Formal Methods
For Design Automation Application Development

by

Hillel E. Bachrach

B.S. Mechanical Engineering, with high honors
School of Engineering and Applied Sciences,
Columbia University in the city of New York, 1995

B.A., Physics
Yeshiva University, 1995

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 1997

Signature of Author:.....
Department of Mechanical Engineering
May 15, 1997

Certified by:.....
Anna Thornton
Assistant Professor of Mechanical Engineering
Thesis Supervisor

Accepted by:.....
Ain Sonin
Chairman, Department Graduate Committee

Formal Methods
For Design Automation Application Development

by

Hillel E. Bachrach

Submitted to the Department of Mechanical Engineering
on May 15, 1997 in Partial Fulfillment of the
Requirements for the Degree of Master of Science

ABSTRACT

Companies are expending significant resources to improve their product design processes. A widely adopted approach is to utilize software that automates tasks in the engineering design process. Many Design Automation applications, such as knowledge based design systems, implement company specific design knowledge, so they are usually unique developments. However, companies are not consistently successful in developing applications that provide significant benefits. In many cases, developers blame the failures on the ad hoc approach to selecting projects, developing software, and deploying the applications. They feel that a more formal approach to developing these applications would help them avoid or minimize these problems.

This thesis describes a benchmarking study of companies in several industries to identify current best practices for developing the applications. In addition, a review of the current literature on the subject was conducted. Based on the study of current practices, past research, and the experience of applications developers, a plan is synthesized to improve the likelihood of success in application development. The plan incorporates formal methods for carrying out opportunity evaluation, project selection, knowledge capture, and application deployment into the current development process.

Thesis Supervisor: Anna Thornton
Title: Assistant Professor of Mechanical Engineering

ACKNOWLEDGMENTS

This work was funded by MIT's Leaders for Manufacturing Group 6-7, *Integrated Analysis and Development Group*.

I want to thank my advisor, Anna Thornton, for her guidance and support in this research endeavor. Many thanks to all my office mates, especially Brad and Mark, who really added life to the office.

I also want to thank developers who gave me their time and experience throughout the project: Mike Harper, Rick Gammons, Rahul Bhatt, Diane Francis, Ken Kuna, Robert Perfetti, Hewon Hwang, Steve Woods, and Steve Martz. Special thanks to Steve Woods for his support above and beyond the call of this research. Thanks also to Jane Page of Concentra Corporation for help finding published case studies.

Special thanks to Avigayil and Baila who tore through the thesis for me.

Last and far from least, I want to thank Mandy for seeing me through some pretty trying times.

TABLE OF CONTENTS

Chapter 1 Introduction

1. Improving Design Through Automation

Companies that develop new products stay competitive by reducing the cost and cycle time of their product development process, while improving or at least maintaining the quality of their products. Usually, however, improved quality requires longer development cycles and costs more. Studies have shown that nearly eighty percent of the product development cost is committed during the design of the product (Suh et al., 1990). Subsequent development activities, from testing through manufacturing, are constrained by decisions that were made in the design of the product. Changes to the design in later stages due to considerations such as ease of assembly or manufacture are usually difficult and costly. If a company can improve these original design decisions, the resulting design can more easily meet higher levels of quality and manufacturability without the need for costly changes. The company will be able to improve quality, decrease cost, and reduce development time as its designs improve.

Research in engineering design has shown that good design decisions depend on the availability of information and experience that will provide the engineer with an understanding of the rules and requirements that constrain a design (Pahl and Beitz, 1988). This information, also called *design knowledge*, may come from disciplines outside the expertise of the design engineer, so the engineers need a way access to this information in order to integrate it with their own expertise. In concurrent engineering, for example, engineers from different disciplines work together on the design, so that their collective knowledge is available and used from the start of the design process. Computer technology stands out as a way to further organize and make available the wide range of information that is of value in making design decisions. A widely adopted approach is to develop software that can automatically complete tasks in the design process. Companies call these systems *design automation* applications.

