Concurrent Engineering through Parallelization of the Design-Analysis Process

Eric Joseph Wardell
Brigham Young University - Provo

Concurrent Engineering Through Parallelization of the Design-Analysis Process

Eric Joseph Wardell

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Brigham Young University

May 2015

ABSTRACT

Concurrent Engineering Through Parallelization
of the Design-Analysis Process

Eric Joseph Wardell
Department of Mechanical Engineering, BYU
Master of Science

The disconnect between the way CAD and analysis applications handle model geometry has long been a hindrance to engineering design. Current industry practices often utilize outdated forms of geometry transfer between these different engineering software applications such as neutral file formats and direct translations. Not only to these current practices slow the engineering design process but they also hinder the integration of design and analysis programs.

This thesis proposes a new, multi-user, integrated design-analysis architecture which allows auxiliary functions such as analysis and computer-aided manufacturing to be better connected with the computer-aided design. It is hypothesized that this new architecture will reduce the time of design-analysis iterations and create more parallelization between CAD and auxiliary programs. A prototype of the proposed architecture was constructed and then tested to evaluate the hypotheses, from which it was discovered that the proposed architecture does indeed reduce the time of iterations in the design-analysis cycle and allows for the parallelization of some design and analysis tasks.

Keywords: concurrent engineering, collaborative engineering, collaborative design, multi-user, FEA, CFD

ACKNOWLEDGMENTS

I would like to thank Dr. C. Greg Jensen from Brigham Young University for not only being my graduate adviser but also for being my mentor throughout my time as an undergraduate and graduate student at Brigham Young University. He has recommended courses to take, given me career advice, and been an invaluable connection in obtaining internships and employment. I would also like to thank Dr. Ed Red and Dr. John Salmon for their willingness to be on my graduate committee, offering suggestions and advice to guide my research and thesis. Additional thanks to Pratt & Whitney for funding the research on design-analysis integration, to Mark Trent for his help in creating the code to implement parameter changes from ANSYS into NX Connect, and to David French for his assistance in understanding NX Connect better and his help in planning the tests. Finally, I would like to thank my wife Rachael for her patience and support of this thesis and of my educational pursuits, and for my son Jonathan's and my angel daughter Elizabeth's love during this time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

API	Application Programming Interface
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CAx	Computer-Aided engineering applications such as CAD, FEA, CFD, CAM etc.
CFD	Computational Fluid Dynamics
FEA	Finite-Element Analysis
MU	Multi-User
SU	Single-User
UDO	User Defined Object

CHAPTER 1. INTRODUCTION

An iterative sequence exists within the engineering design process. This sequence consists of design, analysis, and evaluation. Certain requirements and criteria are established to describe a successful design. A concept, or idea, is generated, and then an engineering design, often in the form of a CAD model, is created from the concept. The design is analyzed in one or more ways (e.g. structural analysis, kinematic analysis, fluid-flow analysis, etc.). The results of the analysis or analyses are evaluated against the requirements and criteria of the design, and if these requirements and criteria are not satisfied then the design, or CAD model, is changed, analyzed again, and reevaluated. The process is repeated until the evaluation confirms that the design's requirements and criteria have been satisfied. This iterative sequence is a useful and necessary tool within the engineering design process, and is a method by which engineering designs are refined. The iteration can be manifest in several different ways throughout an engineering design project.

1.1 Problem Overview

One of the ways in which this iterative sequence is manifest in an engineering design project is through the use of computer-aided engineering tools, often referred to as CAx applications and programs. These CAx applications include Computer Aided Design (CAD) programs, Finite Element Analysis (FEA) tools, Computational Fluid Dynamics (CFD) programs, heat transfer simulations, modal analyses, Computer Aided Manufacturing (CAM) software, and others. The CAD models represent the design, while the FEA/CFD/etc. programs perform the analysis within the iterative sequence. There are two major problems with the CAD-analysis design sequence. First, commercial CAx programs are designed for only a single user to work on any given model at a time, and second, when models are transferred between CAx programs much of the data within the models, including geometric features and parameters, is lost.

Currently in industry the design-analysis sequence is serialized and single-user. For example, the commercial CAD and analysis programs of today allow one user at a time to create or edit a model. For many parts and models this is not a concern, but there are instances in almost every engineering project when the single-user nature of the application causes bottlenecks. One example is the dimensioning/drafting of a complex part, where ideally multiple drafters would work together on the task, but in reality only one can work at a time. Another example of the limitations from single-user CAx programs is the gathering and linking of numerous parts within a CAD assembly file. It is common for assemblies to contain thousands or tens-of-thousands of parts which must all be linked and assembled by the single user who has control of that assembly file.

In an engineering design project the groups of designers, analysts, drafters, etc. are often referred to as "silos" because of their isolation and disassociation from each other. The transfer of data and models between these silos is referred to as "throwing [the data and models] over the wall." This data transfer takes place through neutral file formats and other archaic means, and the models lose much of their important data in the process (parameters, features, part history, etc.). The disadvantages of having models and data "thrown over the wall" are especially evident when models must be updated. For example, if a CAD model has been transferred to an FEA preprocessor where the mesh and boundary conditions have already been set up it is very difficult to instantiate new changes and updates from the CAD model to the FEA model without first deleting the existing mesh and boundary conditions. Often large portions of the finite-element model must be recreated with each update from the CAD model. Models and data are "thrown over the wall" both from CAD to analysis applications and from analysis programs back to CAD, and the transfer of data is equally problematic in either direction.

Though there are multiple definitions for the term "turn-back," within this thesis it is used to describe when a design does not pass the evaluation stage of the iterative sequence. If the evaluation of the analysis shows that the design does not satisfy the established requirements and criteria then a turn-back occurs and the model returns to the design stage. It then must be modified and redesigned, thus beginning another iteration of the design-analysis-evaluation sequence. Often multiple turn-backs (and therefore multiple iterations of the sequence) occur before the requirements and criteria are satisfied. Multiple turn-backs frequently occur when analyses of different disciplines are performed on a design. These different analyses (e.g. structural analysis, kinematic analysis, fluid-flow analysis, etc.) commonly have conflicting constraints. For example, in the design of structural support vanes within a channel where fluid flows the structural analyst will push for the vanes to be wide, for strength, while the fluid analyst will push for the vanes to be thin and narrow, to allow better fluid flow. It is common for multiple turn-backs to occur in a scenario such as this before a design is achieved which satisfies the requirements established for both the structural and fluid analyses. Whereas this example contains just two conflicting analyses disciplines, in the real world a design is not complete until *all* analysis disciplines are satisfied.

1.2 Thesis Objective

The purpose of this thesis is to discover the benefits of collaborative design through parallelization of the design-analysis process. When designers and analysts share a common model the data flow between CAD and analysis programs is greatly improved. It is believed that within such a scenario the time spent by design turn-backs is significantly reduced.

Within this thesis an architecture is presented which encapsulates a parallel design-analysis process. This architecture will support a multi-user environment, in which multiple designers may work concurrently on a single CAD model. The architecture will also include robust data sharing between the CAD and analysis programs. The thesis will attempt to answer the following questions:

1. Does the proposed architecture reduce the time of iterations in the design-analysis cycle?
2. Does the proposed architecture reduce the number of steps in the design-analysis cycle?
3. Does the proposed architecture create more parallelization between CAD and auxiliary programs?
4. Does the proposed architecture create new conflict between designers and analysts?

1.3 Delimitations

It must be understood that the author does not claim the proposed architecture to be the ultimate solution to the problem of CAD-analysis integration. There are several major difficulties which prevent the two types of programs from interacting perfectly [1]. These include the manner in which the geometric data is defined and built, and the necessity for geometry cleanup and idealization in preparation for analysis. The proposed architecture addresses some of these difficulties but it does not solve all of the problems which prevent seamless integration.

Though parts of the software used to test the thesis are commercially available, including the single-user version of NX[1] and the ANSYS Workbench[2] suite of analysis programs, other portions are not, namely the multi-user NX Connect CAD design application. The purpose of the software prototype is to test the method in a specific paradigm, and therefore was not built to function for all scenarios found in the world of engineering design and analysis.

Though the proposed architecture includes a multi-user CAD system in the testing the benefits of having multiple CAD modelers will not be evaluated. The purpose of this research is to test the effectiveness of the design-analysis parallelization, not the benefits of multi-user CAD systems. The benefits of having several designers work concurrently to build the CAD model have been established in previous research [2–8].

The CAD-analysis connection within the proposed architecture allows the analysts to transfer geometry data to the analysis programs even before the designers have completed the CAD model. This would allow the analysts to begin to set up their simulations on early, "rough" versions of the CAD model while the designers are still working, and then update their simulations with the final geometry when the CAD model is complete. This example of parallelization has the potential of yielding significant time savings in the design-analysis process. However, the testing in this thesis will only measure qualitative observations, and not quantitative measurements, regarding this capability. An in-depth, quantitative study of collaboration benefits stemming from this feature is beyond the scope of this research.

It can be speculated that increased communication and collaboration between the designers and analysts provided by the proposed architecture may cause better decisions earlier in the engineering design process and thereby reduce the number of turn-backs and/or cause the creation of "better" designs. However, an investigation into the reduction of turn-backs in the engineering design process because of the proposed architecture is beyond the scope of this thesis.

[1]NX is a trademark of Siemens PLM Software, Inc.
[2]ANSYS Workbench is a trademark of ANSYS Inc.

1.4 Thesis Organization

This thesis is organized into six chapters. Chapter 2 is a literature review which introduces the reader to relevant research related to this thesis. This includes a brief introduction to the need for collaborative engineering, the current state-of-the-art in Multi-User CAx programs, and the work being done to integrate engineering design and analysis programs. It also explains principles of the engineering design process and the design-analysis iterations. Chapter 3 discusses methods to achieve concurrent engineering through parallelization of the design-analysis process and presents two multi-user, integrated design-analysis architectures. Chapter 4 discusses the implementation of these methods and creation of a prototype of the architecture. Chapter 5 presents the tests performed on the prototype and the results of those tests, while Chapter 6 discusses the conclusions and future work of the research.

CHAPTER 2. BACKGROUND

2.1 The Need for Collaborative Engineering

The International Journal of Collaborative Engineering defines collaborative engineering as a discipline which "studies the interactive process of engineering collaboration, whereby multiple interested stakeholders resolve conflicts, bargain for individual or collective advantages, agree upon courses of action, and/or attempt to craft joint outcomes which serve their mutual interests." [9]. In other words, collaborative engineering involves engineers and other invested parties working together towards a common goal. In every complex engineering project there exist conflicting objectives which require compromises to be made. Therefore, communication is key to collaborative engineering, both communication between people and communication between programs. One manifestation of advances in collaborative engineering is the tighter integration of the design-analysis process. This tighter integration allows analyses to be performed earlier in the design process [10]. These early analyses result in faster design times and lower costs [11]. In reference to the need for improved collaboration Y.-S. Ma et al. explained that "global product design and manufacturing has been pushing the adoption of a combined concurrent and collaborative engineering approach" [12]. With the demand for better designs in shorter periods of time the need for collaborative engineering has grown, and these advances in technology are attempting to meet that need. Janeen Hammond et al. explained that "the idea of enhancing the role of technology to make collaboration between team members more effective has definite merit and a promising future" [13].

2.2 Multi-User CAx

A multi-user CAD architecture is essential to the multi-user integrated design-analysis architecture proposed in Chapter 3. Significant research has already been done on the subject of multi-user engineering programs. Edward Red et al. point out the "dichotomy [that] exists in the engineering design process between 1) product teams organized to engineer products collaboratively, and 2) the single-user architectures inherent in computers and computer-aided design applications (CAx)" [2]. "Simply stated - our modern computer operating systems and engineering applications are based on architectures designed for single users: one active application and one active cursor" [14]. The efficiency and progress of engineering projects are hindered by these current single-user CAx architectures [3]. To address this problem a new paradigm has been developed which allows multiple users to work collaboratively on a shared file.

A multi-user program is one in which multiple users on different computers work together over a network using common data. The changes and updates from each user are pushed to all other users in near real time, thereby creating simultaneous collaboration on the shared data. A common multi-user program in use today is Google Docs, in which multiple users can create and edit a shared, cloud-based document. This tool enables near real time collaboration and has shown significant increases in productivity in some scenarios [15].

Extensive research has been performed at Brigham Young University regarding the feasibility of multi-user commercial CAD programs, and the potential benefits of these systems [3–6, 14]. A diagram illustrating the basic architecture behind a multi-user CAD program is shown in Figure 2.1. In the diagram the grey boxes represent individual computers (with one engineer using each computer), the white hexagon represents the CAD server, and the white ovals represent the CAD clients of a common CAD program. Model data is stored on a server and sent to the clients. Each client is an instance of the CAD software which receives updates from the server, and also sends its own updates to the server.

Researchers at Brigham Young University's Center for e-Design have created Multi-User CAD prototypes in NX, CATIA[1], and Inventor[2] [14]. This work has demonstrated significant decreases in modeling time and increases in efficiency. Work has also been done at BYU to overcome inherent difficulties that arise in multi-user systems including data consistency, conflict avoidance, the need for real-time communication, and synchronous vs. asynchronous modeling [3, 4, 16].

[1]CATIA is a trademark of Dassault Systems

[2]Inventor is a trademark of Autodesk Inc.

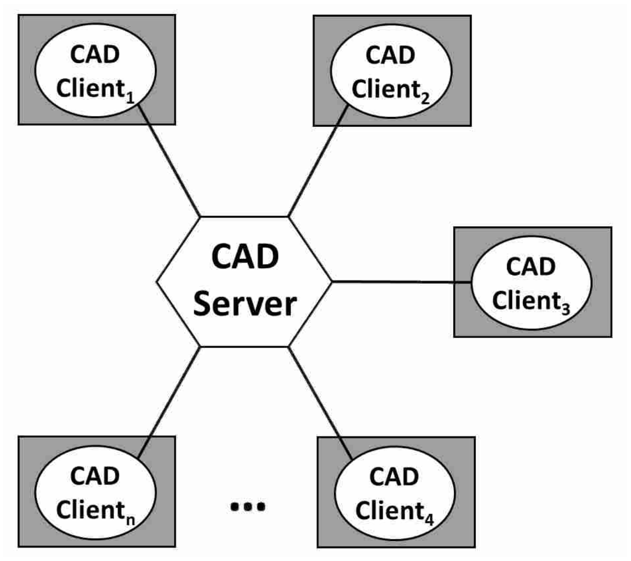


Figure 2.1: Multi-User architecture of a CAD program

Research involving multi-user Finite-Element pre- and post-processors has also been conducted by BYU and others [14]. Prasad Weerakoon et al. showed the advances in collaborative engineering that come from a multi-user FEA program [17, 18]. Their multi-user CUBIT[3] prototype also shed light on difficulties that arise in a multi-user FEA program, including consistency of geometric bodies, nodes, and elements between clients, and the time required for each client to execute time-intensive commands.

Researchers at BYU have also explored the feasibility of transforming the Advanced Simulation FEA tools within Siemens NX into a multi-user pre and post processor [5]. Jared Briggs et al. have shown the benefits of a thick-server, thin-client multi-user finite-element architecture, namely consistency of mesh data and numbering across the multiple clients and the elimination of repeated computations on each user's computer [6]. They tested this architecture by creating a new CAD/FEA program called MUFE (Multi-User Finite-Element), instead of using a commercially available FEA package.

This discussion of multi-user CAx programs is important to this thesis as the methods described in Chapter 3 are built upon a client-server multi-user CAD architecture.

[3]CUBIT is a Geometry and Mesh Generation Toolkit developed by Sandia National Laboratories

2.3 Integrated Design and Analysis

Though research has been performed on multi-user CAD programs as well as multi-user analysis programs, research in the area of analysis programs integrated with the CAD design programs is lacking. As the integration of CAD and engineering analysis programs is a vital element in the multi-user integrated design-analysis architecture proposed in Chapter 3, this chapter describes the current practices and state of the art of design-analysis integration.

CAD and Analysis programs were developed independently of each other, and at their core they represent geometric data in a fundamentally different way [10]. Because of these differences the transfer of geometric data between the CAD and Analysis programs usually takes place through neutral file formats. Geometric data transfer through neutral file formats often happens when the CAD and Analysis programs are on the same computer, as illustrated in Figure 2.2. The geometry data from the design program (CAD) is saved locally on the computer in a neutral file format, and then imported into the analysis program. The grey box represents the computer on which the CAx programs are open, the white shapes represent CAx programs, and the dotted line represents the neutral file which is used to transfer data between the different programs. This setup is often used by a single engineer who oversees the entire design and the entire analysis during a project.

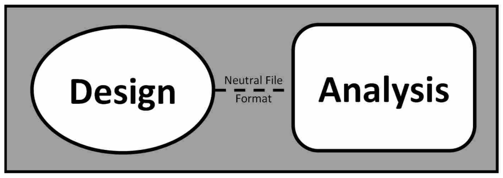


Figure 2.2: Neutral-File Format Geometry Transfer between CAD and Analysis on a single computer

Neutral file formats also allow the design and analysis to be distributed to multiple computers, with the file transfer occurring over a network, the internet, or even physical storage media. This concept is illustrated below in Figure 2.3. In this illustration the grey boxes represent individual computers (with one engineer using each computer) and the white circles represent the design and analysis programs. The dotted line represents the neutral file which is used to transfer data between the different computers and programs. This architecture allows for a single design to be sent to multiple analysts. The analysts are not required in the design portion, nor do they need access to the CAD software. The designers and analysts can work in geographically diverse locations, because the geometry data in neutral file format can be sent over a network or over the internet.

