2010-03-09

Integrating Engineering and Communication Tools for the Automation of Design Rationale Capture

Kenneth John Mix
Brigham Young University - Provo

Integrating Engineering and Communication Tools for the Automation of Design Rationale Capture

Kenneth J. Mix

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

Master of Science

Department of Mechanical Engineering

Brigham Young University

April 2010

ABSTRACT

Integrating Engineering and Communication

Tools for the Automation of

Design Rationale Capture

Kenneth J. Mix

Department of Mechanical Engineering

Master of Science

Product development is continually becoming more challenging as global competition requires more efficient design methods. The reuse of company knowledge, specifically the design rationale that occurs during projects is essential to success.

This thesis presents a method for integrating engineering and communication tools for purposes of automating the capture of communication-based design rationale. The method is based on four basic principles: to integrate, to make data retrievable, to minimize user interaction, and to store as much DR as possible. The core method consists of two primary operations, the first being to capture the design rationale, and the second being to provide for effective retrieval.

An implementation of this method that uses NX as the engineering tool and Skype VoIP software as the communication tool was created for the purpose of testing integration as a means of DR capture. The implementation was evaluated using four separate tests, which focus on efficiency of capture and retrieval, cost analysis, and user satisfaction. These results show that the tool provides improvement in each of the tested categories.

From this testing I conclude that integrating communication and engineering tools is an excellent way to capture communication-based design rationale. The tool presented is more efficient than traditional methods in the test cases and provides a user-friendly solution to DR capture. This tool also has various other important applications, such as global collaboration and expectation management. It also provides an excellent framework for upcoming multi-user CAx tools.

Keywords: design rationale, knowledge capture, design, reuse, global engineering

ACKNOWLEDGMENTS

I would like to thank my committee members for their excellent input on this research. I especially thank Dr. C. Greg Jensen for his unwavering support in every aspect of my academic career. Additionally I would like to thank the members of my family, who have been an incredible support to me throughout the years. Lastly I want to thank my wife-to-be Natalie, for her love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

The product development (PD) world of today is more challenging than ever before. Companies across the globe are shrinking cycle times in an attempt to release new products faster and sustain their competitive advantage. Adding to the challenge, the increased globalization and distribution of tasks aims to reduce cost and cycle times, but often leads to increased difficulty when designing products.

In order to produce products more effectively in this changing environment, most PD firms must rely heavily on their internal knowledge base and resources. Design has changed from companies producing one-off designs to the development and wide-scale adoption of product lines, where the knowledge from each project contributes heavily to future designs. One of the most important sources of knowledge that companies must recognize and use is their employees. The knowledge that employees gain from previous projects allows them to contribute greatly to future products. The thoughts, ideas, intents, and decisions that are made on a project by the employees can be termed design rationale (DR).

Many innovations have emerged that assist in the capture and reuse of company knowledge. For example, the notion of parametric design, where a single CAD model can be parameterized in order to make it dynamic and reusable aims directly at this goal. Additionally, tools such as product lifecycle management (PLM) and product data management (PDM) packages allow for the capture and reuse of design data. These tools focus on the storage and reuse of product definition data. This data could include geometric computer-aided design

(CAD) data, analysis data, manufacturing drawings, bill of materials, and other related product data.

While these tools are very effective for storing and managing data, much of the knowledge gained from the daily process is lost. This thesis presents a method for an integrated communication and engineering tool that allows for the storage and retrieval of communication-based DR during the product development process. The additional capture of this data will allow for more rich and meaningful knowledge data.

1.1 **Problem Statement**

For many years, engineering software firms have aimed to capture PD knowledge. Much of this has been in the form of raw data capture and management, such as with PDM and PLM systems. These systems excel at the storage of raw data. They also allow the raw data to be related and referenced for easier retrieval. For instance, if an assembly file is stored in a PLM environment; its related sub-parts are referenced within the system so that the parts can be easily retrieved upon opening the assembly.

While these tools excel at the capture of this raw data, they fall short when it comes to the capture of design rationale. While revisions of files can often show indirectly the reason why a part was designed the way it was, the discretized nature of revisions show only snapshots in time, rather than showing the entire process. Decisions that take place in short conversations or in design reviews may provide more information as to why certain designs were created how they were, but often much of this knowledge is lost due to poor documentation and the lack of tools for capturing this communication information.

Engineering software companies have not typically focused on capturing this communication-based DR. In fact very few engineering tools directly support integrated communication tools. Even when the support is indirectly included, there are no efforts made to capture this DR and integrate it with engineering data. Because of this, much of the communication data that could be used as an additional resource for the understanding of design data is lost.

1.2 **Research Objectives**

This research attempts to address this issue by developing a tool that can integrate communication functionality into engineering tools. By developing this tool, the capture of this essential communication information is possible. In addition, this information can be directly related to other useful data captured in the engineering tool, such as CAD features, analysis results and other types of engineering data. This effort will augment the excellent tools that already exist for capturing and managing engineering data and DR.

The presentation of this research will take place as follows: Chapter 2 will discuss background information, including previous research efforts and trends. This chapter will also give an explanation of the foundational tools used to implement the capture tool. Chapter 3 will discuss the general methodology developed for capturing, storing, and retrieving this communication based design rationale from within an engineering tool and relating it to engineering data. Chapter 4 will discuss an implementation of this method developed for the purpose of evaluating this approach to communication-based DR capture. Chapter 5 will show the results of this evaluation, and will discuss how the tool augments current DR capture methods. Chapter 6 will present the conclusions from this research and provide conclusions and suggestions for future work.

1.3 **Research Delimitations**

The method developed in this research is intended to be applicable to any set of communication and engineering tools. The functionality available by each implementation of the method is determined by the individual applications that are used to create it.

The implementation presented in chapter 4 uses NX as its engineering tool base, and Skype as its communication tool. These tools were chosen due to their familiarity with the researcher and their excellent application programming interfaces. Using Altair HyperWorks with Microsoft Office Communicator would be entirely possible as denoted by the method. Another example of how the method could be implemented would be to integrate a PLM tool such as Teamcenter Engineering with the Gizmo VoIP software for communication, provided these two tools have application programming interfaces that allow their internal functionality to be accessed.

The implementation presented in this thesis is a subset of the method adequate to evaluate the approach to capturing communication-based DR presented in chapter 3. Some of the functionality that would be desirable in an industry-ready version of the tool will not be contained in the implementation.

2 BACKGROUND

This chapter discusses the background information necessary to understand the method and its implementation. A review of relevant literature will be presented in order to provide explanation of the research done previously and show how this research contributes to the work done by others. A section that discusses foundational tools is also included in an effort to explain concepts and tools used in the implementation that may be unfamiliar to the reader.

2.1 Design Rationale

Design rationale can be explained as the reason something is defined the way it is (Lee and Lai, What's in Design Rationale? 1996), (Gruber and Russell 1991). More specifically, it is the combination of specifications, motivations, and actions for the purpose of creating designs (Moran and Carroll 1996). A major part of DR is the employees' knowledge and experience, which could be termed "Tacit Knowledge" (Komerath 2001). The capture, storage, and retrieval of DR are essential for a company to sustain competitive advantage (Hicks, et al. 2002).

Extensive research has been focused on the capture of this knowledge (Regli, et al. 2000). Because of the many different types of design rationale, the methods developed to capture and use this DR have varied (Dutoit, et al. 2006). The primary motivation of this work is the inability of many data management packages to capture and store these intents and motivations (Klein

1997). Because of this, some researchers have attempted to integrate design rationale systems into PDM packages, as reported by Gao, et. al (Gao, et al. 2003).

2.1.1 Capturing Design Rationale

The primary focus of DR research lies with developing frameworks and tools for capturing and representing design rationale data (Zdrahal 2007). Some tools that have been developed to accomplish this, some examples of these are gIBIS (Conklin and Begeman 1987) and Compendium (Conklin, et al. 2001). These systems attempt to capture data from the user's daily workflow, but often do not directly integrate with the engineering tools.

2.1.2 Barriers to the Capture of Design Rationale

W. C. Regli et al state that "A design rationale system is not effective as a standalone system" (Regli, et al. 2000). The integration of the design rationale capture tool into the worker's daily routine and toolset is essential to the effective capture of this data. Bracewell, et. al have stated that because of this lack of integration to the daily routine, the adoption of DR capture systems into the industry has largely failed (Bracewell, et al. 2008).

Horner & Atwood have discussed the various barriers to the capture design rationale (Atwood and Horner 2007), (Horner and Atwood 2006). They conclude that there are four primary categories for these barriers. They are:

1. Cognitive limitations
2. Capture limitations
3. Retrieval limitations
4. Usage limitations

The method presented in this thesis focuses on addressing the last three limitations. Horner & Atwood further describe that engineers and designers must have incentive to capture and retrieve design rationale. One motivation is time. If capturing the rationale takes too much time, the employee will often fail to do it (Grudin 1996). If the tool can be integrated directly with the user's daily workflow, and automated so that it minimizes user interaction, the user will be far more likely to participate. Some research has attempted to automate the process of capturing design rationale, such as that demonstrated by SRI international (Myers, Zumel and Garcia 1999). The method presented in this thesis builds on this principle, attempting to automate and minimize user interaction.

2.1.3 Re-using Design Rationale

The primary reason for capturing design rationale data is so that it can be effectively reused. Nearly all of the methods discussed previously provide tools for retrieving the data. Lee has discussed the importance of being able to access design rationale by using examples from existing tools (Lee, Design Rationale Systems: Understanding the Issues 1997).

2.1.4 Communication-Based DR Capture

Some tools have also included the ability to capture communication-based DR. Some notable examples are PHIDIAS (Shipman and McCall 1997), REMAP/MM (Ramesh and Sengupta 1995) and InterPOD (Takahashi and Yana 2000). These systems provide the ability to capture and store communication data. This research builds on these tools by integrating communication tools directly into the engineering package so that communication-based DR can be captured in the daily routine and directly inserted into other types of DR.

2.2 **Foundational Tools**

This section discusses some of the concepts and tools that are used in the method and its implementation. It will assist the reader in understanding the means by which the proposed tool was created.

2.2.1 CAx Tools

This thesis makes reference to CAx tools. CAx refers to computer-aided engineering applications (Dankwort, et al. 2004). Computer-aided design (CAD) packages are one type of CAx tools. CAD refers to computer tools primarily used for the purpose of modeling geometric data (Shah and Mantyla 1995). Recently, innovations in the area of parametric modeling in CAD have enabled geometric data reuse, which has allowed increased engineering efficiency (Anderl and Mendgen 1995). Some of the leading CAD packages in the industry are Siemens’ NX, Dassault Systemes’ CATIA, and PTC’s Pro/ENGINEER.

CAx applications also include computer aided manufacturing (CAM) tools, which can be used for automating the process of various manufacturing tasks and for controlling numerically controlled manufacturing systems (Elanchezhian, Selwyn and Sundar 2007).

In addition, CAx encompasses various analysis packages for solving many design related problems that involve computer aided analysis (Alic 1993). This thesis will refer from here on to all CAx tools simply as “engineering tools.”

2.2.2 Communication Software

Communication software refers to any computer application that enables the communication of two or more individuals. Modern communication suites enable multiple types of communication, such as instant messaging, SMS text messaging, voice calls using voice-over-IP, and video conferencing (Stallings 2009). This variation of communication methods allows a user to choose how to communicate according to their preference. In addition, most of these packages include presence information, which is the ability to view whether a person is available, busy, away from their desk, etc. (Sachpazidis, et al. 2006). Some leaders in computer based communication are Skype, a free VoIP based communication service, and Microsoft Office Communicator, a software package that provides an internal secure server for communication within the organization (Sandvine Inc. 2009), (Elliot and Blood 2009).

