2010-09-16

Developing Intelligent Engineering Collaboration Tools Through the use of Design Rationale

Jordan David Ryskamp
Brigham Young University - Provo

Developing Intelligent Engineering Collaboration Tools Through the use of Design Rationale

Jordan D. Ryskamp

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

Master of Science

C. Greg Jensen, Chair
W. Edward Red
Christopher A. Mattson

Department of Mechanical Engineering

Brigham Young University

December 2010

ABSTRACT

Developing Intelligent Engineering Collaboration

Tools Through the use of Design Rationale

Jordan D. Ryskamp

Department of Mechanical Engineering

Master of Science

This thesis presents a new method that improves upon the existing approaches to developing collaborative tools. The new method uses automatically inferred and manually recorded design rationale to intelligently filter the information that is shared by a collaborative tool. This represents an improvement upon the existing state of the art in collaborative engineering tools. To demonstrate the viability of the method three collaborative tools were created. The first is a multi-user collaborative design environment tool named SimulPart and built upon the NX CAD package. SimulPart uses the new method to limit the amount of communication required to keep every user in synch during a multi-user co-design session. The second implementation is a visual history tool named VisiHistory that allows designers to watch time lapse videos of the creation of a design that are automatically generated using the new method. The final tool is an intelligent user directory named SmartHelp that uses the new method to allow designers to identify which of their peers have expertise in certain CAD operations. Validation was performed for each of these tools by benchmarking the tool against the leading commercial solution or industry process. The results of the validation showed that the new method does in fact offer a superior collaborative solution as it outperforms the existing tools and methods in several key collaborative metrics. As a result of this work future efforts are encouraged into both improving upon the quality of the inferred design rationale and increasing the functionality of the three tools created using the new method.

Keywords: design rationale, multi-user CAD, automated acquisition, collaboration, design process, design intent

ACKNOWLEDGMENTS

I would like to first and foremost thank my wife Katelyn for all of her support and understanding while I have pursued my dreams. Without her support and encouragement I would never have been able to accomplish this task. I would also like to thank my advisor Dr. Jensen and my sponsor Pratt and Whitney for all of their help and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Engineering companies are increasingly becoming global enterprises as they attempt to leverage the information technology revolution to create better products at a lower cost that are capable of competing in the global markets. Unfortunately, many of these companies are now facing the realities of global engineering: that despite all of its advantages global engineering requires a complete redefinition of how knowledge and information are shared. For many engineers, gone are the days of being able to just walk down the hall, sit down at a workstation and work out the intricacies of a design. Today, software based tools such as: email, chat, groupware, video conferencing and VoIP have become the key facilitators of collaboration within the engineering community.

1.1 Problem Statement

While digital collaboration tools address all of the different types of collaboration and have seen widespread adoption, the satisfaction with these tools has been mixed. Most tools are found lacking when compared with the gold standard of face to face collaboration. In response extensive research has been done into virtually replicating the experience of face to face collaboration (Billinghurst and Hirokazu 2002). Recent examples include advanced application sharing, hologram presentation technology and video conferencing rooms that mimic the feel of a boardroom (White 2007). Despite these efforts the reality is most existing collaboration tools fall short when compared with face to face collaboration.

1.2 Thesis Objective

The objective of my research was to devise a method for engineering collaboration that once implemented would provide tools which surpassed the existing standard of tools and ideally surpassed even face to face collaboration. The key to achieving this objective was to identify a way to leverage the unique benefits of digital technology rather than just attempting to match existing analog collaborative methods. The ability for a computer to record all of the actions a user is taking while interacting with a computer aided design (CAD) tool, identify patterns in these actions and then later instantaneously recall that information was chosen as the basis for my research. Within the engineering community the "what" and the "why" of a user's design process are known as design rationale (Klein, Capturing Design Rationale in Concurrent Engineering Teams 1993). More specifically, design rationale can be thought of as a combination of some or all of the specifications, actions and motivations used to create a design. The objective of my research is to show that design rationale can be leveraged to provide an improved method for engineering collaboration.

To validate the method created during this research three different collaboration software tools were created. These tools are:

- SimulPart – A tool that enables concurrent CAD modeling.
- SmartHelp – A tool that allows engineers to discover which of their peers have experience with a specific CAD operation.
- VisiHistory – A tool that displays a visual history of a CAD file.

Each tool relies upon the same generalized method but solves a different aspect of the collaboration problem described above. While the purpose of these tools is to demonstrate the usefulness of the proposed method, the overall objective of this thesis is to:

1. Develop a generalized method for capturing and leveraging design rationale to improve collaboration in a broad spectrum of scenarios.
2. Solve a variety of collaboration problems using different tools derived from the generalized method.
3. Prove that the tools provide an improved collaboration experience over existing solutions, thus validating the significance of the generalized method.

1.3 Problem Delimitation

While design rationale is currently and could potentially be leveraged for a variety of purposes, this research is limited to demonstrating only that it can be used to improve collaboration. It is important to note that the tools created for this research are prototypes, not meant for distribution and thus lacking the overall depth and sophistication of enterprise software. For example the complexity and robustness of the inferences in the design rationale is limited to the most basic of assumptions. Also, the method has been tested only on NX 6 CAD running on a Microsoft Windows operating system with version 3.5 of the .NET platform. Within that CAD package only a restricted subset of the complete functionality is supported for any given tool. Of the three tools this limitation is most obvious in the Real-time Collaboration Tool as it supports a very limited subset of NX 6's functionality. While each of these tools provides an improved collaboration experience they may not be the best possible implementation of the tool but are rather an embodiment of the proposed method. Finally, it should be noted that these delimitations are not the result of limitations on the capabilities of purposed method and that future work could extend these implementations to produce production quality collaboration tools.

1.4 Document Organization

In the next chapter the reader will be familiarized with the relative prior research in the fields of design rationale and collaboration and how this research builds upon those efforts. Chapter three presents the generalized method for improving collaboration through the use of design rationale data. Chapter four describes how this method was used to create three different collaboration tools. The results of a series of tests that demonstrate that each of these tools provides important improvements upon the prior research are then presented in chapter five. Finally, chapter six details the conclusions that have been drawn from this research and also presents suggestions for future work.

2 BACKGROUND

This research builds upon the work of others who have pioneered and labored in the fields of design rationale and collaboration. A brief review of the relevant literature is presented to both lay a foundation of knowledge for the reader and also to acknowledge in small part the efforts of those upon whose shoulders my work stands. The review begins with a brief overview of the major research in collaborative design. This is followed by an explanation of the state of the art in design rationale, specifically relating to the: capturing, interpreting and utilizing of design rationale. At each of these steps particular emphasis is given to any prior research that bridges the fields of design rationale and collaboration.

2.1 Collaboration in an Engineering Context

Collaboration is not an activity that is by any means exclusive to engineering. However, collaboration in an engineering context does present unique challenges and thus has been the subject of extensive research. Qiu and Li provide an excellent overview of the key efforts in the field (Qiu and Li 2006). From their work and other survey papers it is evident that, despite the previous efforts of researchers, key shortcomings still exist in a number of the standard engineering collaboration methods and tools (Red, et al. 2010). From the all of these shortcomings a subset was selected to for in-depth analysis. The types of collaboration that are represented in the subset are:

- Asynchronous Collaboration
- Context Sensitive Collaboration
- Real-time Collaboration

This subset was chosen because it had the greatest potential to be improved upon by applying the method presented in this thesis. They also were chosen because they represent a broad range of collaboration scenarios which demonstrates the robustness of the proposed method. For each type of collaboration a tool was created that improves upon the existing state of the art for that tool. The key issues and relevant literature for each of the identified types of collaboration are presented below.

The first collaboration type in which major shortcomings were identified is asynchronous collaboration. Asynchronous collaboration is defined as collaboration that is capable of being performed at a different time and or different place (Edwards, et al. 1997). This is the opposite of synchronous collaboration where the collaboration occurs at the same time and often in the same place. Figure 2-1 is a graphical depiction of the differences between synchronous and asynchronous collaboration. On left are two common examples of synchronous collaboration: editing a part in real-time and verbal communication. On the right we can see two examples of asynchronous collaboration: editing a part and sending an email. In each of these cases the actions of the participants occurs at discrete times. As this figure depicts the difference between asynchronous and synchronous collaboration is a question of time and place.

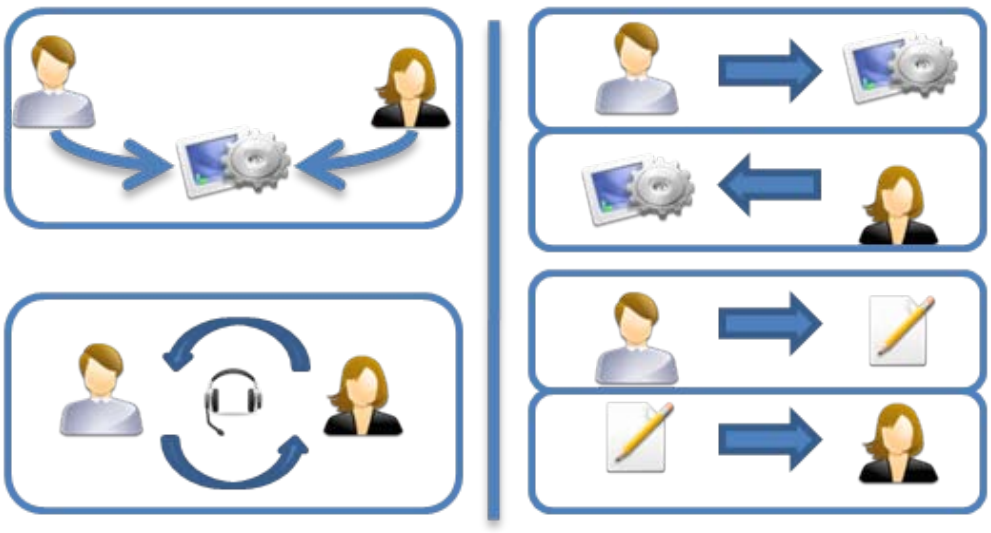

Figure 2-1 Visualization of synchronous (left) and asynchronous (right) collaboration

Currently, the most prevalent example of asynchronous collaboration is email. However, despite its ubiquity researchers have found email systems inadequate for a number of important collaborative tasks including visualization (Wattenberg and Viegas 2006). In fact, despite the obvious benefits of an asynchronous collaboration visualization system, as recently as 2007 only two such systems existed (Marchese and Brajkovska 2007). Neither of these systems was used for engineering specific collaboration. A clear opportunity exists for improving engineering collaboration through the development of an asynchronous collaboration visualization tool for CAD.

The next type of collaboration is context sensitive collaboration. Context sensitive collaboration occurs when the collaboration between multiple individuals is driven by the context of the work of one or more of the participants. A good example of this is when an engineer encounters a problem in the context of the part they are working on and so they ask a fellow engineer for help in solving the problem. While this seems basic, it is in fact the preferred method for getting help as research shows that 70% of all engineers queries for information are

answered by approaching another person where as only 30% are answered from information repositories like documents and drawings (Aurisicchio and Wallace 2004). When an engineering team is large and or spread out over a large distance this approach becomes significantly more challenging. While work has been done on solving this problem, the overwhelming majority of the research has focused on improving the communication of the query by bringing the quality of digital collaboration tools up to the standard set by of face to face collaboration (Fussell, et al. 2004). In contrast no research was found that focused on improving the ability of the engineer to identify the correct person with whom to direct their query in the first place. Improving the process of identifying peers capable of answering engineering queries is an exciting and unexplored area and is the objective of the SmartHelp Tool.

The final type of collaboration that will be explored is real-time collaboration or more specifically real-time multi-user CAD. The existing attempts at multi-user CAD can be divided into two groups: co-viewing packages that allow multiple users to simultaneously view a single model and co-design packages that allow multiple users to simultaneously edit a single model (Li, et al. 2005). A wide variety of architectures have been used to produce these systems including: thin client, client server, peer to peer, and application sharing (Wang, et al. 2007). All of these attempts focused either on developing an entirely new CAD package or on artistic NURBS based packages (Matviykiv, Lobur and Lebedeva 2007). Two of the more promising examples of NURBS based collaborative editors are CoMaya and the Verse Project for Blender (Agustina, et al. 2008) (Ebb n.d.).