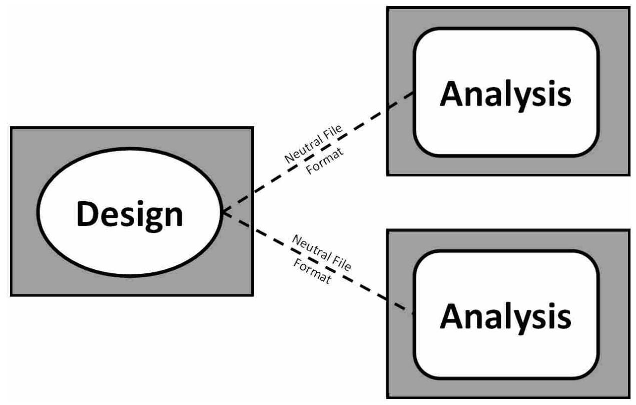


Figure 2.3: Neutral-File Format Geometry Transfer Transfer between different computers

The purpose of the integration of design and analysis programs is the improvement in collaborative engineering, and in the communication between people and programs. It is necessary for the designers and the analysts to "be on the same page," to be able to understand each other's needs and to communicate effectively. McGuire et al. indicated a necessity of vocabulary standards of "mutually agreed upon [engineering] terminology and definitions that are usable by people and their machines" [19]. Olsen et al. also points out the need for general paradigms that foster collaborative engineering [20].

The need for improvements in technology to aid in collaborative engineering has been known for decades, and areas of improvement have been identified and studied by various researchers across the globe [13, 21]. In the mid 1990's researchers at the Massachusetts Institute of Technology developed a multidisciplinary design architecture called DOME (Distributed Object-based Modeling and Evaluation). This architecture utilized the young World Wide Web to allow engineers, analysts, and manufacturers to share data and work together [22]. Since that time much research has been done to facilitate data transfer between different CAx programs, but today the communication still largely relies on neutral file formats (such as IGES and STEP files) and other archaic means. Y.-S. Ma et al. review ways in which geometric data sharing through neutral file formats is used. They reach the conclusion that "to support a comprehensive integration of applications, a more advanced data sharing mechanism is needed than those provided by the existing IGES or STEP standards... Theoretically, all CAx applications operate on their specific features mapped to a common set of data so that the product engineering and management is efficient with respect to changes. However, the available technologies have difficulties in maintaining globally consistent and comprehensive product models." Tony Abbey, an expert in the field of finite-element analysis supports this claim. He explains that the inability to seamlessly integrate CAD and FEA is due to several reasons, namely the required preparation and idealization of CAD geometry before meshing, the lack of CAD models' robustness caused by the hierarchical interdependence of features, and difference in configurations between the CAD and FEA geometry [10].

It is known that successful integration of design and analysis programs is difficult to employ on a general scale to many commercial CAx systems at once, yet there are examples of successful design-analysis layering implemented by specific programs [10]. An image which illustrates this local integration is shown in Figure 2.4.

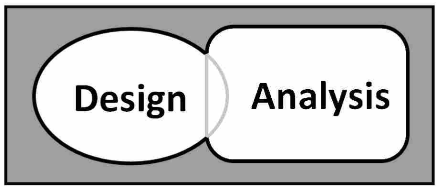


Figure 2.4: Integrated Design-Analysis Architecture on a Single Computer.

One example of an integrated design-analysis architecture is NX Advanced Simulation[4]. NX Advanced Simulation is a suite of analysis programs integrated into NX. It has the capability to perform finite-element analyses, kinematic studies, and other simulations. Its strength lies in the connection it has with the NX CAD geometry. The geometry is transferred from CAD to analysis through internal means, instead of neutral file formats, which allows for more retention of information. Changes in the CAD model can quickly and easily be updated in the analysis model.

Another example of successful integration of design and analysis programs is provided by ANSYS in their Workbench platform. Contained within Workbench is a suite of advanced engineering simulation technology and geometry tools. One such geometry tool is the Geometry

[4]NX Advanced Simulation are trademarks of Siemens Product Lifecycle Management Software, Inc.

Interface Tool[5] which “supports bidirectional, direct, associative interfaces with all major CAD systems” [23]. Geometry transferred from a CAD program to ANSYS Workbench through the Geometry Interface Tool retains key features and the parameters associated with those features. These parameters can be changed within Workbench, thus modifying the geometry in ANSYS. The parameter changes in ANSYS can also be pushed back to the CAD system, updating that geometry as well. This bidirectional interface is able to pass more model information from the CAD program to ANSYS than neutral file formats, and is also able to send information back to the CAD system.

However, these integration tools contained within ANSYS Workbench and NX Advanced Simulation are limited to a single computer, as evidenced in Figure 2.4 above. The CAD file whose geometry is transferred to ANSYS Workbench must be open on the same computer as ANSYS. In Chapter 3 the integrated design-analysis tool is combined with a multi-user CAD system, thereby overcoming this limitation present in the current state of the art.

2.4 The Design Loop

In engineering design there exists a common loop, or iterative process. In the most general sense it consists of three stages: conceptualize, build, and test/evaluate [24]. First, design requirements are established, then a concept is generated. Next a model or prototype is built[6]. Lastly, the model or prototype is tested and evaluated against the design requirements. If the requirements are not met then the concept and model is redesigned and subsequently re-evaluated. This loop is repeated until the requirements are satisfied. With the satisfactory design achieved the design moves to production. This iterative sequence is referred to in Chapter 1 and is illustrated below in Figure 2.5.

The time which this general loop takes can be described in a mathematical formula. The total time is equal to the sum of each step, with the conceptualize, build, and test/evaluate stages repeated a number of times. This design loop is evidenced inside the summation shown below in Equation 2.1.

[5]The Geometry Interface Tool is a trademark of ANSYS Inc.

[6]At times the term “build” implies the construction of a physical object while at other times implies the work to create a detailed design, such as a CAD model. For the purpose of this thesis the term “build” is used in the latter sense, that is, the construction of a CAD design.

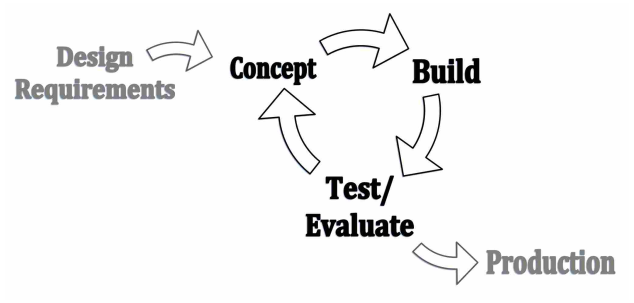


Figure 2.5: Design Loop

$$t_{\text{total}} = t_{\text{establish requirements}} + \sum (t_{\text{concept}} + t_{\text{build}} + t_{\text{test/evaluate}}) + t_{\text{production}} \tag{2.1}$$

This concept of the design loop is important to this thesis as it is the focus of improvement through the use of the proposed architectures in Chapter 3. The design loop will not be done away with, but instead simplified and sped up.

2.5 The Engineering Design Process

The engineering design process is often described as a series of steps or stages. Karl Ulrich and Steven Eppinger describe these stages as: [25]

- Planning
- Concept Development
- System-level Design
- Detail Design
- Testing and Refinement
- Production Ramp-Up

A visual illustration of this engineering design process can be seen below in Figure 2.6.

The general design loop described earlier and found in Figure 2.5 and Equation 2.1 is present within this engineering design process. Indeed, the Concept Development, System-level Design, Detail Design, and Testing and Refinement stages are often repeated multiple times in the

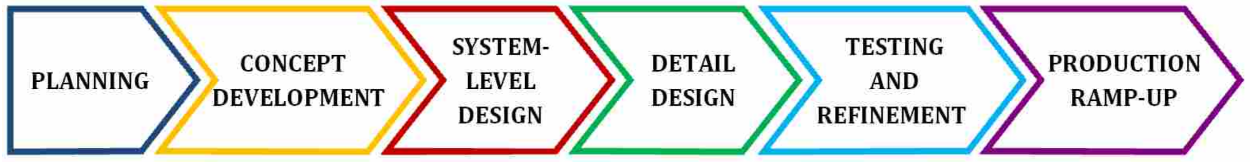


Figure 2.6: The Engineering Design Process according to Ulrich and Eppinger

design process, thereby comprising an alternate representation of the design loop, as illustrated in Figure 2.7.

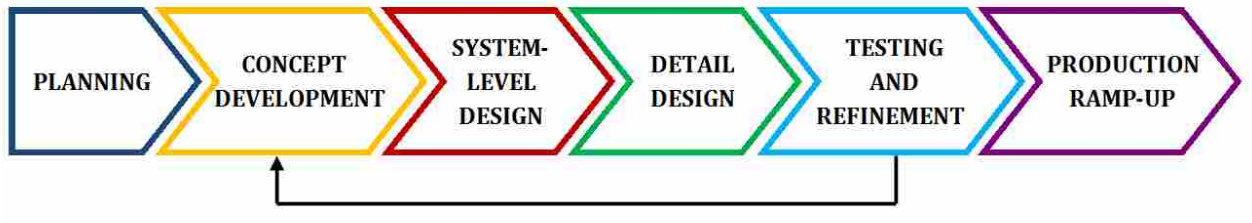


Figure 2.7: Design Loop evident within the Engineering Design Process

It has always been a goal in industry to reduce the total time of the engineering design process, thereby producing finished designs faster. This is often accomplished by reducing the time of the steps, or even the time of operations within steps. The total time required to complete the engineering design process can be described as the sum of the time required for each stage within the process. This is shown in Equation 2.2 below. Note that this equation is a mathematical representation of Figure 2.7, with the same design loop present within the summation.

$$\begin{split} t_{\text{total}} = t_{\text{planning}} + \sum \Big(t_{\text{concept development}} + t_{\text{system-level design}} \\ + t_{\text{detail design}} + t_{\text{testing and refinement}} \Big) + t_{\text{production ramp-up}} \end{split} \tag{2.2}$$

2.6 The Design-Analysis Loop

The iterative sequence of conceptualize, build, and test/evaluate not only occurs on the large scale discussed in the previous section, but also in numerous smaller instances within the stages of the engineering design process. In this sense there are smaller design loops nested inside of larger design loops. One which is the focus of this thesis is the Design-Analysis loop which occurs in the Detailed Design phase of the Engineering Design Process.

The time taken in the Detailed Design phase of the Engineering Design Process can thus be described with two design loops, one nested in side the other, as shown in the summations below in Equation 2.3. This phase begins with the establishment of requirements, and then enters a design loop in which a concept is generated, iterations of design and analysis are performed to refine the design, and finally a prototype is built and tested. If the test shows that the design does not meet requirements then an additional iteration of concept modification is performed, more design-analysis iterations, and finally additional testing.

$$t_{\text{detailed design}} = t_{\text{establish requirements}} + \sum(t_{\text{concept}} + \sum(t_{\text{Design-Analysis}}) + t_{\text{test/evaluate}}) \tag{2.3}$$

The Design-Analysis Loop, which resides within the Detailed Design phase is explained in the following example of the design of a wing rib on a commercial jet airliner. First, the general concept of the wing rib and its functional requirements are developed. Next, a design is built within a CAD program, defining its size, shape, and properties. The design is then subjected to preliminary testing and simulations including structural analyses, fatigue analyses, modal analyses, etc. The results of these analyses are evaluated and compared to the established requirements. If the design does not meet all the requirements, for example if the structural analysis predicts failure in the part, then the wing rib is must be redesigned and re-tested. This process is repeated until the satisfactory design is reached. Figure 2.8 shows this loop taking place for three iterations: an initial design and two re-design cycles. Often this cycle takes place many times before a satisfactory design is achieved.

Looking more closely one can see that there are many smaller steps within the design-analysis cycle. In 2005 Sandia National Laboratories conducted a survey of analysts to learn more about the time spent during this design-analysis cycle [26]. The analysts surveyed were from various fields including heat transfer, fluid flow, modal, radiation, and structural analysis. The survey divided the design-analysis cycle into ten steps:

1. Design Solid Model Creation and/or Edit

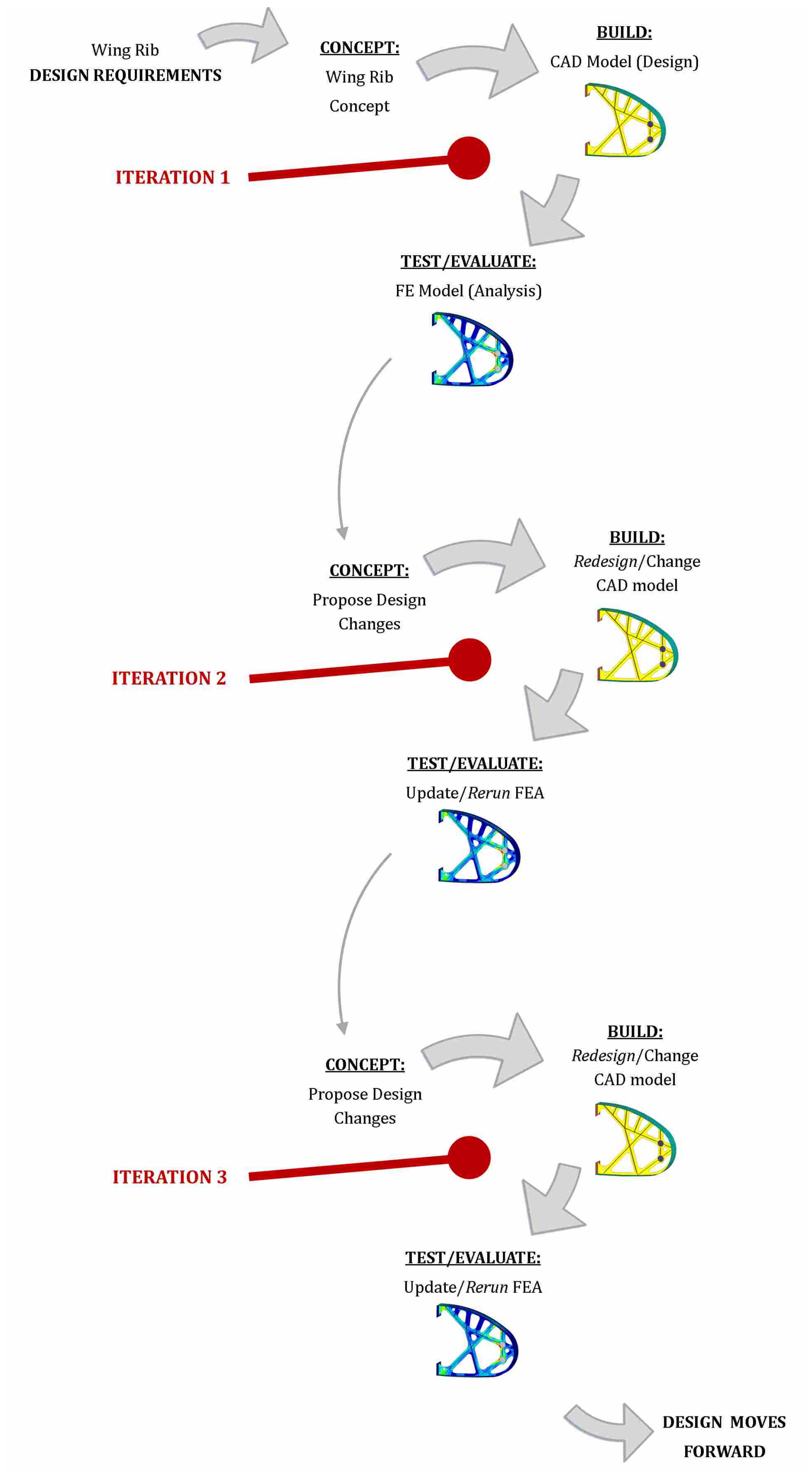


Figure 2.8: Three Iterations of Design-Analysis Loop using CAD and FEA

2. Analysis Solid Model Creation and/or Edit
3. Geometry Decomposition
4. Meshing
5. Mesh Manipulation
6. Assign Model Parameters
7. Assemble Simulation Model
8. Run Simulation
9. Post-process Results
10. Archive Artifacts

The percent of time of each step, when compared to the total time, is illustrated below in Figure 2.9.

The time taken by these ten steps of the design-analysis loop can be described through a mathematical equation, shown below in Equation 2.4.

$$\begin{aligned} t_{\text{Design-Analysis}} = {} & t_{\text{Design Model Creation}} + t_{\text{Analysis Model Creation}} + t_{\text{Geometry Decomposition}} + t_{\text{Meshing}} \\ & + t_{\text{Mesh Manipulation}} + t_{\text{Model Parameters}} + t_{\text{Assemble Model}} + t_{\text{Run Simulation}} \\ & + t_{\text{Post-Process}} + t_{\text{Archive Results}} \end{aligned} \tag{2.4}$$

From this study it can be observed which steps are the "bottlenecks" of the process at Sandia. For example, each of the pre-processing tasks (Analysis Solid Model Creation and/or Edit, Geometry Decomposition, Meshing, Mesh Manipulation, Assign Model Parameters, and Assemble Simulation Model) *individually* took more time to complete than actually solving the model (Run Simulation). Historically the solve times of analysis simulations were notoriously long, a stigma which still exists today. However the processing power of computers continues to grow at an exponential rate, following Moore’s law, which results in the reduction of solve times.