Design automation applications have been developed to configure and design products, select components, compare the characteristics of competing designs, ensure that designs meet established guidelines, manage product data, and automate CAD activity. These applications have been implemented using spreadsheets, standard programming languages, special purpose languages, macro-programming languages associated with CAD systems, and specialized development environments. Companies are especially interested in systems that can incorporate best design practices, company procedures, and design guidelines from other disciplines into the design process. Such systems have been documented as reducing design time and errors by orders of magnitude (Marra, 1997; Procter, 1995), and as providing consistent, rapidly modified component designs. Applications that incorporate design knowledge from many domains into the automated

design process are often termed "knowledge based" design systems, because the knowledge that facilitates design is organized in a knowledge base. The system applies this knowledge to specific problem information supplied by the design engineer to consistently create quality designs.

Automating tasks that are part of a specific design process requires in depth knowledge of how that design process is carried out. The knowledge base of the application will contain best practices and procedures that are specifically for that design task. As a result, these are usually developed by the company that owns the process being automated, and cannot be used by others or even for other activities in the company. In some cases the company hires a consultant to develop the application.

A survey of developers at several companies found that they have had varying degrees of success at developing design automation applications. While there have been many that work and provide large benefits to the company, there have also been development attempts that did not pay off because the developed applications were not valuable, were not or could not be used, or just did not work. The developers believe the root of these problems to be the ad hoc way that applications are developed. Guida and Tasso (1994) also observed that automation projects fail not because of a lack of technical know how, but because of poor management of the development process. If a formal, structured method could be used for selecting, implementing, and deploying applications, problems such as those mentioned above could be avoided or managed. This thesis is motivated by an expressed need to understand the best ways to develop these systems.

2. Research Methodology

The research was based primarily on interviews and surveys of company practice. People involved in the development process were interviewed, and follow up discussions and visits were held. In addition to explaining their companies' development approach, the developers gave their opinions on the value of existing practices and of suggested methods. In addition, a literature search provided ideas for formal methods that supplement current good practices.

This project was funded by MIT's Leaders for Manufacturing (LFM) program, a partnership between MIT and twelve U.S. manufacturing firms. These firms are: Aluminum Company of America, The Boeing Company, Chrysler Corporation, Digital Equipment Corporation, Eastman Kodak Company, Ford Motor Company, General Motors Corporation, Hewlett-Packard Company, Intel Corporation, Motorola, Polaroid Corporation, and United Technologies Corporation. Half of these companies participated in this research, and they are described as companies A, B, C, and so on. In addition, consultants at several leading engineering automation firms provided opinions, on condition of anonymity.

The first step in the research was to survey companies to learn how they currently develop these application and what obstacles they have faced in creating successful applications. The companies' approaches to these development activities were researched and compared, to determine current good practices. Case studies of activity at the companies were written and analyzed to identify where a more formal approach could combat the problems developers had identified. These are compiled in Appendix A. In addition, the literature on design automation application development was reviewed and promising concepts in the literature were identified. Formal methods for carrying out these activities were synthesized by organizing the current good practices and good concepts seen in the literature into a structured development process. The formalized activities were linked to the current development process to form a development plan that provides direction to improve each stage of development.

3. Automating The Design Process

Developers of design automation applications are creating applications that depend on the information and procedures that are used in carrying out the task to be automated. They must organize this design knowledge into a model of the task that can then be used to build an application that carries out the task. In many companies and in past research this process has been called *knowledge engineering*. Some background is presented here, to acquaint the reader with the field of design automation.

3.1 Origins of Automation

Automating design activity was first studied in the development of practical applications of artificial intelligence. One application has been the completion of activities that require human reasoning by computer systems that simulate human intelligence. Researchers worked to develop computer programs that can handle problems whose resolution requires the experienced application of rules of thumb to problem data, guiding the expert to an empirically correct solution. Successful research applications began to emerge in the 1970s when the first *expert systems*, or *knowledge based systems*, were developed. The field expanded, and by the late 1970s many more systems had been developed, including some that were used commercially. Examples of initial applications that reach the stage of commercial use are SPE for distinguishing between causes of inflammatory conditions in a patient; XCON for configuration of Digital Corporation's VAX computers; and DENDRAL, for identifying molecular structures from mass spectral data.

