associated with a steel member model. When a new steel member model is created, a FE model is specified as its parent model.
With regards to the overall structural design a clear distinction is made between analysis and design. The analysis of a structure is done separately from the detail design of the individual members. With such a configuration the engineer has more control over the design of the individual structural members. A structural member consists of one or more finite elements that are connected and in a straight line. In Figure 2.4 a structural member comprising three finite elements is shown. The length of the structural member is defined independently from the finite elements. An offset is specified from the first and last node of the finite elements. These offsets set the start and end of the structural member. Once the geometry of the structural member is defined one can proceed with the design of the structural member.

\[ \text{Figure 2.4: Defining of a Structural Member} \]

Several factors affect the design of a structural steel member. These are the type of forces that act on the member, the start and end conditions, lateral restraints that are placed along the member and whether or not the structure is braced. The steel member model provides all the functionality to take these factors into consideration in the design of structural members.

The forces that act on a structural member determine the type of structural member that is to be designed. Table 2.1 lists the structural member types with a description of the forces that act on that member type.

\[ \text{Table 2.1: Structural Member Types} \]

<table>
<thead>
<tr>
<th>Structural Member</th>
<th>Loads</th>
<th>Picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam</td>
<td>Transverse loads and moments.</td>
<td><img src="image" alt="Beam" /></td>
</tr>
<tr>
<td>Column</td>
<td>Axial compressive loads.</td>
<td><img src="image" alt="Column" /></td>
</tr>
<tr>
<td>Beam Column</td>
<td>Transverse loads, axial compressive loads and moments.</td>
<td><img src="image" alt="Beam Column" /></td>
</tr>
<tr>
<td>Tension Beam</td>
<td>Transverse loads, axial tensile loads and moments.</td>
<td><img src="image" alt="Tension Beam" /></td>
</tr>
<tr>
<td>Tension Member</td>
<td>Axial tensile loads.</td>
<td><img src="image" alt="Tension Member" /></td>
</tr>
</tbody>
</table>
A cantilever with transverse loads and moments.

A supported cantilever with transverse loads and moments.

As mentioned the end conditions for a structural member needs to be defined. These end conditions affect the buckling behaviour of a structural member. Both lateral torsional buckling and Euler buckling is to be considered. Table 2.2 lists the possible end conditions that can be defined for the various structural members.

**Table 2.2: End Condition of Structural Members**

<table>
<thead>
<tr>
<th>Structural Members</th>
<th>Lateral Torsional Buckling</th>
<th>Euler buckling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Beam</strong></td>
<td>Unrestrained</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Partially restrained</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Practically fixed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Torsionally restrained</td>
<td></td>
</tr>
<tr>
<td><strong>Column</strong></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><strong>Beam Column</strong></td>
<td>Unrestrained</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Partially restrained</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Practically fixed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Torsionally restrained</td>
<td></td>
</tr>
<tr>
<td><strong>Tension Beam</strong></td>
<td>Unrestrained</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Partially restrained</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Practically fixed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Torsionally restrained</td>
<td></td>
</tr>
<tr>
<td><strong>Tension Member</strong></td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><strong>Cantilever</strong></td>
<td>Built In (Laterally &amp; Torsionally)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Continuous (Laterally &amp; Torsionally)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Continuous (Laterally only)</td>
<td></td>
</tr>
<tr>
<td><strong>Propped Cantilever</strong></td>
<td>Built In (Laterally &amp; Torsionally)</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>Continuous (Laterally &amp; Torsionally)</td>
<td>Pinned</td>
</tr>
<tr>
<td></td>
<td>Continuous (Laterally only)</td>
<td></td>
</tr>
</tbody>
</table>

*Cantilevers and cantilever beams only have end conditions defined on the one end of a structural member.*
Lateral restraints can be specified along the length of a structural member. In Figure 2.5 three arrows are shown in red, green and blue. Each arrow indicates lateral restraints along the length of a structural member.

![Figure 2.5: Lateral Restraints along the Length of a Structural Member](image)

The red and blue arrows support the weak axis at the top and bottom flange respectively. The green arrow supports the strong axis of the structural member. It is possible to have several lateral restraints along each axis and for both flanges. A lateral restraint is normally a steel member that does not influence the behaviour of the structure, but prevents another member from moving laterally at a point. Lateral restraints decrease the possibility of a member buckling. Finally a structure can be defined as braced or unbraced, which affects the design of all members in a structure. With all these aspects taken into consideration the complete detail design of all members in a steel structure can be done. The design calculations are done according to the SANS 10162-I code that was released in 2005.

![Figure 2.6: Steel Member Model – Defining Structural Members](image)

The figure above shows a screenshot of the structural steel member model; where the structural members are being defined. One structural member has already been defined. The red lines indicate finite elements that have been selected for defining another structural member. The blue lines are finite elements of the remainder of the structure. The figure below shows a list of steel section profiles. The
profile information is obtained from a database provided by steel manufacturers. Profiles are selected from the list and the design calculations for the structural members will be done on these profiles.

![Figure 2.7: Selection of Profiles for Structural Members](image)

In Figure 2.6 and Figure 2.7 the definition of structural members and their profile properties are shown respectively. The design calculations for a structural member are shown in Figure 2.8. When software does design calculations it is important that an engineer can evaluate the calculations done by the computer. To do this the engineer needs to have the information that was used in the design calculations.
DISTRIBUTED COLLABORATION:
ENGINEERING PRACTICE REQUIREMENTS

Figure 2.8: Calculation Results of Adequate Profiles

The information is presented step by step. At the start the type of member design, the load case and the name of the structural member that is being designed is shown. This is followed by a description of the start and end conditions of the structural member. Each member is designed for overall member strength and lateral torsional buckling. Under each of these categories the factors used for the design calculations are shown with the most extreme values that occur in the structural member. This is followed by the maximum allowable forces that the various selected profiles can carry. The interactive ratios are also shown, where these ratios take a combination of forces into account. If the ratio exceeds
one it means that the profile has failed. This information allows the engineer to double check that the calculations done by the computer are correct.

In Figure 2.7 three profiles were selected for the design of the structural member. In Figure 2.8 only two profiles are reported. This means that the one profile that was selected for the design of the structural member inadequate and would not be able to carry the forces acting within it. The calculations that were done for this profile is recorded under failed design results and can be seen in Figure 2.9. The areas where the profile failed are highlighted in yellow.

Figure 2.9: Calculation Results for a Profile that Failed
2.3 The Steel Connection Model

This model is responsible for the design of steel connections, and its development is recorded in reference [9]. To design a connection the steel members that are to be connected need to be known as well as the forces that are acting in the connection. This information has to be obtained from the FE model and the steel member model, for the forces and profile information respectively.

Several types of connections can be designed. The connection type is determined by the forces that a connection has to transmit, the profiles it has to connect and where in the structure the connection is located. Table 2.3 shows four main connection types with subtypes and how these connections look.

<table>
<thead>
<tr>
<th>Connection Type</th>
<th>Connection Subtype</th>
<th>Drawing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base plate connection</td>
<td>Pinned</td>
<td><img src="image1" alt="Pinned Connection Diagram" /></td>
</tr>
<tr>
<td></td>
<td>Moment</td>
<td><img src="image2" alt="Moment Connection Diagram" /></td>
</tr>
<tr>
<td>Beam Column Shear</td>
<td>Welded End Plate</td>
<td><img src="image3" alt="Welded End Plate Diagram" /></td>
</tr>
<tr>
<td></td>
<td>Angle Cleat</td>
<td><img src="image4" alt="Angle Cleat Diagram" /></td>
</tr>
</tbody>
</table>
### Figure 2.10

Figure 2.10 shows a screenshot of the steel connection design application. In this application the connection type must first be specified. Once this has been done the engineer can continue to specify the standard and more advanced parameters.

Standard parameters entail the number of bolts within a connection and their properties. Where plates and angle cleats are used in the connection the material quality of these can also be specified. Advanced parameters involve the exact spacing of bolts and the thickness and the size of plates. These aspects are what comprise the design of a connection. The design calculations can be seen in a design data sheet.

<table>
<thead>
<tr>
<th>Connection Type</th>
<th>Plate Type</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam Column Moment</td>
<td>Extended End Plate</td>
<td>![Image]</td>
</tr>
<tr>
<td>Ridge Connection</td>
<td>Extended End Plate</td>
<td>![Image]</td>
</tr>
<tr>
<td></td>
<td>Flush End Plate (Haunched)</td>
<td>![Image]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>![Image]</td>
</tr>
</tbody>
</table>
Figure 2.10: Application of the Design of Structural Steel Connections

2.4 Final Remarks

The projects that were discussed in this chapter provided the context within which this thesis study began. In this thesis the respective projects were brought together for the first time, and they were arranged into a single software architecture. To achieve this, some of the work that was not covered in the preceding projects had to be done. Furthermore, substantial corrections were made to the member design model. From a software development point of view the next work packages dealt with the development of a GUI for creating FE models, establishing the link between the FE model and the steel member model and developing a software architecture for network-based collaborative design. These aspects are discussed in the chapters that follow.
3 The Finite Element Model Creator

This thesis is about establishing and ensuring proper flow of information in the design software and between the people. When designing structures with software the engineer must be able to interact with the software and create his models. In this chapter the development of a graphical user interface (GUI) for creating FE models is discussed, from here on called the FEM Creator. The FEM Creator is pilot software and supports only fundamental aspects; the full source code can be seen in Addendum C:. Only wire frame two dimensional models are supported.

The components that are supported in the GUI are discussed below:

Nodes:
A node is represented by a circle and contains x and y coordinates.

Finite Elements:
Frame and truss elements are supported and both are represented by a straight line.

Supports:
Supports are represented as triangles and can prevent translational and rotational movement of the structure at a point.

Cross-Sections:
A cross-section is a property of a frame or a truss element. The cross-section component contains the area and moments of inertia properties of an element.

Materials:
A material component is a property of a finite element and influences how an element responds to loading. The material contains the Elastic modulus, Poisson’s ratio and the density of an element.

Loads:
Node loads and point loads are supported in the software. Both are represented by arrows, where node loads are purple and point loads are green. A node load has to be applied to a node and a point load is applied along the length of a frame element.

The rest of this chapter contains screenshots of the FEM Creator application.

Figure 3.1: FEM Creator – Creating Finite Elements

A couple of elements have been created and the properties of the element marked in red are being defined. The element can be defined as either a frame or a truss element. The material and cross-section properties of the elements can also be defined. The materials and cross-sections will have to be created by the user first.
A portal frame is being created and support (pink triangle) has been created at the bottom left of the structure. The support conditions are defined to restrain movement of the structure in either the vertical or horizontal directions or against rotation around a point.

**Figure 3.2:** FEM Creator – Creating Supports

A portal frame has been created with supports at the bottom. Currently loads are being applied to the structure. A popup window is currently shown for a load that will be added to the node that is marked red. The horizontal and vertical components of the load are defined in this window. Before these loads can be created a load case is created to which these loads are added.

**Figure 3.3:** Fem Creator – Creating Loads.

### 3.1 GUI Architecture

The FEM Creator application comprises two main software components which are the FEMFrame and the FEMPanel. These components were created specifically for this application and they will be discussed in the remainder of this section.

#### 3.1.1 FEMFrame

The FEMFrame extends class JFrame and provides the visual layout for the user. Inside the FEMFrame four components are held and these are a menubar, a toolbar, a coordinate label and a FEMPanel. These are discussed in this section except for the FEMPanel which is discussed in section 3.1.2.