Of the existing commercial products, Siemens NX currently offers best in class support for real-time collaboration. Inside of their NX product line multi-user capability is provided by an add-on that is known by the menu item name Collaborate. The related documentation is only

available to licensed NX users. We will refer to this capability as NX Collaborate and note that it uses the same architecture used by Qiang and his colleagues (Qiang, Zhang and Nee 2001). However, NX Collaborate has limited effectiveness as a multi-user MCAD tool because it is not user friendly (Sharma, Raja and Fernando 2006).

Despite the potential advantage of multi-user CAD none of the existing approaches has received widespread adoption in part as a result of the difficulty of real-time interaction over networks (Li, et al. 2005). The problem being in a co-design environment all of the actions performed by each user are shared with all of the other users participating in the session. The SimulPart tool was created to solve this problem by throttling the information that is broadcast to the multiple users who are simultaneously designing a single part file.

The three tools that were created for this thesis each solve a different type of collaboration problem. The solution to each of these problems represents an improvement over the state of the art as identified above. With an understanding of the key problems that exist in engineering collaboration an understanding of the value and contribution of the method presented in this thesis is now possible.

2.2 Capturing Design Rationale

Figure 2-3 shows a typical design process where the user performs a series of actions in a CAE tool as a result of an underlying motivation or intent. Often the motivation is the implementation of particular product design specifications, but it might be based on previous design experiences as well. The phrase design rationale has a variety of meanings and definitions and so an explanation of what design rationale means in the context of this research is necessary. For this research Moran and Carroll's definition is used which defines design rationale as a

combination of some or all of the specifications, actions and motivations that are used to create a design. To further clarify, the depth and complexity of the motivations and actions that comprise the design rationale vary across a broad spectrum. At one end you have the low level tool interactions that define why an engineer chooses or interacts with a tool in certain manner. At the other end you have the high level motivations for why a design was chosen which is driven by engineering experience and judgment as well as a host of other organizational, social, financial, political and cultural factors. This is visually depicted in Figure 2-2 along with examples of low and high level motivations, actions and specifications.

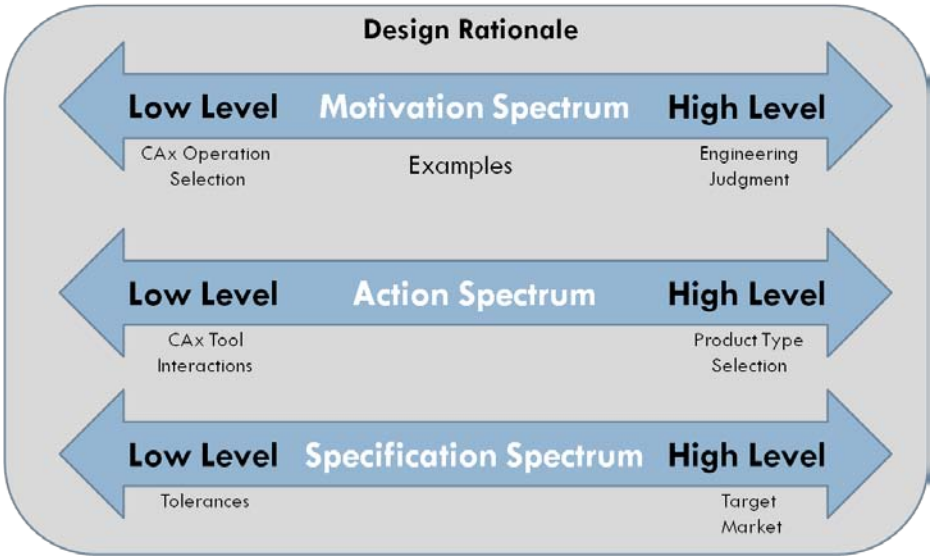

Figure 2-2 The entire spectrum of design rationale with examples

For the past 35 years extensive research has been done into determining the best way to capture and leverage the entire spectrum of design rationale. The motivation for this extensive research comes from the inability of existing data management tools to capture the underlying

intent and logical rationale that occurs throughout a design project (Klein, Capturing Geometry Rationale for Collaborative Design 1997).

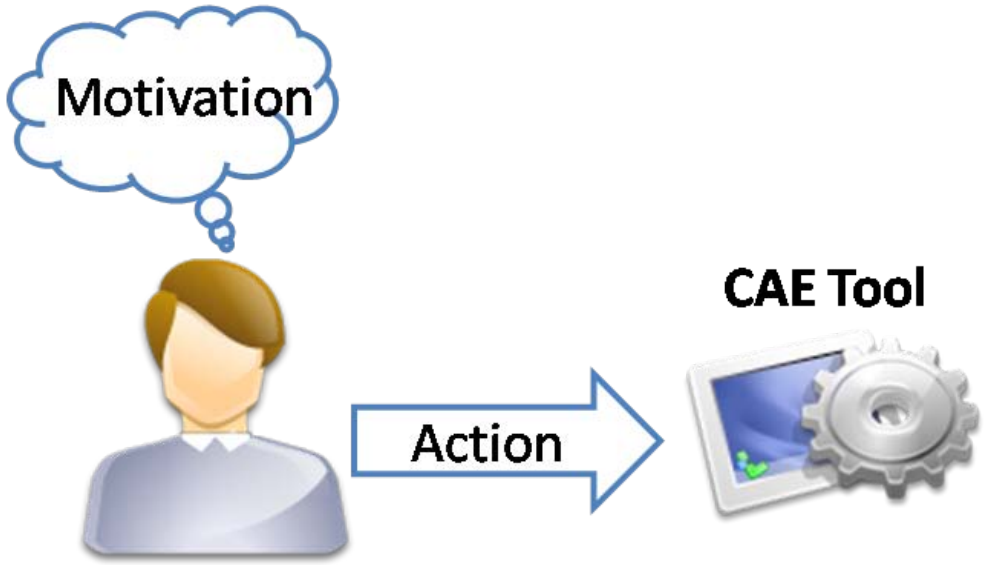

Figure 2-3 The difference between actions and motivations in the design process

Most of the research in the field of design rationale has focused on either providing a framework for representing or a tool for entering design rationale data (Zdrahal, et al. 2007). Illustrative examples of the evolution of design rationale capture tools can be seen in the development of gIBIS (Conklin and Begeman 1987) and Compendium (Conklin, et al. 2001). Figure 2-4 shows how most popular design rationale tools, including gIBIS and Compendium, are integrated into a designer's workflow. This architecture requires the user to manually record the actions they took and the motivations behind these actions.

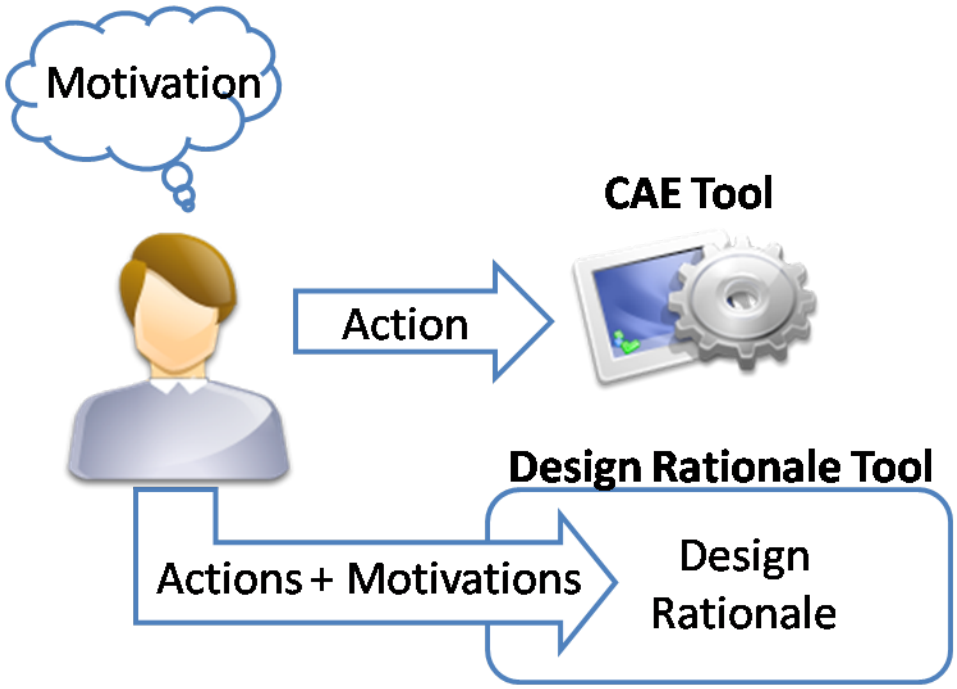

Figure 2-4 The popular approach for capturing design rationale

While research has shown that the traditional methods for design rationale capture provide some benefits they have all failed to receive wide spread acceptance for a variety of reasons including: not fitting into the engineer's existing work process and methods (Bracewell, et al. 2008), the process not being clear, and a reluctance to document mistakes and failures (Rea, et al. 2007) (Rea, et al. 2007). As a result engineers typically don't record the rationale for their decisions. One of the more recent solutions to this issue is to automatically record a portion of the design rationale and then allow the designer to augment or fix the automatically recorded design rationale. This works by recording a designer's actions with the CAE tool and using this information to infer the intent portion of the design rationale. This method was first demonstrated by the Rationale Construction Framework (RCF) designed at SRI International (Myers, Zumel and Garcia 1999). This modified approach allows the designer to continue to use the workflow presented in Figure 2-3 by integrating a computerized workflow shown in Figure

2-5. Capturing the design rationale in a way that is both thorough and yet unobtrusive to the designers workflow is essential for any design rationale tool.

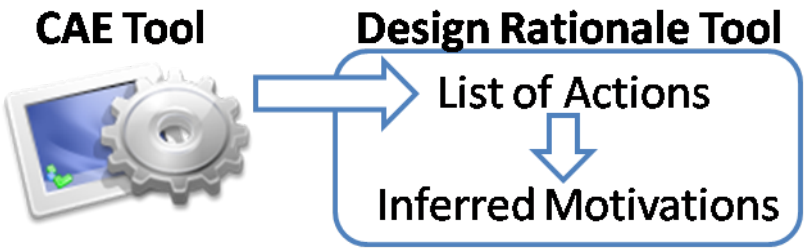

Figure 2-5 The passive design rationale capture architecture

This thesis seeks to build upon the approach pioneered at SRI of passively capturing a portion of the design rationale and then allowing the engineer to refine and augment it. However this approach is not without its limitations. It is only capable of automatically inferring a subset of the complete design rationale. It is also limited to recording design rationale during the detailed design and not the conceptual design phase of the design process. Finally, it is only capable of inferring low-level motivations such as why a designer interacts with a tool in a certain manner. For this reasons it is used in conjunction with an interactive methods for design rationale capture thus allowing the engineer to broaden the spectrum of the captured design rationale and to correct any incorrect inferences. While SRI's approach still requires manual entry it improves upon the completely manual approach since it allows for a more complete set of design rationale to be captured as it automates the capture of the more tedious portions of the design rationale (Rea, et al. 2007) (Rea, et al. 2007). For this reason this thesis will build upon the SRI approach for design rationale capture.

2.3 Interpreting Design Rationale

The primary research performed for this thesis requires inferences to be made about a user's intent based upon their interactions with a CAD file. Multiple methods exist for making inferences from both general engineering data (Johannesmeyer, Seborg and Singhal 2002) (Albazzaz and Wang 2006) and specific information, like CAD event data (Jin and Ishino 2006). These methods work by searching for patterns in the design data (Myers, Zumel and Garcia 1999). The patterns can be identified from design data in batches or in real-time as the user is performing the action. An example of method for real-time inference is WAVE, a fully automatic algorithm for real-time information extraction (Aseltine 1999).