Surveys of knowledge based systems conducted by Harmon (1990) and Durkin in 1992 (Durkin, 1994) demonstrate the dramatic growth in knowledge based systems development. Dozens of applications that were produced in a fields from aerospace and manufacturing to insurance and legislation were at least partially successful at meeting the artificial intelligence goals. It is noteworthy, however, that in the list of nearly 200 systems given by Waterman (1986), only eight were at the stage of commercial

application. Edward (1991) compares several surveys and questions the reliability of estimates of operational systems in commercial environments; he feels the numbers were inflated.

These systems were developed in a variety of fields. Within engineering many, like COMPASS (developed by GTE Laboratories and described in Prerau, 1990), were developed for diagnosing problems. A few, like XCON, were developed for component configuration. None automated the synthesis of new designs or product geometry. Towards the end of the 1980's new approaches to automating design and manufacturing were being explored, approaches that incorporated knowledge and reason into the design process, but were different than other systems. To understand these differences and the current state of design automation, one first must understand how these systems worked, and how systems to automate design were different.

3.2 Expert Systems

Components of Expert Systems

The systems were called expert systems, or knowledge based systems, because the knowledge of an experts in a field was gathered into a knowledge base, to be used to solve the problem like the expert would. The primary components of these systems were the knowledge base, an inference engine and the working memory, shown in Figure 1. The ***knowledge base*** stored all of the information relevant to solving problems in the domain of interest. The developers gathered together the facts, data and rules of thumb that the expert used to solve the problem. This *domain knowledge* and specific techniques and strategies for solving problems were organized into the knowledge base, often in the structure of rules that described what decisions to make on the basis of given facts. The ***inference engine*** controlled the use of the knowledge in the knowledge base by using a reasoning mechanism (Guida and Tasso, 1994) to decide, for example, which rule should be executed next. The ***working memory*** stored all information relevant to the problem being solved. Thus the user input was stored here, and the solution to the problem was placed here.

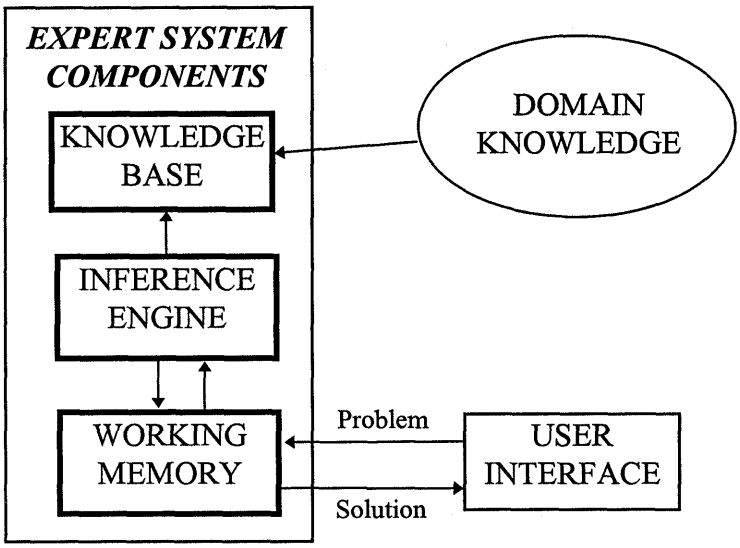

Figure 1 Basic Components of Expert Systems

Expert Systems Characteristics

As the development process matured, it became accepted that there were characteristics inherent to these systems that made their development different from the development of conventional software, as shown in Figure 2. All of these differences are due to the unique way that knowledge of a task is structured and utilized. For this reason, the development of these applications, from gathering knowledge to writing the computer code, has been called *knowledge engineering*.

Expert Systems...	*Conventional Systems...*
employ heuristics	are algorithmic in nature
are modular, easily changed	not modular, hard to change
use symbolic reasoning	make numerical calculations
deal with uncertainty and inconsistencies	need complete information
explain decisions	give result only
separate knowledge from its control	couple knowledge and control in the code
find acceptable solution	give the best or only solution
allow a natural dialogue interface	require exact commands

Figure 2. Comparison of Expert Systems to Conventional Software, based on Waterman (1986), Prerau (1990), Edwards (1991) and others.