**The Menubar:**

The menubar is shown at the top of the FEMFrame. It contains several drop down menus from which the user can select commands that are to be executed. The menus that have been provided are: File, View, Nodes, Elements, Supports, Loads, Model and Select.
The Toolbar:
The toolbar is placed on the left of the FEMFrame and it contains 6 buttons. When a user clicks on these buttons certain commands are executed. The buttons are shown here with a description of the actions are done when clicking on these buttons:

- switching the display of the grid on an off
- switching on and off snap mode
- creating finite elements
- creating support
- creating node loads
- creating point loads

The Coordinate Label:
The coordinate label can be seen at the bottom right-hand corner of the FEMFrame and it displays the current coordinates of the mouse cursor.

3.1.2 FEMPanel
The FEMPanel is the heart of the FEMCreator. In this panel all the information is displayed to the engineer and it is also where the engineer interacts with the software to create FE models. Two programming strategies are used that support users in giving commands to the software to execute. These are implemented through mouse-listeners and popup frames as discussed below.

Mouse-Listeners:
Mouse-listeners "listen" for when an action from a mouse takes place. Actions that mouse-listeners listen for include mousePressed, mouseReleased, mouseClicked, mouseEntered and mouseExited. Only mousePressed will be used here. A mouse-listener needs to be linked to a graphical component, in this case the FEMPanel. If a mouse-listener has been created, but not registered on a graphical component nothing would happen.

In Figure 3.4 the software architecture's code outline is shown. Three global variable mouse-listeners are defined at the top, namely nc, sc and nlc. They allow the user to create nodes, supports and node loads respectively. In the constructor the three mouse-listeners are instantiated. The classes from which these mouse-listeners are instantiated are defined as inner-classes of class FEMPanel. Since, these mouse-listeners were created to work only on the FEMPanel. Having these classes as inner-classes, referencing between the FEMPanel and the mouse-listeners are made easier and all the aspects relating to the FEMPanel are kept together.
Consider Figure 3.4. When the constructor is executed the mouse-listeners are created, but none are registered (or added). This means that no action will take place when a mouse action occurs. An activate method must first be called to register a mouse-listener. In an activate method all the mouse-listeners are first deregistered (even if they weren’t registered) and only the appropriate mouse-listener is then activated. Say the method activateCreateNode() is executed. All mouse-listeners are deregistered, and only the mouse-listener instance nc of class NodeCreator is registered. Now when a mouse action occurs the appropriate method of the mouse-listener nc is executed. If a mouse button is pressed the mousePressed() method of mouse-listener nc is executed. In this case a node would be created (the code for this would be defined in class NodeCreator’s method mousePressed()). If method activateCreateSupport() is then executed, all the mouse-listeners would be deregistered and sc would be registered. Now when a mouse action takes place the appropriate method of mouse-listener sc would be executed and no longer that of nc. The design pattern as described is simple, but extremely effective in the development of a robust GUI. See Addendum C: for the location of source code where this registering and deregistering of mouse-listeners are implemented.
For example, it is possible to have more than one mouse-listener registered at a time. Care needs to be taken when this is done to avoid confusion, i.e. mouse-listeners nlc and nc can be registered at the same time. If a mouse button is pressed the mousePressed() methods of both these mouse-listeners would be executed. Some strange things may happen, in this case the computer would have been given instructions to create a node and a node load at the same time, which does not make sense. To avoid this the code would have to include a lot of logic testing to determine which action has to be performed. Such programming is cumbersome and error-prone, and it is best avoided using the design pattern described above.

**Popup Frames:**

The popup frames are small windows that pop up on the screen to allow the user to define certain input. In this context it is used for defining coordinates of nodes, properties of elements, magnitude and direction of loads and so on. The popup frame for defining a nodes coordinates is shown in Figure 3.5. It can be seen that the popup frame comprises of three main components. It has a title at the top, a done button at the bottom and values that can be defined in the centre.

![Node n1](image)

**Figure 3.5:** Node Property Popup Frame

Popup frames operate similarly to the mouse-listeners. They are created with the execution of the constructor after which none are visible. A show method must first be called i.e. showNodePropertyFrame(). In the show methods first all the popup frames are set to invisible and then the appropriate frame is set to be visible.
Once the user has finished using the popup frame he can click on the done button. The necessary code will then be executed with the information that has been defined by the user in the popup frame. Consider class NodePropertyFrame. A doneButton has been defined at the top of class NodePropertyFrame. An action listener has been added to the doneButton in which the code has been defined for what should happen when the button is pressed. In this case the node’s coordinates will be set and then the popup frame will be set to invisible. The source code for this implementation can be seen in Addendum C.

**Figure 3.6:** Architecture of Popups in the FEMPanel
4 Linking the FE Model and the Steel Member Model

The three models discussed in Chapter 2 were developed separately from one another. They were to be connected at a later stage and operate as a whole. Consider a power station and a kettle. A power station generates power which it transfers through a wire. A kettle operates on power generated by the power station, but it is developed independently from the power station. The power is made available to the kettle through a socket, and the kettle accesses this power by being plugged into the socket. When the plug is connected with the socket the power goes from the power station to the kettle and the kettle can work. Similarly the FE model was developed and it contains the information needed in the steel member and steel connection models. The design models were developed independently from the FE model but they still need information from the FE model. So the models have to be “plugged in” to obtain the information it needs from the other models. This chapter deals with the “plugging in” of the steel member model so that the appropriate information transfers can take place. The steel connection model was not “plugged in” as time constraints did not allow for this to be done.

4.1 Connecting the Steel Member Model to the FE Model

The steel member model is based on the FE model. When the development of the steel member model commenced the FE model had been developed, but was still in a process of changing. It was decided to freeze a FE model that would not change with time and the steel member model was developed based on this frozen FE model. So a link between the FE model and the steel member model was established, however this link became out of date as the FE model changed in time. The steel member model was modified in order to retrieve information from the most up to date FE model. In the process of executing this task some significant changes were made to the architecture of the steel member model, which included the introduction of the Unit Model (see section 4.1.3).

4.1.1 Persistent Identifiers

The FE model and the steel member model both made use of persistent identifiers for all of their components. This means that each component has an unique name and is referenced by that name. No two objects that are persistently identified can ever have the same name. A record is kept of all the names that have been recorded and when a new object is created it is ensured that its name does not coincide with the name of another object. The steel member model and the FE model both had their own way of doing this. The steel member model had a class called “AppObject.java” and the FE model had one named “INamedObject.java”. Effectively this resulted in two files with the same name that does the same work. It was decided to make all components of the steel member model of type INamedObject and the class AppObject.java was completely removed.
4.1.2 Edit GUI of the Steel Member Model

The GUI of the steel member model made use of an event-listener based strategy to allow users to give commands to the software. The steel member model has a class called DrawPanel and is equivalent to the FEMPanel discussed earlier. The concept used for the event event-listener architecture is shown in Figure 4.1. In an event-listener strategy an Event class and a Listener interface has to be created. From the Event class objects will be instantiated that contains information regarding an event that takes place. The interface Listener defines a method that has to be implemented by classes that implement this interface. This method will require that an Event object be sent as a parameter of the method.

The DrawPanel contains a List variable that contains objects of type Listener. Originally when a DrawPanel object is instantiated a List object is also created and it contains no Listener objects at this point. A Listener object can be added or removed from the list variable by add and remove methods, in this case addDrawPanelListener() or removeDrawPanelListener().

```java
public class DrawPanel{
    List panelListeners;

    public void addDrawPanelListener(Listener listener){
        if (!panelListeners.contains(listener))
            panelListeners.add(listener);
    }

    public void removeDrawPanelListener(Listener listener){
        panelListeners.remove(listener);
    }

    // picked method of all registered listeners are executed
    private void firePanelEvent(DrawableShape ds, MouseEvent me){
        Event e = new Event(ds, me);
        Iterator iter = panelListeners.listIterator();
        while (iter.hasNext())
            ((Listener)iter.next()).picked(e);
    }

    public interface Listener(){
        public void picked(DrawPanel.Event e);
    }

    public class Event{
        DrawableShape ds;
        Object object;
        MouseEvent me;
    }

    class ShapeSelecter extends MouseListener{
        public void mousePressed(MouseEvent me){
            DrawableShape ds = selectDrawableShape(me.getPoint);
            if(ds!=null)
                firePanelEvent(ds,me);
        }
    }
}
```

**Figure 4.1:** Event-Listener Architecture used in the DrawPanel
The components that are involved in the event-listener architecture have been described above. Now its operation will be discussed as it was done in the DrawPanel. An Event object represents an event that takes place. In this case the event of interest is when a shape is selected. A mouse-listener, ShapeSelecter, has been registered on the DrawPanel and listens for when a mouse button is pressed. Once a button has been pressed it checks if a shape exists at the position of the mouse cursor. If it does, a shape was selected and the firePanelEvent() method will be executed. In the firePanelEvent() method the Event object is instantiated. Then iteration is done over the List object which contains only Listener objects. For every listener in the list its picked() method is executed.

```java
public class SelectTool implements DrawPanel.Listener{
    // Listener method
    public void picked(Event e) {
        DrawableShape ds = e.getDrawableShape();
        ds.setSelect(!ds.isSelected());
        drawPanel.draw(ds);
        if(ds.isSelected())
            selectedShapes.add(e.getDrawableShape());
        else
            selectedShapes.remove(e.getDrawableShape());
    }
    // adds listener to DrawPanel
    public void registerTool(){ // add listener to drawPanel
        drawPanel.addDrawPanelListener(this);
    }
    // removes listener from DrawPanel
    public boolean de_registerTool(){
        drawPanel.removeDrawPanelListener(this);
    }
}
```

**Figure 4.2:** Code for Tool that Implements Interface DrawPanel.Listener

Figure 4.2 shows the code for class SelectTool.java which implement the interface DrawPanel.Listener. In this class two methods are provided, registerTool() and deregisterTool() which add and removes this object from the DrawPanel’s List object discussed above. If this object is contained in the List object and the firePanelEvent() method of the DrawPanel is executed then the picked method shown in Figure 4.2 will be executed. In this case a shape is selected and would be redrawn in red.

The event-listener architecture described above was used in the GUI of the steel member model. When the link between the steel member model and the FE model was established some problems were experienced with this GUI. This architecture was considered to be less effective than the architecture used in FEMCreator. Consequently the event-listener architecture was replaced by the registering-deregistering of mouse-listeners architecture described in section 3.1.2.
4.1.3 The Unit Model

Some difficulties were experienced in the process of linking the steel member model and the FE model. The steel member model was developed to operate only on one unit type, while the FE model could be created in any units. The FE model left the responsibility to the user to ensure that he/she was using compatible units. Consequently the results produced by the steel member model were nonsense if the correct units weren’t specified in the FE model. To overcome this problem the concept of the Unit Model was introduced.

A unit model is a model that keeps record of the type of units it has been developed in. This is necessary so that the correct conversions can be done when information is transferred between models. As a result it is possible for the two models to have different units, while correct conversion is assured. Consider, for example a FE model that uses $kN$ and $mm$ as its units, while the steel member model uses $N$ and $m$. If a length is requested from the FE model by the steel member model the value returned will be divided by a thousand. It is now possible to have models using different units. The units specified for a unit model is independent from the actual values recorded for the components of the model itself. Consider the following scenario. A user defines an element to be 4.0 units long. Whether he sets or changes the length unit of the model from $m$ to $mm$, the length will remain 4.0. The only purpose of the unit model is for communication between models and has no purpose within a model.

![Diagram of Unit Model](image-url)

**Figure 4.3:** Structural Layout of the Unit Models

In Figure 4.3 the architecture for the unit models is shown. Both the FE model and the steel member model was changed to be subclasses of UnitModel.
The Unit Model is a super class of all the analysis and design models in the integrated software. The layout of the Unit Model is shown in Figure 4.4. The Unit Model has 3 object attributes, namely a length unit, a force unit and an elasticity unit. These attributes can only be set equal to one of the 7 static attributes that have been defined in this class. Of the static attributes, two have been defined for the length unit, two for the force unit and three for the stiffness unit. More units can be defined, but these are the ones that have been required thus far. For example, if there is need to use feet as a length unit, it could be included as well.