2.2.3 Relational Database Management Systems

The implementation of this method utilizes a relational database management system (RDBMS) as a storage medium. These storage systems were originally developed by Dr. E.F. Codd (Powell 2006).

Relational database systems store information in tables. The columns of the tables are the data items related to that table. For example for a "person" table there may be columns for name, age, date-of-birth, gender, etc. Rows of the table represent each distinct entry in the table. To continue the person table example, row one could be information related to "John Doe," while row two would have all the information for "Bill Jones."

A database then relates data to other data by use of relationships. For example, consider a personnel database that is linking together different people in a hierarchal manner. You could create a relationship called "Boss" and a relationship called "Sub-ordinates." Each person in the person table would then have a number in the "Boss" field that was the row number of the personnel table that referred to their boss. Subordinates would require a more complex relationship, since there is only one boss but many subordinates. In this case it would be necessary to create an additional table called “Boss-Subordinates.” This table would store each boss ID with each subordinate ID so they could be linked. There are also relationships called “many-to-many” relationships, in which many rows in one table may relate to many rows of another table. In this case you also need to create an additional relationship table to store the information.

Query languages have been developed to search the database. SQL is one such language, as it allows the user to directly interact with the system through set commands (Connolly and Begg 2009).

All database interactions in the implementation were performed using the Microsoft .NET Language-Integrated Query (LINQ) framework. LINQ is a set of extensions for C# used to simplify the process of interacting with an SQL database (Marguerie, Eichert and Wooley 2008).

2.2.4 API Programming

Some computer applications supply an application programming interface (API) that allows developers to access internal application functionality (Orenstein 2000). The implementation of this method heavily uses both the Siemens NX and Skype APIs to achieve integration.

As an example, one of the main functions of the implementation is the ability to call another individual. The Skype API provides a function which will place a call to the specified user (Kho, Baset and Schulzrinne 2008). Note that this function can be called from any other program. These API’s allow for the direct integration of communication tools into the engineering tools.

2.2.5 C#

The implementation of this method is written entirely in Microsoft's C# programming language. C# was chosen because of its simple, object-oriented approach to creating robust applications (Hejlsberg, Wiltamuth and Golde 2006). C# also utilizes the Microsoft .NET framework, which allows for tighter integration with the Windows operating system. C# was also chosen due to the fact that both Skype and NX provide C# APIs for their respective applications, which allowed for more rapid and robust development of the implementation.

3 METHOD

This chapter introduces the general method for capturing and storing communication data and linking it with engineering data so that it can be easily retrieved and used. Hereafter this method will be termed the method for communication capture (MCC) for simplicity.

The structure for presenting the method will be as follows: first, a series of principles will be presented that will serve to lay a foundation upon which the method is built. Second, the actual method will be presented. Note that the general method presented here is intended to be applicable to any set of engineering and communication tools. While examples will be used to illustrate the method and its foundational principles, the reader should refer to chapter 4 for a full explanation of the implementation created to test the ability to achieve integration between these tools and use this integration to drive DR capture.

3.1 Overall MCC Method Summary

The MCC method is built upon four basic principles. The strict adherence to these principles is essential to any successful implementation of the method. The principles are:

1. Integrate
2. Make Data Retrievable
3. Minimize User Interaction
4. Store as much DR as possible

The principles mentioned will assist the implementers of the MCC method understand the requirements and necessary structure that must be in place in order to successfully implement it. The principles discussion will contain many examples to illustrate the importance of the principles and how they should effectively be addressed.

The core method is divided into two primary operations, each having multiple sub-steps (Mix, Jensen and Ryskamp, Automated Design Rationale Capture within the CAx Environment 2010). Each operation with its respective sub-steps is listed below:

1. Capture Design Rationale
   a. Capture Raw DR
   b. Capture Automatic and User-Supplied Meta-Data
   c. Store Rationale in Storage Medium
2. Retrieve and Use Rationale
   a. Retrieve the DR from the Storage Medium
   b. Display DR such that it can be readily reused

Each operation and its sub-steps will be discussed in the body of this chapter. The core MCC method will be explained verbally, visually and with set notation.

3.2 **Foundational Principles for the MCC Method**

This section will discuss in greater detail the four principles upon which the method is based. Note that these principles were chosen based on the literature review presented in chapter 2. As a survey of the literature was performed, it was determined that certain principles seemed to be constant throughout the published works. Each principle's section will include information from chapter 2 that is relevant to that principle.

3.2.1 Principle #1: Integrate

The first principle is to integrate. As reported in section 2.1.2 above, integration of various types is key to the successful adoption of any DR capture system. Two examples of the principle of integration will be presented in order to illustrate this principle. The first is integration between the communication tool and engineering tool, and the second is integration with the target company.

Communication/Engineering Tool Integration

Perhaps one of the most important integration features is the integration between the communication tool and the engineering tool. The integration of these tools allows the user to seamlessly communicate with others on the project, without leaving the engineering tool. This keeps the design rationale system deeply rooted in the daily routine. As the person uses the communication tool to communicate, the conversation information can be captured. The tighter the integration between these two tool types, the more effective the tool will be. Ideally the communication tool should appear to be a feature of the engineering tool. An illustration of this can be found in figure 3-1. Notice in the figure that the Skype communication tool resides inside the Siemens' NX engineering tool. You cannot separate the two tools from each other, and the communication tool appears to be a feature. Ideally an implementation would provide a way for the user to launch the communication tool directly from the engineering tool interface.

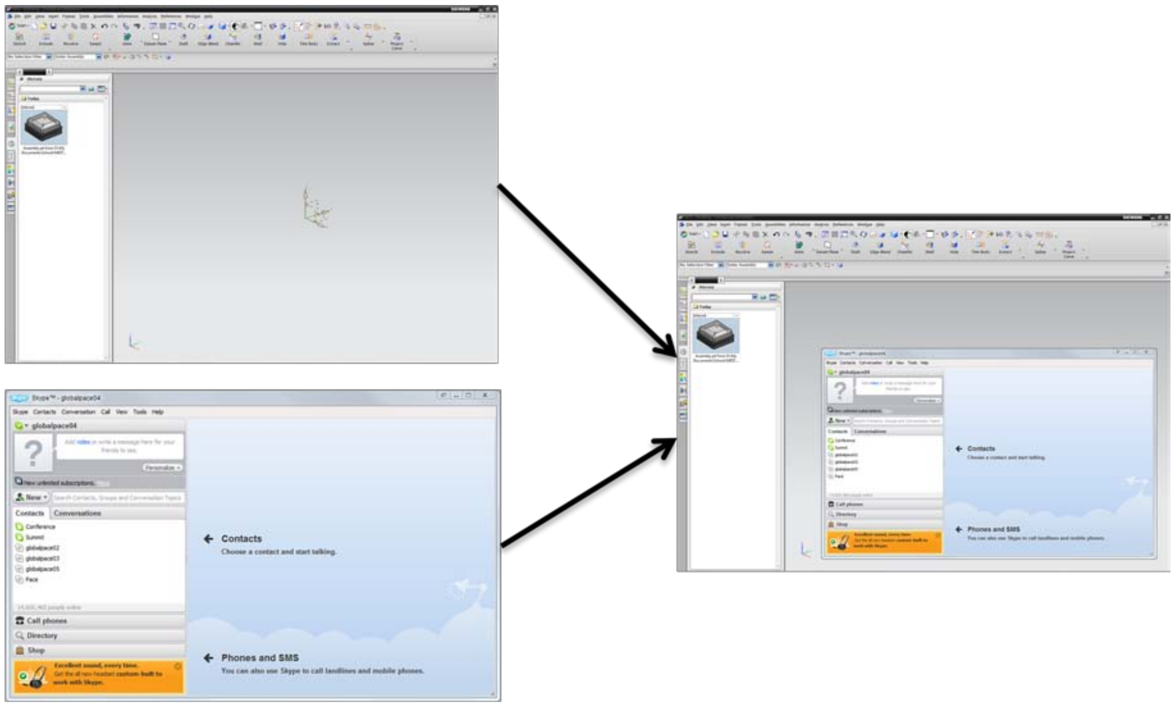

Skype™ - globalpace04
globalpace04
Contacts
Choose a contact and start talking.
Phones and SMS
You can also use Skype to call landlines and mobile phones.
Call phones
Directory
Shop

Figure 3-1: Integrated engineering and communication tools

The implementer should note that while most any communication tool can be integrated in some way, tools that include a robust API will allow for simpler implementation. APIs give more freedom to the integrating developer, and allow the communication package to function from within other programs.

Company Integration

Carefully considering the communication trends of the company and selecting a tool that matches as much as possible those trends is essential to the success of an implementation. Examples of these trends could include, but are not limited to the use of phone calls, email, instant messaging, SMS text, video conferencing and application sharing.

Another way the MCC method can integrate with the company is to integrate well with company policies on privacy and security. For instance, in some cases the conversations that pass through the system should not be captured, due to privacy reasons, security implications such as

IP and export regulations, or other reasons. It is thus essential that the implementation of the tool have a check to see whether data should or should not be captured. Some implementations may choose to automate this process, while others may want to leave the decision up to the user. There are many possible protection methods that could be effectively implemented, and so this feature is left up to the implementer to choose.

The principle of integration aims to encourage use of the method by allowing the user to stay in their daily system. Additionally, effective integration with company principles and policies will increase adoption and promote long-term use.

3.2.2 Principle #2: Make Data Retrievable

The second principle is to make data retrievable. The literature has discussed the need for this principle, and this is reported in section 2.1.3 above. Note that the most important operation of the presented method is the ability to retrieve data. Because of this, the implementer should constantly be striving to make data easily retrievable.

The method builds on this principle by specifying meta-data as a way to ensure data is retrievable. Meta-data is information that augments and helps explain data. In the MCC method, meta-data is information that helps to support and explain the context of conversations and other communication. It is essential to gather meta-data, since often raw conversation data cannot adequately explain itself. By nature, human conversation is quite unstructured (Ferrucci and Lally 2004), and meta-data serves to give context and structure to the data.

In addition, this meta-data serves to relate the conversation data to other types of DR. For example, a CAD designer who is working on a specific part while having a work-related conversation may be discussing the CAD part that is open on his screen. Ideally the file paths or server locations of these parts and even the geometrical features worked on during the conversation should be stored along with the data to assist in reuse.

3.2.3 Principle #3: Minimize User Interaction

The third principle is to minimize user interaction with the tool. Section 2.1.2 above describes that minimizing user interaction can help resolve some of the barriers to design rationale capture.

It is important to reduce direct user input as much as possible, since constant input from the user can cause annoyance over time. This method utilizes the division of meta-data into two types in order to ensure that this principle is met.

The two types of meta-data for the MCC method are automated meta-data and user-supplied meta-data. This division was created in order to give more freedom to the implementer, allowing them to select the level to which the user will be involved in the data structure. For some implementations there may be no user-supplied meta-data in an effort to fully automate the system. This would be ideal in more simplistic implementations, such as those involving only textual conversation. However, in the case of more unstructured communication, such as voice communication, some user-supplied meta-data may serve to define the data more accurately.

In some cases, direct input may be required. Because human beings have the ability to quickly analyze data sets and find patterns (Simon 1990), it enables them to input information more quickly than a computer could do the same. Asking for small amounts of user input can also significantly reduce development time for implementations, as it is often much less time-intensive to write user-input functions that it is to write extensive logic for parsing data and finding relationships. The direct interaction with the user should be minimized where possible, however, to ensure that the system does not become cumbersome over time.