3.3 Expert Systems For Design

Not all design activity can be automated. Researchers noted early on that the design is a high level, cognitive process that is grounded in many disciplines (Pahl and Beitz, 1988).

Designers have a poor understanding of the mental process they use (Miles and Moore, 1994), making it difficult to gather the rules and steps that they follow in creating a design. In attempting to define what kind of design work was suited for automation with expert systems, researchers identified areas that seemed appropriate by modeling the design process (Brown, 1983 and Dixon, 1986) and by attempting to automate some tasks (See for example Dym's descriptions of PRIDE in Mittal et al, 1987 and Dym and Levitt, 1991). A division of the design process that looks at the life cycle of design has been used to identify where the design activity is formal enough to be described completely by the designers. A more in depth description is found in Rychner (1988) and Dym and Levitt (1991). Hopgood (1992) also discusses different design activities and how they might be automated. Briefly, the design life cycle can be divided into five stages of design activity.

- **Conceptual Design:** When a design problem is proposed, engineers first generate concepts that meet the functional requirements of the problem, without identifying the detailed form that the ideas will take. The high level of abstractness makes modeling a standard approach questionable (Brown, 1983), although some continue to work on this problem (Miles and Moore, 1994).

- **Preliminary Design:** To demonstrate how a proposed design concept might work, the engineers begin to synthesize the components that comprise the design. To some extent the problem is subdivided, and methods or components are considered for meeting the requirements of each subproblem. Some preliminary design work is creative and as such, it is difficult to automate. However, where the preliminary design work follows a standard procedure, the process may be a good candidate for automation.

- **Detailed Design:** At this stage in the design process, choices made in the preliminary design are refined, and details such as specific parts and parameter values are determined. The procedures followed are heuristic in nature, as the designer is guided by experience in making design decisions. These activities are often good candidates for automation, provided that the design task process is well understood, as is discussed in Chapter 5.

- **Analysis and Optimization:** Throughout preliminary and detailed design, design parameters must be calculated and tested. This process is often algorithmic in nature, and usually consists of performing calculations that test the properties of a design. In addition, design engineers often skip calculations and rely on their best guess to choose a design parameter value. These activities may be good candidates for automation because they are consistent and well understood. An analysis application can make these calculation quickly so that the engineer knows he or she is using the correct value.

- **Documentation and Detailed Project Planning:** The product design process, like all projects, needs to be planned and organized. To ensure that goals are met, procedures are put in place for documenting plans and for generating the necessary drawings and data. The procedural nature of these tasks makes them good candidates for automation. In addition, as the design process becomes more computerized, and there is more product data to be efficiently managed and controlled, computer applications become the only way that the product data can be managed.

3.4 Problems Automating Design

By the end of the 1980's it became clear that there were certain complexities to modeling design that made it more difficult to implement systems for design than for other fields. Many promising research ideas failed to pay off as usable commercial applications. Developers explained that several difficulties resulted from the fact that the design process differs from other processes.

- **Difficulty articulating the design process:** It is very difficult for the design engineers to describe the detailed steps that are part of the design process (Pahl and Beitz, 1988; Miles and Moore, 1994). As a result of this inability to articulate the steps of design, developers had great difficulty capturing the decisions and heuristics used by the experts. While prototypes seemed to indicate that an application could be developed, making an accurate full scale model proved to be extremely difficult.

- **Knowledge and control cannot be separated:** Developers often had trouble developing a knowledge base in which the facts and rules were not coupled. This meant that the modularity of the application, especially important in a knowledge based design system, was compromised.

- **Formatting of data:** Design data often includes the description of geometry. As companies shifted to CAD models, these systems needed to be able to interpret incoming geometric data, and either provide geometry as output, or link easily to existing CAD systems.

- **Insufficient computing power:** In many cases, the design task contained computations that required more computing power than was available from computers at that time.

- **Poor development process:** As will be examined in Chapter 2, ad hoc development processes led to the selection of tasks that were poorly suited for automation, and the poor planning of these projects.