Figure 4.4: Layout of the Unit Model.

The actual conversion is done by static methods, three of which have been implemented so far. These are calculateForceFactor(), calculateLengthFactor() and calculateEFactor(). These are used to calculate conversion factors. Each of these methods has two input parameters of type byte. The byte values represent the current unit and the required unit. The methods will then return a factor that will have to be multiplied with the current unit to obtain the required unit. Below an example is shown of how these methods work, with some of the calculations:

Say the length of an element is 4 units and its current units are in m. The user needs the units to be in mm. The following needs to be done.

<table>
<thead>
<tr>
<th>Length</th>
<th>Current unit</th>
<th>Required unit</th>
<th>Conversion factor</th>
<th>New length</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>m</td>
<td>mm</td>
<td>1000</td>
<td>4000</td>
</tr>
</tbody>
</table>

```
public static double calculateLengthFactor(byte currentUnit, byte requiredUnit) {
    checkLengthUnit(currentUnit);
    checkLengthUnit(requiredUnit);
    if (currentUnit == requiredUnit) return 1;
    else if (requiredUnit == LENGTH_m && currentUnit == LENGTH_mm) return 0.001;
    else return 1000;
}
```

// code that will have to executed for this example
double length = 4.0;
length = calculateLengthFactor(Length_m, Length_mm) * length;
4.2 Restructuring of the Software

In this thesis all the software code that was written for all the various models of the preceding projects have been brought together for the first time. The software code was structured into one overall package and the overview of the software architecture can be seen in Figure 4.5.

![Software Architecture Diagram](image)

**Figure 4.5: Overview of the Software Structure**

The software is structured into three main categories: Design, Management and Utilities. The Design section contains the code that was developed in the preceding projects and improved and extended in this project as described. It is subdivided into two sections, namely GUI and Models. From a programming point of view it is important to separate the GUIs from the design models. This is to avoid the design calculations becoming imbedded in the GUI code. In the future, should there be a need to replace the GUI of one of the models, it can be done easily and the design calculations will not be affected by the change of in the GUI.

The utilities package contains software code that is used as tools by all the other packages. The tools include mathematical calculations, formatting of text, creating graphical components, etc.

The management package contains the code that supports network communication within the software. This package makes it possible for one person to work in Johannesburg and another in Cape Town while working on the same project. The contents of this package will be discussed in the next chapter.
5 Distributed Collaboration

The work described in the previous chapters brought the software to a point where it was possible to develop a FE model and design the members of a structure on a local computer. This functionality was now to be extended to operate over a network i.e. to make the information contained in the FE model and the steel member model independent of the local workstation on which the software is executed at any point in time. To do this network communication as supported by Java was investigated and implemented, which is discussed in this chapter.

5.1 Remote Method Invocation

Java provides a network communication facility called Remote Method Invocation (RMI). RMI supports communication between two Java Virtual Machines (JVM). A JVM is simply an environment within which Java code is executed on a workstation. It provides the ability for the software to operate on any operating system e.g. Linux and Windows. RMI enables an application one JVM to communicate with an application on another JVM whether it is on the same computer or on another computer on the network.

When using RMI one needs to understand concepts like remote objects, remote object referencing and the RMI registry. A remote object is an object that resides in a JVM and can be accessed from another JVM. A remote object can be considered to be a server to which requests are sent, which it then carries out and returns information.

Before a remote object can be accessed by an external JVM it must be registered in a RMI registry. A RMI registry is a registry that contains references of remote objects and it resides on a port. A port can be likened to a radio channel. Unless a radio is tuned in to the correct frequency there will be no transmission. Similarly a registry can only be accessed on a specific port. When registering a remote object, a name has to be provided for the object by which it will be looked up. When trying to access a remote object, the RMI registry where it is registered must first be accessed. A name is then given to the registry and it will look for the reference of the corresponding remote object. If the name is found, the object reference corresponding to that name is returned. Access has then been obtained to the remote object and direct calls can be made to it, as if it were a local object.
Figure 5.1: RMI Concept

Below sample code for gaining access to a remote object is shown, with requests that are then made to the server. A variable of type IServer is declared. The command Naming.lookup("server", 4000) is then executed. In this command a registry on port 4000 is accessed and instructed to seek a remote object that is registered under the name "server". The registry service then returns this object which is assigned to the variable server. Next a request is made to the server by calling its method storeObject() in which the provided object will be stored in directory "C:\Work". The RMI service will take care of the details of transferring all data as well as the opening and closing of sockets needed to achieve this.

```java
IServer server;
server = (IServer)Naming.lookup("server", 4000);
server.storeObject(object, "C:\\Work");
```

5.2 Communication Architecture

In collaborative design there are three primary functions that need to be supported. In the first place the possibility must be there to assign the responsibility for the overall design and its coordination to a person. Secondly, communication between people and the accessing of files over a network must be facilitated. Lastly a way to manage data that is created during the design process must be supported. It was decided to use a server/client architecture as the networking architecture for collaborative design.
5.2.1 The Server

The server is the heart of the collaborative environment. All communication takes place through the server, whether it is between users or between users and the file system (loading and saving files). The server therefore acts as a mediator between all the clients and the file system. The server makes establishing security measures and a responsibility hierarchy possible. Since all communication takes place through the server, the server can allow or deny rights to a user. The server can define the access rights of the various users, i.e. which users may access and edit which files. The server can address issues regarding accessibility. A file may be currently in use and as a result no other users should be allowed to edit the file.

Implementation:

Management of the file system is the most important role of the server. The server provides access to users to the data files that are created in the design process. This access includes opening and saving of files. To manage the files on the server a remote browser was created. This remote browser provides the functionality to open a file on the server as well as saving a file to the server. Within the browser the client can create directories as well as rename files and directories. The deletion of files is not supported as it is preferred to not allow clients to delete files on the server. Furthermore a root directory is specified on the server and the clients will not have access to the server beyond this directory. The management of files to this point has been restricted to the creation of directories and the opening and saving of files. At present the responsibility lies with the clients to determine where which files are stored on the server.

Figure 5.3 below shows the open dialog of the remote browser. In this dialog the file system of the server is shown and not of the client. The root directory to which the clients have access is Projects. Inside Projects all the files relating to structural design are stored. In the drop down menu a hierarchical layout of the directories can be seen. The current directory is "Projects\FEM files" and all the files contained in this directory is shown. The client can select one of these files and the selected file then will be opened.
The remote browser was developed from Java’s JFileChooser, but it was customized to support remote browsing. A JFileChooser only supports browsing of the file system on a local workstation. Therefore the GUI of the JFileChooser was used and all the commands that it normally executes was intercepted. Consider the file button (third button to the right of the drop down menu) of the file chooser shown in Figure 5.3. That button is for creating a new folder. However when a new folder is to be created that folder now has to be created, its location has to be on the server and not on the local workstation. To achieve this, the code that normally creates a folder was replaced. The functionality of the server was invoked to create a remote folder. In order to customize the JFileChooser one has to understand Java’s File, FileSystemView, FileFilter, and JFileChooser classes. Reference [4] served as guidance to the development of the Remote Browser. The source code for the remote browser can be seen in Addendum C:

The second role of the server is to monitor what is happening in the project. Every time a client connects to the server, the server is aware of it and keeps track of all clients that are currently
connected. It also keeps track of which clients are working on which files and which type of files they are working on. Figure 5.4 is a screenshot of the server application. It can be seen that there are four clients connected to the server. Only Client 89 is working on FE model files and Client 9 is busy with steel member design. The others are connected, but are not doing any design work at present. It is possible to see which files a client is working on by placing the mouse cursor over the client. Client 89 is currently working on two FE files namely, “frame.fem” and “beam.fem”.

The collaborative architecture therefore supports communication between users by the transfer of files. This is the most basic form of communication needed for collaborative design and this can be extended to support more communication means between users i.e. direct messaging between users. The management of files is supported in so far as to where files are located; however users have to do this organization manually. Additional support can be provided in the future to assist users in this. Access rights have not yet been defined. Since the server already keeps track of all clients, implementation access rights will be relatively straight forward. The source code for the server implementation are recorded in Addendum C:.

### 5.2.2 The Client

The client component of the networking software is the point through which users gain access to the working environment. The client has a reference to the remote object (the server) and can call the remote object’s methods. Currently these method calls are related to opening and saving files. Within the client’s GUI the engineer is provided with tools to do his design work as shown below in Figure 5.5.

![Figure 5.5: Client Application](image-url)
The design tools that are made available to the user includes the FEMCreator, the GUI of the steel member model and of the steel connection model. The three icons shown in the toolbar on the left-hand side are buttons that will start up each of those programs. At present the FEMCreator tool is seen on the main screen within a tabbed pane entitled “frame”. This indicates that the file that is currently being worked on is frame.fem. A second tab can also be seen with the name of “Mem Untitled_0”. This is the default name that is provided when a new steel member model is created. The moment the file is saved for the first time its name will change. The source code for implementation of the client can be reviewed in Addendum C. The operation of the integrated software as a whole is described in the next chapter.
6 Systematic Walkthrough of Integrated Software

Up to now all the aspects and functions of the integrated software have been discussed. In this chapter a systematic walkthrough of the software is shown in screenshots with explanations of what is happening in each step of the software.

To begin the software the server application must first be started. When starting the server there are some settings that first needs to be defined as shown in the figure below. The first is the root directory and this is the directory that will serve as the root for all clients. None of the clients will have access to the server outside of this directory. Next is the RMI Location and that is the directory where the command for starting the RMI registry is located. Lastly the server name has to be defined and this is the name by which the remote object (server) will be registered in the RMI registry. When clients want to obtain the reference to the server object they will have to specify the server name as it has been entered here.

After the settings for the server have been defined, the user can click the “Done” button and the server application will be loaded, which is shown below. The server application shows no clients have been registered at present. A printed line is shown “SS_Server bound in registry” and this merely indicates that the server has been successfully registered in the RMI registry.
Since the server has been successfully registered in the RMI registry, client applications can be executed and gain access to the server. Similar to when the server is started, the client will also have to define some settings. Firstly the host name has to be specified where this is the name of the computer that hosts the server application on the network. This name could also be an IP address. Next the server name has to be defined which is the name by which the remote object is registered on the RMI registry. Finally a user name has to be given by which a client is identified. In the future this can be replaced by the IP address of a local computer. One would like the name to be consistent and not change over time for security reasons.

![Client Setup Definition](image)

**Figure 6.3:** Client Setup Definition

Once the client settings have been entered the done button is clicked and the client application is initiated. Figure 6.4 shows the blank client application that is loaded. At this point nothing has been done at the client.

![Blank Client Application](image)

**Figure 6.4:** Blank Client Application

With the client application being loaded, the client is also registered with the server. The server keeps track of all clients that is registered. This is shown in the figure below.
After the client has application has been loaded and registered with the server, the engineer would like to continue with his design work. On the client application three icons are shown on the left-hand side. Each of these icons as shown from top to bottom will load the FEMCreator, steel member model and steel connection model software respectively. The loading of the steel connection model software has not been implemented yet and will be done in future development.