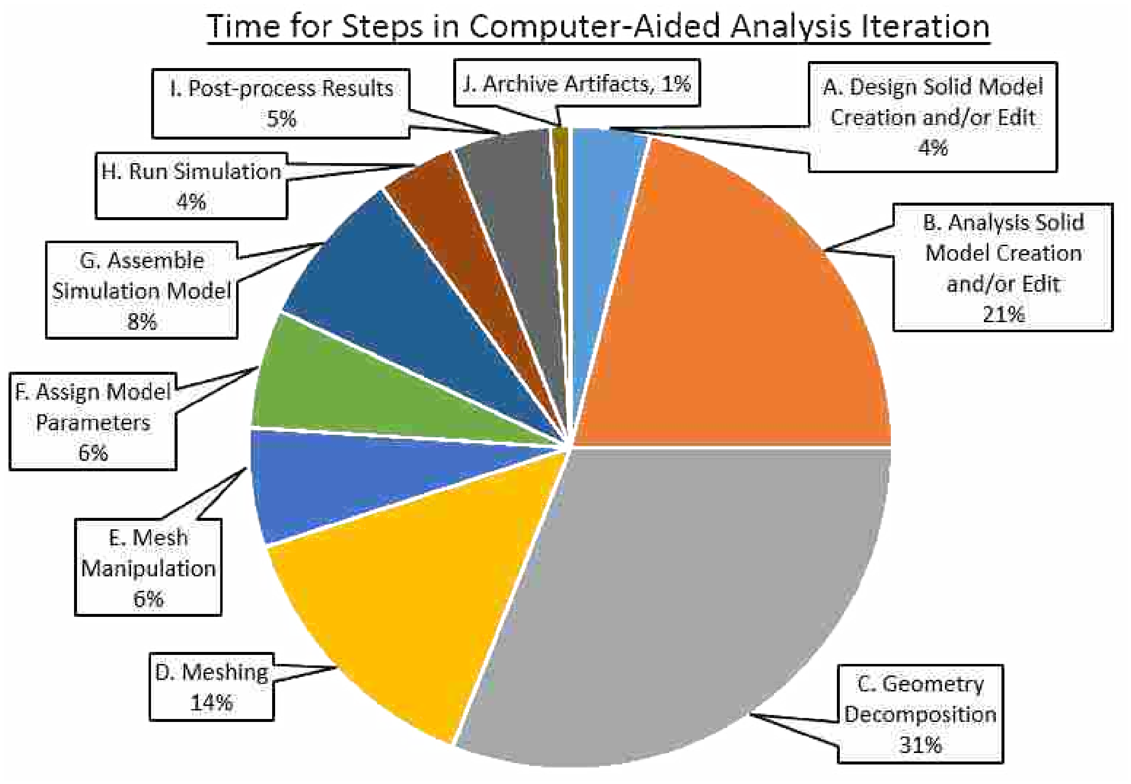


Figure 2.9: Sandia Analysis Time Survey Time Percentages [27]

It should be observed that the four pre-processing tasks of Analysis Solid Model Creation and/or Edit, Geometry Decomposition, Meshing, and Mesh Manipulation reportedly take up 73% of the total time [27]. As described in this section these steps are a necessary part of product design. However, the methods proposed in Chapter 3 seek to reduce the time of these steps and/or eliminate their repetition.

CHAPTER 3. METHOD

As described in Chapter 2 there are limitations in the current design-analysis architecture generally used in industry. Historically, data between design and analysis programs have been transferred through neutral file formats, including IGES and STEP files, and direct readers, which transfer geometric information directly from one application to another. Often this geometry transfer takes place on a single computer, as illustrated previously in Figure 2.2, or between different computers, as illustrated previously in Figure 2.3.

Chapter 2 also describes the current state of the art of design-analysis integrated systems. These integrated systems are limited to a single computer, as illustrated previously in Figure 2.4. This type of commercially available integrated system is well suited for small projects in which all aspects of design and analysis can be performed by one single engineer. However, it becomes impractical for large projects involving multiple disciplines. How can the model data be shared between multiple disciplines if it must remain on one single computer?

This chapter discusses methods to achieve concurrent engineering through parallelization of the design-analysis process. Two multi-user integrated design-analysis architectures are proposed.

The main principle underlying both of the following proposed methods is a centralized database accessed and modified by each discipline in the engineering design process, whether that be CAD design, some form of analysis, CAM, etc. The architectures proposed in the methods differ from today's Product Lifecycle Management (PLM) programs in that the geometric data stored in these architectures receive and send data to and from each user in near-real-time. The different users do not "check out" and "check in" parts as they do in a PLM system, but instead work together in a shared, near-real-time modeling session.

3.1 Method 1 - Multi-User, Integrated Design-Analysis Architecture 1

The first proposed method is built upon the multi-user architecture described in Section 2.2. The geometric data is shared between users and disciplines through a multi-user CAD server. The CAD designers work together to build the model while the other disciplines, whether that be engineering analysts, manufacturing personnel, or another discipline, are connected to the server through a local client of the CAD program. Each individual instance of auxiliary software (analysis, CAM, etc.) is linked to the local CAD client where it receives and sends geometric updates. This proposed architecture is shown below in Figure 3.1.

Though the CAD designers operate in a multi-user environment and the auxiliary functions (analysts, manufacturing personnel, etc.) are connected to the multi-user CAD session, those auxiliary functions are themselves single-user. Each session of analysis, CAM, etc. takes place on one user's computer. In other words, in this first proposed architecture the ability to have multiple analysts participate in the same FEA preprocessing session, or multiple manufacturing personnel to participate together in the same CAM session, is not present.

In the diagram the white hexagon represents the CAD server, the grey boxes represent individual computers (with one user using each computer), the white ovals represent the CAD clients, and the white rectangles represent the other software, including analysis programs and CAM software. Model data is stored on a server and is sent to and received from the CAD clients. In the diagram the grey lines between the CAD clients and the auxiliary programs represent the integrated connection which exists between the CAD and auxiliary programs on each user's computer. This integrated connection allows the sharing of geometric data through means discussed in Section 2.3.

The construction of this multi-user, integrated design-analysis architecture requires two major components: a client-server, multi-user CAD architecture, and CAD-integrated auxiliary programs such as FEA and CAM.

3.1.1 Multi-User CAD Architecture

In order to share geometric data among the different users a multi-user CAD architecture must be set up similar to the architecture described in Section 2.2. Specifically, a thick-client, thin-server system is well suited for the multi-user CAD architecture. Such a system should employ a

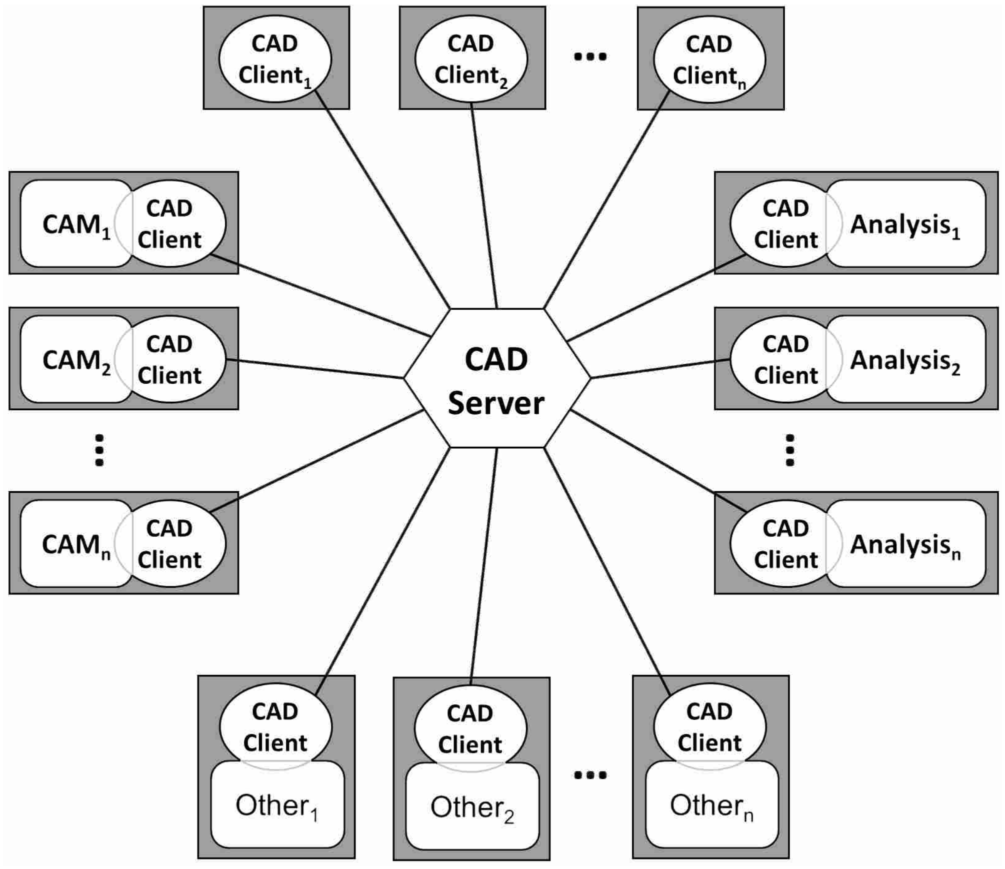


Figure 3.1: Method 1: Multi-User Integrated Design-Analysis Architecture built upon a multi-user CAD architecture

database to store the operations performed on the CAD model, with tracking of parts, features, and attributes of operations, as well as a numbering method to keep track of the order of operations performed.

The architecture must also use a server to handle the commands received from clients, to correctly store the commands in the database, and to push the commands out to the various clients. The connections to the clients must take place over a secure TCP network connection, as to allow only the appropriate clients access to the multi-user server.

The multi-user architecture must also employ clients on each user's computer. These clients are plugins for a specific CAD program which communicate with the server over the secure network connection. The client observes operations in the CAD program made by the user and relays the commands of the operation up to the server. The client also handles the job of receiving commands from the CAD server and executing those commands in the client's CAD system through the API.

More details regarding the construction of a multi-user CAD architecture can be found in the research cited in Section 2.2.

3.1.2 Client-Integrated Auxiliary Programs

The multi-user, integrated design-analysis architecture proposed in this method requires auxiliary programs (FEA, CFD, Thermal Analysis, Modal Analysis, CAM, etc.) which can interface directly with a CAD program in the manner described in Section 2.3 and shown visually in Figure 2.4.

The auxiliary programs used in this architecture must employ a form of geometry interoperability sufficient to transfer geometry from the CAD system to the auxiliary program while maintaining attributes such as geometric features, parameters, and expressions. This transferred geometry must retain a connection back to the CAD program such that when updated and/or modified versions of the CAD geometry are transferred to the auxiliary program it must implement the geometric modifications and/or additions to its geometry without simply "starting from scratch" (as is done with neutral-file format geometry transfer).

There are multiple methods by which this robust geometry transfer and integration can take place. One of which is the recording of the operations which have taken place in the CAD program and the duplication of these operations in the auxiliary program. Thus, as a mirror copy of the CAD geometry, any changes or additions to the CAD model can be easily replicated on the auxiliary geometry, and likewise modifications to the auxiliary geometry, such as parameter changes, may be sent back to the CAD program to modify the CAD model accordingly. However, it should be noted that creating mirror copies by executing the same specific commands on both CAD and auxiliary programs may only work if the kernels and/or internal algorithms are similar; otherwise, differing geometries may result.

3.2 Method 2 - Multi-User, Integrated Design-Analysis Architecture 2

The second proposed method differs from the first in that the auxiliary applications (analysis, CAM, etc.) are not connected to the architecture through local clients of the multi-user CAD session but instead link directly to the server. In this proposed architecture the auxiliary applications send and receive geometric data directly to and from the server and the server's database. The connection takes into account any conversions necessary to interface the CAD system and other software's geometry kernels. This proposed architecture is shown below in Figure 3.2.

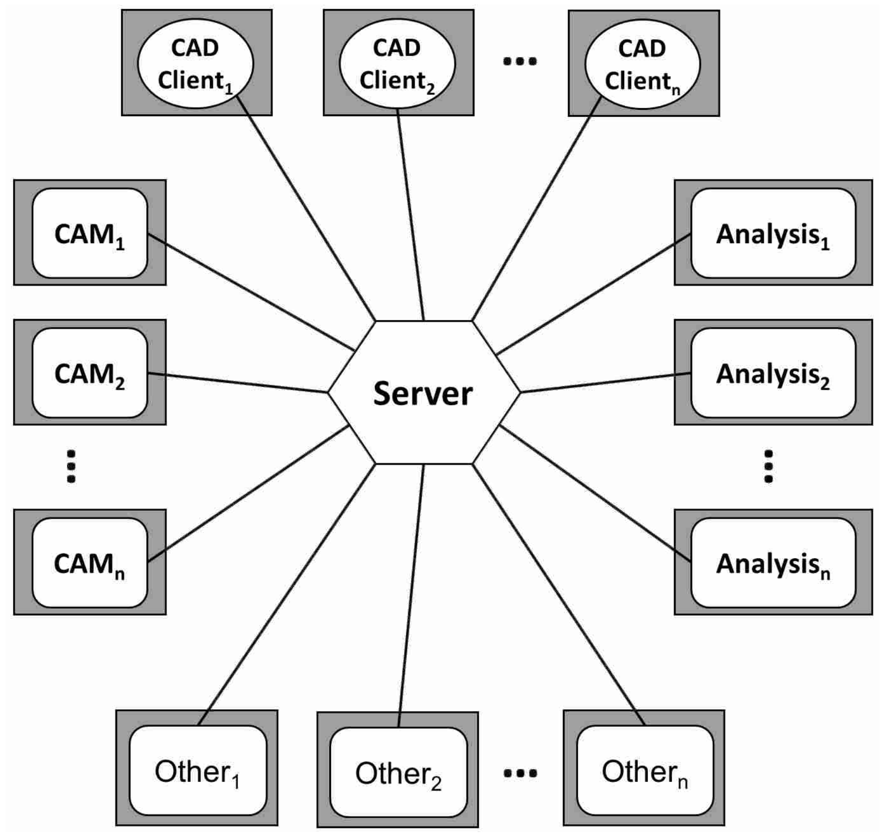


Figure 3.2: Method 2: Multi-User Integrated Design-Analysis Architecture with auxiliary software linked directly to the server

In the diagram the white hexagon represents the CAD server, the grey boxes represent individual computers (with one user utilizing each computer), the white ovals represent the CAD clients, and the white rectangles represent the other software, including analysis programs and

CAM software. Model data is stored on a server and is sent to and received from the CAD clients and the other programs.

Like the architecture proposed in Method 1 the CAD designers in this second proposed architecture operate in a multi-user CAD session similar to the one described in Section 3.1.1, except the server must be set up in a CAD-neutral format so that the auxiliary applications may use the server's data to construct geometry in each application's form of geometry creation. The integrated auxiliary applications described in Section 3.1.2 are not applicable to this architecture. Instead, auxiliary applications which interface directly with the multi-user server are required.

3.2.1 Server-Integrated Auxiliary Applications

In this second architecture the auxiliary applications interface directly with the server, instead of with the clients (as done in the first method). Like those described in Section 3.1.2, the auxiliary applications used in this second architecture must employ a form of geometry interoperability such that they may interface with the CAD database. The auxiliary applications must be able to push and pull geometric data to and from the server.

The means by which data is pushed back up to the servers from these auxiliary clients would take place in a manner similar to the method the CAD clients of existing multi-user CAD architectures push data up to the server. In this proposed architecture the server must handle data consistency and conflict avoidance that may arise from the auxiliary clients.

Like the connections which the CAD clients have with the server, the auxiliary applications' connections to the server must take place over a secure TCP network connection, as to allow only the appropriate clients access to the multi-user server.

Because the auxiliary programs in this architecture interface directly with the server instead of with individual CAD clients they are not limited to single-user sessions. Therefore the server in this proposed architecture could support multi-user capabilities for the auxiliary functions as well as for the CAD session. For example, the server could host one or more sessions of structural analysis, with multiple engineers participating in each session. These engineering analysts would work together to preprocess a common FEA model, in a manner similar to the multi-user FEA described in Section 2.2.

CHAPTER 4. IMPLEMENTATION

This chapter discusses the implementation of the proposed multi-user, integrated CAD-analysis architecture described as Method 1 of Section 3.1. The requirements for CAD and analysis software to construct a working prototype are discussed. The construction of the prototype is also explained, including the software selected and the modifications made to the CAD and analysis applications.

4.1 Requirements

This section outlines the requirements to construct a prototype of the multi-user integrated CAD-analysis architecture described in Method 1 of Section 3.1. Section 4.1.1 explains the requirements of the multi-user CAD system, and Section 4.1.2 reviews the requirements of the analysis application that will be used in the prototype.

4.1.1 CAD Requirements

To create a multi-user, integrated design-analysis prototype a multi-user CAD program is required. For this thesis basic CAD functionality and commands were required, such as sketches, extrudes, revolves, dimensioning, parameters, expressions, instancing, fillets, unions, and subtractions. More complex functionality such as splines, projections, surface modeling, composite properties, etc. were not required. The multi-user CAD program used in the integrated design-analysis prototype must allow multiple users to work on a shared model in near-real time. For the integrated multi-user prototype each user must have his or her own instance of the CAD program running on his or her computer, an attribute most commonly found in thin-server, thick-client, multi-user architectures. A lightweight or browser-based CAD client would only suffice if the analysis program was able to integrate with such a CAD client, a scenario which does not exist today.

The multi-user CAD system required for the integrated prototype must have an Application Programming Interface (API) exposed which grants access to features and parameters within the model. Multi-user applications which plug into commercial CAD programs, making them multi-user, have done so through the API of the CAD programs. This requirement of an exposed API is also necessary in order to add the additional functionality between the analysis program and the multi-user CAD program. This additional integration is needed because the integrated analysis programs are capable of interfacing with a single instance of the CAD program, but are not designed to integrate with the multi-user version of the CAD program. The exposed API allows the necessary changes to be made for integration with the multi-user CAD program.

4.1.2 Analysis Program Requirements

A suitable analysis program is required to construct the multi-user integrated design-analysis prototype. The most important requirement of the analysis program is that it interfaces and integrates well with the chosen CAD program. As explained in Sections 2.1 and 2.3 the realms of CAD and Analysis have been and generally still are disconnected and separated in regard to geometry. However, there are a few commercially available analysis programs which are relatively more integrated with single-user CAD programs. There is a certain level of integration required for the proposed multi-user, design-analysis integrated prototype. The analysis program used must not only be able to take geometry directly from the CAD system, as neutral file format transfer does, but it must also bring data related to the features, parameters, and construction of the CAD model. With this geometry and data also present in the analysis program the model can be modified and edited inside of the analysis program. It must also have the ability to push these parameter/geometry changes back to the CAD program so that the master CAD model is updated to match the analysis model.