One example of a way to balance the two types of meta-data would be to semi-automate some of the generation of information. As an example, consider an engineer who has been using a PLM software package to open PowerPoint files in a version-controlled environment. She has opened a few files and edited them, then checked them back in to the system. During the process she has had multiple conversations about the files with co-workers. The software might suggest to her that these PowerPoints be associated with the conversations in the system, presenting a list for her to see what files are being referred to. This would require only a single click of the mouse to verify, which would serve to minimize interaction.

3.2.4 Principle #4: Store as Much DR as Possible

The fourth principle is to store as much DR as possible in the system. The storage of all raw data along with its meta-data is essential to be able to reuse the data. As an example, consider a short film. If only shown a few frames of the film, it would be difficult to reconstruct the events that took place in the scene. However, if you were able to view the entire film, with every frame being shown you, you could easily reconstruct the events and explain them. The storage of as much DR as possible enables the capture of more frames of the film. For the case of design rationale, the product development process could be considered the film. This concept has roots in the literature, which states that one of the issues with PDM systems that resulted in the development of design rationale systems was that the PDM systems were not able to capture the intents and motivations behind designs. Stated another way, the PDM systems were not able to capture the in-between frames of the video.

The capture of this data can require a great deal of hard drive space, however. Fortunately, recent changes in trends have enabled computers to store a great deal of data inexpensively. A chart showing the cost of computing storage space over time is shown below in figure 3-2 (Historical Notes about the Cost of Hard Drive Storage Space 2008). Notice that the cost has reduced exponentially over the past 30 years. This inexpensive storage enables vast amounts of data to be stored at little cost.

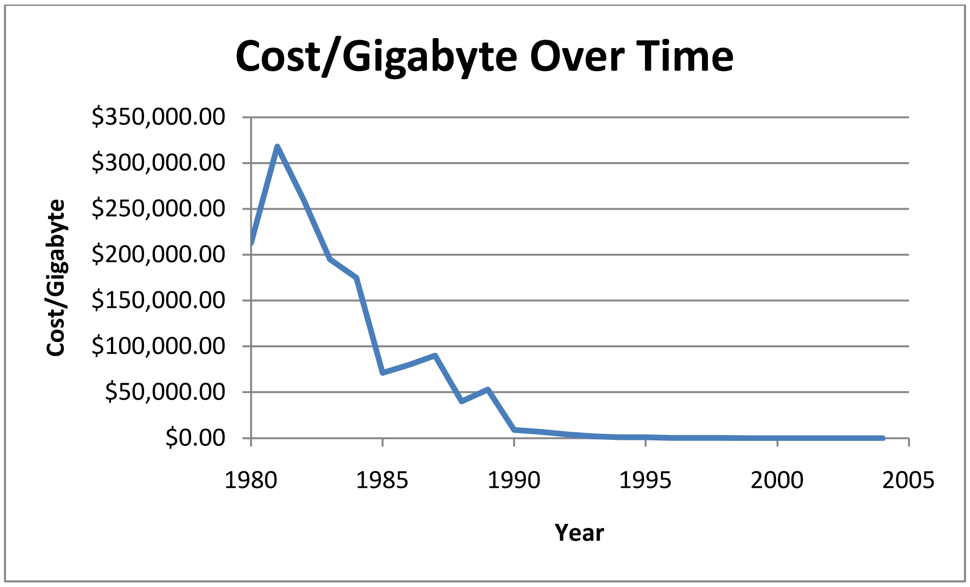


Figure 3-2: Cost/gigabyte of hard drive storage over time

Ideally speaking, any implementation of the MCC method should capture all of the communication pathways that are required for a company. For example, for a company that has selected Microsoft Office Communicator as its communication tool, the software should not only provide all functionality available in that tool, such as voice, video, instant messaging, screen sharing, and email, but should also enable the capturing of this information.

Additionally, this principle is only possible if the selected storage medium includes a few key features. The ability to store large amounts of data is essential, as mentioned above. The ability to add meta-data and relate it to the raw data is also essential. Next, the storage medium should be expandable enough to fit the needs of the company. Some companies may only require a simple storage system, while others may require more complexity. Lastly the database must have the ability to be automated. Since the MCC method takes advantage of the computer's ability to automate the tasks of storing and relating objects, this method would not be a with a system that required manual input of this information.

3.3 **Operation 1: Capturing the Design Rationale**

Now that the reader has a fundamental understanding of the principles that drive the MCC method, it is appropriate to discuss the core method. The core method helps to define exactly what is meant by DR capture and retrieval, and helps define the general steps needed to capture communication-based DR in the proposed integrated environment.

As mentioned the first operation of the core method is to capture the design rationale. Each of the steps of this operation is an essential component of the overall method, and each will be explained in the paragraphs below.

3.3.1 Capture Raw Design Rationale

The first step of the capture operation is to capture the raw communication-based design rationale. Consider C to be the set of all conversations participated in by a user U. Then C can be expressed as:

$$\{C : C1, C2, C3, \ldots, Cn \mid U \in C\} \tag{3-1}$$

The raw data is of little use without the additional capturing of information that explains this data and a primary principle upon which the method is based is to make data retrievable. The following steps ensure that this principle is satisfied.

3.3.2 Capturing Automatic and User-Supplied Meta-Data

If M is the set of all meta-data captured along with a conversation, then:

$$\{\mathrm{M} : \mathrm{M1}, \mathrm{M2}, \mathrm{M3}, \ldots, \mathrm{Mn}\} \tag{3-2}$$

Communication-based design rationale can now be defined using M and C. DR can be considered as the set of combinations of raw conversations with their respective meta-data, and can be written as:

$$\{\text{Design Rationale} : \mathrm{M} \cup \mathrm{C}\} \tag{3-3}$$

A brief note on the division of automatic and user-supplied meta-data will be beneficial to the understanding of the reader. Automatic meta-data is captured without user interaction and typically will take place in the background. The limits of what is captured automatically will vary widely depending on the implementation. User-supplied meta-data is captured explicitly from the user, and requires at least some human input.

The selection of whether to use automatic or user-supplied meta-data and how they are balanced is intentionally left up to the individual implementers and their needs. It is likely that most implementations will contain some elements of both.

3.3.3 Store Data in a Storage Medium

The last step of the capturing operation is permanently storing the information in some type of storage medium. The storage medium S is the set of all DR sets in the system, then:

$$\{\mathrm{S} : \mathrm{DR1}, \mathrm{DR2}, \mathrm{DR3}, \ldots, \mathrm{DRn}\} \tag{3-4}$$

The selection of which type of storage medium to use is entirely up to the implementer, as there are many equally useful tools for storing data in a computer. Please refer to principle number 4 above to ensure proper compliance with the method principles.

Note that the storage medium could also be an external DR management system. Many systems have proposed advanced methodologies for handling DR data, and a powerful tool could be created by coupling the strengths of these systems with the proposed integrated solution for capture.

3.4 Retrieving and Using Rationale

The retrieval of design rationale is a very important aspect of the method. The storage of information is only useful if it is stored with the intent of being reused. Each step of the retrieval operation is discussed below.

3.4.1 Retrieving the DR from the Storage Medium

The first step of the retrieval operation is to retrieve the data from the storage medium. If M is the set of all meta-data, C is the set of raw conversation data, S is the set of all design rationale sets, T is the subset of data that is returned from the set, and R is the search parameters, and in the storage system, then:

$$\{DR : M \cup C\} \tag{3-5}$$

$$\{S : DR1, DR2, DR3, \ldots, DRn\} \tag{3-6}$$

$$\{T \subset DR \mid M = R, C = R\} \tag{3-7}$$

If the search term R matches data from either the meta-data or is contained in the raw conversation data, then it is returned. This is further illustrated by the diagram below:

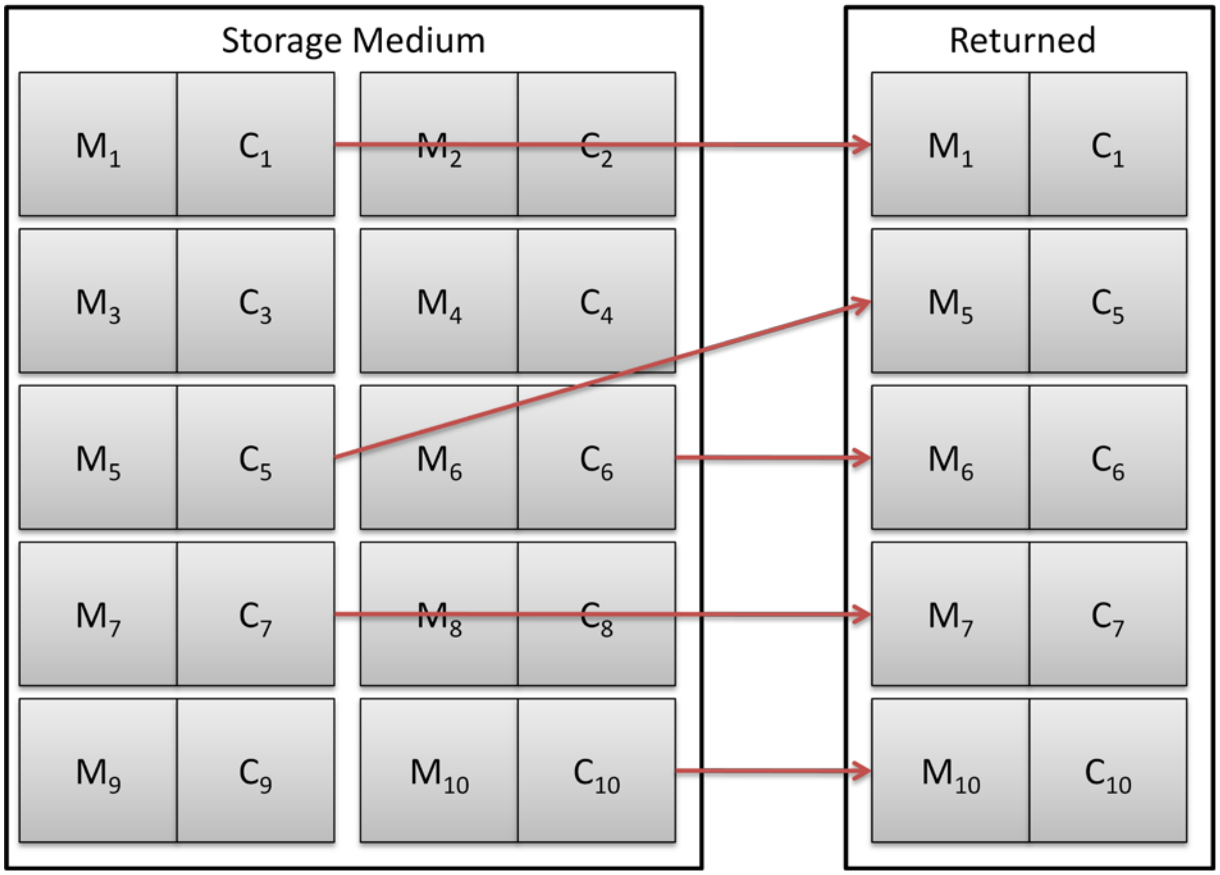


Figure 3-3: Only data matching the search are returned

Notice that the subsets that match the search term are extracted from the storage medium and returned to the user. The method by which data is retrieved from the storage medium is to be decided by the implementer. There are many equally effective methods for retrieving data, and they will vary greatly depending on the storage medium and the needs of the implementation. The search retrieval methods in other data and DR management solutions could also be leveraged effectively alongside this method. This method is intended to function either independently or when coupled with other systems.