Since a wide variety of methods and techniques exist for automatically inferring design rationale, examples are provided from the literature to give the reader a sense of the type of design rationale that can currently be inferred. From SRI's pioneering work two examples are detailed. The first is the *Refinement* inference, which occurs when a user goes back a refines a particular aspect of a model. The second inference function is *1:1 Part Substitution* which occurs when a user swaps out one part file for another (Myers, Zumel and Garcia 1999). By analyzing the parametric iterations an engineer makes on a design due to analysis results the constraints and design rules (Rawson and Stahovich 2009). Jin and Ishino have published a variety of papers on different methods for inference of design rationale and therefore provide a number of excellent examples. In one of their first papers they are able to infer the preferred order in which certain design decisions should take place by looking at the frequency and location of different alterations (Ishino and Jin 2002). In another paper published around the same time they describe the Multiple Genetic Programming (MGP) technique which can identify the tradeoffs made during a design (Ishino and Jin, Estimate design intent: a multiple genetic programming and

multivariate analysis based approach 2002). Later they created an improved method: Value based Mining for Sequential Pattern (VMSP), which allows for the identification of key operations in the creation of a design (Ishino and Jin, An information value based approach to design procedure capture 2006). The final example comes from Iyer, who illustrated how material properties could be used to infer the objectives of the design (Iyer 2007) (Iyer 2007) (Iyer 2007). A variety of examples has been presented; however the method presented in this thesis does not require any specific method or approach for generating inferences but rather is capable of using any method including those listed above.

2.4 Utilizing Design Rationale

Virtually all of the design rationale frameworks and tools that have been designed provide at least some benefits to the end user. The benefits of design rationale systems are well documented and best summarized by Lee as support for: reuse, redesign, extension, maintenance, learning, documentation, management and collaboration of projects (Lee 1997). With regards to collaboration, significant research has gone into capturing the design rationale from collaboration but little work has been done on actually leveraging the design rationale to improve collaboration (Burge and Kiper 2008). This thesis attempts to change this by demonstrating that serious benefits can be gained by leveraging design rationale to improve collaboration.

2.5 Foundational Tools

In developing and implementing this method a large number and variety of tools and techniques were used. A few of these tools and techniques proved indispensible and could be

considered foundation tools. A brief overview of these foundational tools is presented in the following sections.

2.5.1 Foundational Tools: C#

The primary programming language used to create each of the implementations of the method was Microsoft's C# programming language. The C# language was chosen because of its reputation for being a simple, modern, object-oriented approach to creating robust and durable applications (Hejlsberg, Wiltamuth and Golde 2006). It relies upon the Microsoft .NET infrastructure which provides tight integration with the Microsoft Windows operating system (Archer 2001). In the implementation one of the most utilized features was LINQ, a "thoroughly integrated and tremendously powerful" technology for database integration (Kimmel 2009). By leveraging the powerful architecture and integrated features of the C# language significant savings were gained in the time to create implementations of the method. This improved the overall method by allowing for more time to be spent on crafting and refining the method and less on programming the implementation.

2.5.2 Foundational Tools: API Programming

Developing and refining this thesis would not have been possible without the ability to deeply integrate with critical software packages through publically exposed functions. When a software package provides programmatic linkage to their functionality it is known as an Application Programming Interface or API (PC Magazine n.d.). The Siemens' NX Open API provides 3rd party developers with programmatic access to much of NX's functionality (Siemens PLM Software 2007). The C# version of the NX Open API was used in developing this thesis. This

allowed for the creation of powerful and relevant tools that tie directly into an engineer's existing CAD workflow.

2.5.3 Foundational Tools: Relational Databases

A variety of methods and tools exist for storing and retrieving digital data. One of the most powerful and popular of those methods is the Relational Database Management System (RDMS) which was invented at IBM by Dr. E. F. Codd (Powell 2006). The power of the RDMS lies in its combination of simplicity and reliability (Rob and Coronel 1995). SQL is the most common language through which you can interact with an RDMS (Connolly and Begg 2010). Microsoft's SQL Server was the RDMS used in each of the implementations of the method due to the excellent integration that exists between both SQL Server and C#.

3 METHODS

This chapter describes the generalized method for improving collaboration through the use of design rationale. The algorithms presented in this chapter are both verbally explained and visually presented using set notation. Additionally, to provide the necessary context and illustrations a series of examples have been included that reduce the generalized method to practice. These examples use NX 6 as a CAD package but are for illustration purposes only and the reader should refer to Chapter 4 for any details on the actual implementation of the method.

3.1 Overall Method for Collaboration

The proposed method for improving collaboration through the use of design rationale is comprised of two main operations each consisting of a series of steps. This process is presented in its most general form:

1. Capture and record the design rationale
    1.1 Record the user's process for creating their design in an unobtrusive manner
    1.2 Infer and record the design rationale using the process data recorded in the previous step and input from the user.
2. Retrieve and filter a dataset using the associated design rationale
    2.1 Retrieve the relevant dataset and any associated design rationale
    2.2 Filter the dataset using the relevant design rationale

At first glance the method may appear overly simple and lead the reader to question whether such a method is capable of the performance gains described in the objectives section of this thesis. Upon closer inspection, the reader will note that each of the operations listed above lends itself to some degree of computerized automation. In fact arguably the greatest strength of the computer, to store and process large amounts of data, can easily be applied to each of the steps above. For this reason once implemented the simple steps described below are capable of producing powerful tools for engineering collaboration.

3.2 Operation 1: Capture and Record the Design Rationale

As defined earlier design rationale is a set of information including some or all of the motivations, actions and specifications that go into a particular design (Moran and Carroll 1996). The first operation of the method is concerned with capturing and recording this information. The literature suggests that the key to capturing design rationale is to be as unobtrusive as possible upon the engineers existing workflow (Bracewell, et al. 2008). This restriction was a driving principle in the design of the both of the steps used to record the design rationale.

3.2.1 Step 1.1: Record the Design Process Data in an Unobtrusive Manner

The first step in recording the design rationale is to record the design process data. The design process data consists of all of the actions a user takes while interacting with a part file and all of the metadata corresponding to these actions. This metadata could include feature information, user information, mouse actions, screen shots, audio output, command files, log files, undo files, and written notes. For this thesis the combination of action and corresponding metadata is

referred to as an action object. Thus the set of all action objects for a user could be denoted as A or:

$$\{A: \text{set of all action objects}\} = \{A: A_1, A_2 \ldots A_n\} \quad (1)$$

An example of what an action object might look like is shown in Figure 3-1. In this example User1 has performed an extrude operation. The figure depicts both the recording of the operation and the recording of a corresponding set of metadata.

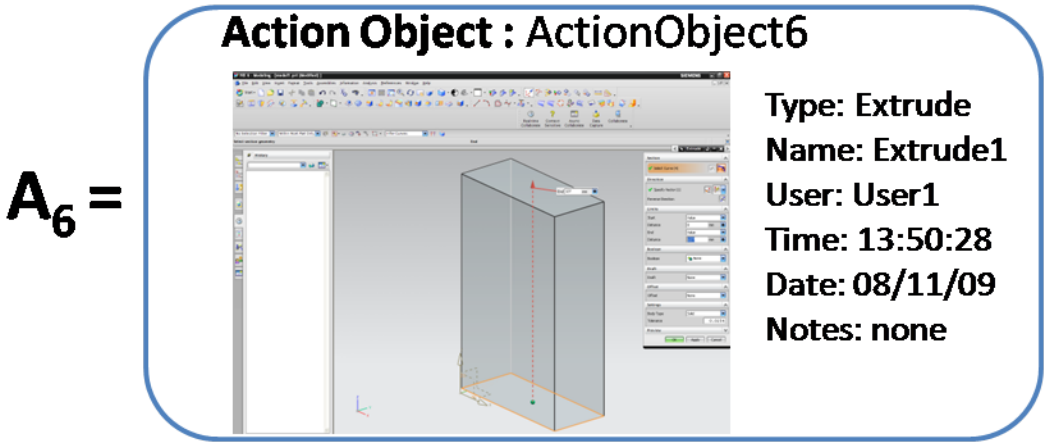

Figure 3-1 A graphical example of an action object

Continuing this example we can see in a generalized depiction of what the set A might look like after User1 has performed a series of actions to create a simple rectangular block using the NX 6 CAD program with its associated terminology. First the user created a sketch, rotated the model, and attempted to create the block by using the "Swept" command. However, after seeing the results of the "swept" command the user deleted the sweep, rotated the view again, and instead performed an "Extrude" command. For every action an action object was created to which a set of metadata was associated as can be seen in Table 3.1.

Table 3.1 The set of action objects used to create a rectangular block

A =

Action Object	Type	Name	Target
ActionObject1	Sketch	Sketch(1)	Model1
ActionObject2	Rotate		Model1
ActionObject3	Sweep	Swept(1)	Model1
ActionObject4	Delete		Swept(1)
ActionObject5	Rotate		Model1
ActionObject6	Extrude	Extrude_1	Model1

For the method to function effectively these actions objects must be recorded passively. The user's normal workflow must not be explicitly interrupted while the action objects are being recorded. The method does not however prescribe either the format or the structure in which the action objects are stored. As will be presented in Chapter 4 the use of an RDBMS like SQL Server is an excellent option for data storage. However it is by no means required. As long as this step is performed passively the exact technologies used are not relevant to the end objective of improving collaboration through the use of design rationale.

3.2.2 Step 1.2: Draw Inferences About Design Rationale

At this point in the process a set of action objects should exist that correspond to the actions a user has taken while interacting with a CAD file. As described earlier design rationale is a combination of some or all of the motivations, actions and specifications that occur while creating a design (Moran and Carroll 1996). Thus the next step is to use the existing action objects to infer the motivations that prompted them and then if needed allow the user to extend and correct these inferences. This method was first demonstrated by SRI as a viable alternative to

interrupting the user's workflow by requiring them to explicitly record all of their motivations (Myers, Zumel and Garcia 1999).

The proposed method takes a set of functions and compares them against a set of action objects to identify patterns. This is the same method used by the majority of the works described in the previous chapter. These inference functions then return a list of inferences using a set of designer's actions. To clarify, let F equal the set of inference functions:

$$\{F: \text{set of inference functions}\} = \{F: f_1(x), f_2(x) \ldots f_n(x)\} \quad (2)$$

These functions could include any of the methods described in Section 2.4 such as MGP or VMSP. Now let I_1 equal the set inferences that have been drawn by running the f_1 function on the set of actions objects A.

$$f_1(A) = I_1 \quad (3)$$

Thus M is equal to the set of all inferences that have been drawn such that:

$$\{M: I_1, I_2 \ldots I_n\} \quad (4)$$

Using the example from the previous step, A equals the set of actions objects depicted in Table 3.1. In this case the entire set of action objects from the design session is being analyzed though this is not necessary as smaller subsets could be analyzed. So let $f_1(x)$ be the inference function corresponding to the "CAD operation mistake" motivation. That is to say that the subset of action objects that the $f_1(x)$ function returns represents actions that are inferred as mistakes. So in this simple example the implementation of $f_1(x)$ can be thought of as looking for the pattern of an action being deleted or undone immediately after it was performed. If the pattern is found, the initial action is returned in the subset of "CAD operation mistake" action objects. Therefore the results of performing the $f_1(x)$ function on the set of actions objects A would be:

$$I_1 = f_1(A) = \{\text{ActionObject3}\} \quad (5)$$

Therefore, the set of all design rationale DR is equal to the combination of motivations and actions or the combination of M and A represented as:

$$\{DR: M \cap A\} \quad (6)$$

Once these inferences have been made they can now be extended or corrected by the user. One example of this interaction might be the recording of voice notes while the user is in a modeling session.

Now that a set of design rationale has been created it must be recorded. In the implementation described in Chapter 4 this is done using the same database where the action objects were recorded. However, this is not a necessity. They could be recorded in any format that facilitates automation. The power of this method comes from the fact that it is highly computerized and a computer is capable of quickly calculating inferences using a large set of actions objects and a complicated set of inference functions. This allows for a larger quantity of data to be processed and ideally a higher number of inferences to be drawn than would be possible with exclusively manual methods.