The analysis program selected for use in the multi-user integrated design-analysis prototype must have the capability to edit the geometry in order to idealize, simplify, and/or modify it in preparation for meshing and analysis. These forms of idealization and geometry modification are different than those described in the previous paragraph; these modifications are only intended to exist in the analysis geometry, and not be pushed back up to the CAD geometry. Often the CAD version of the geometry is more complex than is needed for an certain analysis. At times there exist details in a CAD model which do not have a significant effect on the analysis results. For example, some fillets in non-critical areas of stress may have little-to-no effect on the stress flow patterns in the model. However, such details can cause the analysis model to be more complex than necessary. For example, the finer mesh required to represent extraneous details causes an increase in the analysis solve time. For this reason the analysis program used in the multi-user integrated design-analysis prototype will require this capability to simplify, idealize, and/or modify features of the geometry it receives from the CAD program. A visual example of an idealization of a CAD model is found in Figure 4.1. It should be noted though that not all details within a CAD model should be removed for analysis, but only those judged to be non-critical by the analysts.

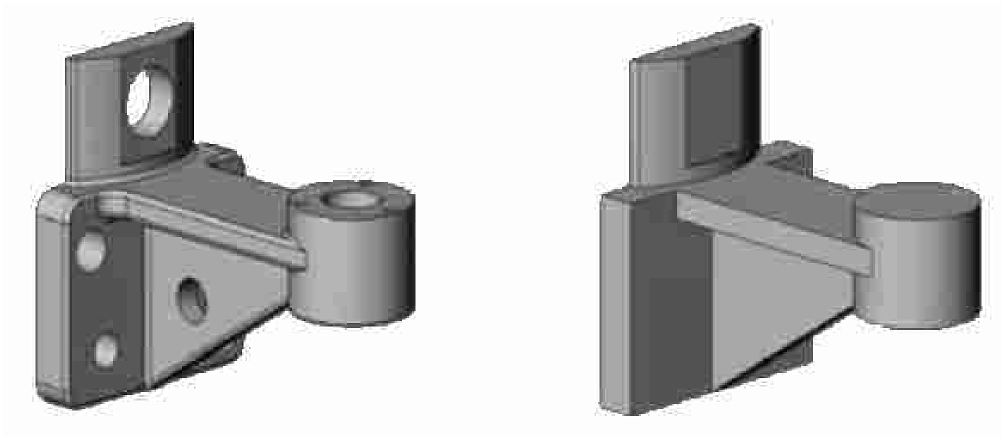


Figure 4.1: The figure on the left shows a fully detailed CAD model, while the figure on the right shows an idealized copy of the original model with some features removed [28]

Though the analysis application must allow geometry idealization and modification it must also have sufficient integration with the CAD program to still allow updates to the CAD model to be reflected in the analysis model, even if geometry modification, simplification, or idealization have taken place. For example, if geometry or parameter changes take place in the CAD model the analysis program must be capable of applying the updates to its copy of the geometry and then reapplying any previously performed idealization or geometry modification on this updated geometry.

4.2 Prototype Construction

A prototype of the multi-user integrated CAD-analysis architecture was constructed in order to test the architecture's feasibility and to test the advantages claimed by the architecture. The selection of NX Connect, the multi-user version of the SIEMENS NX CAD program, and ANSYS Workbench, with its structural analysis, fluid flow analysis, DesignModeler[1] geometry editing application, and Geometry Interface Tool, are explained in Sections 4.2.1 and 4.2.2. The integration of NX Connect and ANSYS Workbench and the modifications made to facilitate parameter exchange between these two programs are explained in Section 4.2.3.

4.2.1 NX Connect

The NX Connect multi-user CAD prototype developed at Brigham Young University's Center for e-Design fulfills the requirements for a multi-user CAD program outlined in Section 4.1.1. As briefly described in Section 2.2, NX Connect is able to perform all the basic functions required for the test performed for this thesis, such as sketches, extrudes, revolves, parameters, instancing, boolean operation, etc. NX Connect is also capable of performing more complex operations beyond those required for this test.

NX Connect was created through the API of Siemens NX. The NX Connect program is modifiable to suit the needs of this multi-user integrated design-analysis architecture. It is a thick-server, thin-client architecture, with full instances of NX used by each user. Because individual instances of NX are operated by each user they each can easily attach an integrated analysis program to their CAD session.

4.2.2 ANSYS Workbench

ANSYS Workbench was selected for use in the multi-user integrated design-analysis prototype. As described in Section 2.3 ANSYS Workbench has the ability to perform detailed structural analysis, fluid-flow analysis, and other forms of engineering analyses. It is very well known and commonly used in the engineering analysis industry. ANSYS Workbench "supports bidirectional, direct, associative interfaces with all major CAD systems," through its Geometry Interface Tool,

[1]DesignModeler is a trademark of ANSYS Inc.

described below in Section 4.2.2.1 [23]. Workbench also has its own geometry editing tool, named DesignModeler, which is described in more detail below in Section 4.2.2.2.

4.2.2.1 ANSYS Geometry Interface Tool

The Geometry Interface Tool is a plug-in developed by ANSYS to facilitate the transfer of geometry and other data between a CAD program and ANSYS Workbench. It is the key application that allows ANSYS Workbench to integrate with a CAD program in a manner described in Section 2.3. As briefly described in Section 2.3, the Geometry Interface Tool is available for several commercial CAD programs including Siemens NX. Within NX the Geometry Interface Tool creates a new menu for use in transferring geometry from NX to ANSYS. This menu is shown below in Figure 4.2.

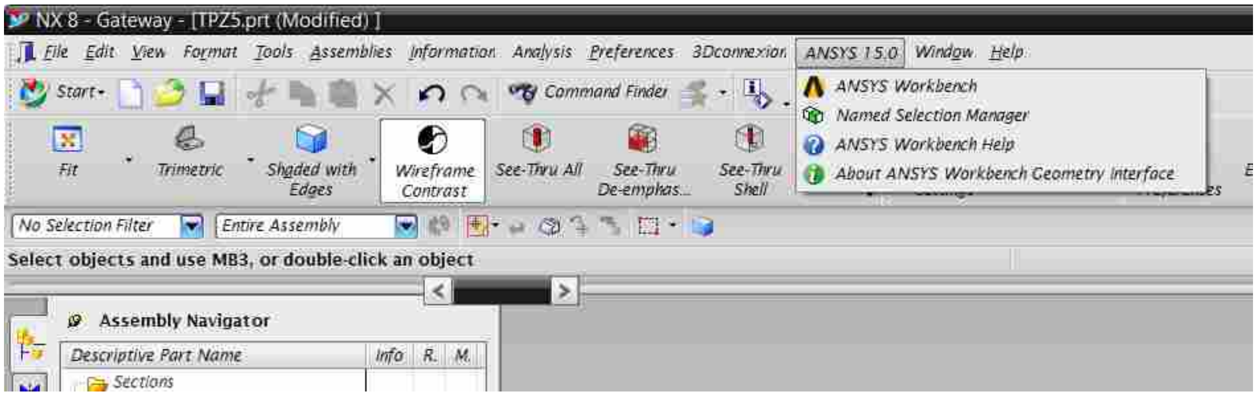

NX 8 - Gateway - [TPZ5.prt (Modified)]
File Edit View Format Tools Assemblies Information Analysis Preferences 3Dconnexion ANSYS 15.0 Window Help
Start Command Finder
ANSYS Workbench
Named Selection Manager
ANSYS Workbench Help
About ANSYS Workbench Geometry Interface
Fit Trimetric Shaded with Edges Wireframe Contrast See-Thru All See-Thru De-emphas... See-Thru Shell
No Selection Filter
Entire Assembly
Select objects and use MB3, or double-click an object
Assembly Navigator
Descriptive Part Name Info R. M.
Sections

Figure 4.2: The ANSYS 15.0 menu present in Siemens NX 8 installed by the Geometry Interface Tool.

The Geometry Interface Tool creates a copy of the NX geometry within Workbench complete with features, parameters, and named selections. The Geometry Interface Tool can be configured to allow access to all or just some model parameters in Workbench. While both NX and Workbench remain open the Geometry Interface Tool maintains a connection between the two, allowing for geometry updates and changes to occur between NX and Workbench. The CAD model in NX can be modified or even expanded with new features, and those changes can be sent to Workbench through the Geometry Interface Tool. Similarly model parameters can be modified within Workbench, and these changes can be sent back to NX through the Geometry Interface

Tool. These updates do not occur automatically but must be manually pushed from one program to the other. This ability to send new geometry, modifications, and parameter changes from NX to Workbench and to send parameter changes back from Workbench to NX fulfills the requirements outlined previously in Section 4.1.2.

The Geometry Interface Tool plugin has no API exposed and is therefore not modifiable. It is designed to work with a single session of NX, and interacts with the geometry present in that NX session. It cannot be modified to interact with geometry from other sources, such as the NX Connect server.

4.2.2.2 DesignModeler

DesignModeler is a program within the Workbench suite which handles all geometries for analysis in the ANSYS programs. It is a feature-based CAD program built on the Parasolid[2] kernel which has the ability to both create and edit geometry. DesignModeler is able to simplify and idealize the analysis geometry, as stated previously in the analysis system requirements. This capability allows a user to manually remove, add, or edit features in preparation for analysis. DesignModeler can also be tasked to automate some of this idealization and simplification through "repair" functions. For example, one repair function within DesignModeler can fill in all cylindrical holes or cavities which are smaller than a specified size, mending geometry afterwards as if there were no hole originally created. These types of changes and modifications are only applied to the analysis geometry, that is, the model existing in DesignModeler and the "downstream" analysis programs within Workbench. These changes are not passed back to NX through the Geometry Interface Tool. Similarly when geometric changes or updates are passed to Workbench and DesignModeler from NX the idealization and simplification previously performed are not eliminated but instead applied to the new/updated geometry. This feature greatly facilitates the transfer of geometry and data between NX and Workbench as it eliminates the need to re-do the idealization modifications performed on earlier iterations of the geometry.

[2]Parasolid is a trademark of Siemens PLM Software, Inc.

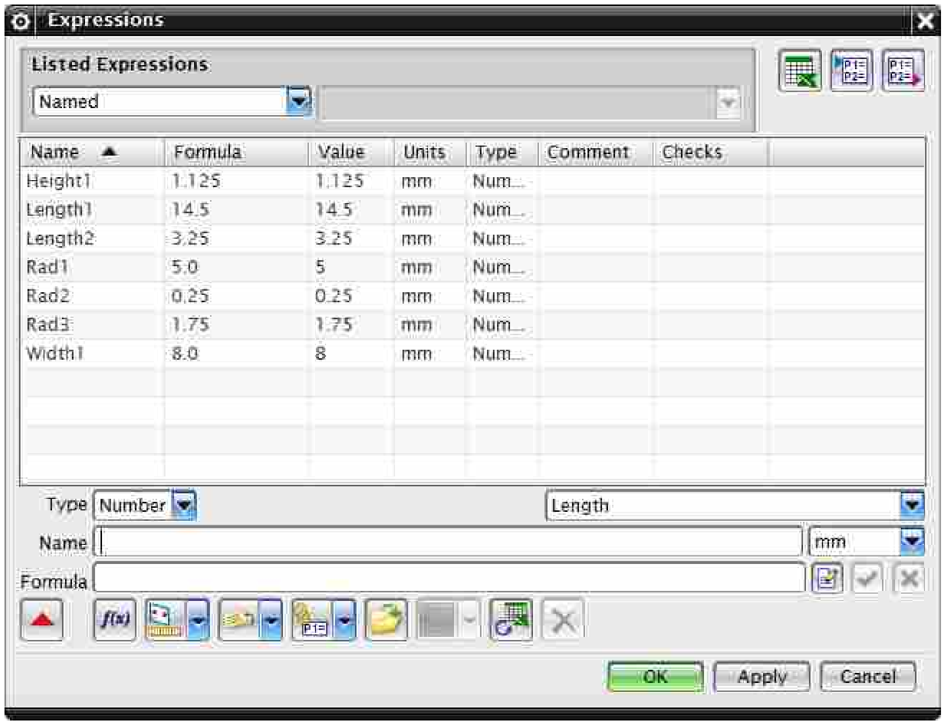
4.2.3 Parameter Exchange

As explained previously, the construction of the prototype multi-user integrated CAD-analysis architecture involved the combining of NX Connect and ANSYS Workbench. In this architecture the collaboration between users occurs through the shared NX Connect model, with each user working in his or her own session of NX. Each analyst then begins an independent session of ANSYS Workbench and attaches it to his or her session of NX through the ANSYS Geometry Interface Tool. As the analyst's session of NX is updated by other users through the NX Connect server the updated geometry can be passed to his or her attached session of ANSYS Workbench.

The ability which the ANSYS Geometry Interface Tools has to pass parameter changes from ANSYS back to NX did not function properly in the NX Connect architecture. The parameter changes sent by ANSYS were implemented in the user's local session of NX, but were not sent up to the NX Connect server and implemented on the other sessions of NX Connect. This created an inconsistency between the NX Connect server and the user's local session of NX. This inconsistency could only be corrected by refreshing the client's model with data from the server, thereby losing the parameter changes previously sent by ANSYS. This problem occurred because of the way NX Connect handled parameter changes. This problem and the remedy are discussed in detail in the following sections.

4.2.3.1 Parameter Limitations Within NX Connect

The NX Connect program assumed that all parameter changes in a model would be made manually by a human user. Therefore it was set up to only detect parameter changes that occur through the Expressions window. An image of the Expressions window in NX is shown below in Figure 4.3. NX Connect contained a menu capture module which modified the API functions performed by NX when a user clicked the menu item or used the shortcut key stroke to open the Expressions window. In addition to opening the Expressions window NX Connect also recorded a copy of all the expressions and their corresponding values at that moment. NX Connect also modified the API functions performed by NX when a user closed the Expressions window. When the Expressions window was closed NX Connect recorded a copy of all the expressions and their corresponding values at that moment. Then NX Connect compared the two lists of expressions and values to detect what was created, changed, and/or deleted by the user. The determined parameter changes were pushed up to the NX Connect server, which in turn sent them on to the other NX Connect clients. The NX Connect client on each of these users' computers would apply the parameter changes through the NX API.



Name	Formula	Value	Units	Type	Comment	Checks
Height1	1.125	1.125	mm	Num...		
Length1	14.5	14.5	mm	Num...		
Length2	3.25	3.25	mm	Num...		
Rad1	5.0	5	mm	Num...		
Rad2	0.25	0.25	mm	Num...		
Rad3	1.75	1.75	mm	Num...		
Width1	8.0	8	mm	Num...		

Figure 4.3: The Expressions window in NX allows a user to create, modify, and delete expressions and parameters.

Because NX Connect assumed that all parameter changes in a model would be made manually by a human user it ignored any changes made through the API. This is necessary because all parameter changes pushed to other clients are executed through the API on each client's computer. Therefore if NX Connect tried to detect API parameter changes and push them to the server it would detect the changes that it makes. By detecting these changes and pushing them up to the server a never-ending loop would ensue on each client's computer every time a parameter was created or modified. For this reason NX Connect assumes that all parameter changes made through the API were done by NX Connect, and so it simply ignores all API parameter changes.

When parameter changes to a model occur in ANSYS the Geometry Interface tool makes those changes in NX through the NX API. This is the reason the NX Connect program does not detect nor pass on these changes from ANSYS. The following section describes the modifications made to NX Connect in order to cause the parameter changes made by ANSYS to be detected and passed on to the NX Connect server.

4.2.3.2 Additions to NX Connect

Additions were made to the NX Connect code in order to detect parameter changes through the API originating from the ANSYS Geometry Interface Tool while ignoring those originating from NX Connect[3]. To accomplish this a User Defined Object (UDO) was created within the NX Connect code which kept track of all the expressions. The code defining the UDO within NX Connect is shown below in Code 4.1.

Code 4.1: C# code establishing the User Defined Object used to track the model's expressions

```
private void startTracking(NXOpen.Expression expr)
        {

            string UDClassName = PartInformation.UDClassName;
            try
            {
                ExpressionTrackingClass =
                        NXConnectUtils.Session.UserDefinedClassManager.GetUserDefinedClassFromClassName(
                        UDClassName);
            }
            catch
            { }
            if (ExpressionTrackingClass == null)
            {
                ExpressionTrackingClass =
                    NXConnectUtils.Session.UserDefinedClassManager.CreateUserDefinedObjectClass(UDClassName,
                    UDClassName);
                ExpressionTrackingClass.AllowQueryClassFromName =
                    NXOpen.UserDefinedObjects.UserDefinedClass.AllowQueryClass.On;
                ExpressionTrackingClass.AddUpdateHandler(update);
            }

            ExpressionTrackingObject =
                NXConnectApp.Parts.WorkPart.UserDefinedObjectManager.CreateUserDefinedObject(
```

[3]Mark Trent, an undergraduate mechanical engineering student in BYU's Center for e-Design, was of great assistance in creating the code necessary to make this change to NX Connect

```
            ExpressionTrackingClass);
        var scalar = NXConnectApp.Parts.WorkPart.Scalars.CreateScalarExpression(expr,
            NXOpen.Scalar.DimensionalityType.None, NXOpen.SmartObject.UpdateOption.WithinModeling);
        List<NXOpen.UserDefinedObjects.UserDefinedObject.LinkDefinition> links = new
            List<NXOpen.UserDefinedObjects.UserDefinedObject.LinkDefinition>();
        links.Add(new NXOpen.UserDefinedObjects.UserDefinedObject.LinkDefinition(scalar,
            NXOpen.UserDefinedObjects.UserDefinedObject.LinkStatus.UpToDate));
        ExpressionTrackingObject.SetLinks(NXOpen.UserDefinedObjects.UserDefinedObject.LinkType.Type3,
            links.ToArray());

        List<string> expressionName = new List<string>();
        expressionName.Add(expr.GetUniqueName());
        ExpressionTrackingObject.SetStrings(expressionName.ToArray());

        List<int> partID = new List<int>();
        partID.Add(Collection.Part.PartID);
        ExpressionTrackingObject.SetIntegers(partID.ToArray());
    }
```

Whenever an expression changes within the user's session of NX Connect the UDO checks two conditions. First, it checks whether NX Connect is performing an operation. This could occur when the parameter change came from the user through the expressions window, in which case NX Connect would be working to push that expression change up to the server. This could also occur when the NX Connect client has received a parameter change from the server and is working to apply that change in the local session. In either of these cases by detecting that the NX Connect client is performing an operation the UDO understands that the parameter change is already being handled by NX Connect, and so it does not try to push the change up to the server.