3.4.2 Display DR Such That It Can Be Readily Reused

The last step of this operation is to display the data in such a way that it can be readily reused. The choice of how to display results, or the options for displaying results will be left entirely up to the implementer.

Some examples to illustrate this step of the method may be useful. One example would be returning the conversation DR with other related data and meta-data, such as CAD or analysis files, office documents and other items. Another example would be to sort retrieved data by various criteria. One criterion could be chronologically. Another criterion could be by project, which would present DR related to a specific project.

4 IMPLEMENTATION

This chapter discusses how the described methodology was used to create an integrated engineering and communication tool that is capable of capturing conversation based DR and linking it with engineering data (Mix, Jensen and Ryskamp, Using Global Communication Trends for Automated Design Rationale Capture 2010). This implementation was created with the intent of validating whether an integrated communication capture system can be functional as a DR capture solution. The implementation would likely require significant changes to be readied for the industry; however it does represent an adequate representation of the method, and so can be readily used to test the integration principle of the method. Hereafter this implementation will be referred to simply as the MCC tool (MCCT).

4.1 Architecture Overview

The MCCT tool uses Siemens' NX 6.0 software as the engineering tool, and Skype VoIP software as the communication tool. Additionally, Microsoft SQL Server was used as the storage medium. The architecture is shown below:

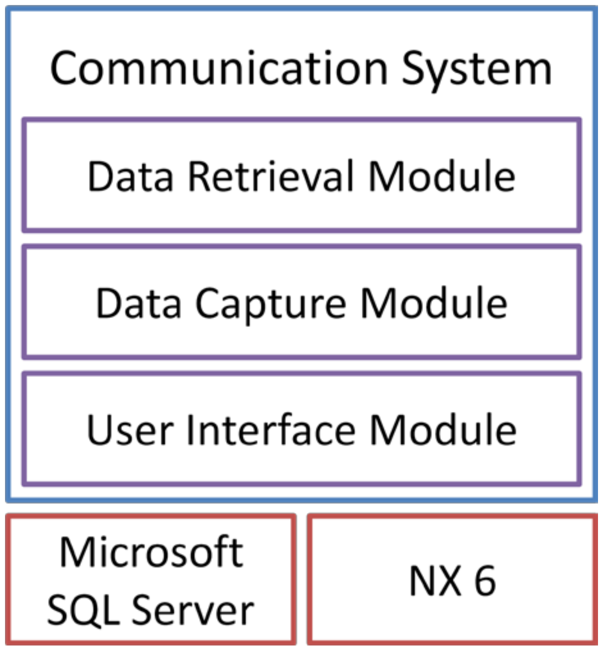


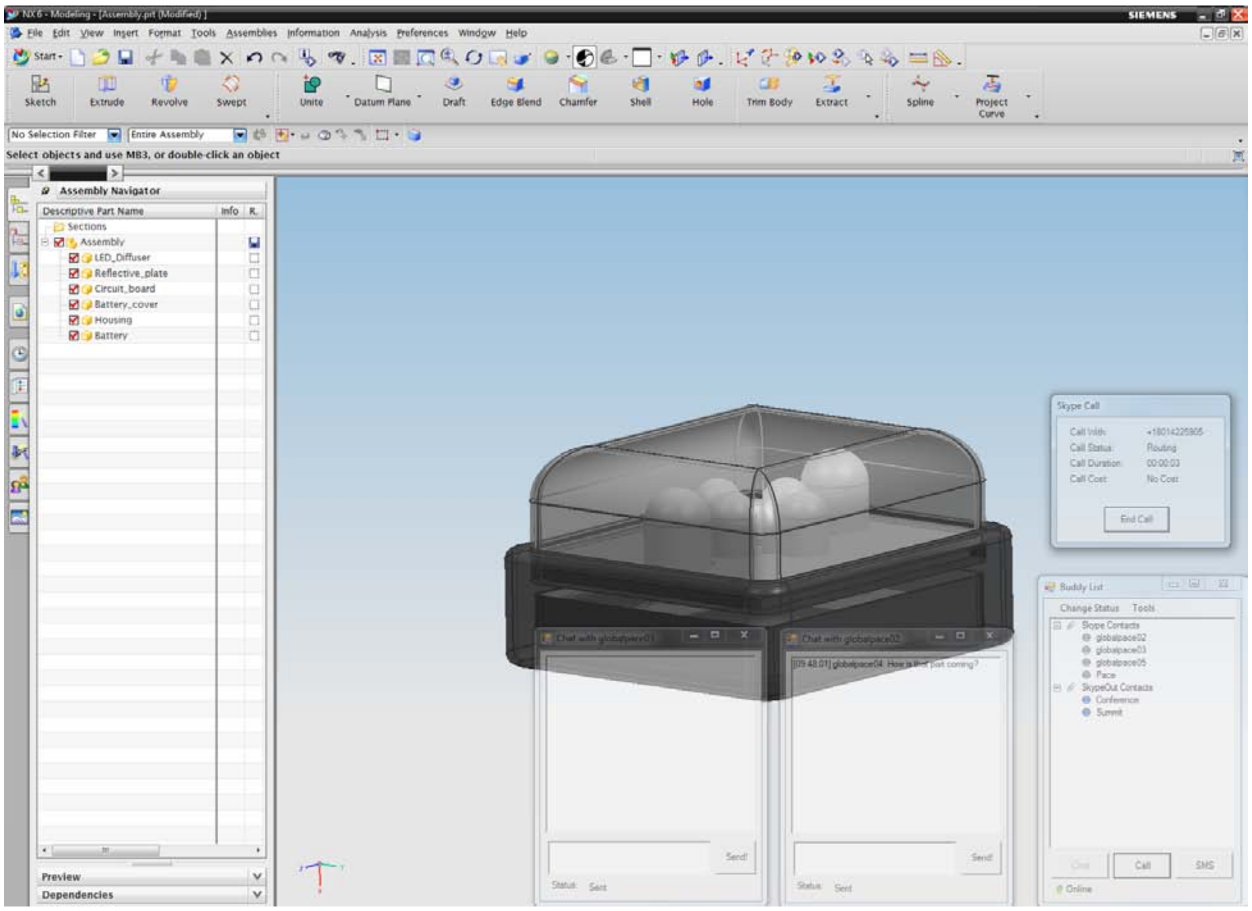
Figure 4-1: MCCT tool architecture

Notice that NX6 and Microsoft SQL server serve as a base, as the modules of the MCCT tool rely on both. The communication system of has three modules as shown.

The MCCT tool was created by following the steps of the method. Because of this, the presentation of this method will proceed as follows. First a discussion of how Skype is integrated with NX and how communication is performed will be presented. Following this, the operations of the core method will be discussed in order. The individual modules from the architecture and their purpose will be discussed in line with the method steps.

4.2 **Communicating**

After evaluating multiple VoIP tools, it was determined that the Skype communication tool possessed all the necessary elements denoted by the method. Specifically it provides a very robust API that allows most of the functionality of the Skype software to be used in a third-party package. This enabled the ability to integrate Skype directly into the NX interface. Integrating Skype adds communication functionality to NX, which enables the engineer working in NX to readily communicate with others without leaving the interface. A screenshot showing the communication interface running in the NX interface is shown in the figure below:



[09:48:01] globalpace04: How is that part coming?

Figure 4-2: MCCT tool running inside the NX interface

Notice that the communication tool is encapsulated in the NX interface and each window is transparent. This was done with the intention that the user would be able to continue their work as they used the communication tool. Each of the elements of this communication interface will be discussed in the sections below.

4.2.1 Launching the Communication Interface

The Skype interface can be directly launched from the NX interface by use of the NX toolbar. A screenshot showing this toolbar with the Skype button highlighted is shown below:

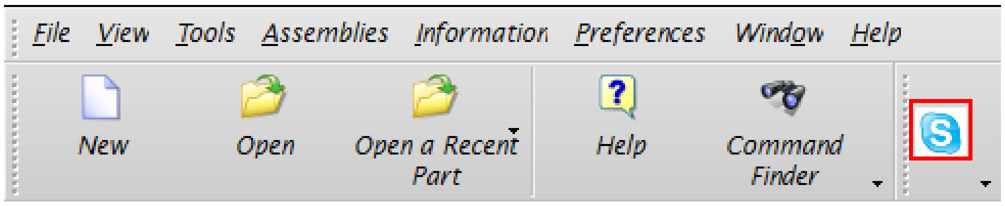


Figure 4-3: Skype button integrated into the NX toolbar

The button was added to NX through use of the NX configuration files. A menu configuration file is used to describe the target and icon for the menu item, and various other configuration files are used to place it in the toolbar.

When the button is clicked, a window is launched in the lower right corner of the NX window that shows a list of people who can be contacted. This list is termed the “Buddy List” for the MCCT tool. A screenshot showing the buddy list is shown below in figure 4-4.

Notice that the buddy list shows individuals who can be contacted. Next to their name is a colored circle representing their presence status. Green means they are online, red that they are busy, gray when they are offline, etc. The user can also set their own Skype status using the “Change Status” menu.

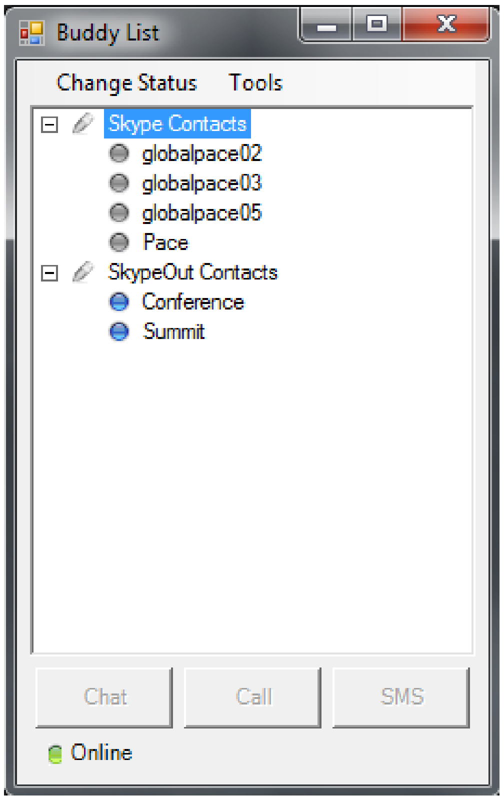


Figure 4-4: "Buddy List" showing users that can be contacted

The buddy list is dynamically populated from the Skype software each time the software runs. This is done using the Skype API and various Visual Studio tools for populating and updating forms. The Skype API is used to access to the names, usernames and statuses of friends in the Skype account that is currently active on the computer. The data is then used to populate the Visual Studio form.

The buddy list and all other user interface elements were created with Microsoft Visual Studio 2008's form designer. The form designer was used to add elements to the various windows, such as menus, buttons, lists, etc.