3.3 Operation 2: Retrieve and Filter a Dataset Using the Associated Design Rationale

The second operation of the method is to retrieve and utilize the design rationale information that was stored in the first operation to improve collaboration. This is done by using the design rationale as a filter on a set of collaborative data. It is important to note that operations one and two (recording and retrieving the design rationale) do not need to be performed simultaneously. In fact two of the three implementations have an indeterminate break between the first and second operations.

The collaborative problems that design rationale can help solve and the ways in which those problems are resolved varies significantly from problem to problem. However, the method used to solve any type of problem can be generalized to always rely upon the same two steps:

1. Retrieve the dataset and any relevant design rationale
2. Filter the dataset using the relevant design rationale

Once automated these simple steps create the second operation of this method for improving collaboration.

3.3.1 Step 2.1: Retrieve the Relevant Dataset and any Associated Design Rationale

As a result of Operation 1 the design rationale should have been captured and stored. The next step is to use a retrieval algorithm to retrieve a specific dataset and the associated subset of design rationale. The precise algorithm for data retrieval can and does vary from implementation to implementation. Regardless of the algorithm the key principle is to identify a relevant dataset and it corresponding subset of the captured design rationale that can benefit a specific collaborative activity.

Using set notation the set of all design rationale DR, which is a combination of all of the action objects and motivations both inferred and explicitly stated, is written as:

$$\{DR: \text{set of all design rationale}\} = \{DR: M \cap A\} \quad (7)$$

From within DR the subset DR_1 is sought. DR_1 is the subset that contains the relevant design rationale necessary to perform the final step of Operation 2.

Using this notation to continue the example from the first operation, let us suppose that we are trying to get a specific subset that will be used to solve one of the key problems described in the literature review: that of helping designers find qualified peers to answer their questions. Our first step is to define an algorithm capable of returning the desired subset of data and any

associated design rationale. In this case we are looking for a dataset of actions objects which can be extracted from the entire set of design rationale DR. So we can use an algorithm $g_1(DR, O)$ that returns both a list of all of the action objects that correspond to an operation O and any additional design rationale that has been associated with these action objects. If we apply this algorithm to the set DR defined in section 3.2 and set O equal to the "Extrude" operation we get the following results:

$$g_1(DR, \text{“Extrude”}) = DR_1 = \{\text{ActionObject6}\}. \quad (8)$$

However, if we run the same algorithm this time setting O equal to "Sweep" we get a result set that includes both an action object and additional design rationale in the form of a motivation inference:

$$g_1(DR, \text{“Sweep”}) = DR_1 = \{\text{ActionObject3, Inference1}\}. \quad (9)$$

A graphical depiction of these results is shown in Figure 3-2 . Once the relevant dataset and design rationale has been retrieved it is now ready to be leveraged to improve collaboration.

DR =

Object	Type	Target	User
ActionObject1	Sketch	Model1	User1
ActionObject2	Rotate	Model1	User1
ActionObject3	Sweep	Model1	User1
ActionObject4	Delete	Swept(1)	User1
ActionObject5	Rotate	Model1	User1
ActionObject6	Extrude	Model1	User1
Inference1	Mistake	ActionObject3	

g_1(DR, "Extrude")

Object	Type	Target	User
ActionObject6	Extrude	Model1	User1

g_1(DR, "Sweep")

Object	Type	Target	User
ActionObject3	Sweep	Model1	User1
Inference1	Mistake	ActionObject3	

Figure 3-2 The relevant subset of action objects and inferences for a given operation

3.3.2 Step 2.2: Filter the Dataset Using the Relevant Design Rationale

The relevant design rationale subset that has been retrieved can now be used to filter that dataset that will drive the collaborative actions. In each application the information being filtered will vary but may include the design rationale itself or separate set of information or a series of actions. Regardless of how it is used the possibilities are endless and depend upon the creativity and ingenuity of the person implementing the method. This is demonstrated below by the termination of the example presented above.

To finish the example from the previous steps we now must take the design rationale subset and use it to filter the dataset of action objects in a way that improves collaboration. As stated previously our objective is to help the user determine which engineers might be qualified to answer a question about a specific CAD operation. In this case we use an algorithm $h_1(DR, O)$ that returns a list of all of the users who have successfully completed an operation O. This works by taking DR_1, which is a combination of the dataset of action objects and their associated design rationale, and filtering out any "unsuccessful" action objects based on the associated design rationale. So if we apply this algorithm to the subset DR_1 defined in section 3.2 and set O equal to the "Extrude" operation we get the following results:

$$h_1(DR_1, \text{“Extrude”}) = \{\text{User 1}\}. \quad (10)$$

However, if we run the same algorithm this time setting O equal to "Sweep" we get an empty set:

$$h_1(DR_1, \text{“Sweep”}) = \{\}. \quad (11)$$

So while the user performed both actions, as can be seen in , only the "Extrude" is considered a "successful" action. The reason for this comes from Inference1: that the "Sweep" action represented by ActionObject3 was in fact a "CAD operation mistake" and was thus not

counted as a successful operation resulting in the empty set. The formatted results of these queries can be seen in Figure 3-3. On the left of the figure is the result of the user searching for an engineer capable of helping with the “Sweep” operation. The right side shows the results when help with the “Extrude” operation is sought.

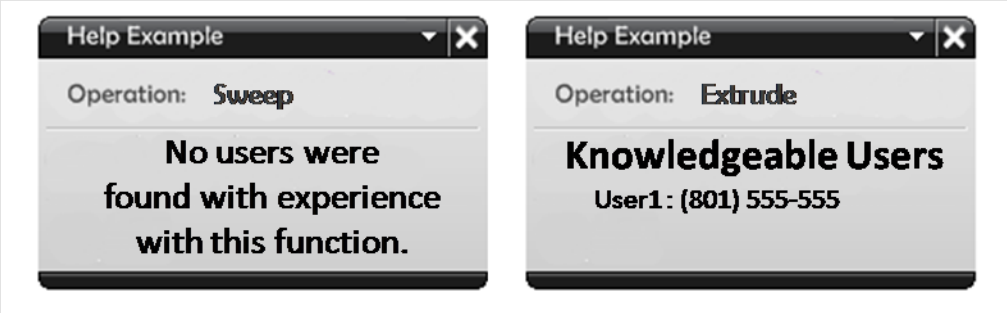

Figure 3-3 Query results for the sweep (left) and extrude (right) use cases

The example presented above highlights the paradoxical but powerful nature of the method. At first glance it appears that the problem this example solves would be better solved by the user just querying her colleagues using the good old fashion face to face method. Is using a computerized method really necessary? Yes. Once you move beyond a few users the ability for a computer to easily record and recall large amounts of data becomes invaluable. Using the method presented above the user could pull from the expertise of thousands or even tens of thousands of her colleagues, something unimaginable with the simple face to face method. By leveraging design rationale to improve collaboration using the method presented above significant productivity gains become possible.

4 IMPLEMENTATION

This chapter presents how the methodology described in Chapter 3 was used to create a series of collaboration tools based upon the objectives and delimitations outlined in the first chapter. The end result of these efforts was the NXConnect collaboration tool suite. NXConnect is a plug-in for the NX 6 CAD package that provides three key functionalities: it enables concurrent CAD modeling, it allows engineers to discover which of their peers have experience with a specific CAD operation, and it displays a visual history of a CAD file. While these tools do not show the sophistication or robustness of a commercial solution they do demonstrate the usefulness and capability of the proposed method. The details of how these tools were implemented are presented below beginning with an overview of the NXConnect architecture.

4.1 NXConnect Architecture Overview

The NXConnect architecture employs a multitier approach to implement each of the three collaborative tools. This multitier approach is presented visually in Figure 4-1. From this figure it becomes clear that NXConnect relies upon four custom parts or modules each of which fulfills a different primary function. Of the four modules three are common to each of the tools within NXConnect. These shared modules are differentiated in Figure 4-1 by the use of a different color background. A common CAx tool, NX6, was also used for each of the implementations. The following sections are devoted to first giving an explanation of how each

of the modules was implemented followed by a detailed explanation of the SimulPart, VisiHistory, and SmartHelp implementations of the proposed method.

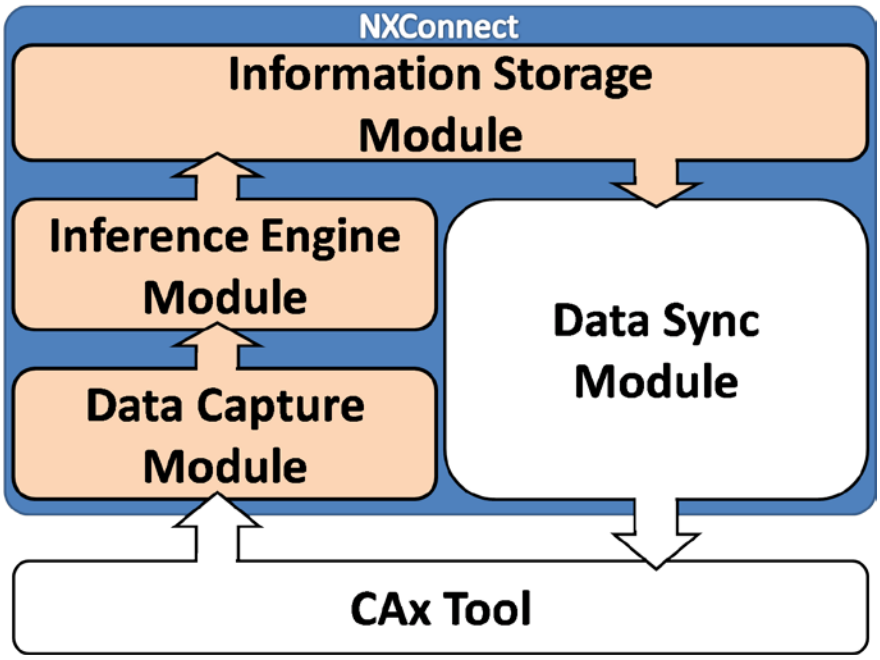

Figure 4-1 The four part multitier NXConnect architecture

4.2 Data Capture Module

Step 1.1 of the methodology presented in Chapter 3 is to record the design process data in an unobtrusive manner. This step is implemented with the Data Capture Module (DCM). The DCM's primary responsibility is to unobtrusively capture the process the user is performing while interacting with the NX 6 CAD package. The information that the DCM gathers includes: feature information, audio annotations, user information, mouse actions, screen shots, and undo files. To accomplish this task a special class was created within the C# programming language that was capable of interacting with NX 6 through the NXOpen API. Using API commands the status of the part could be retrieved at any given time.

An example of the types of methods that were created within the DCM can be illustrated by two methods, one for capturing the current feature and another for capturing a screenshot of the design session. The first method getCurrentFeature( ) uses the NXOpen API to extract feature information from the model. The second method getCurScreenShot( ) might be used to capture a screenshot of the action that a user is performing. This method is an example of how the .NET API was leveraged through the use of the C# programming language.

The exact timing of when information is captured is determined using an adjustable timer. This was done for convenience as well a performance. The more heavily the NXOpen API is used the less responsive NX becomes to the user. Therefore a balance must be struck between performance and synchronization delays. By implementing Step 1.1 of the methodology presented in Section 3 the DCM of the SimulPart tool accomplishes the task of unobtrusively gathering design process data.

4.3 Inference Engine Module

Once the design process data has been captured by the DCM it is the responsibility of the Inference Engine Module (IEM) to then draw inferences from that data. This corresponds to Step 1.2 of the generalized method for improving collaborative design environments through the use of design rationale. Using C# the IEM was created using the architecture outlined in Figure 4-1. As can be inferred from the figure the IEM takes the design process data captured by the DCM and uses it to draw inferences. These inferences are then stored in the Information Storage Module for future use. This is done by using the methodology presented in Section 3.4. An example of how one of the algorithms which would be run against the set of data objects captured by the DCM is provided next. The getVisActionsInfers (List Actions) method parses a

list of actions looking for actions that are contained within the list of pre-designated visual actions. Once these visual actions are identified they are returned by the function. This is an example of one of the many inference algorithms that was implemented using this method.