3.5 Knowledge Based Design and Design Automation

Towards the end of the 1980's new approaches to modeling design knowledge were being tested, and prototype applications were developed. These applications were focused on what was described earlier as routine design, and directly addressed the problems described above. Companies built knowledge based design applications that incorporated geometric data and used techniques for structuring the knowledge that were better suited for design. To do this they used development tools like ICAD (Concentra Corp.) and Concept Modeler (Wisdom Systems), which directly addressed geometry modeling and also provide a high level language of design for easier representation of product structure and design constraints. Others, like STONErule (Prescient Technologies) incorporated product modeling capability into existing CAD systems. Tools were also developed using the programming languages that came with CAD systems, and with conventional programming languages.

As computers grew more powerful, and computer aided drafting grew to replace hand drawings, design automation applications were developed to ease the drawing process, to manage the data associated with these computer models, and to analyze design properties. To distinguish the budding field of design automation from past work in artificial intelligence, developers redefined "knowledge based design systems" to be those in which structures other than rules were used for modeling the design process (Davies and Anderson, 1991). Not all of the applications being developed can be classified as knowledge based design applications, but all are systems that employ design knowledge to carry out activity that was previously done by hand.

4. Summary

Design automation applications are being used to improve the product development process. Their development process is currently not structured, and projects often do not produce an application that works as expected. The remainder of the thesis reports on research carried out to identify how the development process can be improved, and the formal methods to do so. This research is presented in the remaining chapters, as follows:

- Chapter 2, "Formalizing The Development Process", presents the current development process, and the proposal for improvement through the use of formal methodologies for parts of the process.

- Chapter 3, "Organizational Structures For Development," describes the effect of the company's organization for developing design automation applications has on the kinds of methodologies that may be used for different stages.

- Chapter 4, "Opportunity Evaluation", presents a structured approach to considering how design automation technology can be implemented.

- Chapter 5, "Task Review and Selection", presents methods and considerations for assessing the feasibility and worth of a candidate task so that the selected project will be likely to succeed and be of value to the company

- Chapter 6, "Knowledge Acquisition", presents methods for carrying out the capture of design knowledge.

- Chapter 7, "Deployment," presents a formal approach to considering what happens before, during, and after a design automation application is introduced into the design environment. Suggestions for approaching these activities are given.

- Chapter 8, "Conclusion", presents a brief summary of this research, direction for how this research can be utilized by developers, and suggestions for future work.

Each chapter consists of five sections. The objective of each chapter is summarized in an introduction. In the second section, current practices are presented. In the third section concepts for formal methods in the literature are described. In the fourth section the current practices and concepts in the literature are analyzed, and suggestions for formal methods that will improve the development process are presented. The chapters conclude with a summary of how the methods will improve that step in the development process.

Chapter 2 Formalizing the Development Process

1. Introduction

Developers of design automation applications believe that the development process could be improved if formal methodologies were utilized. However, the different development processes currently in use have never been reviewed to find where the formal methodologies could help. In order to determine how the development process can be improved, the current development process needs to be reviewed, and activities that can be improved need to be identified.

The current development process is presented and analyzed in Section 2. Literature on application development is reviewed in Section 3. Suggestions of where formal methodologies can improve the development process are presented in Section 4. The chapter is summarized in Section 5.

2. Current Activity

2.1 The current development process

Developers at several companies were asked to describe how they develop applications, from selecting the project through deploying the applications. The developers described a development process that consists of four basic steps, as illustrated in Figure 3.