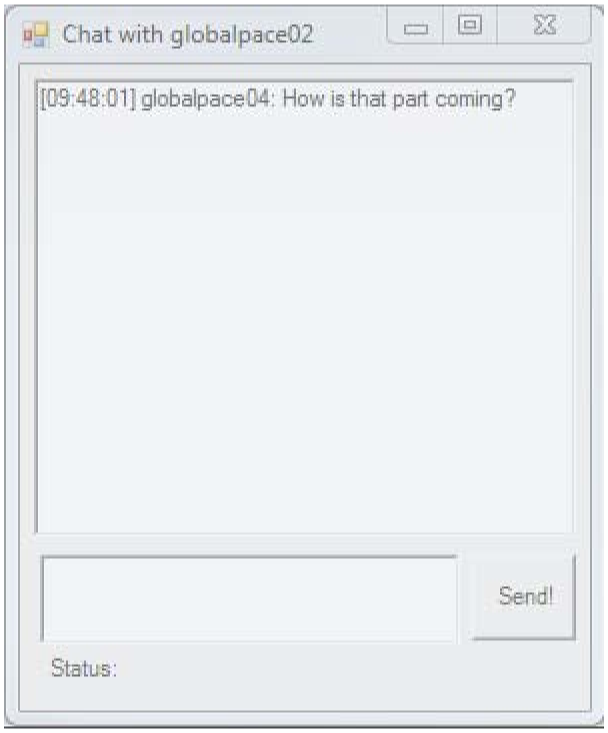
In the event that the buddy changes status, a pop-up window in the lower right informs the user of this change. This pop-up window contains the friend's username as well as the status they have changed to.

4.2.2 Using the Communication System

Once the buddy list is launched from the window, it can now be used to communicate. To initiate communication with someone in the buddy list, the user clicks on their name in the list. Each individual has different communication options depending on what type of contact they are and what type of information is in their account on Skype. For example, if they are a Skype contact, they can be instant messaged, by clicking on "Chat." They can also be called, and in the case where they are offline, the user can leave a voicemail. If they are a SkypeOut contact, meaning they only have a phone number associated with them, they can be called directly. If any user has a cell phone on file, they can be SMS text messaged. The buttons at the bottom of the interface activate and deactivate depending on which type of user is selected.

There are three communication options. One option is "Chat." Upon clicking a name and then clicking the chat button, the user is presented with the window shown in figure 4-5.

Notice that the window is transparent, so that the CAD data behind it can be seen and interacted with. This is an important feature of the tool, as it allows the user to continue work on CAD while communicating. The windows can be moved to any location on the screen. The user can type messages as with any instant messaging program and send them to the selected buddy. The buddy's messages appear on the screen as they are received.

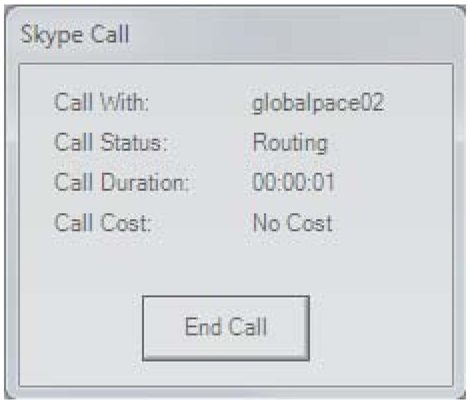


[09:48:01] globalpace04: How is that part coming?

Figure 4-5: MCCT tool chat window

The chat tool was also made possible with the Skype API. One excellent feature of the Skype API is event handling. Event handling allows a developer to trigger certain actions when an event occurs. For instance, when a message is received, the API recognizes this and a set of actions are performed, such as retrieving the message text and displaying it to the screen. Also, various function calls can be used to send messages, namely Skype.SendMessage(). In addition, the MCCT tool retrieves the history of a conversation. This allows a user to close the window and open it at a later time in the day and see the same conversation. When the user terminates the window, they are given the option of whether or not they would like to save the conversation in the database.

Another option the user has is to call a buddy. This can be a free phone call, in the case that both participants are Skype users, or a paid phone call, if calling an external number or cell phone. Rates for external calls are determined by the Skype software according to the locations of both call participants. After a user clicks the call button, they are presented with the following small window:



Call With:	globalpace02
Call Status:	Routing
Call Duration:	00:00:01
Call Cost:	No Cost

Figure 4-6: MCCT tool call window

Notice that the window informs them of who they are calling, how long the call has been happening, and the cost of the call. The cost increments constantly, showing a second by second update. This component is rather simple from a programming standpoint. A single Skype API call initiates the call, and the status is updated with an event handler. The duration display uses a simple timer from the Visual Studio Forms Toolbox, and the cost is based on the call rate retrieved from Skype and the duration timer. If the “End Call” button is clicked, the call terminates, and the user is asked if they would like to save the conversation.

The last option simply allows the user to send SMS text messages. This option is only available if a buddy has a cell phone listed in their account on the Skype network, since SMS messages cannot be received otherwise. This option looks and functions nearly identically to the chat window above. The major difference is a character counter that shows how many messages are being sent (the maximum characters for an SMS text message is 160). This is useful since each message that is sent is individually charged to the user's account. The Skype API enables the sending of SMS text messages through the Skype.SendSms() function.

4.2.3 User Interface Module Elements

The buddy list, the chat and SMS windows, and the call window make up the user interface module. In addition, there is an additional user tool available in the tools menu of the buddy list which allows a user to manage their buddy list by adding and deleting buddies directly from the MCCT tool, rather than having to use the separate Skype interface.

4.3 **Capturing Design Rationale**

The first step of the capture operation of the MCC method is to capture the design rationale from the user's conversations. As mentioned earlier, the call and chat functions each prompt the user to save conversation data when they close the window. If the user chooses not to save the conversation, the software simply returns them to the buddy list. If they choose to save the conversation, the MCCT tool then begins the process of capturing the rationale.

4.3.1 Capture Module Architecture

In order to fully understand the capture module, a brief explanation of this structure is necessary. The structure of the Microsoft SQL Server database for the MCCT tool is shown in the diagram below:

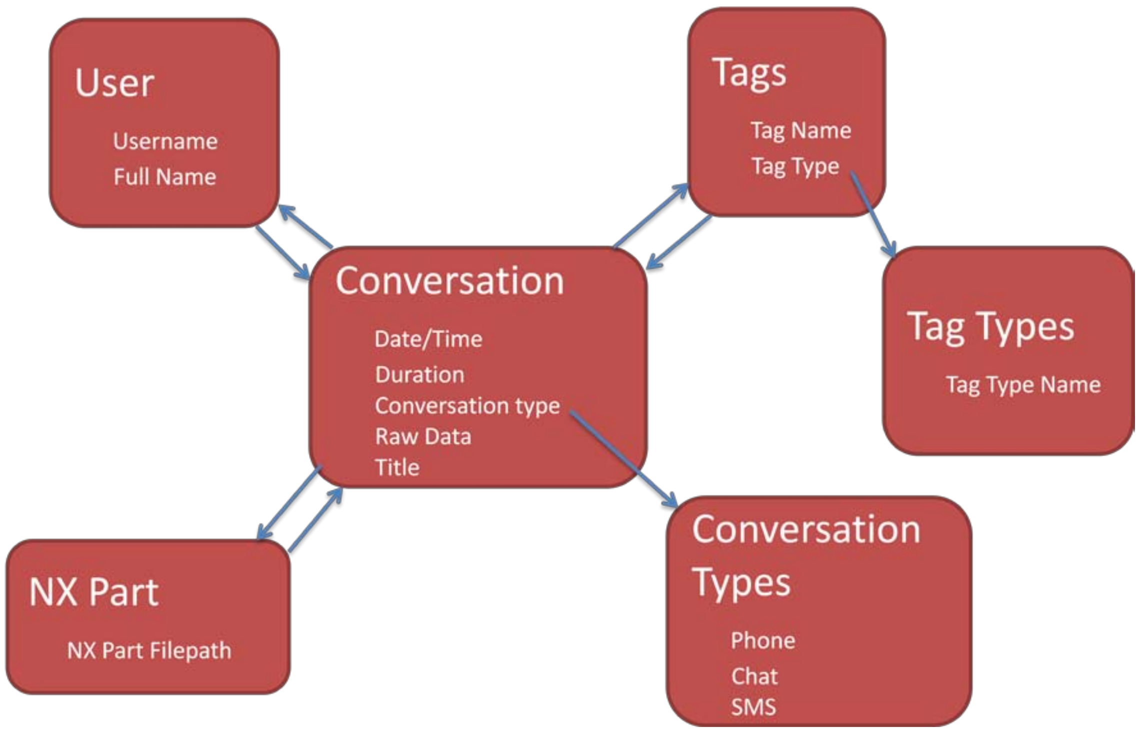


Figure 4-7: MCCT tool database structure

The conversation table serves as the central table of the database. This table stores some meta-data, such as date, time, duration, etc. In addition it stores all the raw data from the conversation. It is also linked to nearly all other tables of the database. Some of the other tables contain automatic meta-data that is gathered from the design session, while other tables, such as tags, are explicitly defined by the user. The process by which each table is populated will be discussed in greater detail below.

4.3.2 Capturing Raw Conversation Data

There are two types of raw conversation data that can be stored in the system: audio and text. When a user completes a phone call, audio data from participating parties is captured. Currently the MCCT tool only supports conversations between two people, although the ability to capture data from a conference call is also possible with identical function calls. Skype has built-in functions for recording audio data called Call.set_CaptureMicDevice() and Call.set_OutputDevice(). CaptureMicDevice() captures the audio that gets spoken into the computer's microphone. Set_OutputDeviceAudio() records the audio information that passes through the output device of the computer (i.e. speakers or headphones). This means that the data is captured separately, and must be mixed together for final storage. There are multiple methods available for doing this, including the one used for the MCCT tool (Sujoy 2007). Audio raw data is stored in the Microsoft SQL Server database in the conversation table. A table row that supports binary data is used to store the audio.

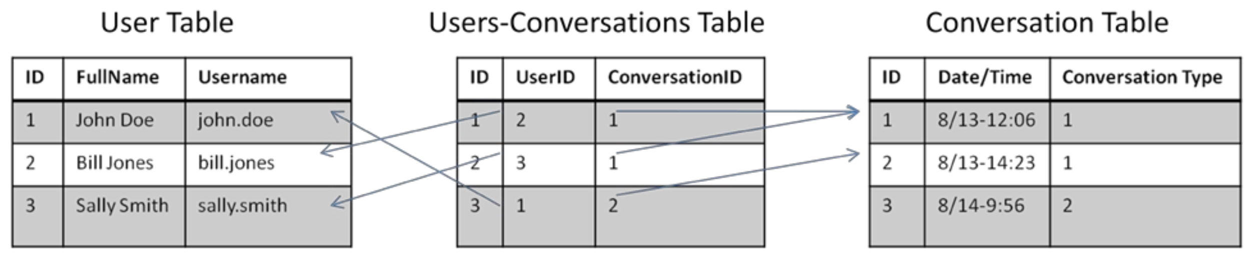
If a user completes an instant message conversation and chooses to save, the raw text data from the conversation is stored in the conversation table in a row that is formatted to accept textual data.

4.3.3 Capture Automatic Meta-Data

The next step of the capture operation is capturing automatic meta-data. The automatic meta-data that is captured is nearly identical between all conversation types. However, the method of linking each type of meta-data does differ, so each type will be discussed individually.

A few of the items captured automatically help make up the conversation table. As can be seen in figure 4-7 above, the conversation table has columns for date/time, duration and conversation type. Each of these columns is populated automatically when the user chooses to save the conversation. They are added to a row of the conversation table along with the raw data.

The next type of automatic meta-data item is the usernames of each person involved in the conversation. In this case, it was required to create a separate table called "User" which holds both the full names and usernames of each person that has participated in the conversation. The list of users in this table steadily grows, and each person in the list has a unique ID associated with them. Another table was created to store the relationship between users and conversations. The tables with example rows are shown in figure 4-8 below:


User Table

ID	FullName	Username
1	John Doe	john.doe
2	Bill Jones	bill.jones
3	Sally Smith	sally.smith

Users-Conversations Table

ID	UserID	ConversationID
1	2	1
2	3	1
3	1	2

Conversation Table

ID	Date/Time	Conversation Type
1	8/13-12:06	1
2	8/13-14:23	1
3	8/14-9:56	2

Figure 4-8: User, users-conversations, and conversation tables

In this case, there is a need to set up a "many-to-many" relationship between the data, since users will likely participate in more than one conversation, and the conversations will have multiple users. The relationship table in the figure, called "Users-Conversations" simply holds a User ID and Conversation ID in each row.