4.4 Information Storage Module

Once the action objects and inferences have been made they must be stored away to be used at a later point by the second operation of the proposed method. The storage and retrieval of both the action objects and any inferences is done in the Information Storage Module (ISM). The primary qualification for designing an ISM is the ability to quickly and efficiently store and recall small and large amounts of data.

The ISM for SimulPart was implemented using Microsoft SQL Server, an enterprise class RDMS. A table was created for both the action objects as well as the inferences. Additional tables were also created to manage the session and user information. Each of these tables is accessible through the SQL language. While any one of a number of data storage approaches could have been used to create the ISM, MS SQL Server allowed for the creation of a quick and efficient data solution capable of delivering a near real-time experience to a multitude of users.

4.5 Data Sync Module

According to the method presented in Chapter 3 once the data has been stored a subset of the data that includes any associated design rationale must be retrieved and with the design rationale being used to filter the corresponding dataset. In NXConnect this process is performed by the Data Sync Module (DSM). The DSM performs two principle actions which correspond to

both Step 2.1 and Step 2.2 of the method, namely retrieving the dataset with any relevant design rationale and using it to filter the different sets of information. For each of the implementations the DSM accomplished the two principle actions but did so in a different way. As a result the precise manner in which the DSM was implemented is deferred to the section in which each implementation is explained.

4.6 SmartHelp Implementation

The SmartHelp tool was created to allow designers to identify peers in their organization who have experience using a certain functionality of the NX6 CAD tool. For example a designer may be unsure of how to use the Ruled Curve function and wish to discuss the function with a colleague skilled in using the function. The problem becomes identifying which colleagues have experience with a specific function. To help solve this problem a custom GUI was created where users could select from a list of functions and see which users in their organization were most likely to be able assist them with that particular function. Each colleague is given a potential helpfulness score for each function which is calculated by summing the total number of times a user successfully performed a specific function. A screenshot of the GUI can be seen in Figure 4-2. In this case the user has queried the system to determine which users have successfully used the Sketch feature of NX. A list of 3 users is returned which are ranked from most likely to least likely to be of assistance. For a more detailed example of how the SmartHelp tool implements the proposed method the reader is referred to the example that is given to illustrate the method as presented in Chapter 3.

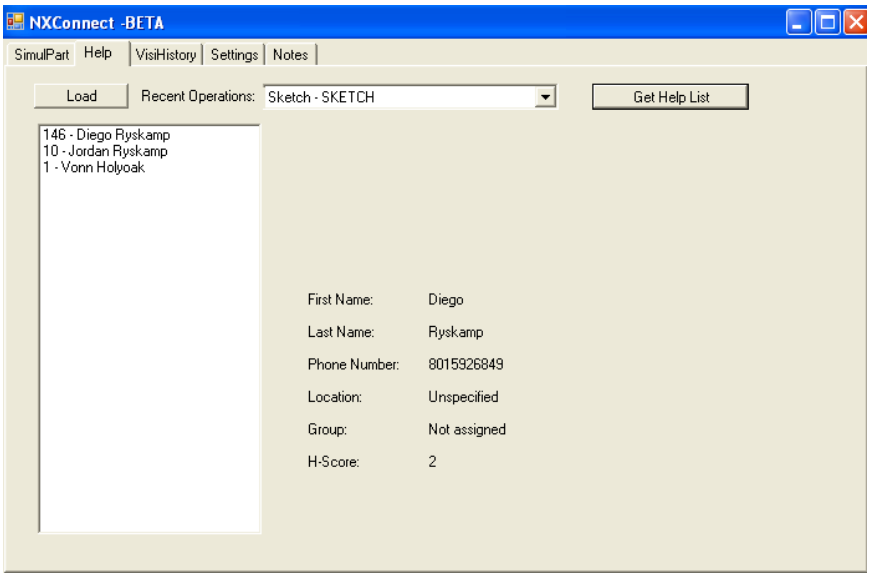

Figure 4-2 A screenshot of the SmartHelp graphical user interface

4.6.1 SmartHelp Data Sync Module Implementation

The SmartHelp DSM is tasked with pulling down from the ISM a list of the relevant users who have experience with the specific function and ranking them. To do this inferences made by the IEM about whether a function was performed successfully are used to eliminate any actions that weren't successful. This is necessary because many times a user will use a specific function only to delete it and attempt to use the function again or even use a different function. If these attempts were included in the rankings they might skew the data so they are eliminated. Once the filtered result set has been created by the DSM the users are ranked by the number of times they have performed an action with the assumption being that a user who has successfully performed an action many times is more likely to have mastered it than one who only performed the action a few times. The user is then able to select from the list and see contact information for the select user so they might contact the user and get assistance with the function they are struggling with.

4.7 SimulPart Implementation

The SimulPart tool is the most complex and arguably impressive tool of the NXConnect suite. The tool uses the proposed method to intelligently limit the amount of information shared with each of the users in a multi-user co-design session. In doing so it demonstrates that complex MCAD files can be simultaneously edited by a multitude of users. The result is the creation of the first ever commercial MCAD real-time co-design environment. To give the reader a better idea of how this is accomplished the reader is presented with a description of both how the DSM was uniquely implemented for SimulPart and an example of how SimulPart functions.

4.7.1 SimulPart Data Sync Module Implementation

The first principal responsibility of the SimulPart DSM is to retrieve the relevant design rationale. More specifically it is to keep the part synched up with the relevant actions in the database at all times. The relevancy of the actions is determined by using the inferences that were created by the IEM. The timing of how frequently the database is queried was determined by using a timer that was set to a specific wait time of $1/10^{th}$ of a second. As this can present a performance bottleneck a dynamic timer that adjusts based upon the network load conditions would be optimal. However, this was not necessary for the SimulPart implementation and thus a fixed time was utilized.

The second key function the DSM provides is to update the part file to reflect the relevant changes that have been performed by other users. This was done using a series of custom C# methods that relied heavily upon the NX Open .NET API. These methods take the

relevant action object and update the NX part file to reflect the creation of the action object on a different user's computer.

4.7.2 SimulPart Example

To illustrate how the SimulPart tool works an example is provided. This example details the implementation of each of the steps of the proposed method outlined in Chapter 3. Where the SimulPart example differs significantly from the example given in Chapter 3 extra detail is given. In this example we are trying to get a specific subset that will be used to solve the network bandwidth and latency problems that plague multi-user CAD. In any co-design session multiple users are simultaneously performing a large number of actions which must be broadcast to each of the users in the session. This example shows how design rationale can be used to intelligently filter the amount of information that is shared in the session by identifying which operations are most important and should be immediately pulled from the ISM and reflected in the part being shared in the co-design session.

For our example we have two users, User1 and User2, who are participating in a co-design session using the NX Connect SimulPart module. As each user independently interacts with the same part on their separate computers any changes are synchronized between each of the users in the session. In our example the users are creating a rectangular block. As they create the block the design rationale is recorded as action objects for each user. The set of all action objects for both User1 and User2 is denoted as A or:

$$\{A: \text{set of all action objects}\} = \{A: A_1, A_2 \ldots A_6\}. \quad (12)$$

This set is visually depicted in here as Figure 4-3.

A =

Action Object	Type	Name	User	Target	Timestamp
ActionObject1	Sketch	Sketch(1)	User1	Model1	1/15/2010 3:48:24 PM
ActionObject2	Rotate		User1	Model1	1/15/2010 3:48:26 PM
ActionObject3	Sweep	Swept(1)	User2	Model1	1/15/2010 3:48:29 PM
ActionObject4	Delete		User2	Swept(1)	1/15/2010 3:48:30 PM
ActionObject5	Rotate		User1	Model1	1/15/2010 3:48:40 PM
ActionObject6	Extrude	Extrude(1)	User1	Model1	1/15/2010 3:48:42 PM

Figure 4-3 A set of action objects with corresponding user and timestamp information

In this example the capturing of these actions objects is still automated and performed passively by the Data Capture Module so as not to interrupt the user's workflow.

The next step is to take the action objects and use them to infer the motivations that prompted them. This is done by the Inference Engine Module and is performed by taking a set of inference functions and comparing them against the set of action objects. The inference functions then return a list of inferences that correspond to those actions. Let us say that $f_1(x)$ is the inference function corresponding to the "Visual Adjustment" motivation. That is to say that the subset of action objects that the $f_1(x)$ function returns represents actions that are inferred as adjustments to the user's view only. So in this simple example the implementation of $f_1(x)$ can be thought of as looking through each action and determining if the action is a visual adjustment such as rotate, zoom etc. Therefore the results of performing the $f_1(x)$ function on the set of actions objects A would be:

$$I_1 = f_1(A) = \{\text{ActionObject2, ActionObject5}\}. \quad (13)$$

For both ActionObject2 and ActionObject5 a "Visual Adjustment" inference would be created which would be named Inference1 and Inference2 respectively. These inferences would then be merged with any manually inputted design rationale. For example if the user was recording a voice note during the visual adjustment then the "Visual Adjustment" inference would be

adjusted to reflect the additional information gathered in the voice note. This design rationale would then be recorded into the Information Storage Module which in this case is an SQL Server RDMS.

At this point the design rationale has been captured by the DCM, interpreted by the IEM, and stored in the ISM. The next step is to use a retrieval algorithm to retrieve a specific subset of the inferred design rationale. In our example the retrieval algorithm is denoted as $g_1(DR, T)$ and returns the subset of the design rationale data with a timestamp greater than time T and any inferences that have been made on these action objects. Let us suppose that it is 1/15/2010 3:48:43 PM and User2's part was last synchronized at 1/15/2010 3:48:39 PM. So we run the $g_1(DR, T)$ setting T equal to the date and time we last synchronized which returns the following results:

$$g_1(DR, \text{“3:48:39 PM”}) = DR_1 = \{\text{ActionObject5, ActionObject6, Inference2}\}. \quad (14)$$

This is done at the ISM using a simple select where SQL statement. Now that the relevant subset has been identified at the ISM we are ready to move onto the next step of the method.

The final step is to use the returned subset DR_1 to limit the amount of information that is transferred by filtering out irrelevant information. This can be thought of as performing the function $h_1(DR)$ which returns a list of all of the action objects in the given set which are not just visual adjustments. If we apply this algorithm to the subset DR_1 defined above we get the following results:

$$h1(DR_1) = \{\text{ActionObject6}\}. \quad (15)$$

Notice that ActionObject5 was filtered out as a result of Inference2. This was done using a nested SQL "*select where*" statement with the limiting criteria being the action object not having an associated "Visual Adjustment" inference. By using a nested statement the ISM is

able to perform both Step 2.1 and Step 2.2 of the method simultaneously. So while User1 performed two actions, as can be seen in Figure 4-3, the “Rotate” action is not considered relevant to User2 because it was just a “Visual Adjustment.” If however the user had added a voice note the “Visual Adjustment” motivation would have been adjusted and ActionObject5 would not have been filtered since the additional manually added design rationale would have identified it as of importance.

ActionObject6 would now be used to update User2’s CAD part to reflect the actions taken by User1 since the last synchronization. The update process can be seen graphically in Figure 4-4. As this figure shows only half of the information that is generated by User1 is actually transferred to User2. The rest is filtered out as a result of the proposed method. While this example was very simple with only a few users and actions once you move beyond a few users the ability to minimize the amount of information traveling between users becomes invaluable. Using this methodology a major bottleneck in collaborative design environments is overcome by intelligently reducing the amount of information required to sync multiple models into a single session.