The second condition the UDO checks is whether the user is in an idle state within NX, the state that is ready to push parameter changes up to the server. For example, if the user is in a sketch or in a feature dialog then NX is not in the idle state, and thus not ready to send nor receive updates to or from the NX Connect server. In this case the NX Connect client must wait to send or apply any changes until the local session returns to the idle state.

If these two conditions are met, that NX Connect is not currently applying changes and that the user is in a state that is ready to push the expression changes, then the UDO knows that the parameter changes came from ANSYS and thus pushes those changes to the NX Connect server. This code which was added to NX Connect is shown below in Code 4.2.

Code 4.2: C# code which detects parameter changes sent from ANSYS and pushes those changes on to the NX Connect server

```
public static int update(NXOpen.UserDefinedObjects.UserDefinedLinkEvent eventObject)
        {

            if (NXConnectApp.Parts.WorkPartInfo.ConsistencyManager.applyingQueue == false
                && NXConnectController.CurrentStateIsIdleState())
            {

                // push the expression
                BasePartInfo part = NXConnectApp.Parts[eventObject.UserDefinedObject.GetIntegers()[0]];
                ExpressionInfo ourExpressionInfo =
                     part.Expressions[eventObject.UserDefinedObject.GetStrings()[0]];
                Data.Expression oldData = ourExpressionInfo.data;
                Pushers.ExpressionPusher pusher = new Pushers.ExpressionPusher(ourExpressionInfo);

                part.ConsistencyManager.unverifiedItemList.RemoveAll(x => x.name ==
                     ourExpressionInfo.GetUniqueName());
                Undo.ExpressionChange expressionChange = new
                     Undo.ExpressionChange(NXConnectApp.Parts.WorkPartInfo.Expressions, oldData,
                     pusher.ObjectToPush);
                ConsistencyManagers.ConsistencyManager.unverifiedItem unverifiedItemToAdd =
                            new ConsistencyManagers.ConsistencyManager.unverifiedItem(
                                   NXConnectApp.Parts.WorkPartID,
                                   ConsistencyManagers.ConsistencyManager.unverifiedItem.OperationType.isEdit,
                                   ourExpressionInfo.GetUniqueName(),
                                   expressionChange);

                part.ConsistencyManager.unverifiedItemList.Add(unverifiedItemToAdd);
                pusher.SubmitToServer();
            }

            return 0;
        }
```

With this addition to the NX Connect code the parameter changes sent to NX through the ANSYS Geometry Interface Tool were detected by the NX Connect client, sent to the server, and ultimately distributed to the other clients. This new addition enables analysts to update the shared NX CAD model from their linked ANSYS session.

CHAPTER 5. RESULTS

Two scenarios were developed to test the proposed multi-user integrated CAD-analysis architecture presented as Method 1 in Section 3.1. The performance of this new architecture is compared against current industry practices. These two test scenarios are described below.

5.1 Test Scenario 1 - Jet Engine Front Frame

In the first test the principle investigation is concerned with the time taken to perform multiple iterations of the design-analysis cycle, as described in Section 2.6. The total time taken for each iteration of analysis, not the steps that make up each iteration of analysis, was the focus of this test.

The test scenario incorporated three users, one as a CAD designer and the other two as analysts. One analyst performed structural analysis while the other performed fluid flow analysis. The two differing analysis disciplines were representative of many disciplines which can be present in the design of a mechanical component, including structural, fluid flow, thermal, and modal, etc.

5.1.1 Test 1 Procedure

The test began with all three users present in a NX Connect session with a CAD model already completed. In the test there was no time overlap between the designers creating the CAD model and the analysts performing analysis on the model. The model used in the test was a generalized version of a front frame of a jet turbine engine, as shown below in Figures 5.1 and 5.2.

In the test scenario three iterations of the Design-Analysis Loop occur. Each iteration follows the steps described in Equation 2.4. As described previously the first step of building the CAD model (Design Model Creation) has already taken place before the start of the test. In the scenario the key parameters which the analysts were focusing on were the thickness of the cylindrical sheet of material and the width of the vanes. The width of the vanes was a parameter

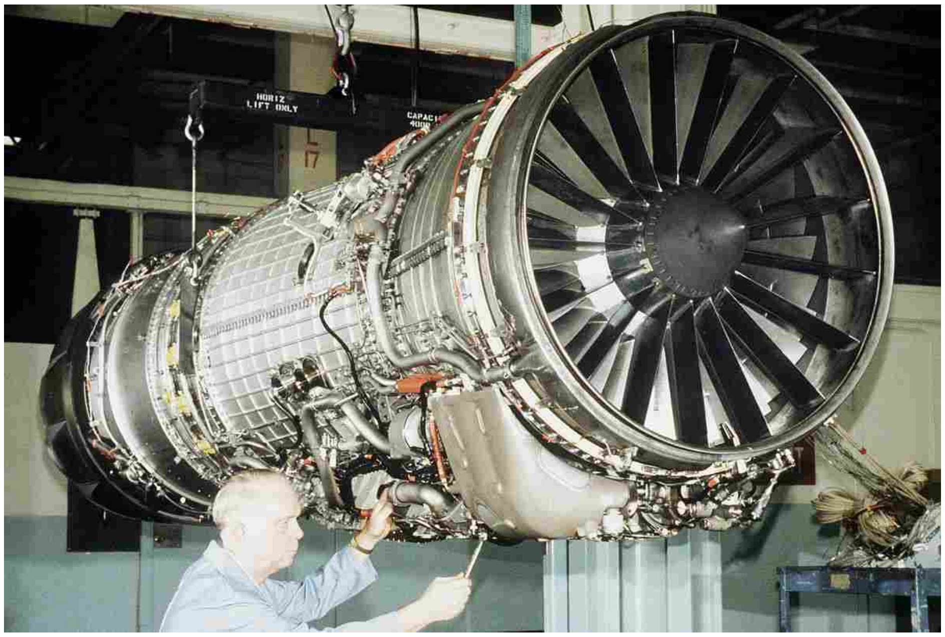


Figure 5.1: F110-GE Turbofan Engine with front frame visible. [29]

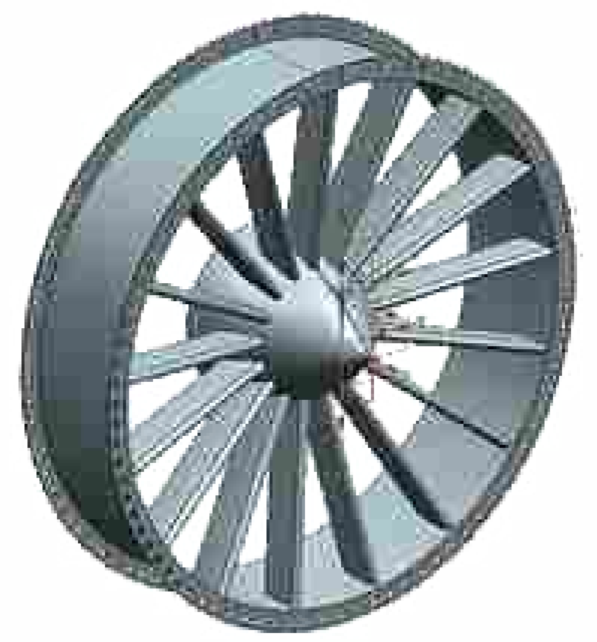


Figure 5.2: NX model of jet turbine engine front frame

in the model whose value the structural analysts and the fluid flow analysts differed in their desires and recommendation. In a general sense the structural analyst wanted very thick vanes for strength while the fluid flow analyst wanted thin vanes for better fluid flow through the channel. In the three iterations described below these two parameters, the thickness of the cylindrical sheet of material and the width of the vanes, were the deciding factors in whether the analysts approved the geometry or recommended changes.

First Iteration: In the first iteration both the structural analyst and the fluid flow analyst receive the model from the CAD program, thereby creating a linked copy for analysis (Analysis Model Creation). The analysts then proceed to prepare the geometry for each of their respective analyses (Geometry Decomposition). In this step the structural analyst idealizes the geometry by removing non-critical features, such as holes, and by dividing some geometric bodies to facilitate meshing. In comparison, during this step the fluid flow analyst uses the CAD geometry to extract a fluid geometry. In the next step each analyst meshes their respective analysis geometries, and adjusts or improves the mesh as necessary (Meshing, Mesh Manipulation). Next the boundary conditions and loads are set up, and the simulation is prepared for solving (Model Parameters, Assemble Model). The structural and fluid flow analyses are solved (Run Simulation). After the solution is reached they post-process the results and provide feedback and recommendations to the designer (Post Process, Archive Results). Images of the FEA and CFD results can be seen below in Figures 5.3 and 5.4.

Second Iteration: The second iteration begins with the implementation of the two analysts' recommended geometry changes in the CAD model. This updated model is then brought back to the analysis programs. Each analyst integrates the updated geometry into his or her analysis model, replacing old geometry, and reconnecting boundary conditions and other features to the updated geometry. When ready, each analysis simulation is run again, and the new results are post-processed. At the end of the second iteration of the test scenario the structural analyst recommends a geometric change while the fluid-flow analyst informs the designer that the design is satisfactory from a fluid-flow perspective.

Third Iteration: The third iteration begins with the implementation of the structural analyst's recommendation in the CAD model. Then, once again, the updated geometry is sent to the analysts. Each analyst again integrates the updated geometry into his or her analysis model, and

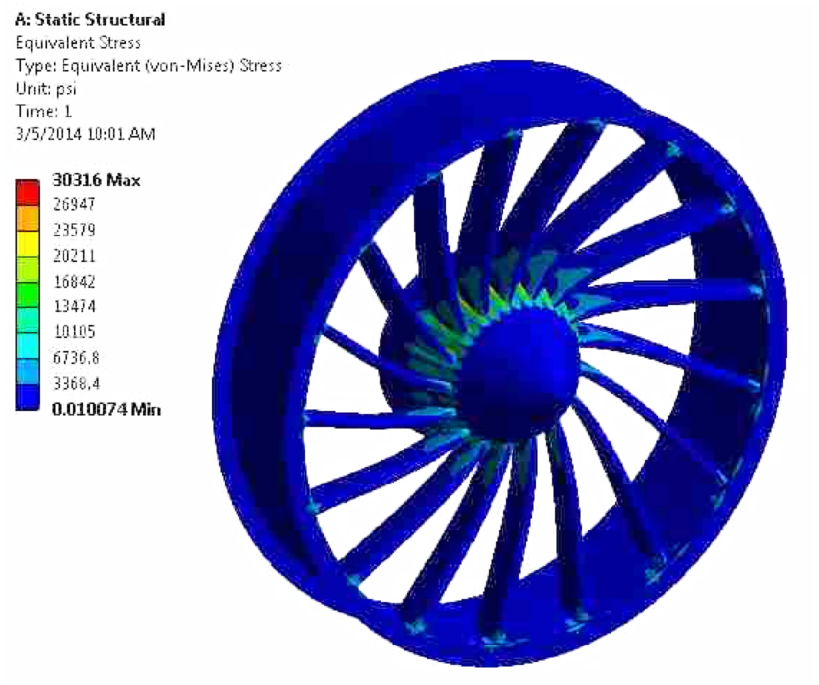


Figure 5.3: Von-Mises Stress Contour Plot from Structural Analyst at conclusion of Iteration 1 of Front Frame Test.

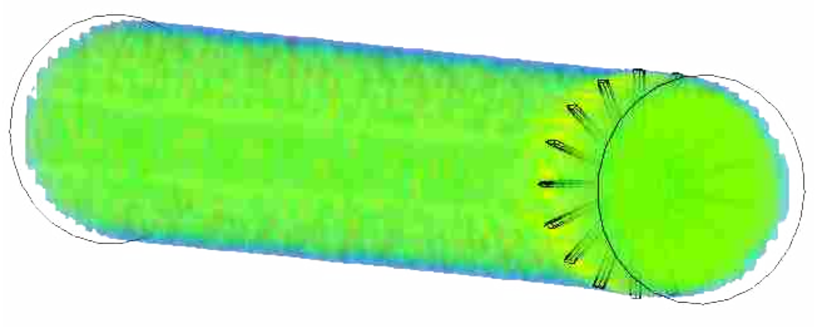


Figure 5.4: Air Velocity Countour Plot from Fluid-Flow Analyst at conclusion of Iteration 1 of Front Frame Test.

each simulation is run again. At the end of this third iteration both analysts inform the designer that the geometric design is satisfactory and the test concludes.

5.1.2 Integrated Architecture vs. Current Industry Practices in Test 1

The test scenario was performed with two design-analysis architectures, the method of transferring geometric data through neutral file formats which is currently used in industry, and

Method 1's multi-user, integrated design-analysis architecture. These two architectures are discussed in more detail in Sections 2.3 and 3.1 respectively. During trials that used the current industry-standard practice of using neutral file formats to transfer geometric data, an IGES file was used. The designer exported the CAD model as an IGES file and the analysts imported the IGES file into the analysis model to either create new geometry or replaced existing, outdated geometry. The trials using the multi-user, integrated design-analysis prototype transferred geometric data through the ANSYS Geometry Interface Tool. Geometric updates to the analysis model during the second and third iterations of the design-analysis loop were also handled by the ANSYS Geometry Interface Tool.

5.1.3 Results of Test Scenario 1

A total of twelve trials were performed in the test scenario 1. Six of these trials were performed by using the current industry practice of IGES geometry transfer, while the other six trials used the integrated design-analysis prototype[1]. The times taken to complete each iteration of FEA analysis in the scenario were recorded. The average times for the six trials of each scenario were calculated. The average times for the iterations of finite-element analysis are shown numerically in Table 5.1, and the average times for the iterations of computational fluid dynamics analysis are shown numerically in Table 5.2.

Table 5.1: Average FEA Times and Percent Difference between IGES geometry transfer and integrated design-analysis prototype.

	Iteration 1	**Iteration 2**	**Iteration 3**
IGES Scenario	20.5 min.	18.8 min.	7.8 min.
Integrated Scenario	18.5 min.	4.0 min.	2.2 min.
Percent Reduction in Time	*9.8%*	*78.8%*	*72.3%*

There is an interesting trend in the average times for each iteration. In this first test scenario the average times using the integrated system were shorter than the average times using the tradi-

[1]A seventh trial was performed using the integrated design-analysis prototype. However, the fluid flow analyst experienced errors during the trial which created a large delay. Due to this error the times recorded from this trial are not included in the data

Table 5.2: Average CFD Times and Percent Difference between IGES geometry transfer and integrated design-analysis prototype.

	Iteration 1	**Iteration 2**	**Iteration 3**
IGES Scenario	23.2 min.	18.2 min.	13.3 min.
Integrated Scenario	21.8 min.	4.5 min.	5.5 min.
Percent Reduction in Time	*5.8%*	*75.2%*	*58.8%*

tional IGES file geometry transfer method. This trend was true for both the finite-element analysis as well as the fluid flow analysis. In the first iteration of analysis there was virtually no difference in time between the use of traditional neutral formats and the integrated system. However, the time taken for the average second and third iterations using the integrated system were greatly reduced from the traditional IGES geometry transfer method (79% and 72% reductions for FEA, 75% and 59% reductions for CFD). In other words, the integrated architecture made no reduction in the analysis time for the first iteration, but caused very large reductions in time for subsequent iterations. This trend can be seen visually in Figures 5.5 and 5.6.

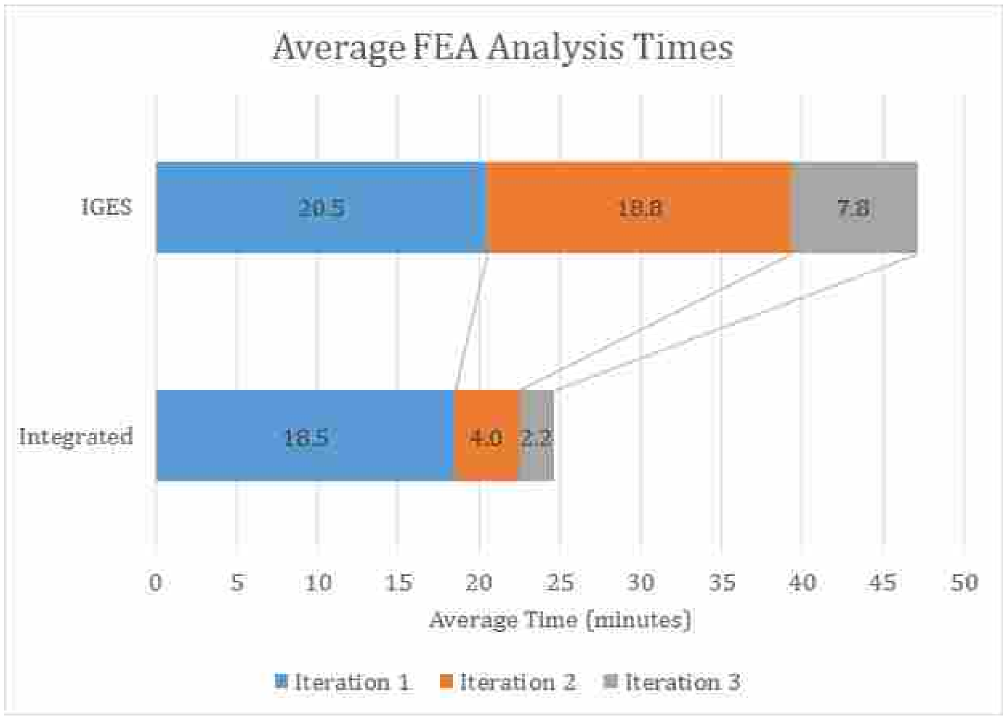


Figure 5.5: The average time to complete each finite-element analysis iteration of the test 1 scenario.