In the above figure the client application is shown where FEMCreator software has been loaded. The FEMCreator software is not discussed in detail here. Chapter 3 explains in more detail the functionality of the FEMCreator and what is possible in the software. After the FE model has been developed and analysed, the user can store the model on the server. Below the save dialog of the remote browser is shown. The file system of the server is shown in the remote browser. It can be seen that the save dialog is also opened at the root directory that was previously defined when the server application was loaded.
At this point it is possible for the client to create directories on the server, or to rename files and folders on the server. This gives the clients the ability to organize the working environment and storage of files on the server. The next screenshot shows where the user is creating a new directory to store the file that he has just created.

The user can then navigate into the directory that has just been created and store the file there. One can see in the next directory screenshot that the current directory is Portal Frame and that the user is about to save his file as frame1.fem. It is not necessary for the user to specify the extension, the appropriate extension will be added.
In the next step one can see the client application and where the steel member model software has been loaded. This means that the second button of the toolbar on the left has been clicked. There are two tabs at the top, where the current tab is named “Mem Untitled_0”. This indicates a new steel member model is being created, since “Mem Untitled_#” is the default name for a steel member model that is being created. The first tab is named “frame1” since the FE model that was created in the FEMCreator was entitled “frame1”.

In the screenshot above a new steel member model is being created. In order to create a steel member model and FE model has to be specified. In this case the steel member model is based on the FE model that has just been created. Figure 6.11 shows the remote browser where a the FE model is being selected for the steel member model.
Figure 6.11: Selecting the FE Model for the New Steel Member Model

The figure below shows the steel member model that is based on the FE model. A structural member has been defined and the user can continue with the detail design of the structure. The working of the steel member model is discussed in greater detail in section 2.2.

Figure 6.12: Structural Steel Member Design Being Done in the Client Application

After the user has finished his work on the steel member model he can save it like with the FE model as was done earlier. The same procedure as before can be followed, where the user can navigate to the appropriate place to store the file. Figure 6.13 shows the remote browser where the two files that have been stored are shown. Both are labelled “frame1” but the extensions are different where “.fem” files are for FE model files and “.mem” files are for steel member model files.
In Figure 6.14 the server application is shown again. It can be seen that Client 30 is shown in the lists for “FEM Design Active” and “Mem Design Active”. This indicates that Client 30 is doing both FE modelling and structural steel member design. When the FEMCreator and steel member model software was originally loaded by Client 30, was he registered in the FEM Design Active and Mem Design Active lists respectively.

That concludes the functionality that is supported by the integrated software at present. Much work would have to be done in future developments. However, it is important take into account what engineers believe to be important in design software and what they deem to be important in design software that supports network collaboration. In the next two chapters the findings of the interviews that were done with practicing engineers are discussed.
7 Interviews: Collaboration Types

The fundamental design features were implemented in the design software. The design can begin with the creation of FE models and their analysis, followed by the design of the structural steel members. Files can be exchanged over a network which makes the most basic collaborative design possible. Furthermore it is possible to see the design activity that is taking place at the server terminal. Consequently it is possible to demonstrate basic concepts of collaborative engineering in the field of structural steel design. At this point it becomes important to determine what the practicing engineers think about the design software, designing structures in particular, as well as general engineering in a collaborative environment.

A survey was drawn up that was to be used to interview practicing structural engineers. The survey addressed how people work together in the design of structures, how they think collaborative designs could be done, what are the aspects that need to be considered when doing collaborative design and whether or not they believe there is a need for software that supports collaborative design. Aspects of the survey and its results are discussed in this chapter. The survey also addressed some additional topics regarding design software and other possibilities regarding collaborative work; those topics are discussed in chapter 8. In total 27 people were interviewed with various backgrounds, from young to old, from small firms to large firms, from structural engineers to software distributors to contractors and project managers. The survey was intended to obtain as wide a perspective as possible within the time available. The questionnaire used in the survey can be seen in Addendum A:

There are essentially three ways in which collaborative work can be done. These are sequential, parallel and interactive collaboration. The collaboration types were used to structure the survey i.e. the engineers were interviewed regarding their perceptions of and preferences of these collaboration types. In this chapter the collaboration types are explained and the findings of the interviews. The findings that are reported here are as perceived by the author. The raw data of the interviews, including voice recordings, can be reviewed in Addendum B:. On some questions a wide range of views were given by the engineers. In such cases only the most prominent views are reported.

The interviews also addressed some general topics that may influence the structure of the software or require that additional functionalities be included in the software. These topics are discussed in chapter 8.

7.1 Sequential Collaboration

Sequential collaboration is where two or more people work together on consecutive design tasks that are related. Sequential collaboration follows naturally in any design, as one task may have to be completed before another can be started. Sequential collaboration can be done within a phase or between phases.
7.1.1 In-Phase Sequential Collaboration

Figure 7.1: Sequential Collaboration within a Phase.

Figure 7.1 demonstrates sequential collaboration within a phase. An architect has developed the conceptual design of the structure which he then hands to an engineer. The engineer then begins to develop a FE model based upon the conceptual design. The engineer may develop the model up to a point which he then sends to another engineer to complete. The work done by the two engineers was in a sequence and within a phase, namely the FE modelling phase.

7.1.2 Between-Phase Sequential Collaboration

Figure 7.2: Sequential Collaboration between Phases.

In this scenario design tasks are also done in sequence, however the tasks are executed in consecutive design phases. This is demonstrated in Figure 7.2. In the first phase the architect develops the conceptual design. This is then followed by an engineer developing the FE model. In the last phase an engineer does the detail design of the structure. These phases take place in succession and the information created in the one phase is required as input for the next phase. If a change is then made to a model developed in the earlier phases, the models of the subsequent phases may have to be updated.

7.1.3 Interview Findings

In-phase sequential collaboration is not of interest for this thesis. To support sequential collaboration within a phase no special software support is required. One person can send a file to another, they
will use the same software and the second person can open that file with the software as is. The interviews therefore focused on between-phase sequential collaboration. In this section reference will be made to the hierarchy of models and a model may be referred to as a parent model or a subsequent model. The parent model is a model that provides information for a subsequent model and the subsequent model is dependent on the parent model.

The major issues that are of concern in sequential collaboration are the updating of existing models. Firstly, when a parent model is changed, a lot of work may have to be done to accommodate changes in subsequent models. The second issue is the communication of changes that have occurred. One person could make a change to a model and it could affect the work of a second person. If the second person is not informed he may continue with his work which may have to be redone later if that change was not communicated at the earliest time possible. The last issue deals with the updating of parent models due to information obtained through the subsequent models, for instance the engineer does the structural design on the architect's conceptual design. He determines that the conceptual design is inadequate and that it would have to be changed.

Consistency of consecutive design models:
Consistency of consecutive design models is a major problem in current design practice. Inconsistency between consecutive design models exists when a subsequent model is not based on the most up to date parent model. How this problem arises is described by an example. An engineer receives a conceptual design from an architect. Based upon the architect's drawings he develops a FE model, analyses it and does his detail design. The architect then makes changes to the conceptual design. At this point in time the concept of the architect, which is the parent model, and the model from which FE model is developed is no longer the same. The consecutive design models are now considered to be inconsistent. For these models to be made consistent again, the FE model would have to be updated according to the current version of the conceptual design.

To update subsequent models may be difficult and may be a source of frustration for engineers. Communication of changes that are made to parent models needs to be done as soon as possible. The sooner a change is communicated the sooner it can be accommodated and prevents unnecessary work being done. Current practice is limited by a lack of accommodation of changes that are made to parent models. At present there is little to no support to update subsequent models due to changes in parent models. Currently these updates have to be done manually by the engineer.

Forward feed of information:
A topic that is being considered for research and implementation in design software is the “feeding forward of information”. Feeding forward of information is when an intelligent link exists between a parent model and a subsequent model. When a change is made to the parent model the subsequent
model is informed. Support is then provided to assist the person responsible for the subsequent model to accommodate these changes in his/her model.

On this topic engineers stated that having such a facility available would be useful. When updating the subsequent model, the updates must be done subject to the approval of the engineer. There may be scenarios where engineers may want updates to be done automatically and a choice should then be given to the engineer whether or not he would like the updates to be done automatically.

**Backward feed of information:**
Another topic also considered for research and implementation is the backwards feed of information. This is similar to the forwards feed of information, however in this case the parent model is updated from information that has been derived in a subsequent model. This is specifically considered between FE models and steel member models. When a FE model is developed assumptions are made regarding the elements’ properties. When the FE model is analysed these elements’ properties affect the results of the analysis. In the steel member model the specific properties of a steel member is determined. In “backwards feed of information” the elements’ properties determined in the steel member model is imported into the FE model for reanalysis. This is to determine if there are major changes in the force distribution of the structure due to the change in elements and whether or not the steel members from the steel member model need to be redesigned. This feedback could also be used to determine if the steel members meet the serviceability requirements of the design codes.

The engineers considered the possibility of feeding information backwards as essential for future structural design work, but not in all projects, depending on their sensitivity to changes. When backwards feed of information is done it must not be done automatically, but under the control of the engineer. This functionality could be extended to sending design information to the architect so that he may investigate if the detailed design fits into his conceptual design. This is considered to be of less importance.

### 7.2 Parallel Collaboration

Parallel collaboration is where two or more portions of design work are done at the same time by different people, where the various portions of work add to the overall design of a structure. In structural design there are three possibilities of parallel collaboration that needs to be considered: parallel collaboration in the development of FE models, in the detail design of structures and in the design of independent sections.
7.2.1 Parallel Collaboration in the Development of Finite Element Models

In this scenario the development of a FE model is to be carried out by two or more people. Specific sections of the design are allocated to different engineers. Each engineer develops his section independently and once the sections are completed they are merged together to form a complete model. When these sections of work are merged together there are certain merging requirements that will have to be fulfilled.

Consider the multi-storey building shown in Figure 7.3. The building is separated in two parts, part A and part B, where part A is the upper section and part B the lower section of the building. One engineer will develop part A and another engineer will develop part B. The bottom nodes of the part A and the top nodes of part B will have to coincide. An interface will have to be defined for these nodes. When the files are merged together a check will be executed to ensure that these nodes do coincide, if not an error message will be displayed stating that the models do not align. Either the merging process will be cancelled or an opportunity will be given to the engineers to correct the discrepancies in the models. The errors could be highlighted to assist the engineer in correcting the errors.

7.2.2 Parallel Collaboration in Detail Design of Structures.

This scenario is similar to the previous. The difference between the scenarios is that now the detail design of a structure is to be done instead of developing a FE model. Sections of work are also
assigned to different engineers and the models developed by the engineers will be merged as well. Similarly certain merging requirements will have to be satisfied.

In a multi-storey building one would prefer to have one column size or profile run through a number of storeys of the building. When the design of such a building is separated the possibility arises that two engineers will choose two different size columns. When the different sections are merged, a check will be run to see if the column sizes are consistent. Should it happen that the profile sizes are not uniform a message will sent to inform the engineers of this. The engineers will then have the choice to ignore the message and keep the design as is, or choose an appropriate profile for both sections. The software will then redo the design calculations where necessary to check if the profile that has been selected satisfy the design requirements.

7.2.3 Parallel Collaboration in the Design of Independent Sections.

In this scenario a project is subdivided into smaller sections that are designed by different engineers. The most significant difference between this scenario and the previous two is that the sections are independent from one another. Consider the design of the airport shown in Figure 7.5. The piers and the entrance hall are the two sections which have to be designed. The piers will be considered independently from the entrance hall and once they have been designed there is no need to merge the models. To do this no special software support is required. From this point forward parallel collaboration will only refer to the first two scenarios.