The last type of automatic meta-data that is captured is the file paths of currently open NX parts. These are stored similarly to usernames, since it is possible to have many NX parts associated with a conversation, and many conversations associated with a single NX part. A table named NXPart was created that had a one description column, namely "NXPartFilePath" that contained a text representation of the path on the system. A table named "NXPart-Conversations" was created with columns "NX Part ID" and "Conversation ID." When the user chooses to save a conversation, all open NX parts are linked to that conversation.

Next a conversation type is stored for each conversation. ConversationType is a static table that has just one description column, "Type." The first row type is "PhoneCall," the second row type is “InstantMessage” and the third row type is “SMS.” This table is referenced from the conversation table to determine what type of conversation is stored in each row.

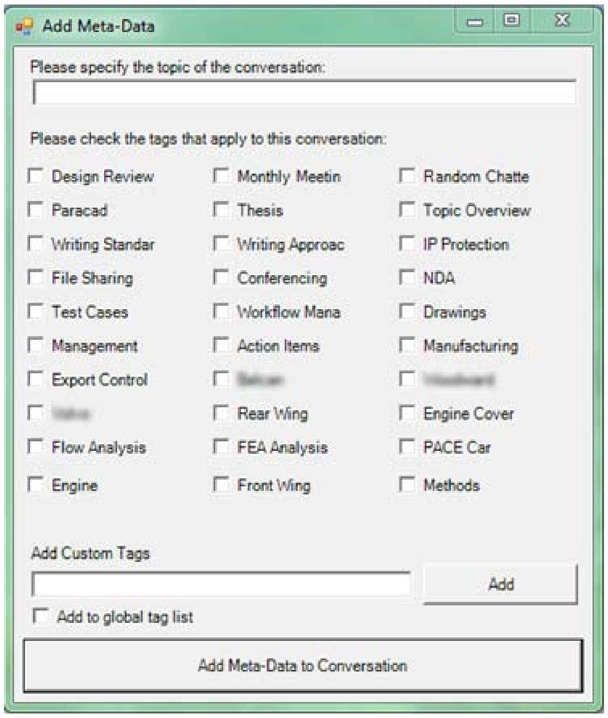
Note that for this implementation, all of this meta-data is gathered very quickly. There is no noticeable time-delay as these actions are performed. This builds on the principles of the method, to minimize user interaction and to automate as much as possible.

4.3.4 Capture User-Supplied Meta-Data

The next step of the capture operation is to capture user-supplied meta-data. This implementation attempts to follow the principle of minimizing user interaction as much as possible. However, some information supplied by the user was deemed beneficial to the ability to retrieve the data, another important principle. There are only two tables of the database that are populated with user-supplied meta-data. Each will be discussed in detail below.

In order to capture user-supplied meta-data, a custom form was created in Visual Studio. This form is shown below in figure 4-9. At the top of the form there is a field to input the topic of the conversation. In addition, there is a list of tags shown that have been added to the global database. The list of tags shown in the window updates each time the form is shown to include the latest changes in the database.

The purpose of these tags is to ensure that the principle of making data retrievable is met. In this implementation, much of the actual rationale is contained in the conversations, but often is embedded and not easily retrievable from viewing only the raw data. Tagging indexes the raw information much more effectively, and allows a user to find the data they need even if the raw data of the conversation does not contain certain keywords.


Add Meta-Data

Please specify the topic of the conversation:

Please check the tags that apply to this conversation:

☐ Design Review	☐ Monthly Meetin	☐ Random Chatte
☐ Paracad	☐ Thesis	☐ Topic Overview
☐ Writing Standar	☐ Writing Approac	☐ IP Protection
☐ File Sharing	☐ Conferencing	☐ NDA
☐ Test Cases	☐ Workflow Mana	☐ Drawings
☐ Management	☐ Action Items	☐ Manufacturing
☐ Export Control	☐ [redacted]	☐ [redacted]
☐ [redacted]	☐ Rear Wing	☐ Engine Cover
☐ Flow Analysis	☐ FEA Analysis	☐ PACE Car
☐ Engine	☐ Front Wing	☐ Methods

Add Custom Tags

Add

☐ Add to global tag list

Add Meta-Data to Conversation

Figure 4-9: MCCT tool user-supplied meta-data form

Tagging also can provide additional rationale, in the case that it is tagged with notes such as “decision” or “intention.” These might be special keywords that denote certain things that took place in the conversation. Note that for industry ready implementations, the indexing of the raw data would need to be more robust.

In the event that the user wants to add additional custom tag names that don’t already appear in the list, an interface at the bottom of the form provides this capability. If they choose, they can also add their tags to the global tag database, by clicking a checkbox and clicking the “Add” button. This is useful for building a library of tags and providing a consistently updating tag database. They then click the “Add meta-data to conversation” button to complete the process of supplying meta-data.

A “Tags” table was created that has two description columns, “TagName” and “TagType.” The TagName column contains the text description of the tags. TagType refers to a TagType table that is static and holds various tag types. This was intended for expandability, so that tags could be differentiated further.

There is a table called “ConversationsTags” that relates conversations with their tags. Again this is a many-to-many relationship very similar to the “UsersConversations” table above. The tags table is constantly updated with global tags input from users. This allows the system to be dynamic and constantly updated. It requires no IT management to add new tags to the database.

Again, this implementation attempts to minimize user interaction. Chapter 5 will discuss some illustrative examples of how quickly data can be captured. It will be shown that the time is very brief when compared to traditional methods of communication capture.

4.3.5 Store Rationale in Storage Medium

The last step of the capture operation is to store the rationale in the database. In the case of this implementation, this step occurred throughout the entire capture process, and so there is no discrete module for this part of the method.

The data and meta-data are stored in a central database. This allows for the capture of vast amounts of data, and always keeps tag lists and data up to date. Additionally, it allows for anyone in the corporation to access the data as they need it (provided they have access to the database). The implementation of the final operation of the method, retrieving rationale, is described in the next section.

4.4 Retrieving and Using Rationale

The final operation of the method is to retrieve and use the rationale that was gathered. This implementation provides a search system that allows a user to quickly search the database and retrieve conversations. The search system is simplistic, but provides an adequate representation of the method, and so is sufficient for evaluating how the tool might be used in industry.

4.4.1 Retrieving the DR from the Storage Medium

The first step of the retrieval operation is to retrieve the data from the storage medium. This is achieved through SQL query commands. In this implementation, the search system can be launched from the buddy list window by clicking the "Tools" menu and selecting "Search Conversation Data." The following window shows the search interface:

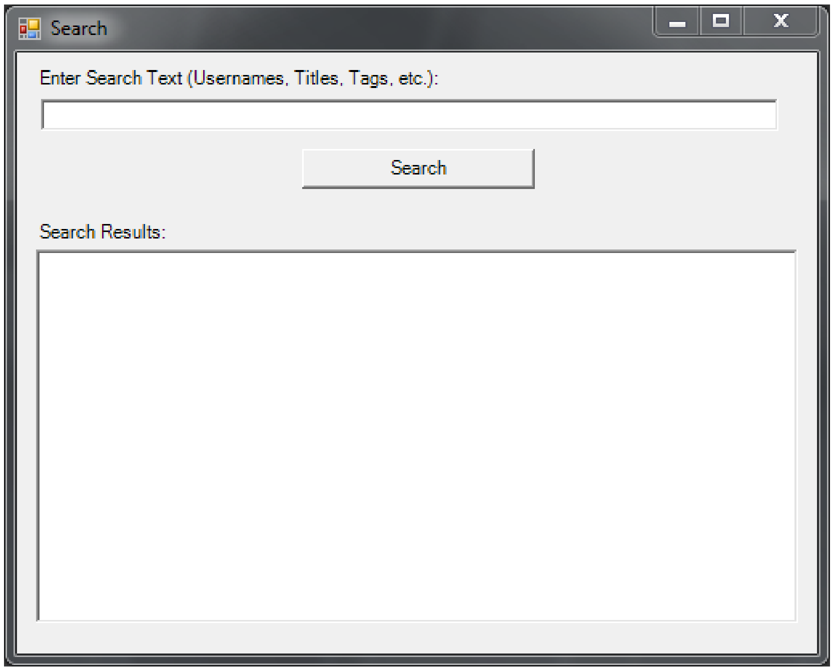


Figure 4-10: MCCT tool search window

There is a single field where a user can enter keywords and search for DR. The single field approach was chosen due to increased familiarity with and affinity to web-based search engines, which typically use a single field for search queries.

4.4.2 Display DR Such That It Can Be Readily Reused

The last step of the retrieval operation is to display DR data such that it can be readily reused to benefit future designs. The intention of the search tool was to ensure that users could quickly locate conversations stored in the tool. Because of this the data that is displayed to the user is relatively simple.

When the user enters a search term and clicks the search button, the database is queried. Currently the MCCT tool parses the following tables:

- Users
- Conversations
- Tags

If data is found in these tables that contain the keywords typed into the search bar, that data is returned. The data is organized by category, so the data is separated for the user before it is displayed to the screen.

The user can then view the data in the list below the search box. For textual data, they can click on a conversation and see it displayed on the screen. Because this implementation was focused primarily on testing the speed at which individuals could retrieve the data, audio playback support was not added.

4.5 **Implementation Summary**

The MCCT tool adequately mimics the method. It provides an excellent integrated solution for communication that allows the users to launch the system and communicate with others in their contact list while staying in the engineering tool. Additionally it follows the steps of the core method, as it allows them to store raw data in the database, including audio and textual data. It automates the process of adding meta-data quickly, and provides a rapid method for adding user-supplied meta-data to the system to assist with retrieval. Lastly it supplies a method to retrieve data.

The MCCT tool serves as an excellent example of an integrated solution for capturing communication-based design rationale. Chapter 5 will discuss this integration in more detail and will provide some additional tests that were performed that show how this tool could be used to benefit industry.

5 RESULTS

The MCCT tool was created in order to evaluate the MCC method and provide proof that the integration of engineering and communication tools for means of DR capture is achievable. This section outlines numerous tests designed to show that the integration is possible and to show how the tool might be used to benefit the industry. The goals of the testing were to answer the following questions:

1. Can the MCC method be effectively implemented in an engineering environment?
2. Is the MCCT tool more efficient than traditional methods for capture and retrieval?
3. Is the MCC method cost-effective as a DR capture solution?
4. Is the MCCT more user friendly than traditional methods?

For each question listed above, a test was designed that could either qualitatively or quantitatively answer the question. The tests were based on the principles upon which the MCC method is built. Test one aims to answer question one, and aims to show that engineering tools can be integrated with communication tools by using their application programming interfaces. Test two answers question two and focuses on ensuring that the tool satisfies principles two through four of the methods chapter, by ensuring that data can be retrieved effectively and efficiently, and that the capture of the data indeed does minimize user interaction while storing as much communication-based DR as possible. The third test is designed to address question three, and to address specifically the integration into the company, ensuring that the tool can fit well within a company from a cost-effectiveness standpoint. The fourth test answers question four, and ensures the basis behind principle three is satisfied, that tools that are not natural and easy to use will be neglected. Each test will be discussed in the paragraphs below.