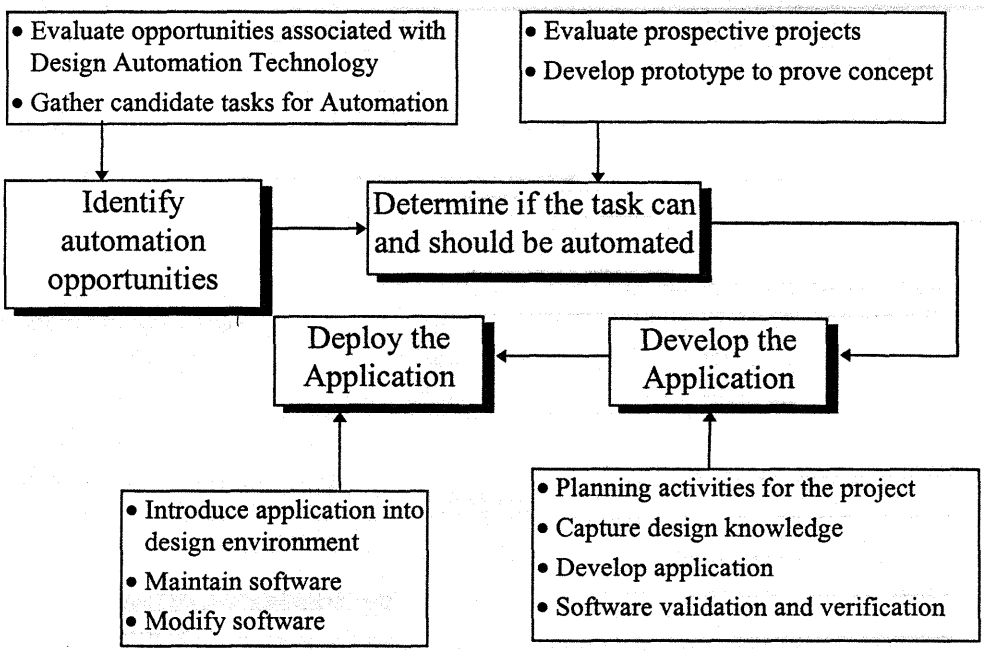

Figure 3 Current Development Process

The details differ from company to company, but the figure shows generally the activities that comprise each of these steps. Although developers did not explicitly include deploying the application as a step in the development process, problems with deployment were identified, so it made sense to incorporate it in the development process being studied.

Developers were also asked about the quantity of work and the use of formal methods for carrying out each of the activities. Their responses are shown in **Table 1**, and are on a scale from 0 (no effort) to 5 (significant effort). The activities of **Figure 3** are described in detail in subsequent chapters, and summarized below.

Table 1 The use of methods during development

Formal Methods in the Current Development Process	***A***	***B***	***C***	***D***	***E***	***F***	***G***
Quantity of Activity (0 = none, 5 = a lot)	3	4	3	5	5	2	0*
Task identification methods	2	0	4	2	0	1	1
Opportunity evaluation methods	3	3	3	4	4	3	1
Task selection methods	4	0	4	3	1	1	2
Methods for planning development	4	3	4	4	2	2	2
Knowledge acquisition techniques	1	0	2	2	0	0	1
Use development tools	2	1	3	4	5	1	2
Methods for developing software	3	2	3	4	2	1	1
Methods for validating software	5	5	4	4	5	3	2
Methods for deploying applications	3	2	4	3	2	3	1

* This company has placed all development on hold

Overview of Activity

Projects take from a few weeks to several years, with many being six to twelve months long. While all of the companies are developing knowledge based design (KBD) applications, some companies are doing more work than others. Company A listed several applications, with very little knowledge based work going on. Company B indicated a lot of work is being done, some of which is in expert systems (not for design). Company C is working on projects in different groups, and so ten or more projects are completed or being worked on. Company F listed two or three applications being developed, compared with more than a dozen at both companies D and E. The companies that are doing a lot of development have dozens of developers working on different projects, while the companies doing only a few projects have five or six people. Company G placed all development on hold. While no explanation was forthcoming, the costly failure of the first design automation application developed is likely to be at least partially responsible. Although there was insufficient data from the company for it to be included in the tables for each chapter, the developer who was interviewed provided some information on the company's activity.

Identify automation opportunities

Two activities that make up this step are opportunity evaluation and task identification. Some companies evaluate the opportunities presented by design automation technology. Developers said that to evaluate the opportunity to utilize design automation technology, they consider the choices of automation techniques mentioned in the introduction. They also consider where these techniques can be applied to the company's product design process. A third area of consideration is the resources in the company that can be harnessed for developing design automation applications.