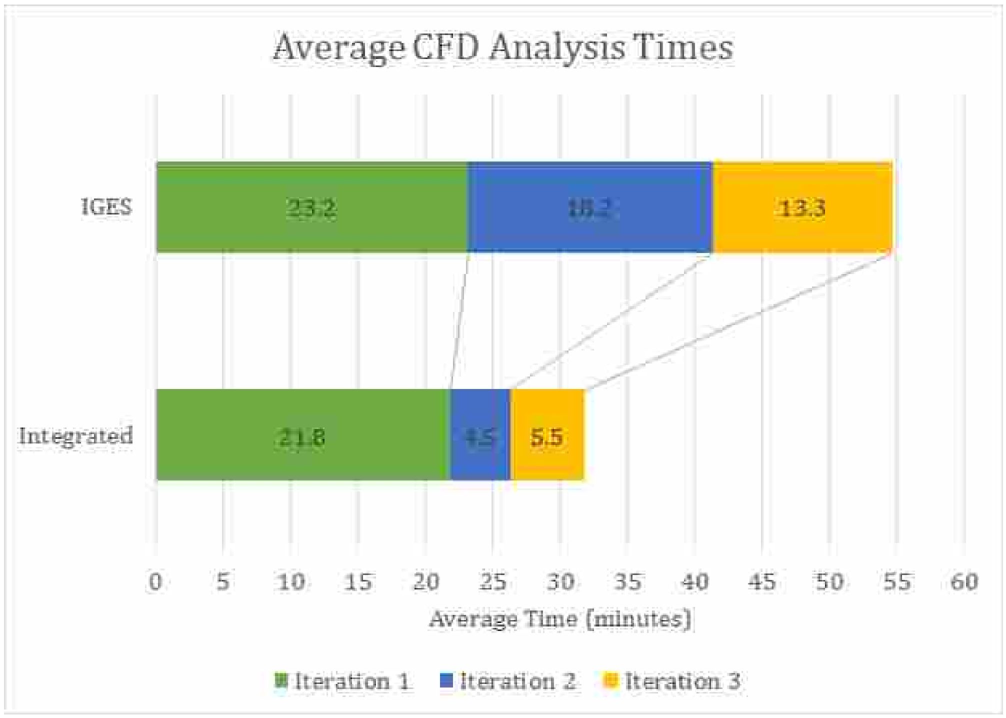
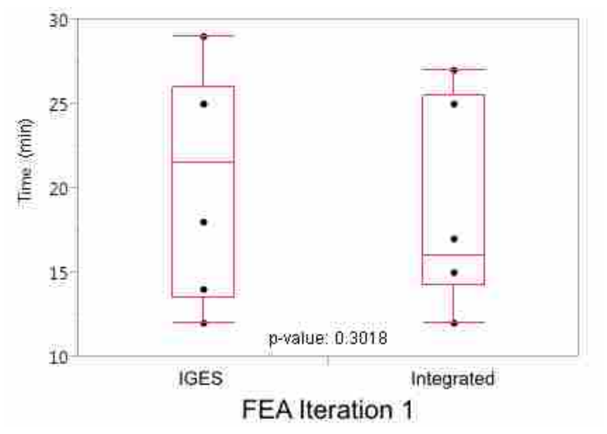


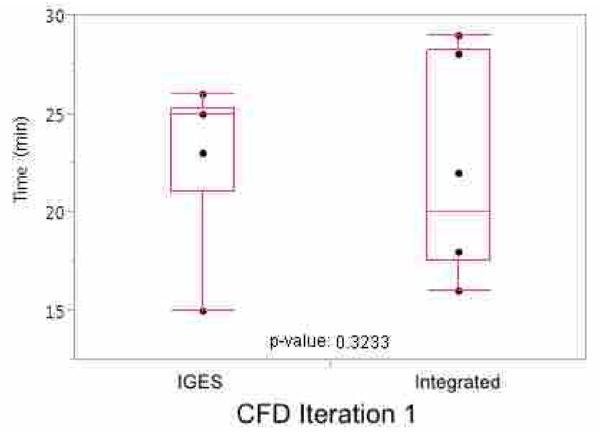
Figure 5.6: The average time to complete each CFD iteration of the test 1 scenario.

Statistical analysis was performed on data of recorded times in this first test scenario. The six recorded times of the first iteration of analysis using the IGES method of geometry transfer was compared to the six recorded times of the first iteration of analysis using the integrated architecture. Box-and-whisker plots comparing the iteration 1 times for the IGES geometry transfer and the integrated architecture can be seen below in Figures 5.7a and 5.7b. The p-values of 0.3018 and 0.3233 indicate that the difference in time between the use of IGES files and the integrated architecture for the first iteration is not statistically significant for finite-element analysis nor the fluid-flow analysis. Therefore, even though small differences in time for the first iteration are observed in Figures 5.5 and 5.6 these differences are insignificant.

A statistical analysis was also performed for the second iteration in the first test scenario. The six times taken from the IGES Geometry transfer method were compared to the six recorded times from the integrated architecture. This comparison was done for both the finite-element analysis and fluid flow analysis. Box-and-whisker plots comparing the iteration 2 times for the IGES geometry transfer and the integrated architecture can be seen below in Figures 5.8a and 5.8b. P-values of 0.0011 and <0.0001 indicate that there is a statistically significant difference in time



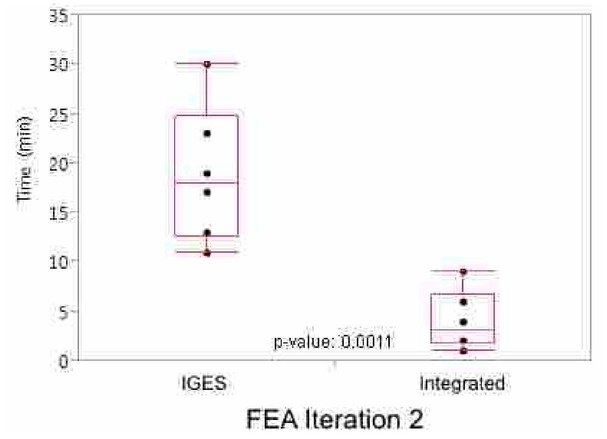
(a) Statistical Comparison of recorded times for Iteration 1 of FEA during Test Scenario 1



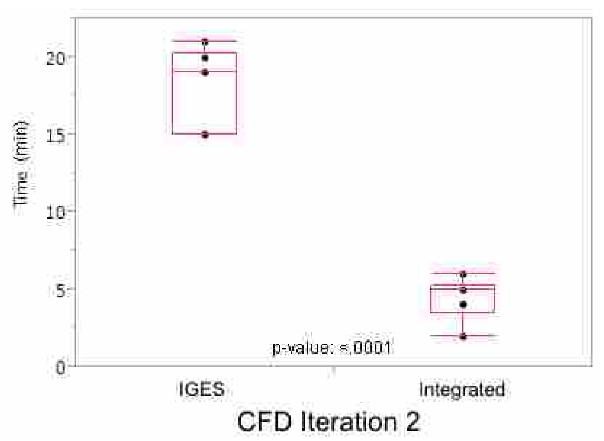
(b) Statistical Comparison of recorded times for Iteration 1 of CFD during Test Scenario 1

Figure 5.7: Statistical Analysis - Test 1 Iteration 1

between the use of IGES files and the integrated architecture for both finite-element analysis and the fluid-flow analysis. With a 95% confidence level it is determined that in iteration 2 of this test scenario the integrated design-analysis architecture completes the FEA tasks quicker by 7.5 - 12.2 minutes, and completes the CFD tasks quicker by 10.9 - 16.4 minutes.



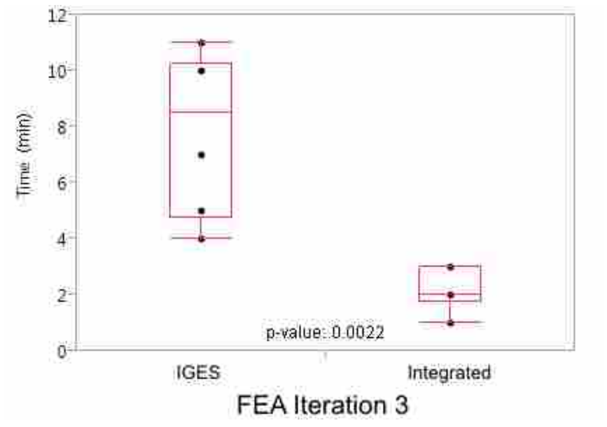
(a) Statistical Comparison of recorded times for Iteration 2 of FEA during Test Scenario 1



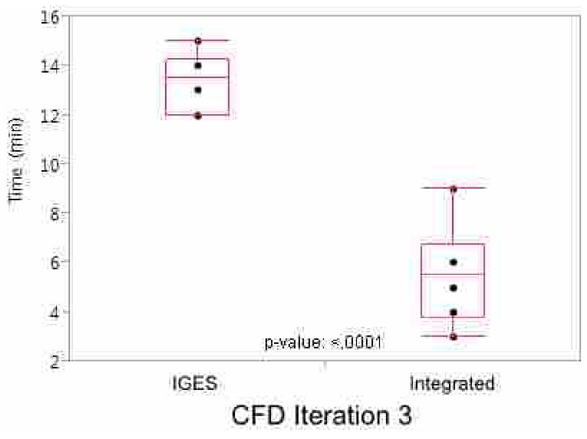
(b) Statistical Comparison of recorded times for Iteration 2 of CFD during Test Scenario 1

Figure 5.8: Statistical Analysis - Test 1 Iteration 2

A statistical analysis was also performed for the third iteration in the first test scenario. Box-and-whisker plots comparing the iteration 3 times for the IGES geometry transfer and the integrated architecture can be seen below in Figures 5.9a and 5.9b. P-values of <0.0001 and 0.0022 indicate that there is a statistically significant difference in time between the use of IGES files and the integrated architecture for both finite-element analysis and the fluid-flow analysis. With a 95% confidence level it is determined that in iteration 3 of this test scenario the integrated design-analysis architecture completes the FEA tasks quicker by 2.6 - 8.7 minutes, and completes the CFD tasks quicker by 5.5 - 10.1 minutes.



(a) Statistical Comparison of recorded times for Iteration 3 of FEA during Test Scenario 1



(b) Statistical Comparison of recorded times for Iteration 3 of CFD during Test Scenario 1

Figure 5.9: Statistical Analysis - Test 1 Iteration 3

It was hypothesized that the proposed, multi-user, integrated design-analysis architecture would reduce the time spent in iterations of the design-analysis cycle due to its integrated connection between the CAD model and analysis geometry. The integrated geometry facilitates geometry updates between the two programs. For example, when updated versions of geometry are sent from the CAD to the analysis program, the decomposition and idealization can be automatically applied to the updated geometry, and unchanged portions of geometry can retain the elements of their mesh as well as the applied boundary conditions, thereby reducing redundant work by the analyst.

This first test was designed to test this theory of time reduction. As described in this section the analysts recorded the time taken for each of the three iterations of analysis. As seen is Figures 5.5 and 5.6 the time taken to complete the first iteration of both FEA and CFD was nearly the same, while the time taken for each of the subsequent iterations of both FEA and CFD were significantly reduced. The statistics displayed in Figures 5.7, 5.8, and 5.9 show that for this test scenario the difference in time between the neutral file format method of geometry transfer and the integrated architecture are statistically significant for the second and third iterations of analysis. It can be supposed that this difference in times would also occur for a fourth, fifth, and more iterations. From the data gathered of the first three iterations of analysis there appears to be convergence of the times down to zero, as the second iteration is shorter than the first and the third is shorter than the second. However, the iteration times for the fourth iteration, fifth iteration, and beyond are not expected to reduce to zero, for there are still operations that the user must perform and computations which the computer must perform during each iteration. It is expected that the iteration times for subsequent iterations will become level, both for the neutral file format method and integrated architecture method. However, as shown in the data the time of iterations using the integrated architecture will be statistically significantly less than those using neutral file formats.

These findings answer the research question regarding the time of design-analysis iterations. This test proves that beginning with the second iteration there is a significant reduction in analysis time when the multi-user, integrated design-analysis architecture is used with CAD models of similar complexity as the one used in the test.

5.2 Test Scenario 2 - Automobile Engine Block

In the second test the principle investigation concerns the steps taken to perform multiple iterations of the design-analysis cycle, as described in Section 2.6. The time taken for each step within the loop was the focus of this test.

The test scenario incorporated six users, three as CAD modelers and three as analysts. One analyst performed structural analysis, the second performed thermal analysis, and the third performed a modal analysis. The three analysis disciplines were representative of many auxiliary functions which can be present in the design of a mechanical component, including structural, fluid flow, and thermal analyses, as well as Computer-Aided Manufacturing operations.

5.2.1 Test 2 Procedure

The scenario followed in this second test was the model construction and analysis of a generalized automobile V8 engine block, shown in Figure 5.10. This model was chosen because it represents a different industry than that represented in test scenario 1, and because it is aptly suited for multiple disciplines of analysis.

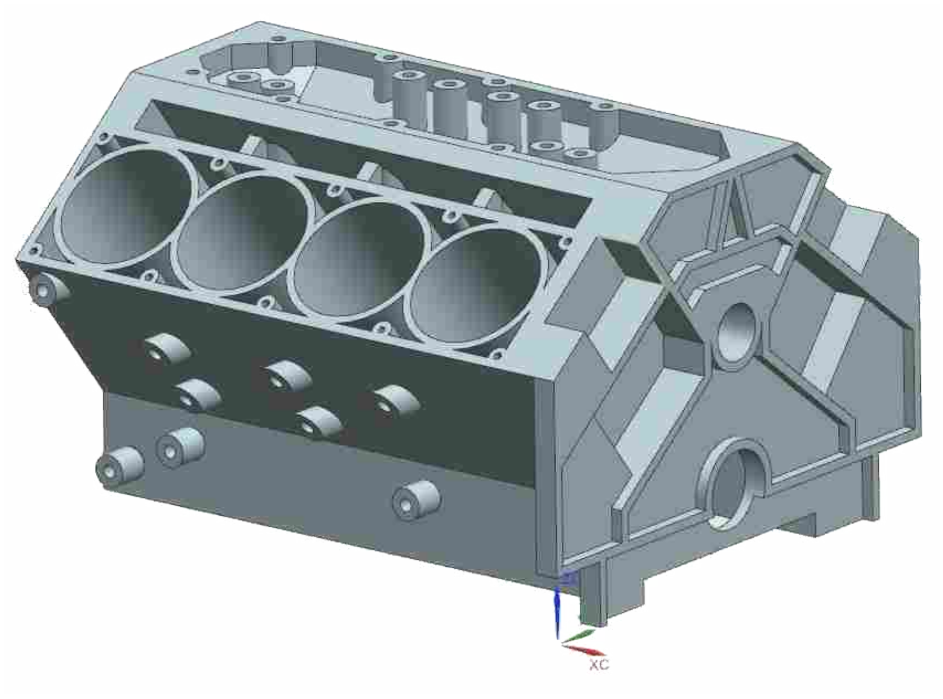


Figure 5.10: Engine Block CAD model

The test scenario would began with the three designers constructing the engine block model in NX Connect. The portions and features in the modeling plan were divided among the three designers in a manner already demonstrated in the research described in Section 2.2.

At the appropriate time, as described in Section 5.2.2, the analysts would import the CAD geometry into their individual sessions of ANSYS Workbench to begin pre-processing.

In ANSYS Workbench the three analysts would each perform geometry decomposition and/or idealization in the DesignModeler geometry editing tool. Each analyst would modify the geometry to suit his or her individual analysis needs. The analysts then proceeded to perform the subsequent steps of meshing, analysis setup, solving, and post-processing. While post-processing the analysts would compare the results to allowable criteria originally given to them (maximum stress and deformation for structural analysis, maximum temperatures at certain locations for thermal analysis, etc.). Examples of resulting contour plots for structural, thermal, and modal analysis on the engine block model are shown in Figures 5.11, 5.12, and 5.13 respectively. Upon completion of the first iteration of analysis the analysts would provide feedback to the designers by recommending geometric changes to improve the model with regards to the allowable criteria they were originally given. Some of the recommended changes included the thickness of the cylinder walls and the rate of flow of coolant through the cooling channels.

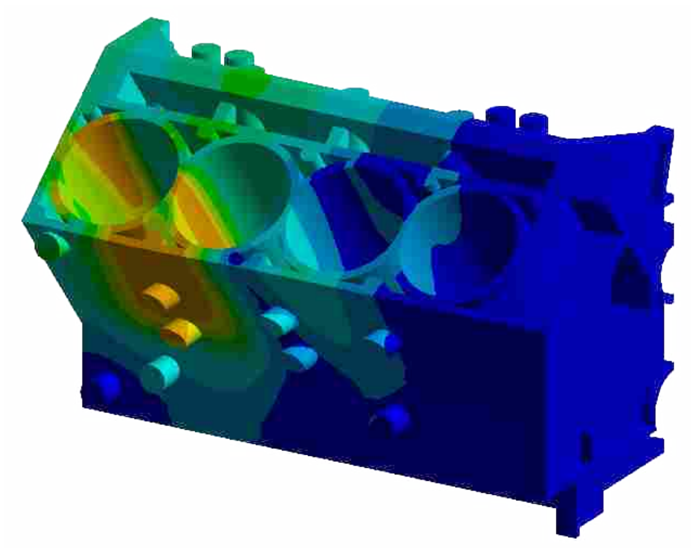


Figure 5.11: Von-Mises Stress Contour Plot from Structural Analysis of a combustion cylinder pressure load in the Engine Block model

After the first iteration of analysis the designers would modify the engine block model according to the suggestions from the analysts. Then the analysts would import this updated geometric model into ANSYS Workbench and follow the steps of pre-processing, solving, and post-processing again in order to perform a second iteration of analysis on the engine block model. The analysts and designers continued to perform iterations of the design-analysis process until a satisfactory design had been achieved.