5.1 Test #1: Can the MCC Method Be Effectively Implemented?

The implementation itself serves as its own test in this case. The test was simply to see whether it was possible to successfully integrate communication tools with engineering tools, capture communication-based design rationale, and provide for retrieval.

As discussed in chapter four, the NX – Skype implementation was successful through the use of their respective APIs. The MCCT tool is capable of providing communication functionality to a user directly from the NX interface, which follows the integration principle of the method. It also allows for the capture and storage of communication raw data along with associated meta-data. Lastly it provides a solution for retrieving this information from a storage medium.

5.2 Test #2: Is the MCCT Tool More Efficient Than Traditional Methods?

This test was designed to show an example of how this tool can improve on current industry methods for communication capture. While some tools exist for capturing communication information, such as the basic functionality of recording phone calls, there are few industry-ready tools available that capture this information and correlate it with engineering-specific meta-data. Additionally, for many engineering firms, including the sponsors of this research, the tools available to the individual engineers do not allow capture of even raw communication data. Communication data is often captured informally by taking notes from a meeting or conversation and emailing them to colleagues or in some cases storing them on a central server. Often however, communication-based DR is often not captured at all, but resides in the minds of the engineers who made the decisions or created the designs.

Even with conversation recording systems there are many problems. For instance, the data is in some cases not centrally stored by default unless configured programmatically. Secondly, engineering meta-data is not stored alongside the data.

The method of note taking has many problems with both capturing and retrieving communication-related DR. One problem is that it takes considerable time to take notes and transcribe them in a form that can be centrally stored and searched. Additionally in the case of a non-centralized storage system, such as notes in a notebook or email, retrieval is slow or often impossible. One way that this method improves upon note-taking is that in many cases notes only allow for a single person’s view of a meeting’s events to be captured, and often do not portray an accurate depiction of the actual process. The MCCT tool allows for the entire conversation to be archived and indexed, allowing the full set of events to be recorded.

Two sub-tests were created to see if the MCCT tool solved these problems. The first refers to the efficiency with which the MCCT tool can store conversation-based DR with associated meta-data. The test was designed to compare the MCCT tool to the traditional note-taking method, due to the fact that this is a method with which most engineers are very familiar, which made the testing process very simple for participants. The second test was designed to analyze the efficiency of the MCCT tool’s retrieval from a centralized server, as compared with traditional methods that utilize decentralized storage.

These tests are not intended to compare the purposes of note-taking directly to the purposes of the MCCT tool. Note-taking has many different purposes besides data archival, such as showing a concise summary of events, or highlighting especially important moments in the conversation. This research does not seek to replace the taking of notes; it only seeks to provide a better solution for archiving the sequence of events and design rationale that can be gleaned from a meeting.

5.2.1 Capture Efficiency Comparison

This test was performed with nine young engineers who are familiar with computing environments in general. While a test with engineers unfamiliar with computing could show some interesting data, it was determined that typing speed and general computer familiarity would provide a fair comparison for the two methods.

Each participant was asked to assume that they had just completed an hour-long meeting, and had generated a single page of notes from that meeting. The notes were written neatly on a sheet of paper and given to each participant. They were then asked to transcribe the notes into a digital format and email this digitized file to five people. This part of the test mimics the current practice of many engineering firms, to share notes after a meeting as an informal way of storing DR. Each participant was timed as they performed these activities.

Following this, the participants were then asked to use the MCCT tool to capture the data from the meeting. A mock phone call was set up when they arrived and they were told that the meeting had just completed. They were then instructed to end the phone call and follow the prompts on the screen to document the conversation. They were shown the user-supplied meta-data screen described in chapter 4, asked to supply a title for their conversation and check a few tags that related to the mock meeting to complete the archival process. All of these instructions were given them before beginning the test, and only supplemental assistance was given where needed. They were timed as they performed these actions.

The result from these two activities is shown in table 5-1. The taking of notes and emailing them is simply referred to as "Note-Taking."

Table 5-1: Comparison of time (in minutes) required for capture

Participant	Time required for note-taking	Time required for MCCT
Participant 1	0:10:32	0:01:29
Participant 2	0:08:15	0:01:27
Participant 3	0:10:54	0:01:00
Participant 4	0:07:06	0:01:02
Participant 5	0:09:02	0:01:19
Participant 6	0:06:00	0:02:18
Participant 7	0:09:38	0:00:45
Participant 8	0:11:50	0:00:26
Participant 9	0:08:18	0:00:45
Average	**0:09:02**	**0:01:10**

On average the MCCT tool was roughly 8 times faster than the note-taking method. The time required to perform the capture of DR is significantly less with the MCCT tool than with the traditional note taking method.

This data takes into account that the time each participant saw the MCCT tool was when they tested it. Their times using the tool would likely greatly decrease as they became more familiar with it.

Also of note, the amount of time required for the note-taking is proportional to the amount of notes recorded. The test performed was for a single page of notes, but for two or three pages of notes, the time would increase close to two or three times. This is primarily due to the time-consuming nature of transcribing. The time required for the MCCT tool does not increase for longer conversations or meetings significantly, which allows it to perform more efficiently in any scenario. This is because the speed at which the computer records the data increases only a negligent amount when handling more data.

5.2.2 Retrieval Efficiency Comparison

The next test of efficiency is directed at retrieval. This test is intended to provide an example for how the MCCT could provide an improvement in retrieval time over current methods. There are two basic categories for storage systems. The first is a de-centralized system, meaning that the data is spread across multiple locations. This method makes retrieval more difficult, since a person seeking data must find the right person before they can find the right data. The second type is a centralized storage system, where most people in an organization have access to a single source of data. This type makes retrieving information much easier, since a person only has to visit a single place to get the information they need.

The note-taking method by nature does not automate the process of capturing the data. In some cases policies within a company can dictate the storage of this information, and often it is stored on a central file server. File servers excel at storing data centrally, but often are poorly managed and are difficult to parse by hand and search effectively.

In many cases, however, the note taking method does not store data in a central system. Often email is used to distribute documents from one place to another. Sometimes notes are stored on local machines and not distributed. Worst of all, conversation-based DR often resides in notebooks of employees or remains undocumented.

This decentralized nature of the note-taking method hinders the retrieval of data. Often it can take days to weeks to retrieve needed information, since it requires knowing the right people to ask, which can often be difficult and time-consuming. Because of the time required for this process, most people will end up re-inventing the wheel for their project, rather than spending the time to find the data that could be used to recreate work.

The decentralized method is difficult to test, since it is difficult to recreate the complicated industry environment that causes these challenges. Instead, the researcher evaluated the sponsor company to find how long this type of data typically took to retrieve. In one example, involving locating a person within the company who was responsible for a specific software area, it took nearly two months to locate the correct person. It is easy to see that with the MCCT tool, it would be a simple search about that particular software area that would return a high volume of conversations from a small group of people. This would assist in finding the right person quickly.

To test the MCCT tool against note-taking methods, the participants were asked to locate a conversation in the database using the MCCT tool. They were asked to give the conversation title that they supplied in the previous test to see if they could retrieve it from the database. In every case, the MCCT tool successfully found the data they were searching for and showed the data to the user. A table showing the amount of time for each user to retrieve this data is shown in table 5-2.

On average, it took under 30 seconds to retrieve the conversation from the MCCT tool. The fastest user was able to find the data in 10 seconds, and the slowest took 52 seconds. The retrieval time represented a significant improvement over traditional note-taking methods. This is primarily due to the centralized nature of this data, as well as the efforts made in the creation of the MCCT tool to include meta-data such as a conversation title and tags.

Table 5-2: Retrieval times for MCCT search

Participant	Retrieval Time
Participant 1	0:00:15
Participant 2	0:00:52
Participant 3	0:00:51
Participant 4	0:00:16
Participant 5	0:00:16
Participant 6	0:00:10
Participant 7	0:00:24
Participant 8	0:00:13
Participant 9	0:00:18
Average	**0:00:23**

The researcher recognizes that the time required for users to find data might increase if a different set of participants were used for the retrieval test. The test was designed to emphasize the advantages of centralized storage for data retrieval. The centralized nature of the data storage system, as well as the tagging system that helps with the indexing and retrieval of the data would likely still represent a significant improvement over de-centralized systems if the test were repeated with different participants.

5.3 Is the MCC Method Cost-Effective as a DR Capture Solution?

The purpose of the next evaluation was to illustrate some of the potential cost savings that could result from implementing a centralized DR capture and storage system. This evaluation was performed since a primary factor of adoption into industry is cost. It is thus important to attempt to quantify these costs.

The software costs for the MCC method will differ depending on the implementation. However, we assume in the principles of the method that we want to match as much as possible the current software set of the company. From a licensing standpoint, this means that the company will not have to spend additional dollars to acquire the tool. However, some money will need to be spent in development in order to integrate the communication and engineering tools and create a system that follows the steps of the method.

While detailed information about time savings for an industry standard tool has not yet been determined, this section will attempt to quantify some of the potential cost savings for a company from the results of the previous capture comparison tests. These numbers are simply illustrative, and are shown here to further emphasize the potential benefits of using the MCCT tool.

In order to calculate the potential cost savings there are certain assumptions that should be stated. The first is that average engineer will spend at least one hour in their day in meetings, from which an average of one page of notes will be generated. The second is that an engineer's wage per hour, including the cost of overhead, is assumed to be $90/hour (Occupational Employment and Wages: Aerospace Engineers 2008).

Consider that the average for capturing information from the note-taking method was roughly 9 minutes. The average for the MCCT tool was roughly 1 minute. The difference between these times is about 8 minutes. If we use the simple equation:

$$C = W * T \tag{5-1}$$

Where C is the cost for the company, W is the hourly wage, and T is the time. Using the assumption of a page of notes per day, then the average daily savings for each engineer is roughly $12. This is a significant amount, considering that this translates into a yearly savings of roughly $3000 per employee. When considering a large engineering firm with many engineers on staff, this savings would be very significant.

Obviously this is not the only time savings that could be considered. Retrieval, while not as easy to quantify, is drastically improved with the MCCT tool. The time required is significantly less than with a decentralized system. Other factors, such as the reduction in cost for using newer and cheaper communication systems, such as VoIP, are also difficult to quantify, but would ultimately lead to greater savings.

5.4 Is the MCCT Tool Easier to Use Than Traditional Methods?

This test was aimed at illustrating that the MCCT tool could improve on current methods in the area of user satisfaction. The test was performed using a simple survey. Each participant was asked to rank each method at the conclusion of the above tests in the area of ease of use. They specified a score between one and five for each method, one being very difficult to use, and 5 being very easy to use. The results are shown in the table below:

Table 5-3: Ease of use scores comparison

Participant	Note-Taking	MCCT
Participant 1	3	5
Participant 2	4	4
Participant 3	3	5
Participant 4	4	5
Participant 5	2	5
Participant 6	3	5
Participant 7	3	5
Participant 8	1	5
Participant 9	2	5
Average	**2.78**	**4.89**

Notice that the MCCT tool consistently scored better in the test. The average score for the MCCT tool was nearly twice as high. This is likely due to the time-consuming and mundane nature of transcribing notes, and the hassle of locating these notes. The score for the MCCT tool would likely increase with the users familiarity with the system.

5.5 **Results Summary**

Each of the initial questions that were asked was answered through the series of tests. Chapter 6 will discuss the conclusions drawn from each of these tests, and provide some suggestions for future work in this area.

6 CONCLUSIONS

This chapter will focus on conclusions drawn from the testing of the MCCT tool. First, there will be a few sections that address each conclusion drawn from the testing and discuss the benefits that can be gained from implementing the MCC method. Second, some suggestions for future work will be provided.