The companies in the survey do not use formal methods to search for candidate problems. In most cases, people at different levels in the company suggest design activities they want improved, or applications that they think would be useful. The use of design automation is mostly driven by engineer's needs, and not pushed by the technology. Developers described several ways in which tasks are identified:

- Engineers suggest activities that need improvement (companies A and E).
- Multidisciplinary teams study the design process and recommend areas to be improved through automation (companies A, C, E, and D).
- Research concept appeals to developers (company F).
- Management decides that an activity should be automated (companies A and B).
- Developers identify opportunities in the process of assisting engineers (company B).

Often the project choice depends on the interest of the developers or of a manager in a particular problem. A developer at company A said that "most projects have been initiated by someone within our group looking for a problem", as opposed to being selected for their value to the company. At some companies, however, there is a formal review process to identify activity for improvement, though not necessarily through design automation. Design automation is considered as a possible solution technique. Companies C, D, and E all have programs of this nature, although ideas continue to be identified by the engineers.

Determine if the task can and should be automated

Developers described two significant activities that comprise this step. First, prospective projects are reviewed, and a project selected. Developers described reviews that include assessing development cost, the value of expected benefits, and the technical difficulties associated with automating the task. The selection of the task is usually formalized with a project proposal. Second, before a task is selected for automation, a proof of concept is often built. The proof of concept may be a paper demonstration of a concept, or it may a scaled down version of the application being developed. This step takes between 10 and 20 percent of development time, depending on the complexity of the prototypes that are built.

The effort put into reviewing a task varies. Most companies incorporate some business plan in the review process, and carry out a needs analysis as well. At some, such as company D, there is a documented process for assessing a project. However, the quality of the analysis depends on person conducting the study. In companies where a business case is presented, it usually qualitative and focused on the estimate of development cost. There are also many situations where decisions are based on other factors. Company B described situations where the project was not economically sound, but was carried out anyway because of the demands of the customer.

Develop the application

In this step, developers turn the project proposal that was generated during task selection into a complete project plan. They then gather the knowledge of the task from experts, case studies, and written documents, and model the task. The model is then used by the developers to create the software. Often the developers implement the knowledge iteratively, capturing some knowledge, creating a part of the application, testing it, and then repeating. They may even deploy the application in phases, and return to complete it in later phases. Before any part of the application is allowed to be used, most companies put strong effort in to validating its ability to perform, and some effort into ensuring that the software is bug free. This step takes about seventy percent of the development time.

There are very few formal techniques for gathering design knowledge, although the companies usually have a method for gathering requirements as part of software development. A developer from company B said that "we have some processes for formally gathering requirements," but gathering core design knowledge from domain experts is "done in a very ad hoc manner" and takes a long time. Developers at company D are beginning to use a more structured approach to capturing design knowledge.

The companies manage the software development aspects of the project differently. While some, such as company A and company D, put effort into using established methodologies like the Capability Maturity Model, most do not. Similarly, verifying the software is more formal in some companies. Developers at all of the companies recognize that they must convince the users that the software works, and have in place methods for testing with the user.

The companies that are developing a lot of applications often use specialized commercial tools for developing these applications. Examples of tools are the ICAD System ™ from Concentra Corporation, STONErule from Prescient Technologies, and PowerModel from Intellicorp. Consultants also used these kinds of development tools. The tools provide some structure to the development process, as they help the developers organize the knowledge and write the software.

Deploy the application

Once the development phase is completed, the developers release the application to the engineers. Usually the engineers have already been testing the application, but now it is integrated into the design process. Developers explained that they cannot just pass the application to the engineers and walk away. Some applications are complicated enough that training in its use is needed. In addition, developers usually have to put effort into convincing the engineers to use the application. They must also keep it up to date and ensure that it remains usable even as design technology and the product change.

The companies all have their own methods for deploying applications. Where the engineers develop their own applications (companies D and E), the transition to using the application is swift and does not require special effort. Other companies rely on training and the involvement of higher level management to get the applications into service. Some companies, anticipating that changes in the software will be necessary, set up procedures for modifying and maintaining the system (company C). In most cases the developers remain responsible for maintenance of the application, although more general maintenance groups may take charge as well. Some companies try to measure the benefits, while others rely on customer feedback. Even the evaluations are mostly qualitative reports about savings that resulted from application use.