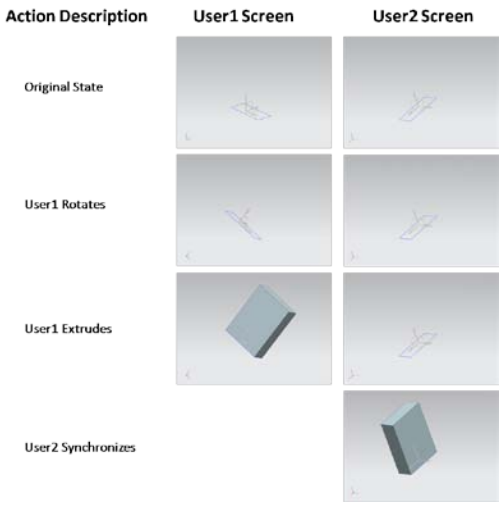

Figure 4-4 Graphical depiction of the screens of each user as the actions are performed.

4.8 VisiHistory Implementation

The VisiHistory tool solves many of the issues that occur when two or more users asynchronously collaborate on a single part file which include duplicated efforts and disjointed design rationale. To solve this problem VisiHistory presents the user with a video of all of the key events that occurred during a previous users design session. Unlike existing tools that just replay through the existing feature or assembly tree, the VisiHistory tool replays all the key actions a user takes including deleting and editing existing features. Additionally, these actions are recorded from the perspective of the original user which adds a richer context for all subsequent users.

The VisiHistory tool is accessed using the user interface seen in Figure 4-5. As the figure shows the user is presented with a large depiction of the previous users design screen on the right with controls and supplemental information on the left. The supplemental information includes a description of the exact action that is being displayed at any given moment as well as a list of all the features for the part. As new features are added or edited they are visually highlighted in the feature list so that the user can easily track what actions are being performed on what features. Supplemental information that is recorded by the user during the design session including audio and notes are also displayed and played back during the video. The controls include the standard play, stop, etc as well as a slider tool that allows the user to quickly find any part of the video. Using these capabilities a user can avoid duplicating efforts or creating parts with disjointed design rationale.

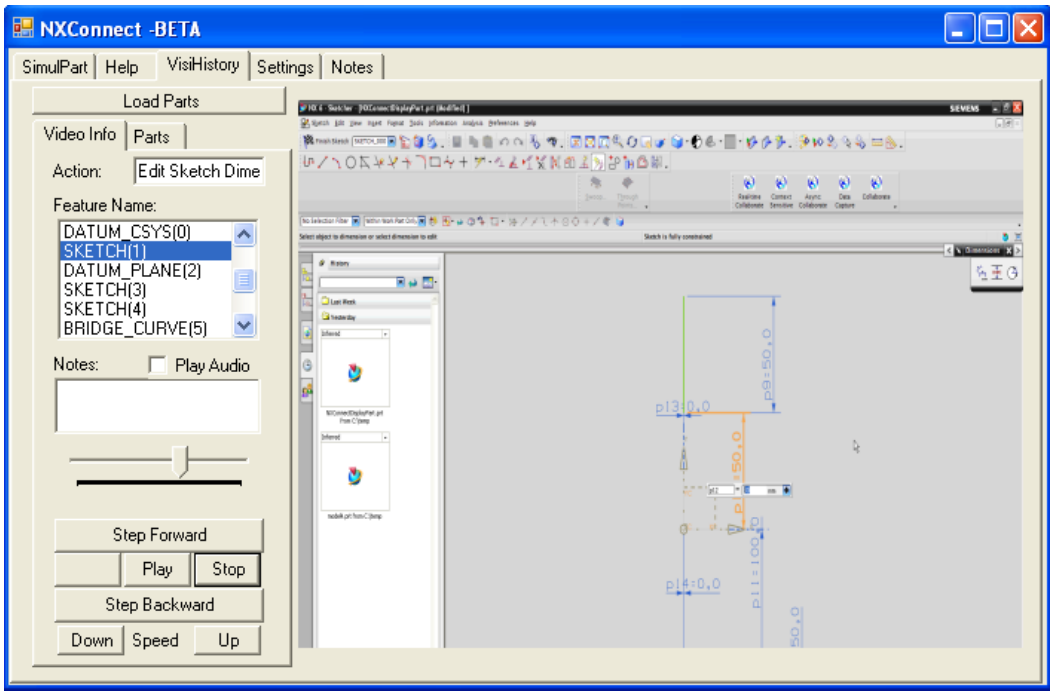

Figure 4-5 The VisiHistory user interface.

4.8.1 VisiHistory Data Sync Module Implementation

The DSM for the VisiHistory implementation is different from both the SimulPart and SmartHelp DSM implementations. The VisiHistory DSM is primarily focused on retrieving all of the important action objects and stitching them together into a video that allows the user to see the history of the part without having to watch unimportant or irrelevant events. Again, the relevancy of the actions is determined by using the inferences that were created by the IEM. To do this an SQL statement is used that selects the action objects from the database that correspond to a specific part and do not have one an inference made about it that would indicate it was irrelevant. Once the action objects have been returned they are loaded into the custom built player interface where they can be reviewed by the user.

4.8.2 VisiHistory Example

A simple example is given of how the VisiHistory tool functions and implements the method outlined in Chapter 3. This example uses a situation that is very similar to the previous example given for the SimulPart tool. Again, User1 and User2 are creating a block part, except this time they are asynchronously working on the same part. So in this case the design rationale will be used as a filter to select the most important events from previous design session so they can be shown to the user of the VisiHistory tool.

As both users take turns opening and working on the part their actions are recorded as actions objects. These actions objects are visually depicted in Figure 4-6. Special notice should be taken of the time gap that exists between User1's and User2's actions. The figure also includes the inferences that are made by the IEM as described in the SimulPart example. Once captured these action objects are then stored in the ISM.

A =

Object	Type	User	Target	Timestamp
ActionObject1	Sketch	User1	Model1	1/15/2010 3:48:24 PM
ActionObject2	Rotate	User1	Model1	1/15/2010 3:48:26 PM
ActionObject3	Sweep	User2	Model1	1/16/2010 1:27:19 PM
ActionObject4	Delete	User2	Swept(1)	1/16/2010 1:27:20 PM
ActionObject5	Rotate	User1	Model1	1/17/2010 5:11:03 PM
ActionObject6	Extrude	User1	Model1	1/17/2010 5:11:05 PM
Inference1	Visual		ActionObject2	1/15/2010 3:48:26 PM
Inference2	Visual		ActionObject5	1/17/2010 5:11:03 PM

Figure 4-6 The set of action objects and inferences generated by two users.

Now let us suppose that a day has passed since User1 last closed the part and User2 goes to open the part for editing. After opening the part the user also opens the VisiHistory tool. At this point the VisiHistory DSM goes to work. First it creates an SQL select statement that

selects the relevant subset of data which corresponds to Step 2.1 of the proposed method. In this case the subset would be all of the actions and inferences that had occurred since User2 last opened the part consisting of ActionObject5, ActionObject6, and Inference2. Like the SimulPart example Step 2.2 of the method is performed in conjunction with Step 2.1 for the VisiHistory tool. The results are identical as well since ActionObject5 would be filtered out of the result set that is displayed in the VisiHistory tool. So as a result the only action object that User2 would see would be ActionObject6 which corresponds to the extrude operation performed by User1.

The example above presents a very simple and easily understood scenario. As a result of the simplicity and limited number of actions both filtered and displayed it is somewhat difficult to see the true usefulness of the VisiHistory tool. However, as the number of actions increases in each design session the usefulness of the tool increases. This is best seen by the VisiHistory test case which is explained in the results section of this thesis which is presented in the next chapter.

5 RESULTS

To verify the true benefits of the proposed method a series of analyses was performed on each of the implementations described in the previous chapters. These analyses were designed to accomplish two main objectives. First, to show that these tools are capable of solving some of the most significant collaborative problems that engineers currently face and second that they do so in a way that surpasses the state of the art described in Chapter 2. Based upon these constraints the analyses were created and performed with the results of each analysis being presented in each of the following sections of this chapter.

5.1 SmartHelp Results

An analysis was performed to determine the benefits of the SmartHelp tool which resulted in the creation of two equations. These equations predict the approximate amount of time it would take a user with or without the use of the SmartHelp tool to find a peer in their organization with expertise regarding a certain NX function. To create the equation a number of employees from a variety of engineering companies were consulted to determine conservative estimates for the following variables:

- The average number of individuals in a work group = n = 5 individuals.
- The average time required to query your work group. = t_{local} = 10 seconds
- The average time required to query each additional work group = $t_{external}$ = 5 minutes

It should be noted that the average time to query additional work groups is very dependent upon distance and the layout of a particular building. However, 5 minutes was deemed by all of the interviewees to be a very conservative estimate of the time required to query each work group. Using this information the following equation was created for the total time required to verbally query x number of individuals in an organization:

$$f(x) = \{ g(x-1,n) * t_{external} \} + t_{local} \quad (16)$$

where g(x,y) = the integer portion of x/y

This can be compared with the function that was derived for querying x number of individuals in an organization for help using SmartHelp. This equation was derived based on actual experience using the SimulPart tool and used the following variables:

- The average incremental time per user = $t_{incremental}$ = .5 seconds
- The average time to open the SmartHelp tool = $t_{opentool}$ = 25 seconds

Using these variables the following equation was created:

$$f(x) = (x * t_{incremental}) + t_{opentoo} \quad (17)$$

The results of these two equations were then plotted for an organization with 100 users as can be seen in Figure 5-1. Obviously the majority of large engineering corporations have more than 100 engineers. However, at higher numbers it becomes impossible to see how initially the verbal method is actually more efficient than the computerized SmartHelp method. However, as the number of individuals to query increases the time to perform this search increases significantly faster for the verbal method as compared with the SmartHelp method.

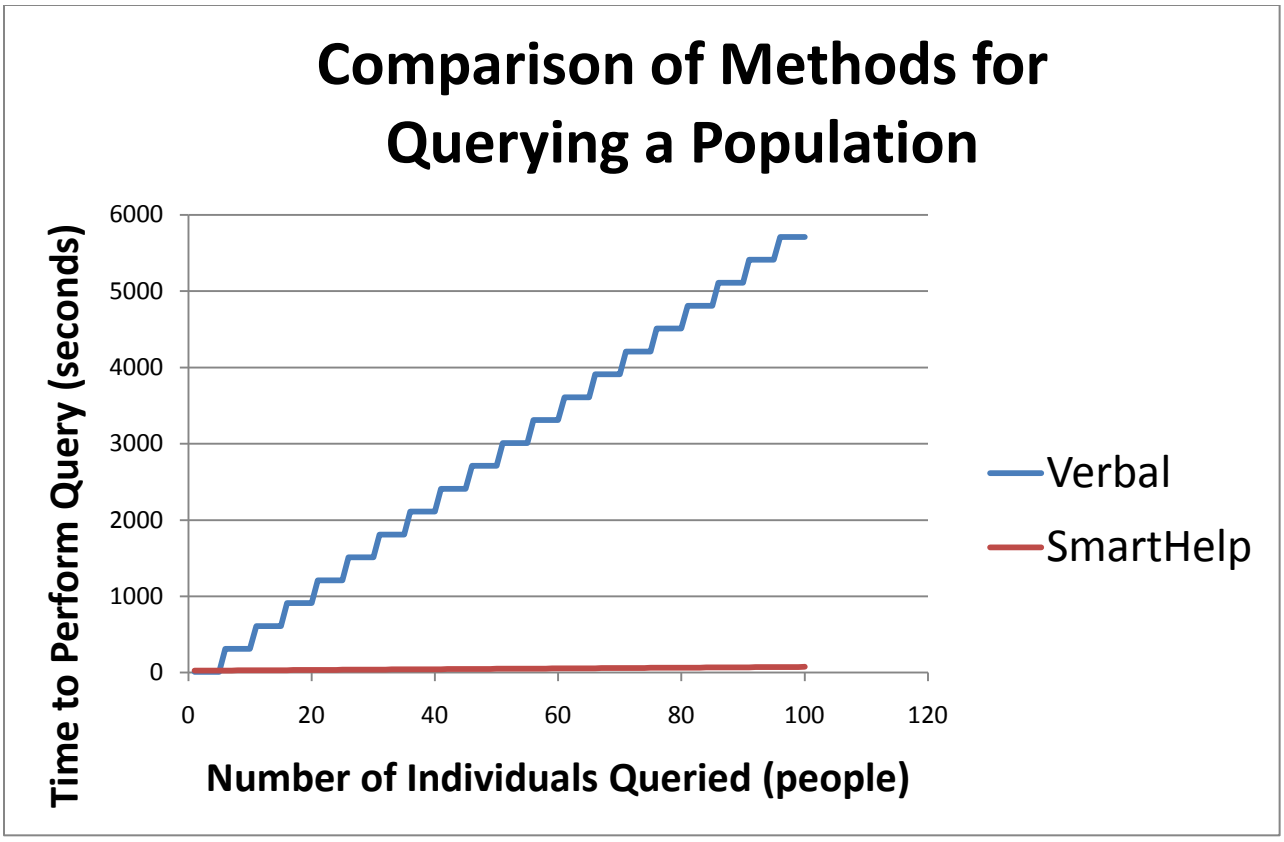

Figure 5-1 A comparison of SmartHelp and verbal methods for querying a population

5.2 SimulPart Results

To the knowledge of the author the transformation of a commercial single user MCAD package such as NX into a real-time multi-user MCAD design had never before been accomplished prior to SimulPart. This alone being a significant accomplishment, still additional verification was sought to demonstrate the benefits of both the tool and the underlying method. As a result a series of tests were performed to compare SimulPart to the leading commercial collaborative MCAD solution.