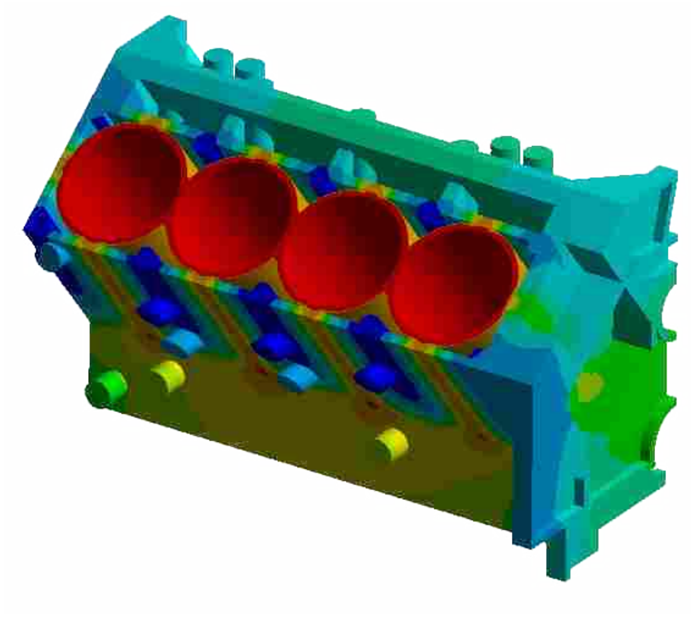


Figure 5.12: Temperature Contour Plot from thermal analysis of combustion temperatures in cylinders of the Engine Block model

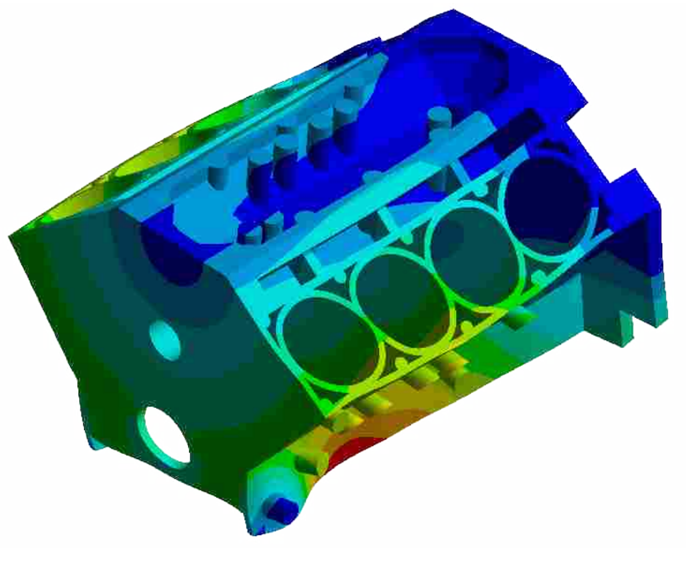


Figure 5.13: Contour Plot showing one of the free-vibration modes of Engine Block model

5.2.2 Integrated Architecture vs. Current Industry Practices in Test 2

As done in Test Scenario 1 this second test scenario was performed with two design-analysis architectures: the method of transferring geometric data through neutral file formats which is currently used in industry and Method 1's multi-user, integrated design-analysis architecture. These two architectures are discussed in more detail in Sections 2.3 and 3.1 respectively.

In the tests which utilized the method of geometry transfer through neutral file format the designers would send the model geometry to the analysts in a Parasolid file, which is a kernel-specific geometry format. This would occur once the model was fully completed in the multi-user NX Connect session. The geometry transfer for each subsequent iteration was performed in a similar manner, with the designers sending the Parasolid geometry file to the analysts once they finished making the necessary geometric changes. The analysts updated the geometry in their analysis models by importing the new Parasolid file into their session of ANSYS Workbench, making the necessary geometry cleanup/idealization, and setting up the analysis with this new geometry.

In the scenarios which utilized the integrated design-analysis architecture the analysts did not wait for the designers to send them the geometry. Instead, while the designers built the model from scratch the three analysts were present in the multi-user CAD session, though they did not contribute to the construction of the model. As observers they witnessed the engine block's construction, and, when the model was completed to a point at which each felt they could begin working, they linked their NX Connect client to ANSYS Workbench and transferred the geometry to the analysis suite through the ANSYS Geometry Interface Tool. These analysts were not required to wait until the designers sent them the model, nor were they even required to wait until the designers were finished with all of their tasks. Instead they had the freedom to transfer the CAD geometry to their analysis program when they felt they could use it. In the times when the analysts transferred over a partially-completed model they were free to update their analysis model to match the CAD model at nearly any point in pre-processing.

It should be noted that the way designers plan to build a model may not be a sequence which the analysts could pull "rough" versions of the geometry to perform preliminary analyses. Therefore there must exist communication between the analysts and the designers before the CAD model is begun. The analysts must be involved in the Design Requirements and Concept stages described in Section 2.4, so that they understand what is to be built. By understanding the design requirements and understanding what will be built they can know what features will need to be analyzed, then, they can explain to the designers which features are important to them, or which key features they would like to analyze first. That way the designers can plan the model construction in such a way as to complete those key features as soon as possible, so that the analysts can perform their analyses as soon as possible.

In subsequent iterations the analysts would update their analysis geometry to match the modified CAD geometry through the Geometry Interface Tool. This method caused many of the geometry modifications/idealizations and pre-processing steps to be automatically applied to this new, updated geometry.

5.2.3 Results of Test Scenario 2

During each test scenario the designers and analysts recorded the times taken to perform the steps of the design-analysis cycle described in Section 2.6 and shown visually in Figure 2.9. Specifically, the designers recorded the time taken for the Design Solid Model Creation step (both their portion of the modeling session and the total modeling time), and the analysts recorded the time taken in each of the subsequent steps of the design-analysis cycle except Archive Artifacts. One may recall that these steps are Analysis Solid Model Creation, Geometry Decomposition, Meshing, Mesh Manipulation, Assign Model Parameters, Assemble Simulation Model, Run Simulation, and Post-process Results.

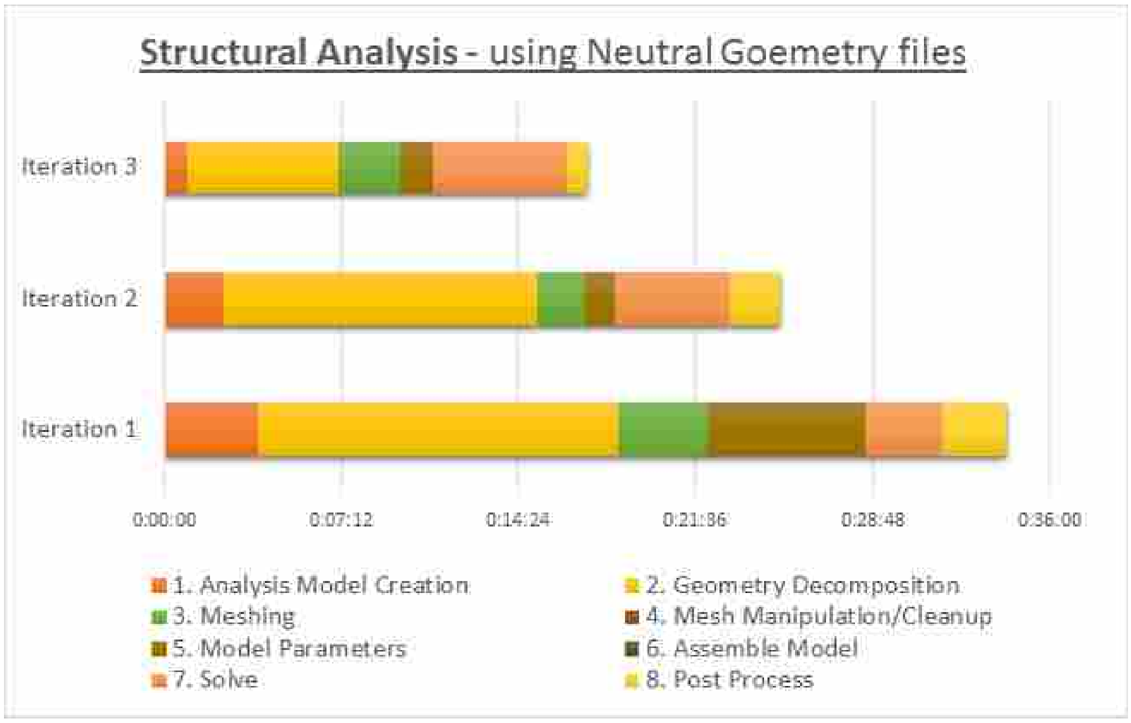
Two trials were performed by using neutral-file formats in the geometry exchange and two trials were performed by using the integrated design-analysis architecture. However, one of the trials of the neutral-file format scenario was not performed correct to the instructions given and thus did not reflect a scenario as intended by the test. Also, one of the trials using the integrated design-analysis architecture experienced errors in the multi-user CAD software that prevented the design-analysis integration from functioning properly, thereby negating the trial as an accurate representation of the integrated architecture. Therefore the testing produced only one set of data for the neutral-file format scenario and one set of data for the integrated design-analysis architecture scenario. The structural analysis times gathered from both the neutral-file format scenario and integrated architecture scenario are displayed below in Figure 5.14, the thermal analysis times gathered are displayed below in Figure 5.15, and the modal analysis times gathered are displayed below in Figure 5.16.

The second research question posed in Chapter 1 addressed the potential for the integrated architecture to reduce the number of steps in the design-analysis cycle. It was hypothesized that some of the analysis steps in Equation 2.4 would be reduced or even eliminated by the proposed integrated architecture. This is due to the fact that the proposed architecture retains much of the geometric data and facilitates the importations of updated geometry.

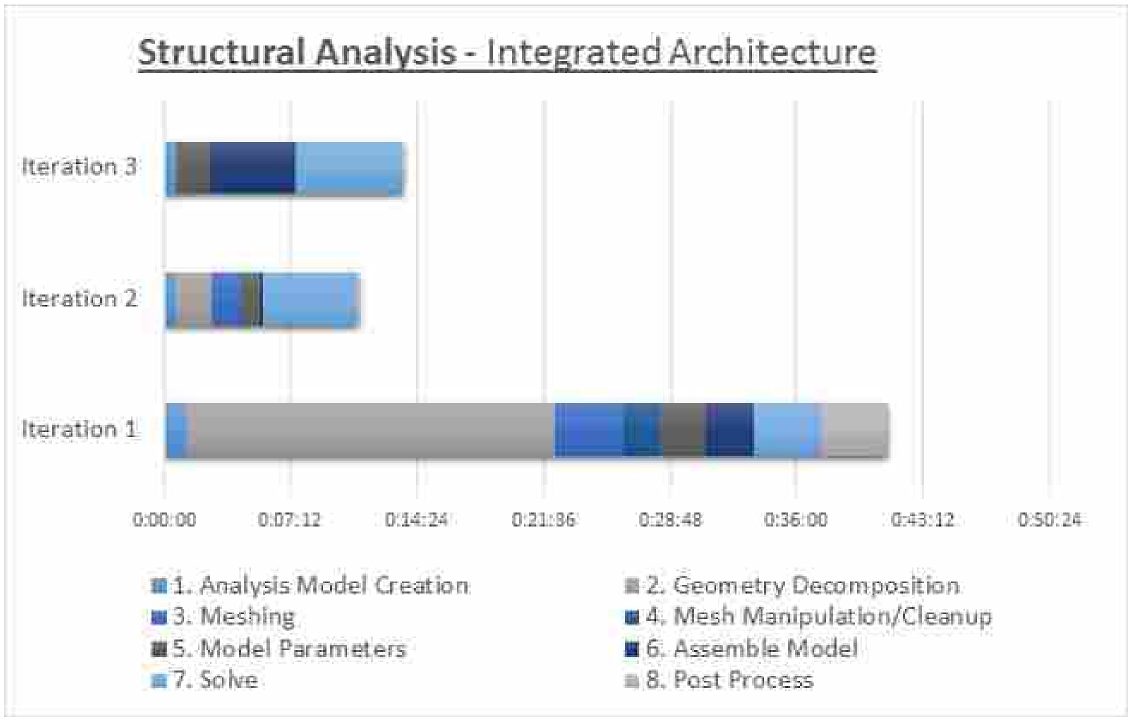
During the second test, described in Section 5.2, the analysts measured the time of each step within the analysis process for each iteration. Results were obtained, though statistically significant differences were unable to be determined from the data due to the lack of quantity of data. The data shown in Figures 5.14, 5.15, and 5.16 are from only two data points (one trial using neutral file formats and one trail using the integrated architecture). Because this data comes from only one trail of each architecture there is variability witnessed. This variability is expected to be eliminated as additional trails are performed and more data is gathered. An example of this variability is that in the integrated architecture scenario all three analysis disciplines took longer to complete the third iteration than the second, when it would be expected that the opposite would be true. In these single trials the data is effected by other variables such as the skill of the users and different computers' performance capabilities. With more trails and more data actual trends will begin to show and the variability witnessed in this first trail will be reduced.

The third research question posed in Chapter 1 inquires about the ability of the proposed architecture to create parallelization between the multi-user CAD program and the attached auxiliary programs. As explained in Section 1.3, a quantitative study of this phenomenon was not a goal of this research. However, in this second test scenario several observations were made in regards to this parallelization of design and analysis.

The proposed architecture does not cause design and analysis to exist completely in parallel, that is, the two processes cannot begin at the same time. Indeed, design and analysis will always contain elements of a serial process since the analysis step relies on the creation of design geometry first. However, as demonstrated by this second thesis test, the proposed multi-user, integrated design-analysis architecture described as Method 1 of Section 3.1 allows some overlap of the work performed by the analysts and the designers. During testing it was observed that the ana-

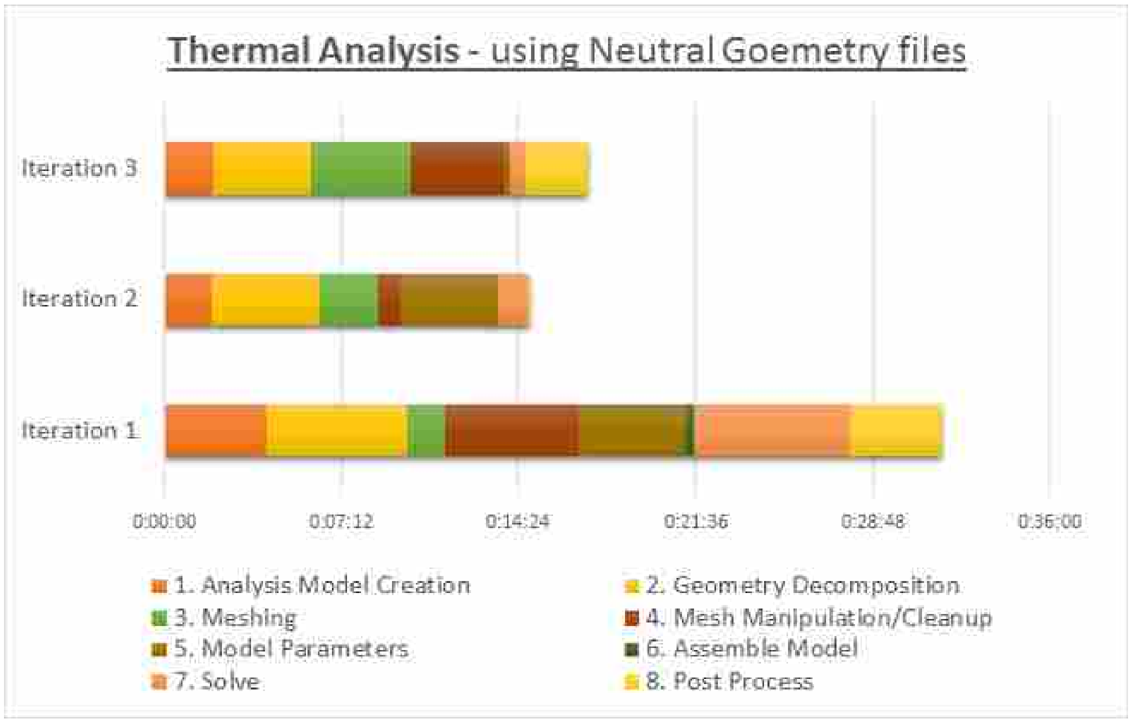


(a) Times of each step in Structural Analysis using Neutral File Format geometry transfer. All three iterations are shown.

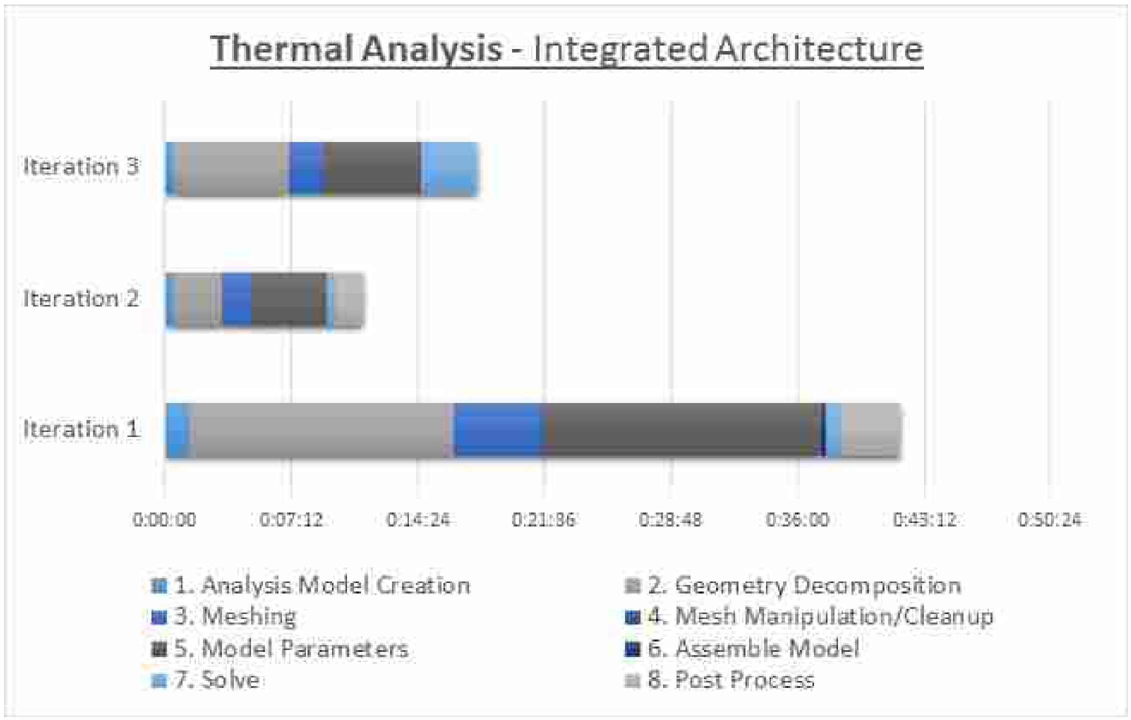


(b) Times of each step in Structural Analysis with geometry transfer through the integrated architecture. All three iterations are shown.