7.2.4 Interview Findings

Regarding parallel collaboration engineers were interviewed specifically surrounding its usefulness and its possible usages. The issue of usefulness is to determine whether or not engineers believe that support of parallel collaboration is worthwhile. The “usage of parallel collaboration” is to find out what parallel collaboration will be used for and what functionalities need to be supported.
7.2.4.1 Usefulness of Parallel Collaboration:

The opinions of engineers varied considerably in their beliefs of whether or not parallel collaboration is useful. Some said that it was and other said it was not. Some considered it useful for FE modelling and other said that it can only be used in detail design of structures. The table below indicates the percentage of engineers that believed that parallel collaboration is useful for the given aspect of structural engineering.

Table 4.1: Usage of parallel collaboration

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite element modelling</td>
<td>53.8%</td>
</tr>
<tr>
<td>Detail design</td>
<td>76.9%</td>
</tr>
<tr>
<td>No usage</td>
<td>11.5%</td>
</tr>
</tbody>
</table>

88.5% of the engineers stated that some form of parallel collaboration is useful. A general provision for the usage of parallel collaboration is that the project would have to be large enough.

Those who were against parallel collaboration felt that it was too risky and that miscommunication could take place too easily. This could have some disastrous consequences. Concerns were raised with regards to parallel collaboration used for FE modelling, as the engineers believed this to be dangerous. It became apparent that where parallel collaboration is done in FE models care must be taken in dividing the sections of work. The critical sections of the structure must first be determined and some sections must not be separated.

52% of engineers believed that parallel collaboration is essential for future design and 48% believed it was not. 63.5% said that the engineering practice is not hindered by a lack of software support tools for parallel collaboration. It was evident that parallel collaboration may not be essential to do design work, but that it could be helpful to the engineering practice. It is particularly useful for allowing people to work away from the office.

In parallel collaboration merging support has to be provided. 62.5% of the engineers said that it is worthwhile to find solutions to problems related to merging. With a limited number of people available the integration and automation of the more mundane tasks becomes more important. The reasons why people were against development in parallel collaboration is that it is risky, that the projects on which they worked were not big enough or that there are other things that they deem to be more important.

7.2.4.2 Usage of Parallel Collaboration

During the interviews engineers made some general comments as to what parallel collaboration might be used for. They were also asked about applications for parallel collaboration or how the design process might be improved. From the engineers answers the following uses for parallel collaboration was recorded. The usages are categorized as follows: specific usages within structural design, usage
of parallel collaboration between disciplines and general improvements and additional recommendations.

1) Structural Design
   a) Separation of the design of the global structure from the local elements

![Figure 7.6: A Local Floor Being Designed of a Tall Building](image)

A local element is designed separately from the global structure. Checks are done to ensure that the local elements fit into the global structure and that no misalignments occur.

b) Building with a section below ground level and above ground

![Figure 7.7: Building with Sections Below and Above Ground Level](image)

One person is responsible for designing the top section and another for the bottom section. Both engineers need to look at how the two parts are connected. The engineer designing the bottom needs to know what forces are transferred at the connections. The engineer designing the top needs to know how the top is supported at the connections as it affects the structure’s stability. The software should support checking if the connections used by the two engineers are the same. If there are discrepancies, these need to be highlighted and corrected.

c) High rise buildings

In high rise buildings a building can be subdivided into sections that different engineers can then design. The software will need to integrate the sections into one model. It will need to ensure that nodes on interfaces align and that columns are in line with uniform profiles.
d) Stadiums

Stadiums have similar connection interfaces as that of a building having a section below and above ground level. In this scenario the separation takes place between the roof and the seating. The connections will have to be investigated similarly as what has been done in scenario 1 (Figure 7.8 a). Another possibility in stadium design is that a person could begin at one half of the stadium and define its elements and another person starts at the other end. They could work their way around until they meet up and connect the sections together to get a complete model (Figure 7.8 b).

![Figure 7.8: Parallel Collaboration in Stadiums](image)

2) Between Disciplines:

It is possible to include all the disciplines that are involved in designing and building a structure into one model. All the role players can then add their design information in the design model and have access to the design information of the others involved. This becomes useful where clashes may arise between the designs of the various disciplines. Each role player is mentioned below, with a description of their roles and the information they provide.

**Architect:** He is responsible for the conceptual design of the structure, its general appearance and functionality. He interacts directly with the client and puts into drawings what the client wants. As a client makes changes the architect edits his drawings and communicates these changes to the rest of the professional team. His work provides the outline in which all the other disciplines work.

**Engineer:** Based upon the conceptual design of the architect the engineer develops an analysis and design model of the structure. He checks that the elements of the architect are structurally adequate and he does the detail design of the structural elements. Together, the architect and the engineer makes decisions regarding the location and sizes of structural members.
**Contractor:** He is responsible for the construction of the building. His perspective is more practical, in how the structure will be built. He advises the architect and engineer on changes that may need to be made to the structure’s design for construction purposes. Changes are also to be communicated by him during the construction process, i.e. certain assumptions that were made in the design process is not true to actual conditions, for instance the soil conditions or a specific material may not be available.

**Manufacturer:** He retrieves the information regarding structural members that are required for the structure. He then quotes the cost of the design. Should it happen that the design is too expensive the architect and engineer will need to discuss possible alternatives to make the design cheaper.

**Mechanical and Electrical:** They need to ensure that there is place to insert their equipment for the building within the designs provided by the engineer and architect. If they find no space for their equipment they need to consult the engineer and architect on solving this problem. Heavy equipment may have major impacts on the engineer’s structural design.

One way of storing the information for such an integrated model is to make use of a centrally located database. The role players can extract the necessary information from the database and define their input in a section of the database that has been allocated to them. In the end one model then contains all the design information of all the disciplines. With this it is possible to run checks and see to it that no clashes occur in the input of the respective disciplines.

**General Recommendations:**

Some recommendations were made by the engineers of aspects that need to be considered for parallel collaboration.

- Find means to involve contractors in the design process as early as possible to obtain their ideas and take them into account when design decisions are made.
- Inform the various role players of changes that have been made to the design, so that everyone does their design work from the most up to date information
- The engineer responsible for the overall design of the project should define basic design parameters for a project. Where other engineers support the head engineer they will have to comply with the basic parameters that he defined. Say the head engineer specifies three column sizes as basic design parameters. Every time a column is designed by one of the engineers only those column sizes specified by the engineer can be used. This is a way to ensure conformity in structural design where several engineers are involved in the design.
7.3 Interactive Collaboration

In interactive collaboration only one model exists and all users edit and work from that model. When one user makes a change to the model everyone sees the change and the change is updated everywhere. Such a model could be used within a specific discipline (the detail design of structural members) or between several disciplines (architectural drawings and structural design).

Interactive collaboration requires good communication tools and clear definition of access rights for all the users. Not just any user should be able to change somebody else’s work, but there must be a means to communicate if one user needs another user to make changes to his work. The absence of these facilities could be a source for great frustration. A messaging system that informs users of changes have been made or a log of most recent changes could be very helpful.

One way to define access rights in the software is to specify layers within the working environment and assign each user to a particular layer. Consider Figure 7.9 and let’s say that User A is designing the foundations while User B does the structure. User A and User B are interested in what the other is doing, however User B would not want User A to change his work. Therefore User A works on layer 1 and he can only make changes to layer 1. User A can see the work of User B in layer 2, but not edit it.

7.3.1 Interview Findings

Like with parallel collaboration the engineers were asked whether or not they would use interactive collaboration in design and more specifically what they would use it for.

7.3.1.1 Usefulness of Interactive Collaboration:

In using interactive collaboration most of the engineers felt that a project would have to be large before something like this is required, however some saw it as the way of the future in design. Those who stated that they would not use such a model were not set against it, but they saw no scope for it within their working environments. They said that it might be useful in special scenarios that require intensive modelling and testing. Two concerns were raised by the engineers. Each person involved in the design of a structure should be responsible for the work that he has done. A record would have to be kept of the work each person has done and if something goes wrong they would be held
responsible. The second concern was establishing a hierarchy in the design environment. It must be possible to assign the responsibility of overall design to one person who would then coordinate the design and assign work to all the role players involved.

7.3.1.2 Usage of Interactive Collaboration

The main areas for which the engineers said they would use it are: Documentation and Drawings, Communication of changes, and Development and Design. Some did say they would not use the interactive model at all. Figure 7.10 shows a pie chart of the percentage engineers that stated they would use interactive collaboration and for what.

![Pie Chart of Interactive Collaboration Usage]

**Documentation and Drawings:**
Documentation and drawings denotes the systematic development of a project’s construction drawings. The documentation part is about recording decisions that are made during the design process. This is so that a history of the design process is available and that people can be held responsible for their design work. The drawings have to be versioned, and may be logged (i.e. frozen) at a certain point in time. Once a model has been logged it cannot change and may be referred to at a later stage to see why certain construction was done the way it was done. If these tasks are supported by an interactive model all participants are automatically locked into a disciplined way of working. The result will be less confusion and frustration, and more confidence in the results of the work.

The drawings part deals with tracking the evolution of drawings and creating an environment for users to concurrently develop drawings required in the design and construction processes. Tracking the evolution of drawings is always helpful when users wish to return to a previous point in time. Having the ability to develop drawings concurrently, could be particularly useful as it is probably the single aspect of design that is the greatest volume of work and requires the most time to complete.

**Communication of Changes:**
Communication of changes is important when a user edits a design model. The most recent changes made to a model are communicated to all people involved in the design, and the impact of these
changes can then be investigated. Say an engineer changes the foundation of the structure which results in a redistribution of forces. A check will have to be done to see if the structural members that have been designed are still adequate for these redistributed forces. If not they will need to be redesigned by the responsible engineer, otherwise the changes in the foundation has to be undone. This communication of changes therefore assists engineers in making decisions regarding the design of the structure. In an interactive model all the changes are communicated immediately; thus improving the integrity of the design process.

Development and Design:
Development and design comprises the entire structural modelling that is done in designing a structure. This starts with the development of the FE model, running analyses on this model and then doing the detail design of the structural members, such as columns, slabs, connections, etc. The engineers were more inclined to use interactive modelling in the detail design of structures rather than in FE modelling.

7.4 Feasibility of Collaborative Design
After the three collaboration types were discussed with the engineers, they were asked what they thought the impact would be of providing support that would allow engineers to do collaborative design from remote locations. In other words the software would enable one engineer in Cape Town and another in Johannesburg to actively work together on the design of a structure. Figure 7.11 shows the responses of the engineers to this question.

![Figure 7.11: The Impact of Software Supporting Design Collaboration from Remote Locations](image)

69% of the engineers believed that enabling engineers to do collaborative design from remote locations would have a positive impact in design practice. People would be able to do their design work from home. It would be easier to get more people to help out on a project as they would not have to travel to the design office. People from other cities could help out more easily. This is particularly useful to obtain specialist support. However, with all these benefits the design process will have to be carefully managed between the people that are involved.
The engineers were asked which of the collaboration types they regarded as the most important and should receive the most attention for future development. Figure 7.12 shows the percentage of engineers that prioritized the respective collaboration types.

![Figure 7.12: Priority of Future Software Development According to Engineers.](image)

Interactive collaboration, somewhat surprisingly, received the most support, followed by sequential collaboration. Some stated that interactive collaboration is the way in which future design work will be done. Parallel collaboration did not receive much support. This may have due to concerns about the merging operations that needs to be done in parallel collaboration and whether or not the merging operations are reliable. One engineer said that he saw no future for parallel collaboration.

Engineers were asked if they thought that there was scope for design software that supported network collaborative design in today’s design environment. 86% of the engineers responded positively and said that current design practice would welcome such design software. They did say that some persuasion may be required before engineers would adopt this way of designing structures. The following reasons were mentioned why engineers may hesitate to use the design software:

- If the software is not proven and the results produced by the software cannot be trusted.
- If the software does black-box design. In other words one is unable to determine how the software arrived at a certain result. The engineers needs to be able to determine to design decisions and assumptions that are made by the software,
- If the software is not user-friendly and the design steps that has to be followed is not logical.
- If the support for the software is inadequate.
- If the software it too expensive.