As previously stated one of the best commercial collaborative co-design environments is available in Siemens NX (Sharma, Raja and Fernando 2006). Specifically the technology is

available through the NX Collaborate add-on. Like SimulPart, NX Collaborate overcomes the network limitation by filtering the changes that are broadcast to each of the users in the session. Unlike SimulPart, NX Collaborate requires the user to manually perform this filtering.

While both solutions provide efficient usage of network resources through filtering SimulPart improves upon the prior state of the art by doing so in a less obtrusive manner. To quantify this, a number of key criteria for unobtrusiveness were identified. These were selected based on generally acknowledge best practices for collaborative engineering in both academia and industry. The essential criteria were identified as:

- Minimal required tool count
- Minimal setup time and effort
- Minimal effort to maintain a collaborative session

For each of these criteria the SimulPart tool was compared to NX Collaborate with the results being presented below.

Engineers want collaborative solutions that require them to master or even be familiar with the least number of tools possible. As a result the number of tools required to create and open a SimulPart and NX Collaborate were compared. The results are presented in Table 5.1. This table is divided into two categories, the number of server tools required and the number of end-user tools. A server tool is one that is required for the collaborative tool to function but that the end user is not required to explicitly open or interact with. As can be seen in Table 5.1 SimulPart does a better job than NX Collaborate at minimizing the number of tools that an engineer is required to use to participate in a collaborative design session. The fact that SimulPart can achieve just as capable of a collaborative design environment with half the tools is made possible as a direct result of the simplicity of the method presented in Chapter 3.

Table 5.1 A comparison of the number of software tools required to create and open a collaborative design session in NX Collaborate and SimulPart.

	SimulPart	NX Collaborate
Server Tools	1	4
End-user Tools	1	2

Another requirement for unobtrusiveness is that the time and effort required to setup a collaborative design session be kept to a minimum. While the time and effort required to perform an action within a software tool can be difficult to accurately measure, the number of mouse clicks required to perform the same action can provide a reasonable substitute. For both SimulPart and NX Collaborate the number of mouse click required to create a session and join an existing session were calculated and can be seen in Table 5.2. SimulPart simplifies the time and effort required to setup and join a collaborative design session. As before this is a result of the simplicity of the method presented in Chapter 3.

Table 5.2 The number of mouse clicks to perform vital functions in SimulPart and NXCollaborate.

	SimulPart	NX Collaborate
Create Session	4	11
Join Session	4	8

The final key criteria engineers use when evaluating a collaborative tool is the amount of effort required to maintain the collaborative session. In this aspect SimulPart has a clear advantage. NX Collaborate requires the user to explicitly choose when and which feature operations they wish to share. It also requires all of the users in the session to use a single

“token” to determine which user is permitted to make changes at any given time. If a different user wishes to make changes they must go through the process of requesting and passing the token. In contrast SimulPart requires no actions from the user to maintain the collaborative session. Instead it uses the design rationale to infer when and which actions should be broadcast thus providing no effort for the user to maintain the collaborative session.

The results show that the SimulPart tool outperforms NX Collaborate in each of the crucial collaborative tool metrics. It requires the user to learn fewer tools and spend less effort creating and maintaining a collaborative session.

5.3 VisiHistory Results

To verify and test the efficacy of the VisiHistory tool an experiment was performed. For the experiment an engineer is presented with a hypothetical situation in which they work for a global product development company that develops brackets. The company has an office in both India and Provo and the test subject works in the Provo office. In this hypothetical example an engineer in India has started working on a part during their shift but has not been able to complete it and so they leave it for the Provo office to perform the necessary steps to finish the job. To help the test subject understand what they are expected to create they are shown a finished version of a similar part as depicted in Figure 5-2. They are also given an engineering drawing which calls out the desired locations for the bolt pattern on top of the flange for the part being created. Using this information the test subject, who represents an engineer in the Provo office, should be able to take any partially completed design from the Indian office and complete it without requiring any additional help or information.

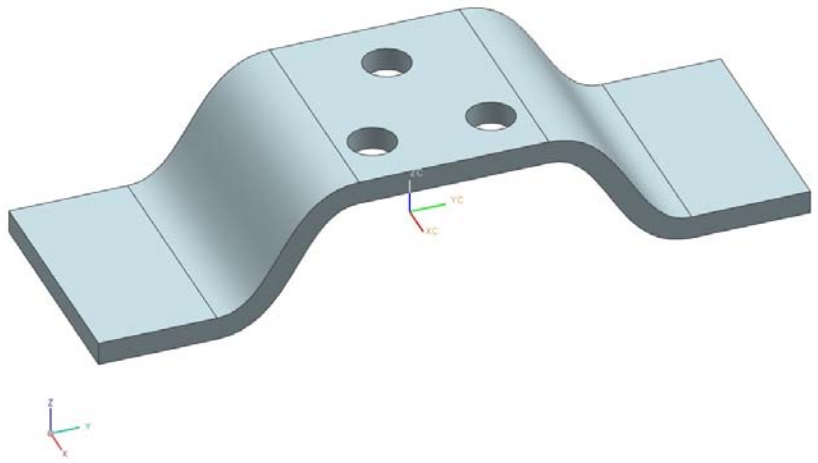

Figure 5-2 A finished version of a similar part

When the user opened the flange part file that was supposedly created earlier in the day by the engineer in India they found the bracket almost entirely completed with the exception of the bolt pattern. A visual depiction of what the Provo office engineer would see upon opening the part can be seen in Figure 5-3. Upon visual inspection it becomes clear that the engineer in India has already created the flange portion of the part and all that is missing is for the Provo engineer to add the bolt pattern. In addition to the visual information, the Provo user also has access to the feature history tree through which they can determine both the process taken to create the part and the features that were created. Using these two resources the Provo office engineer should be able to continue the process followed by the India office engineer and successfully complete the part.

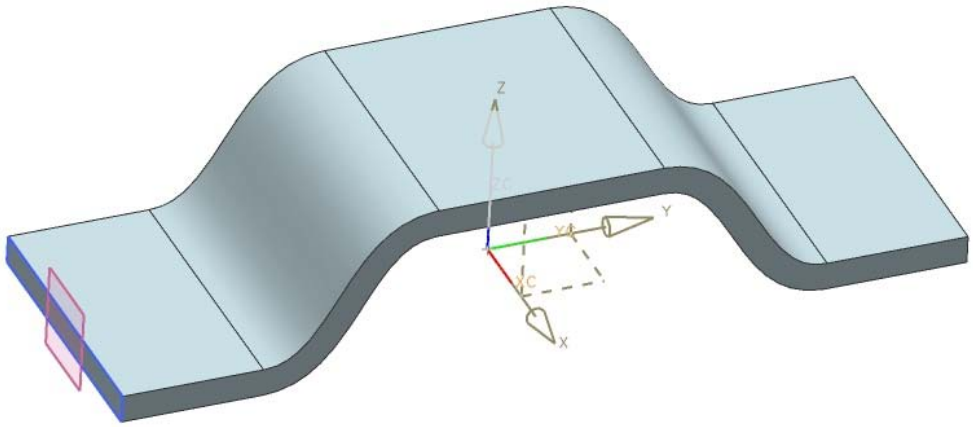

Figure 5-3 What the test user sees when first opening the example flange part file.

The key to finishing the model is recognizing that the bolt pattern sketch already exists but is in fact hidden by the solid model. This sketch can clearly be seen in wireframe depiction of the part in Figure 5-4. Because the flange was made by sweeping the rectangle on the left of the figure and the guide curve goes through the same plane that the bolt pattern was drawn on the bolt pattern sketch is covered up. For the designer in India this is a logical and completely justifiable design decision. However, for the second designer this presents a potential problem common to asynchronous collaboration. This problem is the basis for the experiment testing the efficacy of the VisiHistory tool in eliminating the common asynchronous collaboration problem of duplicated efforts.

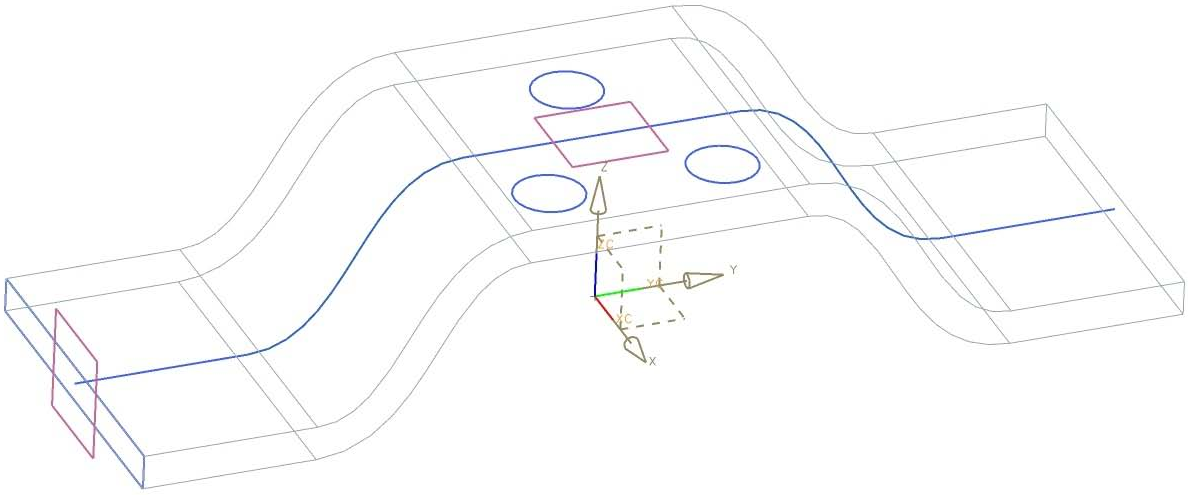

Figure 5-4 A wireframe depiction of the original flange part file.

For the experiment the sample population was divided into two separate groups A and B. Each group was given the exact same scenario with all of the same information. The one difference was that Group A was also allowed to use the VisiHistory tool while group B was not. Using the tool Group A was able to see a brief dynamically generated video of the key events of designer in India's design session. Otherwise both groups were given the same information and asked to complete the model with their actions being observed to determine if they had identified the existence of the hidden sketch.