Figure 5.14: Test 2 - Times of Analysis Steps for Structural Analysis, comparing the use of Neutral File Formats and the Integrated Architecture

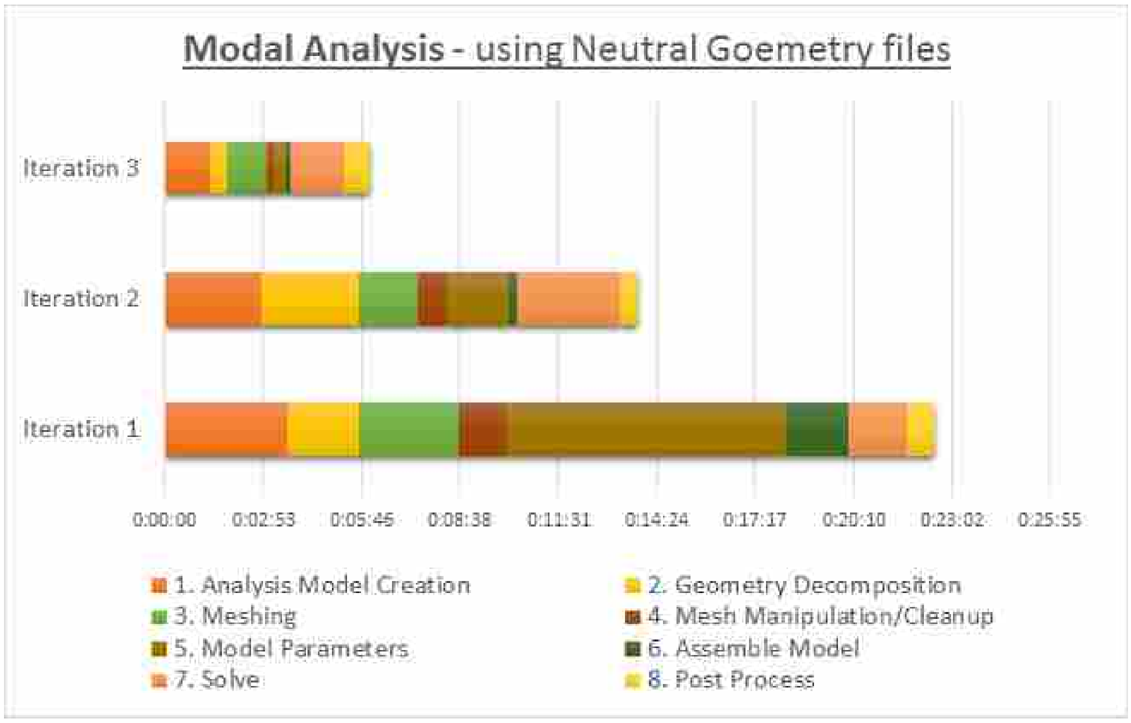


(a) Times of each step in Thermal Analysis using Neutral File Format geometry transfer. All three iterations are shown.

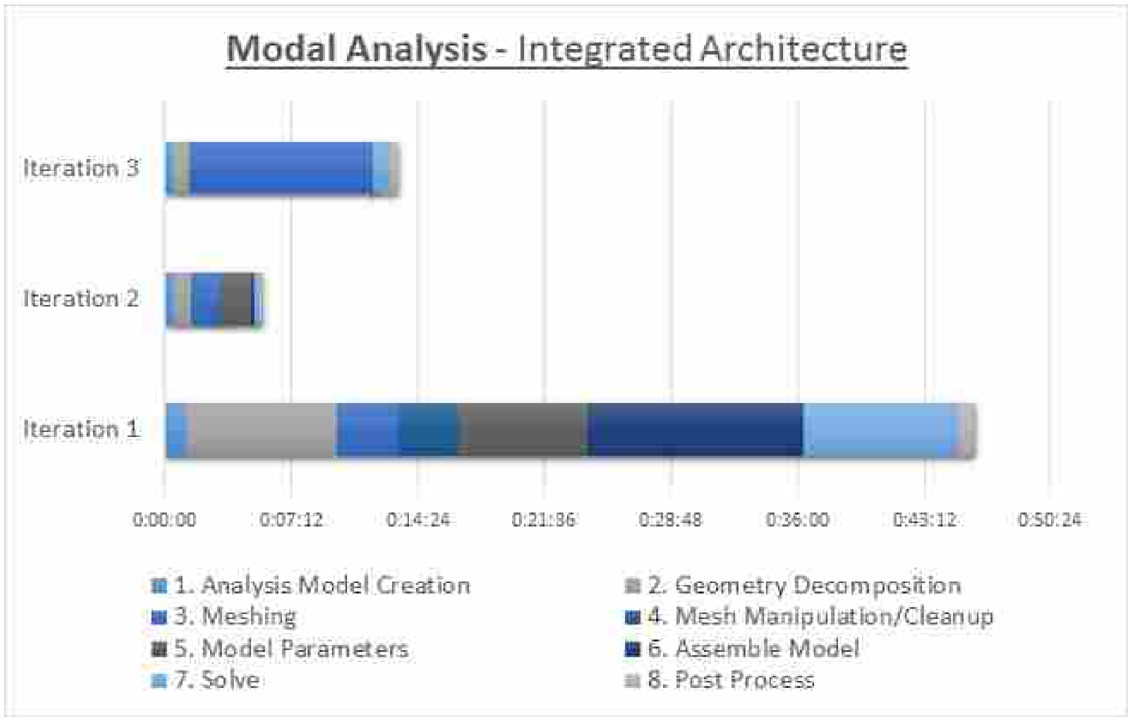


(b) Times of each step in Thermal Analysis with geometry transfer through the integrated architecture. All three iterations are shown.

Figure 5.15: Test 2 - Times of Analysis Steps for Thermal Analysis, comparing the use of Neutral File Formats and the Integrated Architecture



(a) Times of each step in Modal Analysis using Neutral File Format geometry transfer. All three iterations are shown.



(b) Times of each step in Modal Analysis with geometry transfer through the integrated architecture. All three iterations are shown.

Figure 5.16: Test 2 - Times of Analysis Steps for Modal Analysis, comparing the use of Neutral File Formats and the Integrated Architecture

lysts needed only to wait for the completion of the geometric features relevant to their analysis. For example, the structural analyst, who in the test scenario set up simulations of combustion pressures within the engine block's cylinders, needed only to wait for the completion of the piston cylinder geometry to then perform his analysis. In effect, this analyst understood which geometric features were required to be created in a serial process (the cylinder geometry he needed to analyze), and which geometric features could be created in a parallel process (all other geometric features not affecting his analysis).

In reality the parallelization of design and analysis occurs to an extent in industry today, even with the use of neutral file formats and direct translation as mediums for geometry transfer. Often "rough" CAD models are created first and sent to analysts while the designers continue to add details to the geometric model. However, not only does the proposed architecture facilitate this transfer of geometry and parameters back and forth between the CAD program and auxiliary programs, but this second test scenario demonstrates that the multi-user nature of the architecture allows the analysts to begin work *before the geometric feature they require are even completed.* Within the proposed architecture the analyst may start on other tasks, such as material creation or the idealization of certain geometric features, while the designers concurrently create additional geometric features required in the analysis. The analyst would then pull those new features into his or her analysis model when they are completed. This additional overlap of time and parallelization of design and analysis work is made possible by the proposed architecture described in Section 3.1.

In addition to the observations of parallelization new forms of conflict between designers and analysts were discovered in the proposed architecture. These conflicts were independent from the forms of conflict already observed in multi-user CAD architectures, as described in the research cited in Section 2.2.

One of the forms of conflict observed was the requirement of designers to cede some of their control over the CAD model to the analysts. Under a traditional form of geometry transfer (neutral-file format or direct transfer) the analysts themselves have no access to the design model. They can only provide recommendations and suggestions to the designers who make changes to the model. In the proposed architecture however, the analysts are given direct access to the shared CAD model. Besides simply making recommendations they themselves can edit and modify the design geometry. This breaks the designers' exclusive control over the CAD model, and could lead to data inconsistency and syntactic conflict within the model. The analysts' new ability to modify the master geometry without adequate communication with the designers could lead to semantic conflicts as well.

CHAPTER 6. CONCLUSION

From the results discussed in Chapter 5 several notable facts about the proposed multi-user, integrated design-analysis architecture were discovered. First, it was determined that for CAD models of similar complexity to the Front Frame model used in Test Scenario 1 the proposed architecture does not decrease the time required to perform the first iteration of engineering analysis but does reduce the time of subsequent iterations of analysis by about 70%. This reduction in time is caused by the improved method of geometry data transfer which in effect “connects” the analysts better to the design model and facilitates the process of updating analysis geometry. These time savings in analysis in turn contribute to an overall reduction of time in the entire engineering design process.

The proposed multi-user, integrated design-analysis architecture was also observed to create more parallelization between the design and analysis processes. The proposed architecture allows analysts to begin working even before the geometry they need to analyze is completed. The ability for designers and analysts to work concurrently is facilitated by the proposed architecture, an overlap of time which likely yields more reductions of time in the entire engineering design process.

6.1 Recommendations

It is recommended that research continue in this topic of concurrent engineering through the parallelization of the design-analysis process. More information should be learned about some of the ideas presented in this thesis.

In this thesis the benefits of a multi-user, integrated design-analysis architecture were first demonstrated using simple models. However it is only speculated that these benefits will also be witnessed in more complex, real-world designs. Therefore, it is recommended that tests of the proposed architecture be performed on more complex engineering designs to investigate whether the benefits in the design-analysis process continue as the complexity of models increases.

Quantitative results were not achieved with regards to the reduction of time or complete elimination of steps within the analysis process, as investigated in the second test scenario. Therefore it is recommended that further research and testing be performed to determine the quantitative effects on the time of individual steps of the analysis process.

Though the effects of concurrent work of designers and analysts were not the principle investigation of this thesis, observations were made regarding the qualitative benefits of parallelization. Further investigation into the quantitative benefits of parallelization, namely the reduction of time of the overall design process, would be useful in understanding when a multi-user, integrated design-analysis architecture would be of most use in different engineering design situations.

As this thesis only observed visible conflicts between designers and analysts and commented on other unseen but potential conflicts, further research into the conflict between designers and analysts in a multi-user, integrated design-analysis architecture would be required to better understand the sources of and solutions to these conflicts.

Another fertile field of future research is the exploration of the second proposed method described in Section 3.2. This second proposed architecture incorporates connections from the different analysis programs directly to the multi-user server, instead of connecting through the local CAD clients as done the architecture presented in Method 1. This second architecture could have the ability to attach multi-user analysis or multi-user CAM to the multi-user CAD session, thereby expanding the possibility for cooperation and parallelization beyond the levels achieved through the architecture presented in Method 1. One critical question to be answered in this further research would be how to actually construct a prototype of this second architecture. How will the auxiliary programs (FEA, CFD, CAM, etc.) share geometric data directly with the multi-user CAD server? The answer to this question, which is critical to the creation of a prototype of this architecture, would be a great advancement towards answering a core question that researchers and engineers have asked for decades: how do we fully integrate computer-aided design and other computer-aided applications such as FEA and CAM?

REFERENCES

[1] Hughes, T. J., Cottrell, J. A., and Bazilevs, Y., 2005. “Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement.” *Computer methods in applied mechanics and engineering,* **194**(39), pp. 4135–4195. 4

[2] Red, E., Holyoak, V., Jensen, C. G., Marshall, F., Ryskamp, J., and Xu, Y., 2010. “ν-CAx: A research agenda for collaborative computer-aided applications.” *Computer-Aided Design and Applications,* **7**(3), pp. 387–404. 4, 7

[3] Xu, Y., Red, E., and Jensen, C. G., 2011. “A flexible context architecture for a multi-user GUI.” *Computer-Aided Design and Applications,* **8**(4), pp. 479–497. 4, 7

[4] Moncur, R. A., Jensen, C. G., Teng, C.-C., and Red, E., 2013. “Data consistency and conflict avoidance in a multi-user CAx environment.” *Computer-Aided Design and Applications,* **10**(5), pp. 727–744. 4, 7

[5] Cannon, L., Nysetvold, T., Phelps, G., Winn, J., and Jensen, C. G., 2012. “How can NX advanced simulation support multi-user design?.” *Computer-Aided Design and Applications, PACE,* **2**, pp. 21–32. 4, 7, 8

[6] Briggs, J., Hepworth, A., Stone, B., Coburn, J., Jensen, G., and Red, E. “Multi-user, integrated design and analysis: A fundamental approach.” *(Not Yet Published).* 4, 7, 8

[7] Marshall, F., 2011. “Model decomposition and constraints to parametrically partition design space in a collaborative CAx environment.” Masters thesis, Brigham Young University, December. 4

[8] Xu, Y., 2010. “A flexible context architecture for a Multi-User GUI.” Masters thesis, Brigham Young University, December. 4

[9] Inder Science Publishers, 2009. International Journal of Collaborative Engineering. 6

[10] Abbey, T., 2013. “Integrate FEA and CAD.” *Desktop Engineering,* **9**, pp. 48–51. 6, 9, 11

[11] Hoffman, C. M., and Joan-Arinyo, R., 1998. “CAD and the product master model.” *Computer-Aided Design,* **30**(11), pp. 905–918. 6

[12] Ma, Y.-S., Chen, G., and Thimm, G., 2008. “Paradigm shift: unified and associative feature-based concurrent and collaborative engineering.” *Journal of Intelligent Manufacturing,* **19**(6), pp. 625–641. 6

[13] Hammond, J., Koubek, R. J., and Harvey, C. M., 2001. “Distributed collaboration for engineering design: A review and reappraisal.” *Human Factors and Ergonomics in Manufacturing,* **11**(1), pp. 35–52. 6, 10

[14] Red, E., Jensen, G., French, D., and Weerakoon, P., 2011. “Multi-user architectures for computer-aided engineering collaboration.” In *Concurrent Enterprising (ICE), 2011 17th International Conference on*, IEEE, pp. 1–10. 7, 8

[15] Ograph, B. T., and Morgens, Y. R., 2008. “Cloud computing.” *Communications of the ACM,* **51**(7). 7

[16] Shaojin, Y., Jianjun, C., and Jindou, L., 2010. “An asynchronous CAD collaborative design model.” In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, Vol. 6, IEEE, pp. V6–563. 7

[17] Weerakoon, P., Wu, J., Bright, T., Teng, C.-C., Red, E., Jensen, G., and Merkley, K., 2013. “A networking architecture for a multi-user FEA pre-processor.” *Computer-Aided Design and Applications,* **10**(3), pp. 527–540. 8

[18] Weerakoon, P., Wu, J., Bright, T., Red, E., Teng, C.-C., Jensen, G., and Merkley, K., 2012. “Multi-user FEA pre-processing and workspace assignment.” In *ASME Early Career Technical Conference, 2012 ECTC*, ASME, pp. 1–6. 8

[19] McGuire, J. G., Kuokka, D. R., Weber, J. C., Tenenbaum, J. M., Gruber, T. R., and Olsen, G. R., 1993. “SHADE: Technology for knowledge-based collaborative engineering.” *Concurrent Engineering,* **1**(3), pp. 137–146. 10

[20] Olsen, G. R., Cutkosky, M., Tenenbaum, J. M., and Gruber, T. R., 1995. “Collaborative engineering based on knowledge sharing agreements.” *Concurrent Engineering,* **3**(2), pp. 145–159. 10

[21] Lu, S.-Y., ElMaraghy, W., Schuh, G., and Wilhelm, R., 2007. “A scientific foundation of collaborative engineering.” *CIRP Annals-Manufacturing Technology,* **56**(2), pp. 605–634. 10

[22] Pahng, G.-D., Bae, S., and Wallace, D., 1998. “A web-based collaborative design modeling environment.” In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 1998.(WET ICE’98) Proceedings., Seventh IEEE International Workshops on*, IEEE, pp. 161–167. 10

[23] ANSYS, Inc., 2009. "ANSYS Geometry Interfaces". Brochure. 12, 29

[24] Mattson, C. A., and Sorensen, C. D., 2013. *Fundamentals of Product Development*. CreateSpace Independent Publishing Platform. 12

[25] Ulrich, K., and Eppinger, S., 2011. *Product Design and Development*. Product Design and Development. McGraw-Hill Education. 13

[26] Hardwick, M., 2005. “Dart system analysis presented to simulation sciences seminar.” *Sandia National Laboratories, Albuquerque, NM June,* **28**, p. 2005. 15

[27] Owen, S. J., Clark, B. W., Melander, D. J., Brewer, M., Shepherd, J. F., Merkley, K., Ernst, C., and Morris, R., 2008. “An immersive topology environment for meshing.” In *Proceedings of the 16th International Meshing Roundtable*, Springer, pp. 553–577. 18

[28] Hamdi, M., Aifaoui, N., and Benamara, A. “Design and analysis integration model based on idealization of cad geometry.”. 27

[29] Wikipedia, 2015. General electric f110 — wikipedia, the free encyclopedia [Online; accessed 19-January-2015]. 37