### 7.5 Future Development

In this section the view of the author is reported with regards to the future development of the integrated software. The section specifically addresses future development with regards to the collaboration types that are to be supported by the software and which types should be prioritized in the future.
As seen in the previous section, most of the engineers believe that future development should focus on interactive and sequential collaboration. Sequential collaboration should be properly researched, developed and implemented in the design software. Only once sequential collaboration is properly implemented should attention be given to interactive collaboration. It is not that interactive collaboration is less important, but that sequential collaboration provides the framework in which interactive collaboration can be developed.

In the design of structures there are basic steps that have to take place that cannot be altered. First an analysis has to be done, to determine forces and displacements. A structural member cannot be designed unless the forces acting on it is known. Nor can a connection be designed unless the members that are to be connected are known. This design process takes place in a sequence and sequential collaboration takes this process into account. Interactive collaboration takes place within the design phases not between the phases. Many people can work interactively on a specific phase. This is shown Figure 7.13.

![Sequential and Interactive Collaboration](image)

**Figure 7.13**: Sequential and Interactive Collaboration

Both sequential and interactive collaboration deals with the updating of models and the accommodation of changes. In sequential collaboration a parent model is developed and then a subsequent model is developed based on the parent model. If the parent model changes then the subsequent model needs to be updated. Interactive collaboration extends the functionality provided by sequential collaboration. When several users are working on a model, the moment that model is edited everyone sees the change. Interactive collaboration supports the immediate update of changes within a model. Say a group of engineers are busy with a FE model and a group is busy with a subsequent steel member model. For every change made to the FE model the steel member model is not updated. Only once the engineers have finished editing the FE model will it be necessary to update the steel member model and this operation is a function of sequential collaboration.

Currently the software does support some aspects of sequential collaboration. The steel member model is developed from a FE model. The links to the steel connection model should be established,
where a connection can be designed based on the information provided by the FE model and the steel member model. How the data of the models is linked should also be addressed. The steel member model is based on the FE model and the steel member model needs to obtain information from the FE model. Currently, the path of where the FE model is stored is saved as a property of the steel member model. When the steel member model is loaded, a command is executed to first load the FE model, from the path it has recorded. The danger comes in if the name of the FE model changes or if the FE model has been moved to another location. This would render a steel member model useless. These models should be linked more effectively.

It may be possible to store the data of all the models inside a database. This way the information is contained as a unit. Therefore whenever the steel member model is loaded the information needed from the FE model is readily available inside the database. At a later stage this would also make the implementation of interactive collaboration easier, especially where interactive collaboration between disciplines may be considered.

A study should be done on the impact of changes in the parent model on subsequent models, specifically between the FE model and the design models. Currently if a change is made to the FE model the steel member model will become corrupted. A couple of changes that might happen will be mentioned.

- A structural member comprises of several finite elements, say 3. If one finite element is deleted but replaced by another with the same properties as the one before, the structural member would be corrupted, if not the entire model.

- The nodes of the finite elements from which a structural member is built up may be moved. The nodes may be moved in two ways. Either the nodes will remain in a straight line, or it will form a bend, as shown in the two figures below. In the first scenario no problem should be experienced in the definition of the structural member, however in the second scenario two new structural members would have to be defined.

![Figure 7.14: FE Nodes Moved of a Structural Member (a) in a straight line and (b) in a bend](image)

- Support conditions or the loads in the model may have changed. The geometry of the structure would then remain unchanged. Therefore no new structural members need to be
defined. However, the forces would be redistributed. This would mean that the design of the structural members have to be redone.

- Finite elements may be added to the structure. This, like with the altering the supports and the loads would result in the redistribution of forces, but it should not affect the existing structural members. However, they may have to be redesigned.

These problems could be addressed by highlighting the changes made to the FE model in the steel member model when it is loaded. Every structural member for which the finite elements have changed are highlighted, say red. The old and the new finite element geometry could be shown and the engineer could then be given the opportunity to redefine the structural members. All the structural members for which only the forces in the finite elements have changed could be highlighted in green. For these the design calculations could be redone in order to check if the profile chosen for that structural member is still adequate. It may be worthwhile to provide a functionality that displays the force diagrams before and after the change in forces. If the forces have become significantly less, the profile selected for the member will still be adequate, but the engineer may want to choose a lighter profile. Such support for accommodating changes would have to be extended to the design of connections as well.

In this thesis the interaction between models were done at model or file level. Model files were transferred over the network. The doctoral study, Reference [7], dealt with network-based exchange at object level. This functionality makes interactive collaboration possible. The moment a change is made, an object can be sent over the network and the view someone else may have of the model is also updated. Research should be done on how this functionality can be incorporated in the integrated software to support interactive collaboration.

It is the author's belief that specific attention needs not be given to parallel collaboration before the interactive and sequential models have been thoroughly researched and developed. Comparatively, the engineers showed relative little interest parallel collaboration, however parallel collaboration can still be supported. In section 7.3 the significance of a history record is discussed and in that section a scenario is described as to how parallel collaboration could be supported. Therefore parallel collaboration should not be the focus of attention, but can be supported as a function of other aspects that are implemented in the software.
8 Interviews: General Aspects

In the previous chapter the sections of the interviews that dealt with the collaboration types were discussed. This chapter deals with some general topics that were discussed during the interviews. The findings that are reported in this section were not only derived from specific questions that were asked, but also side comments that were made by the engineers while discussing the three collaboration types.

8.1 File Management

File management refers to both the storing of files and their accessibility. File management addresses the following questions: where should files be stored, who should have access to which files, what type of access should users have to a file and when should files be accessible to users?

Storing of files:
Engineers preferred one of two options for the storage of files; files stored on the server only or a temporary copy stored on a local computer and the server is then updated at a later stage. Half the engineers preferred the first option, half preferred the other and it appears to be a function of how one’s design office is structured. Those in favour of the first option did not like the idea of many temporary copies since one person could overwrite the work of another. Duplicate files are being created and it becomes difficult to determine which copy or version is the most recent copy, and which version should be used. Where engineers do make use of temporary copies, they need to ensure that the server copy is updated regularly. To assist engineers in this an automatic update could be set at appointed times or whenever a user logs on or off the server. Temporary copies do have the benefit that it cuts out the possibility of one’s work being slowed down when the network connection is slow or lost.

Accessibility:
Accessibility refers to the access rights that a person has to a file, data or section of work. In this section accessibility will be addressed in terms of access to files; however this can be interchanged with data or sections of work. This is important as in some scenarios an office may not necessarily work with several files, but with one large integrated database. Then instead of having access to a file a person will have access to a section of the database.

Accessibility is about assigning access rights to various users to a set of files. Some users may only view files, others may edit files, some will have access only to some files while others may have access to all the files. Engineers were asked to select the option they agreed with most, out of the following:
a) Multiple users must be able to view and edit a file at the same time.
b) Multiple users must be able to view a file, but only one may edit it.
c) Only one user may view and edit a file.
d) All users must be able to view and edit a file, but one user has to approve all the changes.

60% of the engineers selected option b, while 35% selected option d. It is clear that when design is done that engineers want to have a measure of control, or someone must take responsibility. Everyone should not just do as they please, either one person edits the file or someone approves all the changes.

Many of the engineers stated they would like a hierarchy to be established within the software, particularly with interactive and parallel collaboration. A project coordinator should be allocated and he would then be able to define the access rights of the other users i.e. the project coordinator will give editing rights to the architect for the conceptual design and editing right to the structural engineer for the structural design. These two may have viewing rights to each others work, but would not be allowed to edit each other’s work. The structural engineer may have a junior engineer that will help him in the design. He could then give the junior engineer access rights to a portion of the design which he will edit. If people need access to a portion of work, they will only be able to get access of that work through a responsible person that will give them the access rights. This definition of access rights is important for the protection of the integrity of the design.

8.2 Networks

Most of the engineering practices operate within a local intranet, even within the larger companies; one branch would have its own intranet separate from the rest of the company. Their software applications generally operate independently from the internet. Sometimes the internet is used for transferring large files, but the transfers are not done through the design software. Currently the internet is not a major factor in design software as used in South Africa.

Engineers considered design software being dependent on a network as a good attribute since it is then possible to store files on a server. The batch files on the server could then be backed up regularly. Network-based software also allows people to transfer files through the network. However, 90% of the engineers did say that the software must be able to operate independently from the network. Engineers must be able to continue with their work should a network connection not be available. This means that design software must be able to retain local copies of files away from the server. Therefore it should be possible to update a file on the server from a local copy and vice versa.
8.3 Data Recording

Many of the engineers emphasized the importance of documenting designs and that there is room for improvement when it comes to the recording of data. Specific areas that were mentioned are recording design assumptions, calculations not done on a computer, logging of files, history record of the design and recording activity of what is happening in the project. Each of these will be discussed shortly, except for the history record which is discussed in section 7.4.

Design Assumptions:
During the design process assumptions are made to execute a design. When the foundations are designed the exact conditions of the soil may not be known, whether there is sand, clay or rock. Assumptions will have to be made and these need to be recorded with a motivation. Past experience may show that soil in that area is sandy; hence the foundation design was done for sandy soils.

Calculations Not Done on a Computer:
Some calculations are not done on the computer, but they will affect the design. These calculations may include; metal fatigue, vibration tests and hand calculations for checking computer designs.

Logging of Files:
Files that are sent between the professional team should be logged. A structural engineer does the structural design of a building and sends the structural drawings to the mechanical engineer. The file is then logged, and the logged file cannot be edited. The mechanical engineer fits his equipment into the structural framework. Say the structural engineer changes the structural design, without informing the mechanical engineer and construction then takes place. The equipment of the mechanical engineer no longer fits into the structural framework of the structural engineer. The structural engineer is then held responsible since he never communicated the change, and the files that have been logged are used to see if the changes were communicated or not.

Project Activities:
Records could be kept of all activities that take place in the design process i.e. conversations that take place between the professional team, on site visuals like films and photographs. A database is then kept with all these visuals and each entry is stored with a reason why it was taken. Conversations might have been about a specific design aspect or instructions of changes that need to be made, etc. Photographs could have been taken of cracks that occurred in walls, videos to record the construction process of slabs for quality control, and so on. Visuals can also be useful in recording a specific problem, which could then be sent to an expert in order to get their advice on addressing the problem.
8.4 History

Most of the engineers believe that having a history record of the design process available would be beneficial to the design practice. There are times when one would like to view what a model looked like at a previous point in time. Alternatively one may want to make significant changes to the model that may not work. If it turns out that these changes do not work, one could simply return to the point where the changes were made and continue from there. In that case it would be as if the changes were never made.

Today, the most common practice for keeping record of model development is by storing a series of versions of the model. This requires large volumes of storage space, while the same data is stored time and time again. An alternative approach to this is to store the changes that are made to a model, instead of storing all the data within the model. Its history then consists of a series of changes. The figure below shows the development of the structure where the red sections indicate the changes that have been made in consecutive steps, these steps could have been consecutive days. Currently three models would have to be stored in order to look back at these models at a later stage. If the model is stored as a series of changes, only one model would be required. Later when loading a model a user can specify the step he wants to view and the model is updated to that point. From now on a series model will refer to a model that is stored as a series of changes and a data model is a model of which only the data is stored.