6.1 **Test Conclusions**

This section focuses specifically on tests from chapter 5. Each conclusion is a response to the questions asked in chapter 5.

6.1.1 MCC Method Can Be Effectively Implemented into a Tool

It has been shown that the MCCT tool is a successful implementation of the MCC method. As discussed, the implementation used Siemens NX 6.0 software as the engineering tool, and Skype VoIP software as the communication tool. The two were successfully integrated such that Skype resided in the NX environment. Additionally, the tool acts very well as a communication capture tool.

The implementation of this specific set of tools also provides additional insight into the future integrations of this method. Specifically this test shows that any set of engineering and communication tools could be integrated, provided they have APIs that allow their internal functionality to be accessed.

6.1.2 The MCCT Tool Can Be More Efficient Than Traditional Methods

On average, the speed at which data was captured was about 8 times faster with the MCCT tool than with the traditional method. Additionally retrieval time was reduced by orders of magnitude, as the traditional methods often took weeks to months to retrieve data, while the MCCT tool took on average less than 30 seconds.

From these tests I conclude that MCCT tool can provide an efficient way to capture and store conversation-based DR as compared to current methods. Additionally it provides functionality not available in current methods.

6.1.3 The MCCT Tool is a Cost-Effective Solution for DR Capture and Storage

It has been illustrated that the additional time savings provided by the MCCT tool can translate into greater cost savings overall. The time saved from both capture and retrieval is significant and can also provide significant savings. Additionally, chapter 5 included a discussion of the savings from other sources, such as the savings realized by changing to VoIP software and other factors.

From this I conclude that the MCCT tool has the potential to provide a significant reduction in day-to-day engineering costs if implemented correctly. Retrieval time will be reduced which will also help to shrink cycle times and lead to better products overall.

6.1.4 The MCCT Tool is Easier to Use than Other Methods

A survey taken from participants who had used both methods showed that the ease of use of the MCCT tool outranks traditional methods two to one. It was shown that this is primarily due to the fact that the MCCT tool saves the engineer time and does not cause significant irritation. From this I conclude that the MCCT tool represents a more user friendly from the standpoint of the engineer.

6.2 **Suggestions for Future Work**

While the focus of this research has been to capture communication-based DR, the method and tool presented in this paper have other possible areas of application. I leave it to the effort of other researchers to extend the work that I have presented. A few suggestions for future work are mentioned in this section in order to provide a starting point for interested researchers.

The communication capabilities that are a part of the MCCT tool provide for some additional research opportunities. Implementing video and screen sharing into the MCCT tool would be an excellent way to capture more detailed communication. These additional communication pathways could also be used to gather more detailed information about the product development process, especially the capture and categorization of application sharing information. Ultimately the addition of these functionalities would provide a better tool for DR capture.

Another research activity would be to augment the functionality of the MCCT tool to include more detailed CAD data in the capture. For example, one could tie conversations directly to the features of an NX part, so a person could review the exact features that were being created during the conversation, which would likely provide additional insight as to why certain design decisions were made. Another augmentation would be the automatic categorization of conversations based on content. This could be readily implemented for textual communication, and possibly for verbal communication provided that audio transcription technology becomes more sophisticated. Currently this is not possible, since these systems generally are inaccurate (Holstein and Gubrium 2003).

Another area where this work could be readily applied is in the improvement of global engineering. Globally dispersed product development teams face challenges in communication, both in language and in cultural differences that could be assisted by the capture of communication data. The ability to record and replay conversations from meetings and other encounters would allow these teams to review and verify the communication of expectations. This could help alleviate some of the miscommunication that often hinders global product development.

This work could also be readily applied to a new and exciting field, multi-user engineering design. Today engineering software only allows a single user to edit a single file at a time. However recent research has focused on allowing these tools to accommodate more than one user at a time. This would allow parts to be edited and created by a larger group of people, ultimately leading to faster design times. Communication would be an essential component of this new tool. The MCCT tool could function extremely well as a way to provide an integrated communication tool for multi-user tools. It could also provide a way to capture the actions and conversations that take place in the multi-user design session. This would provide a history for how parts were created and would assist a user in reviewing the events that took place.

The culture of the workplace and its effect on communication tools has been briefly discussed in this thesis; however a deeper study into the adoption and use of these tools in industry would benefit this work. A full test of this tool in the industry would provide additional insight into the most effective way to have the MCCT tool adopted into industry.

REFERENCES

Alic, J.A. "Computer-assisted everything? Tools and techniques for design and production." *Technological Forecasting and Social Change*, 1993: 359-374.

Anderl, R., and R. Mendgen. "Parametric design and its impact on solid modeling applications." *ACM Symposium on Solid and Physical Modeling archive.* Salt Lake City, Utah: ACM, 1995. 1-12.

Atwood, Michael E., and John Horner. "Redesigning the Rationale for Design Rationale." *Human-Computer Interaction. Interaction Design and Usability.* Beijing, China: Springer, 2007. 11-19.

Bracewell, R., K. Wallace, M. Moss, and D. Knott. "Capturing Design Rationale." *Computer Aided Design*, 2008: 1-14.

Conklin, Jeff, Albert Selvin, Simon Buckingham Shum, and Maarten Sierhuis. "Facilitated hypertext for collective sensemaking: 15 years on from gIBIS." *Proceedings of the ACM conference on Hypertext and Hypermedia.* 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, T., and C. Begg. *Database Systems: A Practical Approach to Design, Implementation, and Management.* Boston: Addison-Wesley, 2009.

Dankwort, C. Werner, Roland Weidlich, Birgit Guenther, and Joerg E. Blaurock. "Engineers' CAx education—it's not only CAD." *Computer-Aided Design*, 2004: 1439-1450.

Dutoit, Allen H., Raymond McCall, Ivan Mistrik, and Barbara Peach. "Rationale Management in Software Engineering: Concepts and Techniques." In *Rationale Management in Software Engineering*, by Allen H. Dutoit, Raymond McCall, Ivan Mistrik and Barbara Peach, 1-52. Netherlands: Springer-Verlag Berlin Heidelberg, 2006.

Elanchezhian, C., T. Sunder Selwyn, and G. Shanmuga Sundar. *Computer Aided Manufacturing.* New Dehli: Laxmi Publications, 2007.

Elliot, Bern, and Steve Blood. *Magic Quadrant for Unified Communications.* Gartner, 2009.

Ferrucci, David, and Adam Lally. "UIMA: an architectural approach to unstructured information processing in the corporate research environment." *Natural Language Engineering*, 2004: 327-348.

Gao, J. X., Hayder Aziz, P. G. Maropoulos, and W. M. Cheung. "Application of Product Data Management Technologies for Enterprise Integration." *International Journal of Computer Integrated Manufacturing*, 2003: 491-500.

Gruber, Thomas R., and Daniel M. Russell. *Design Knowledge and Design Rationale: A Framework for Representation, Capture, and Use.* Stanford, CA: Stanford University, 1991.

Grudin, Jonathan. "Evaluating Opportunities for Design Capture." In *Design Rationale: Concepts, Techniques, and Use*, by Thomas P. Moran and John M. Carroll, 453-470. Hillsdale, N.J.: L. Erlbaum Associates, Inc., 1996.

Hejlsberg, A, S. Wiltamuth, and P. Golde. *The C# Programming Language.* Boston: Pearson Education, 2006.

Hicks, B. J., S. J. Culley, R. D. Allen, and G. Mullineux. "A framework for the requirements of capturing, storing and reusing information and knowledge in engineering design." *International Journal of Information Management*, 2002: 263-280.

Historical Notes about the Cost of Hard Drive Storage Space. January 21, 2008. http://www.littletechshoppe.com/ns1625/winchest.html (accessed January 12, 2010).

Holstein, J.A., and J.F. Gubrium. *Inside Interviewing: new lenses, new concerns.* Thousand Oaks, CA: Sage Publications, 2003.

Horner, J., and M. Atwood. "Effective Design Rationale: Understanding the Barriers." In *Rationale Management in Software Engineering*, by A.H. Dutoit, R. McCall, I. Mistrik and B. Paech, 73-90. The Netherlands: Springer Berlin Heidelberg, 2006.

Kho, Wookyun, S.A. Baset, and H. Schulzrinne. *Skype Relay Calls: Measurements and Experiments.* New York: INFOCOM Workshops 2008, 2008.

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

Komerath, N. *Knowledge Management Techniques in Experimental Projects.* American Society for Engineering Education, 2001.

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

Lee, Jintae, and Kum-Yew Lai. "What's in Design Rationale?" In *Design Rationale: Concepts, Techniques, and Use*, by Thomas P. Moran and John M. Carroll, 21-51. Hillsdale, NJ: L. Earlbaum Associates, Inc., 1996.

Marguerie, Fabrice, Steve Eichert, and Jim Wooley. *LINQ in Action.* Greenwich, CT: Manning Publications, 2008.

Mix, Kenneth, C. Greg Jensen, and Jordan Ryskamp. "Using Global Communication Trends for Automated Design Rationale Capture." *Tools and Methods for Competitive Engineering.* Ancona, Italy: TMCE, 2010.

Mix, Kenneth, Greg Jensen, and Jordan Ryskamp. "Automated Design Rationale Capture within the CAx Environment." *Accepted for Publication in Computer-Aided Design*, 2010.

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

Myers, Karen L., Nina B. Zumel, and Pablo Garcia. "Automated Capture of Rationale for the Detailed Design Process." AIII Press, 1999.

Occupational Employment and Wages: Aerospace Engineers. May 2008. http://www.bls.gov/oes/2008/may/oes172011.htm (accessed December 2009).

Orenstein, David. *QuickStudy: Application Programming Interface(API).* January 10, 2000. http://www.computerworld.com/s/article/43487/Application_Programming_Interface (accessed January 12, 2010).

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

Ramesh, Balasubramaniam, and Kishore Sengupta. "Multimedia in a design rationale decision support system." *Decision Support Systems*, 1995: 181-196.

Regli, W. C., X. Hu, M. Atwood, and W. Sun. "A Survey of Design Rationale Systems: Approaches, Representation, Capture and Retrieval." *Engineering with Computers*, 2000: 16: 209–235.

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

Sachpazidis, Ilias, Roland Ohlb, George Kontaxakisc, and Georgios Sakasa. "TeleHealth networks: Instant messaging and point-to-point." *Nuclear Instruments and Methods in Physics Research*, 2006: 631-634.

Sandvine Inc. *Research Report: 2009 Global Broadband Phenomena.* Basingstoke, U.K.: Sandvine Incorporated, 2009.

Shah, Jami J., and Martti Mantyla. *Parametric and Feature Based CAD/CAM.* New York: John Wiley & Sons, Inc., 1995.

Shipman, Frank M., and Raymond J. McCall. "Integrating Different Perspectives on." *Artificial Intelligence for Engineering Design Analysis and Manufacturing*, 1997: 11(2), 141-154.

Simon, Herbert A. "Invariants of Human Behavior." *Annu. Rev. Psychol.*, 1990: 41:1-19.

Stallings, William. *Business Data Communications.* Upper Saddle River, NJ: Pearson Education, 2009.

Sujoy, G. *The Code Project.* July 19, 2007. http://www.codeproject.com/KB/cs/WAVE_Processor_In_CSharp.aspx (accessed January 14, 2010).

Takahashi, Kenji, and Eiji Yana. "A Hypermedia Environment for Global Collaboration." *IEEE MultiMedia*, 2000: 7(4), 36-47.

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.