The experiment was replicated with 10 different students of a variety of skill levels and experience. Each was randomly assigned to either Group A or Group B so that there were 5 in each group. Of the five students in Group B who did not use VisiHistory none of them identified the existence of the bolt pattern sketch. Of the five students in group A who did get to use VisiHistory four identified the existence of the bolt pattern sketch. A two-sample unequal

variance t-test was performed on these results to determine if the VisiHistory tool had a significant impact on the user's ability to identify the sketch. The results indicate that with a 95% confidence level we can reject the null hypothesis and state that the VisiHistory tool does have significant impact on the user's ability to identify the hidden sketch.

6 CONCLUSIONS & SUGGESTIONS FOR FUTURE WORK

The underlying objective of this thesis is to devise a new method for using design rationale to improve engineering collaboration and to show that this method can be leveraged to provide tools which surpass the existing standard for engineering collaboration tools. This standard as well as several of the critical issues with the state of the art was presented in Chapter 2. The generalized method that was devised for leveraging design rationale to improve collaboration in a broad spectrum of scenarios was then presented in Chapter 3 of this thesis. Chapter 4 detailed the implementation of the three tools that were created using the proposed generalized method:

- SimulPart – A tool that enables concurrent CAD modeling.
- SmartHelp – A tool that allows engineers to discover which of their peers have experience with a specific CAD operation.
- VisiHistory – A tool that displays a visual history of a CAD file.

The fifth chapter of this thesis was dedicated to demonstrating that each of the tools provides an improved collaboration experience over existing solutions, thus validating the significance of the generalized method.

To validate the SimulPart tool an analysis was performed in which it was compared to one of the best commercial collaborative co-design environments. Based on this analysis SimulPart outperformed the existing solution. In doing so SimulPart demonstrates that design rationale can in fact be used to provide a more efficient usage of network resources for

collaborative design environments. The benefit of this approach is that it can be applied to existing methods and tools for co-design, including all of those reviewed by Li and his colleagues (Li, et al. 2005).

The validation of the SmartHelp tool was done by comparing it to the currently preferred manual approach to seeking help. As Figure 5-1 shows if a users local workgroup does not possess the desired expertise the SmartHelp is by far the better method for seeking help. From these results it can be concluded that for help on simple or common topics manually querying your local workgroup is still the best solution but for anything more difficult SmartHelp is the preferred solution.

An experiment was performed to validate the VisiHistory tool. The results of this experiment indicate that with a 95% confidence level the VisiHistory tool is better at avoiding rework than the current alternative of visual inspection of the history tree. Based upon these results it can be seen that the VisiHistory tool is in fact a better solution for avoiding rework in asynchronous collaboration scenarios.

Overall these results indicate that the generalized method for improving engineering collaboration through the use of design rationale does in fact improve upon the state of the art. However, while this analysis has proven the value of these tools and more generally the proposed method they are both not without limitations and weaknesses. These limitations and weaknesses provide the basis for the recommendations that are presented in the next section of this thesis.

6.1 Recommendations

While the value of the method has been successfully proven the implementation of the method is also not without its faults and weaknesses. The first such limitation is that the

implementations have been limited to the NX6 CAD package. Future work is suggested into researching the requirements to implement of the method into additional CAx tools.

Another limitation that the SimulPart implementation demonstrates is that the method currently only filters the data. This may result in important data not being transferred. One way to overcome this limitation would be to instead use the design rationale to prioritize the transfer data. This way crucial information still gets immediately shared while less important information is broadcast later when excess network bandwidth becomes available. Another limitation is in the quality of the design rationale created by the inference functions. Further research is recommended into improving the quality of the design rationale inferred from these functions. SimulPart is also limited in the functions it currently supports. However, these limitations are a result of implementation time and not the underlying architecture. Thus, recommended future work would increase the number of functions that SimulPart supports to match those currently supported by NX6. An additional line of work would be to take the architecture and design a tool from the ground up that takes advantage of the unique nature of multi-user MCAD. Either of these efforts would benefit from using the method presented in this thesis.

REFERENCES

Agustina, Fei Liu, Steven Xia, Haifeng Shen, and Chengzheng Sun. "CoMaya: Incorporating Advanced Collaboration Capabilities into 3D Digital Media Design Tools." *Proceedings of the ACM 2008 conference on Computer supported cooperative work.* San Diego: ACM, 2008. 5-8.

Albazzaz, Hamza, and Xue Z. Wang. "Historical data analysis based on plots of independent and parallel coordinates and statistical control limits." *Journal of Process Control*, 2006: 103-114.

Archer, Tom. *Inside C#.* Redmond: Microsoft Press, 2001.

Aseltine, Jonathan H. "WAVE: An Incremental Algorithm fro Information Extraction." *AAAI Technical Report*, 1999: 21-24.

Aurisicchio, M., and K. M. Wallace. "Information requests and consequent searches in aerospace design." *The 8th International Design Conference.* Dubrovnik, 2004. 18-20.

Billinghurst, Mark, and Kato Hirokazu. "Collaborative Augmented Reality." *Communications of the ACM*, 2002: 64-70.

Bracewell, Rob, Ken Wallace, Michael Moss, and David Knott. "Capturing Design Rationale." *Computer-Aided Design*, 2008: 1-14.

Burge, E Janet, and D James Kiper. "Capturing decisions and rationale from collaborative design." *Design Computing and Cognition*, 2008: 1-19.

Conklin, Jeff, Albert Selvin, Simon Buckingham Shum, and Maarten Sierhuis. "Facilitated hypertext for collective sensemaking: 15 years on from gIBIS." *Proceedings of the 12th ACM conference on Hypertext and Hypermedia.* Arhus: ACM, 2001. 123-124.

Conklin, Jeffrey, and Michael L. Begeman. "gIBIS: A Hypertext Tool for Team Design Deliberation." *Proceedings of the ACM conference on Hypertext.* Chapel Hill, 1987. 247-251.

Connolly, Thomas, and Carolyn Begg. *Database Systems: A Practical Approach to Design, Implementation and Management.* Boston: Addison-Wesley, 2010.

Ebb, Matt. *Project Verse.* http://verse.blender.org/ (accessed August 06, 2009).

Edwards, W. Keith, Elizabeth D. Mynatt, Karin Petersen, Mike J. Spreitzer, Douglas B. Terry, and M. Marvin Theimer. *Designing and Implementing Asynchronous Collaborative Applications with Bayou.* Banf: Association for Computing Machinery, 1997.

Fussell, Susan R., Leslie D. Setlock, Jie Yang, Jiazhi Ou, Elizabeth Mauer, and Adam D.I. Kramer. "Gestures Over Video Streams to Support Remote Collaboration on Physical Tasks." *HUMAN–COMPUTER INTERACTION*, 2004: 273-309.

Hejlsberg, Anders, Scott Wiltamuth, and Peter Golde. *The C# Programming Language.* Boston: Pearson Education, 2006.

Ishino, Yoko, and Yan Jin. "Acquiring engineering knowledge from design processes." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 2002: 73-91.

Ishino, Yoko, and Yan Jin. "An information value based approach to design procedure capture." *Advanced Engineering Informatics*, 2006: 89-107.

Ishino, Yoko, and Yan Jin. "Estimate design intent: a multiple genetic programming and multivariate analysis based approach." *Advanced Engineering Informatics*, 2002: 107-125.

Iyer, Ganeshram Ramji. "A context-aware inferece system, to capture design rationale from legacy MCAD." *Doctor of Philosophy the University of Texas at Arlington.* Arlington: The University of Texas at Arlington, December 2007.

Jin, Y., and Y. Ishino. "DAKA: design activity knowledge acquisition through data-mining." *International Journal of Production Research*, 2006: 2819-2837.

Johannesmeyer, Michael C., Dale E. Seborg, and Ashish Singhal. "Pattern Matching in Historical Data." *AIChE Journal* 48, no. No. 9 (September 2002).

Kimmel, Paul. *LINQ Unleashed.* Indianapolis: Pearson Education, 2009.

Klein, Mark. "Capturing Design Rationale in Concurrent Engineering Teams." *IEEE Computer*, 1993: 39-47.

—. "Capturing Geometry Rationale for Collaborative Design." *Proceedings From the Sixth IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises.* Cambridge, 1997. 24-28.

Lee, Jintae. "Design Rationale Systems: Understanding the Issues." *IEE Expert*, 1997: 78-85.

Li, W.D., W.F. Lu, J.Y.H. Fuh, and Y.S. Wong. "Collaborative Computer-aided design—research and development status." *Computer-Aided Design*, 2005: 931-940.

Marchese, Francis T., and Natasha Brajkovska. "Fostering Asynchronous Collaborative Visualization." *Proceedings of the 11th International Conference Information Visualization.* Washington, DC: IEEE Computer Society, 2007. 185-1960.

Matviykiv, Oleh, Mykhaylo Lobur, and Olga Lebedeva. "Virtual Collaborative Design Environment for Distributed CAD Systems." *CAD Systems in Microelectronics.* Lviv-Polyana: CADSM, 2007. 538-540.

Moran, Thomas P., and John M. Carroll. *Design Rationale: Concepts, Techniques, and Use.* Mahwah: Lawrence Erlbaum Associates, 1996.

Myers, Karen L., Nina B. Zumel, and Pablo Garcia. "Automated Capture of Rationale for the Detailed Design Process." *Proceedings of the Eleventh Conference on Innovative Applications of Artificial Intelligence.* AIAA, 1999. 876-883.

PC Magazine. *PC Magazine Encyclopedia - API.* http://www.pcmag.com/encyclopedia_term/0,2542,t=application+programming+interface&i=37856,00.asp (accessed August 14, 2009).

Powell, Gavin. *Beginning Database Design.* Indianapolis: Wiley Publishing, 2006.

Qiang, L., Y. F. Zhang, and A. Y. C. Nee. "A Distributive and Collaborative Concurrent Product Design System through the WWW/Internet." *Advanced Manufacturing Technology*, 2001: 315-322.

Qiu, Z.M., and W.D. Li. "State-of-theart Technologies and Methodologies for Collaborative Product Development Systems." *International Journal of Production Research*, 2006: 2525-2559.

Rawson, Kevin, and Thomas F. Stahovich. "Learning Design Rules With Explicit Termination Conditions to Enable Efficient Automated Design." *Journal of Mechanical Design*, March 2009.

Rea, Heather J, Jonathan R Corney, James M Ritchie, Raymond Sung, and Csaba Salamon. "Automating Digital Capture of Engineering Knowledge." *Proceedings of DET2007 4th International Conference on Digital Enterprise Technology.* Bath, 2007.

Red, Edward, Vonn Holyoak, C. Greg Jensen, Felicia Marshall, Jordan Ryskamp, and Yue Xu. "v-CAx: A Research Agenda for Collaborative Computer-Aided Applications." *Computer-Aided Design & Applications*, 2010.

Rob, Peter, and Carlos Coronel. *Databse Systems: Design Implementation, and Management.* Danvers: Boyd and Fraser Publishing Company, 1995.

Sharma, Manglesh, Vinesh Raja, and Terrence Fernando. "Collaborative Design Review in a Distributed Enviroment." *Intelligent Production Machines and Systems*, 2006: 65-70.

Siemens PLM Software. "NX programming and customization." *Siemens PLM Software.* November 2007. http://www.plm.automation.siemens.com/en_gb/Images/nx%20programming%20and%20customization%20fs%20W%203_tcm642-4564.pdf (accessed August 14, 2009).

Wang, Lirong, Jiacai Wang, Lixia Sun, and Ichiro Hagiwara. "A Peer-to-Peer Based Communication Environment for Synchronous Collaborative Product Design." *Cooperative Design, Visualization, and Engineering* . Berlin: Springer Berlin / Heidelberg, 2007. 9-20.

Wattenberg, Martin, and Fernanda B. Viegas. "Communication-minded visualization : A call to action." *IBM systems journal*, 2006: 801-812.

White, Bobby. "Cisco's Homegrown Experiment." *The Wall Street Journal*, January 23, 2007: A14.

Zdrahal, Zdenek, Paul Mulholland, Michael Valasek, and Ansgar Bernardi. "Worlds and Transformations: Supporting the Sharing and Reuse of Engineering Design Knowledge." *International Journal Human-Computer Studies*, 2007: 959-982.