![Figure 8.1 History Record of the Development of a Building](image)

A downside of the series model is that it requires a longer time to recreate than a data model. When loading a series model, the model has to run through all the changes up to the point that the user has specified, whereas a data model has all the information immediately available and loads it. This problem can be addressed by specifying a number interval of changes after which a data model is stored. When loading a model the nearest data model is loaded followed by the changes to the desired step. Say that 9140 changes have been made to a model and a data model is stored after every thousand changes. Then data model 9 would be loaded followed by the last 140 changes to get to the latest version of the model. If the user wants to load the model at a point of 5300 changes, then data model 5 would be loaded followed by the next 300 changes.
Understandably it is impossible to remember what the model looked like after every change. Therefore it is possible to associate a specific change with a label i.e. “Storey 41 complete” or “Section A and B connected”. One could then search through these labels to find the appropriate model version to load. Another way of finding the appropriate model to load is to specify a date, and the model version on that date would be loaded.

Storing models as a series of changes has more benefits than merely saving storage space. It is possible to make use of this facility to allow multiple users to edit the same file at the same time. In other words it would make parallel collaboration a real possibility. Consider Figure 8.2. Say User A and User B both access version 20 of a file. They continue their work independently of each other and User B completes his work before User A. User B then submits his work, the server model is updated and it becomes version 21. When User A finishes he wants to submit his model. However User A used version 20 and a version 21 already exists on the server. User A’s model as he worked on it is now out of date. All the changes that were made to the model in the meantime is loaded up to User A and he then has to ensure that the work he has done is compatible with the changes that have already been added to the model. Once User A has done this he can submit his file and version 22 of the model will be stored on the server.

![Figure 8.2: Editing Models Simultaneously and Managing Conflicts.](image)

Another useful feature that could be included is the recording of notes on changes that are made in the design of a structure. When a change is made the person responsible adds a note that motivates the change in design. Therefore when somebody else evaluates the design and notices the change, the note would be there to inform him of the change.

It is also possible to include aspects surrounding liability. When a user updates a model all the changes made by him can be logged and every change can be associated with a person. Therefore each person can be held responsible for the work that they have done in the design of a structure.
8.5 Recommendations for Design Software

Recommendations were made by the engineers regarding aspects that they felt should be included in design software or aspects that they saw was lacking in current design software. Some of these aspects related specifically to analysis and design of structures, others were additional functionalities that is helpful in the design process. The following categories will be discussed; FE modelling, detail design, links between disciplines and design software extras.

8.5.1 FE modelling

- Defining loads in FE models is a time consuming process. Ways should be investigated on how this input process can be made faster. A suggestion was that loads might be drawn in a CAD model, which could then be imported into the FE model.

- At present it is difficult to model combined steel and concrete structures in the same model. Concrete is usually modelled by shell elements and steel by beam elements. These shell and beam elements do not interact properly in FE models. A solution should be found to support the interaction between shell and beam elements.

- It should be possible to define pile supports where the soil conditions and stiffnesses can be taken into account. In the case where large foundations need to be modelled the user should not have to break up the large foundation into a number of smaller sections, where he will have to calculate a spring support for each of these sections. A more user friendly way should be found to define these pile support conditions.

- Modelling of cable structures are not handled well at present and they are becoming more popular. Modelling of cable structures should be improved.

- The calculation of long term deflections would be helpful in the design of structures.

- In the analysis of FE models the redistribution of moments should be supported according to the design principles that are used in the SANS codes. The load concentrations found in the supports has to be spread out to the rest of the structure.

- When designing steel frames it should be possible to define the endplate bolts with the foundation moments. The foundation takes the soil conditions with rotation into consideration together with the endplate when considering deflections. At present designs are very conservative. In the ultimate limit state design 2 bolts in a foot-piece is considered to be to be pinned. This is fine, however is the serviceability limit state design one can consider this connection to be practically fixed. This is not taken into consideration and often the members are then over designed.

8.5.2 Detail Design

- In the detail design of structural members a function could be included in the software in which the five lightest adequate profiles are determined from a database. The engineer is then given the option to select the profile that he wishes to use in the design.
• In the design of structural members there must be a wider variety of support conditions that can be defined, not only fixed or pinned. With limited support options available a mistake can easily be made in the design of a member, particularly for lateral buckling. (already supported in the design software)

• Design software could be extended to support the design of wood structures according to the SANS codes.

• The design software should have proper calcsheets for documenting the design. When designing a member the bending moment diagram, axial force diagram and shear force diagram should be displayed so that one can see what is happening in that structural member. With the design calculations that are executed by a computer it should be possible to see the decisions that were made by the software, which formulas were used and the process that was followed in the design of a structural member. The entire design process does not have to be shown all the time, but the user should have an option of how detailed the design output should be.

8.5.3 Links Between Disciplines

Architects:
It would be useful if an intelligent link could be established between architectural drawings and engineering FE models and design models. When changes are made to the architect’s drawings, it should be possible to update the engineering models from the changes made by the architect. A way in which this could be supported is by defining intelligent lines in the CAD environment. A line in a drawing is no longer just a line with a start and end point. That line could represent a beam element or a series of lines could represent a slab in a FE model. The architect would not define the structural properties, but the engineer would define these properties based on the architects drawing. The structural information that has then been defined by the engineer on the architect’s drawings would then be transferred to the design software. The architect would then still be able to change the lines, and when he does, the engineer can be informed of the change.

Manufacturers:
The design information could be linked with the High Level CAD systems used by manufacturers. Manufacturing schedules could then be determined from the design information which could save time.

Suppliers:
A link is established to a supplier’s database, so that engineers could do their design with materials and components that are currently available. For FE modelling and analysis the steel profile information could be obtained directly from the local steel suppliers.
8.5.4 Design Software Extras

Messaging System:
Most of the engineers stated that network-based collaborative design software must have a proper messaging or notification system. Whenever changes are made to a model that could affect the work that is done by another person, that person must be informed. Every effort should be made to keep all the people involved in the design informed of design changes or updates. In practical terms it may be annoying to receive live prompts for every change that is made to a model. In such a scenario a user should be able to choose not to see the prompts or to specify a time period after which he would like to see all the prompts.

It would be helpful to send messages directly to other users, where this sending of messages is a function of the design software. If the person to whom the message is sent, is offline when the message is sent, then that user will receive the message when he logs in again.

Help System:
A good help system should be included in the software. It needs to be easy to understand and a user must be able to find the information that he is looking for with relative ease. The definitions used should be clear; in order that users may know exactly what is being talked about i.e. in the design software reference is made to several models, namely the FE model, the steel member model and the connection model. When talking about a model it should be clear to the user which model is being talked about.

When doing analysis or design there are times when certain parameters need to be defined for a certain analysis. When these parameters or default values have to be defined, users must be guided in the selection of parameters. Users should be informed of the significance of each parameter that is entered and examples of what good values may be should be given.

Interaction with Word Processors and Spreadsheets:
The software should be able to interact with word processing and spreadsheet software packages like MS Word and MS Excel. It should be possible to copy data to and from the one to the other.

Quantities and Cost Estimation:
The calculation of quantities and cost-estimation may be included in the software. Cost optimization could then be considered in the design. It is possible to obtain the materials and steel components that are available from local suppliers, and the cost of each. A design might have been done using a steel profile that is not locally available and this will lead to increased costs. It may then be worthwhile to make use of a larger profile that is locally available at lower cost. In most scenarios the construction industry is cost driven.
Process Modelling:
Proper evaluation of the work distribution and the project scheduling has to be done. Where possible
the work should be evenly distributed between the people available for the design of a structure. A
situation should be avoided where some people could be over worked while others are able to
manage more work. Project scheduling is very important. The right work needs to be done at the right
time. Certain tasks can only be executed after another task has been completed i.e. a connection
cannot be designed until the members that are connected has been designed. Therefore one wants to
ensure that the work is done in such a way that the overall project is done in the least amount of time
possible. A decrease in time will save money in the project.

8.6 Future Implementation

This section contains the views of the author regarding the inclusion of general aspects into the
design software.

The management of files and the record keeping of a model’s development history are the two most
important aspects to be considered for future implementation in the design software. The file and
data management system is the single most important aspect of design software that supports
collaborative design over networks. The transfer of information is the lifeblood of collaborative design
software and it is affected by the management and recording of data. It is suggested that all the
design information be stored in an integrated database, and all the users will retrieve the information
they need from this database.

A critical aspect that must be included in the design software is the definition of access rights of the
various users; especially where multiple users work on single models that contain all the information
of the design of a structure. Liability is a major concern in the design practice where people are held
responsible for their design work. There can be very severe consequences when things go wrong in
the design of structures. Therefore it is of the utmost importance to engineers that other people do
not interfere with their design work. Defining the access rights of people in software is not a new
study; it is merely a critical aspect that must be included in the design software.

Maintaining the history record as a series of changes, is new to structural design software and gives
rise to several new possibilities. The first benefit is that less storage space is required to keep a record
of the design model’s development, secondly it makes the editing of a file by multiple users possible.
Having a history record available also gives engineers the freedom to explore more in their design
work. When an engineer explores a new design possibility, it may go wrong and the model may
become useless. Now it is easier for the user to revert to a previous point in the model’s history
before the changes were made and continue his work from there. An additional benefit to this type of
model storage is that smaller volumes of data can be sent to update files over a network. Since
models are stored as a series of changes only the changes that have been made to the model has to
be sent to update the same model on another computer. This makes designs being done over the internet a more feasible option.

The implementation of a history record would require a significant change to the architecture of the design software. The software will have to implement a command based architecture. MS Word is an example of a software package that implements such an architecture. Say a portion of text is accidentally deleted in MS Word. It is possible to give a command to the software to undo the action, deletion of text in this case. Furthermore if an action was undone it is possible to redo that action. In command based software every task that is done, is done through a command object. In order to record the design history of a model, the actual commands that are executed have to be stored and not only the data. MS Word does not do this, it only stores the data. When an existing MS Word document is loaded it is not possible to undo any actions, which means no commands have been executed. This is not true, commands had been executed, but none have been recorded. MS Word does not record what happened earlier in the document, which is what is aimed to be supported in the design software.

A decent communication tool and help system is required in the design software. The communication system will have to support the notification of changes made to models between users. It will also have to support the direct messaging between users. The help system must be able to effectively guide users to information that they need. It needs to be clear in its definitions and explanations. These two aspects, though very important, do not require immediate attention. Once the fundamentals, like storage of data, keeping history records, linking between design models are established, then attention can be given to these. Once these aspects have been researched and implemented it would be necessary to re-evaluate the direction of the development of the design software.

Two additional aspects that are worth mentioning for inclusion the software are process modelling and establishing design links between CAD model and FE models. Currently research is done at the University of Stellenbosch in these fields, and pilot softwares for supporting these aspects are being developed. It should be possible to integrate these functionalities into the design software.
9 Conclusions

This thesis forms part of an extended project in which software is developed to support network-based collaborative design of structures. Three projects preceded this thesis study, which provided the context in which this thesis commenced. The work done in this thesis built upon the work done in these preceding projects. In the preceding projects software architectures for a FE model, a steel member model and a steel connection model were developed. These models form part of an integrated software and the aspects that were not covered in the preceding projects were addressed in this thesis.

This final chapter is subdivided into two sections. The first section comments on the software development that was done in the thesis and the second summarizes the findings of the interviews conducted with the practicing structural engineers.

9.1 Software Development

Firstly a GUI was developed to support engineers in the creation of FE models. The GUI that was developed is only a pilot software. The purpose of its development was to contribute in demonstrating to engineers the possibilities for collaborative design supported by software. The software is therefore not of value for commercial purposes. In the future a more advanced GUI should be developed, that supports a 3D viewing and the creation of a wider range of FE models. However, the two concepts “registering an deregistering of mouse-listeners” and “hiding and displaying popup frames” are effective software strategies for allowing users to interact with the software. These strategies may be useful for the development of a more advanced or commercial GUI.

Links had to be established between the models that were developed in the preceding projects. The link between the FE model and the steel member model had been established earlier, but it was updated in this thesis. The steel member model was connected to a FE model that had become out of date. The information transfer from the FE model to the steel member model had to be altered in order to design the structural members from the most recent version of the FE model software. The updated link transfers the information from the FE model to the steel member model adequately, but it is flimsy. The two models are linked by recording the location of the FE model as an attribute of the steel member model. The FE model can then be loaded through the steel member model when it requires information from the FE model. However, if the name of the FE model changes, or if the file is moved to another location the steel member model will be rendered useless. In the future research should be done in how this link can be strengthened. A possible solution to this problem may be by storing the data of the various models in an integrated database.

Unfortunately time did not allow for the links to the steel connection model to be established. The steel connection model requires input from both the FE model and the steel member model. The FE
model contains the forces that act within a connection and the steel member model contains the information regarding the profiles that are to be connected. The steel connection model has to be linked to these two models in order to get a hold of this information. When the link between the steel member model and the FE model was established, some architectural changes were made to the steel member model (discussed in section 4.1). Many of these changes will have to be made in the steel connection model, when it is linked to the FE model and the steel member model.

A software architecture was built into the integrated software in order to support communication over a network. This was done so that a user would not be confined to a local workstation. The network communication ability of the software allows the user to work away from the office and still have access to the relevant files and design information. Currently the network communication capabilities are confined to the transfer of files (loading and saving files). With the existing framework it should be easy to extend this functionality to support a wider range of communication capabilities.

At a later stage in the software development attention must be given to the aspect of security. Currently no restrictions are placed on client applications that log onto the server. Therefore, they have the ability to access and change anything on the server. This is potentially dangerous and will have to be addressed, but it does not need immediate attention.

9.2 Interview Findings

In the latter part of the thesis, engineers were interviewed with regards to how they collaborate in the design of structures, how collaborative design could be done and what needs to be taken into account when doing designs in collaboration, specifically when supported by software. During the interviews three collaboration types were discussed namely: sequential collaboration, parallel collaboration and interactive collaboration. The engineers were asked which of the collaboration types they considered to be the most important. Interactive collaboration was prioritized by most of the engineers and sequential collaboration was second.

Despite interactive collaboration being prioritized by the engineers, sequential collaboration should be prioritized for future development. The reason for this is that sequential collaboration provides the platform on which interactive collaboration is to be developed. One of the aspects that must be addressed in future research is how changes made in parent models can be accommodated in subsequent models. In section 7.5 suggestions are made as to how changes can be accommodated between the FE model and the steel member model.

Once sequential collaboration has been well researched and implemented, then interactive collaboration can be developed from the basis of sequential collaboration. The research done in Reference [7] can be used for the development of an interactive model.
Additional aspects that affect the effectiveness of network-based collaborative design software were also discussed in the interviews with the engineers. The most important aspects that have to be taken into consideration for the future development of the software are the data management system and maintaining a history record of a model by storing it as a series of changes.

Much attention needs to be given to the development of an effective data management system. Information transfer is the lifeblood of the integrated software and it is directly affected by the data management system (storing, loading and distribution of files or data). It has been suggested earlier that storing the models in an integrated database may be a way of organizing the data. If an integrated database is not used, research must be done to find alternate solutions.

Developing the ability to store model as a series of changes is also a worthy aspect to include in the software. There are several benefits to this. Firstly a history record of the model’s development is made available, with not much additional space being required for this. Engineers have more freedom to explore new design possibilities, as changes made to models, can more easily be disregarded. Allowing multiple users to edit a file at a time also becomes a more feasible option. Research in this is a worthwhile pursuit.

Lastly some other aspects that may also be included into the integrated software are:

- Proper messaging and notification system
- User friendly help system with clear definitions
- Process modelling within the structural design process
- Establishing intelligent links between CAD models and analysis models.
Bibliography


[3] HEWETSON, C.G. Object-Oriented Steel Member Design Framework, University of Stellenbosch, 2005


[7] VAN ROOYEN, G.C. Structural Analysis in a Distributed Collaboratory, University of Stellenbosch, Department of Civil Engineering, 2002


[10] Book on surveys
## Addendum A: Survey

### PARALLEL COLLABORATION:

#### Essay Questions:

<table>
<thead>
<tr>
<th></th>
<th>Question</th>
<th>(Y/N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Would you use parallel collaboration as described in the scenario of the 5-storey building?</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Do you consider such collaboration as essential for future design work?</td>
<td>(Y/N)</td>
</tr>
<tr>
<td>3</td>
<td>Do you think current practice is hindered in its work by lack of software support tools for parallel collaboration?</td>
<td>(Y/N)</td>
</tr>
<tr>
<td>4</td>
<td>Are there other applications where you would use parallel collaboration if merging support is available? (i.e. The development of drawings)</td>
<td></td>
</tr>
</tbody>
</table>
5. Can you think of what merging requirements would be necessary for such an application?

6. With the design of large buildings all engineering offices perform parallel work. This is done by separating the parts that can be done in parallel which is then merged again in the end. If possible, how could this process be improved by supporting software?

7. The greatest challenge in parallel work is to merge the parallel streams of work that is generated. In certain cases this is relatively easy in others conflicts occur that makes this merging process problematic. To find solutions to this problem is rather difficult. Do you think engineering practices would use such an application?

8. Would you say it is worthwhile to find solutions to this problem?
### SEQUENTIAL COLLABORATION:

Assign a value from 1-5 to the following statements, where 1 you disagree strongly, 3 neutral and 5 you agree strongly:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9.</td>
<td>Consistency of consecutive design models is a major problem in current design practice?</td>
</tr>
<tr>
<td>10.</td>
<td>The feeding forward of information (CAD -&gt; FEM -&gt; Design) would be used in practice.</td>
</tr>
<tr>
<td>11.</td>
<td>Existing practice is limited by a lack of accommodation of changes that are made to the parent models.</td>
</tr>
<tr>
<td>12.</td>
<td>The designer of the design model must be informed when changes are made to the parent FEM model.</td>
</tr>
<tr>
<td>13.</td>
<td>A design model must be updated automatically according to the changes that have been made to the parent FEM model.</td>
</tr>
<tr>
<td>14.</td>
<td>You would like to see that changes made in the FEM model is highlighted in the design model, so that the necessary adjustments can be made in the design model, subject to the approval of the engineer.</td>
</tr>
<tr>
<td>15.</td>
<td>The backwards feed of information from the design model to FEM model for reanalysis is essential for structural design.</td>
</tr>
</tbody>
</table>

### Essay Questions (Technical):

16. Can you think of a way how information can be “bound” in CAD to information in FEM models to support the feeding forward of information?

17. What do you think of the following proposal? You define “structural components” in CAD (e.g. floors, shear walls, beams), and bind these to components in the FEM model (the FEM model will have finite elements that represent each structural component). So, when any change occurs to the CAD structural component, the analysis engineer can see that his FEM model is no longer consistent. He is then supported in re-meshing the component in order to obtain a consistent FEM model again.
**Essay Questions:**

Would you use the software as in the subsequent situations, provided adequate communication tools are available?

18. FEM:

19. Member / Connection Design:

20. Finishing Stages:

21. Are there other scenarios where you would use such a tool?
GENERAL:

Select one that you agree with most:

22. On which type of collaboration should the developers of software focus?
   a) Parallel Collaboration
   b) Sequential Collaboration
   c) Interactive Collaboration

23. Which type of network would you use for the software?
   a) Intranet (fast in one office building)
   b) Expanded intranet (fast, distributed and expensive)
   c) Internet (slow and geographically distributed)

24. Storing of files:
   a) Files that you are working on must be stored on your computer
   b) All files should be stored on a server
   c) A temporary copy must be stored on your computer and as soon as you are finished with it, the file on the server side must be updated.

25. Management of files:
   a) Multiple users must be able to access the same file at the same time and be able to edit it.
   b) Multiple users must be able to access the same file at the same time, but only one may edit it.
   c) Only one user may access and edit a file at a time?
   d) All users may access and edit a file at a time, but one user has to approve all changes made to the file

Assign a value from 1-5 to the following statements, where 1 you disagree strongly, 3 neutral and 5 you agree strongly:

ACCESSIBILITY:
Users must be able to set the type of accessibility to a file.

Users must have the freedom to access each others work and make changes.

Users must be able to send work to one another for the other to complete.

Users must have access to one another’s work, but not be allowed to edit it.

Users must be able to make comments on one another’s work (i.e. propose changes)
### HISTORY:

26. During the design process you would like to return to a previous point in time to see what the model looked like at that time.

27. Engineers must be able to make comments that declare why certain changes were made in the design process.

28. It makes no difference to you whether or not a history record is available.

29. The engineering practice would benefit a lot by having a history record available.

### NETWORKS:

30. You are satisfied with software that is dependent upon a network. (It cannot operate without a network)

31. Users must be able to continue their work should a network not be available.

### INTERNET:

32. The internet speed is currently a hindrance in South Africa.

33. You are satisfied with the internet speed as it is in South Africa.

34. High speed networks are rapidly entering the country to such an extent that the internet speed will no longer be a hindrance.

### GENERAL:

35. There is place for such a distributed design software.

36. Current engineering practice would welcome such a distributed design software.

37. You see no real difference in the proposed software than that which is currently available.
**Essay Questions:**

38. Are there any reasons why you would be hesitant to use such a design software?

39. What utilities do you think should be included in the software? (i.e. Communication between users, Server and Client)

40. Are there any shortcomings in current software tools that you would like to see addressed?

41. Do you think allowing an engineer in one office to be actively involved in a design taking place in another office will have an important impact?
Addendum B: Interview Data

The content of the interviews are recorded on an additional DVD and a CD provided at the back. The DVD contains all the information regarding the interviews; this includes the raw data files, processed data and the actual voice recording of the interviews. As for the CD, it only contains the raw data and the processed data.
Addendum C: Source Code

The content of the source code is located on an additional disc provided at the back. Only the most prominent portions of code that were developed in this thesis are mentioned here. The path names of the various sections of code are located will also be referenced. The path name will start with "D:\" where this refers to the drive letter of the local computer’s CD-Rom drive.

FEMCreator (Chapter 3)
Path: "D:\Code\Integrated Software\design\gui\fem\"
There are two files that make up the FEMCreator software FEMFrame.java and FEMPanel.java. Both of these files are found in this directory.

Registering and Deregistering of Mouse-Listeners (Section 3.1.2)
This concept was implemented in two places in the software namely, in the FEMPanel of the FEMCreator and in the SSMemPanel of the steel member model GUI. The location of the FEMPanel is mentioned in the previous paragraph on the FEMCreator. SSMemPanel is located at:
Path: "D:\Code\Integrated Software\design\gui\members\"

Remote Browser (Section 5.2.1)
Path: "D:\Code\Integrated Software\management\utils\filechooser\"
There are five files that are used in the remote browser software. These are:
ClientFileChooser.java
FileChooser.java
IFilterFileChooser.java
RDSfile.java
RemoteDirectoryService.java
The work developed in this section is based upon the work done in Reference [4]. The file RemoteDirectoryService.java extends one of the files that were developed in this reference, which is DirectoryService.java. DirectoryService.java is located at
"D:\Code\Integrated Software\utilities\directoryService\"

Popup Frames (Section 3.1.2)
The source code location for popup frames is the same as for Registering and Deregistering of Mouse-Listeners that is discussed earlier in this addendum.
Server (Section 5.2.1) and Client (Section 5.2.2)

Path: "D:\Code\Integrated Software\management\server_client\"

The section formulates the software architecture that allows for network communication within the software. The files included in this section are:

Client.java
ClientProfile.java
IServer.java
Server.java
ServerSettings.java