2.2 Problems with the current development process

The surveys and discussions with developers provided a picture of what the development process is currently like. To understand where the process could be improved, the developers were asked about the problems they experienced in developing and deploying applications. These are described below.

- **Application not needed**: Several projects were discontinued or resulted in applications that were not necessary. Developers described two circumstances where this has occurred. One case where the application may not be needed is where the application was developed without proper consideration of what the engineers can use and need. At company F, for example, most applications reach prototype stage before any engineering group looks at it. The result is that some expensive projects turn out to be of no use to the engineering groups, and are discontinued (the company is currently working to address this problem). An application also can be rendered unnecessary if the process being automated was not stable. Changes in the design process may render the application obsolete. For example company D developed an application to automate tooling design, only to find that the way the part was manufactured was changed. This meant that the tooling design process also needed to change, and the application could not be used.

- **Application of little value**: In some cases, the developed application did not provide the engineers with capabilities that are of value to them. This occurred because the task was not complex enough to be enhanced by automation, or because the developers did not ensure that they provided the functions that engineers need. A case study described a company that discontinued knowledge based engineering even though applications worked because the payoff did not justify the cost of knowledge based engineering (Guida and Tasso, 1994).

- **Tasks that were unsuitable for automation**: Several projects failed because it was too difficult to properly model the task. For example, developers at company A tried to develop an application to automate mold design, but found that they had not understood the complexity of the task when they began. They consider the resulting difficulty in modeling the application to be part of the reason the application was not completed. Company G had the same problem when developers tried to automate the design of an electrical system.

- **Slow Development**: Developers explained that design automation applications require design knowledge to be captured, modeled, and represented in the software. The application cannot be developed until some amount of design knowledge has been captured. Consultants consider this step to be very important and noted that projects can even be abandoned if knowledge capture is too time consuming. Developers also noted that development is slower than it need be because each project needs to begin from scratch; the code that is developed cannot be used to speed the development of the next application.

- **Problems in Deployment**: Another common problem is that even though an application works as intended, engineers do not want to use it, or have difficulty using it. In some cases described by developers at companies D and E, the engineers did not trust the application's results and did not want to use it. In other cases, such as a project at company G, the application was too slow, so it could not be used. Most developers do not currently track the use of applications, and have no way of knowing if the application is being used as it was designed. Another problem can occur when the system works as expected, but the format of input data is incompatible with the application, or the format of the application's output data is incompatible with other systems.

3. Application Development Methods In The Literature

Development stages have been researched and described in literature on expert systems development. While these descriptions can help a novice learn about knowledge based technology, their focus is on the use of the expert's knowledge to model a task, only one step in the process shown in Figure 3. They do not describe how the development process should be carried out or organized. For example Waterman's (1986)

development stages, shown in Figure 4, is not intended to be the outline of a development methodology (Edwards, 1991).

Stage	Activity
Identification	Determining problem characteristics
Conceptualization	Finding concepts for producing a solution
Formalization	Determining the structures for representing the knowledge
Implementation	Formulating the rules that embody knowledge
Testing	Validating the rules

Figure 4 Stages of expert systems development according to Waterman (1986)

More recently there has been research that describes formal methodologies for developing knowledge based systems in general (Edwards, 1991; Wielinga et al., 1992; Guida and Tasso, 1994) and for design in particular (Miles and Moore, 1994). They stress the advantages of using a structured approach to developing knowledge based systems. For example, Edwards (1991) gives this list of advantages:

- Developers have greater control over the development process.
- The application will function according to an agreed upon manner.
- Even when not used as intended, the application will be robust and functional.
- The application is more likely to be able to produce reliable results.
- The application code will be more maintainable; problems can more easily be corrected, and future functionality can be added more easily.

A very comprehensive development methodology for knowledge based systems, including documentation requirements was developed by Guida and Tasso (1994). Their three part - six phase framework, shown in Figure 5, stresses that a structured approach requires the developers to explore the possible projects, review candidates and select one, develop the application using a formal method, and incorporate some deployment issues into the development process.

Part	Phase
1. ANALYSIS (phases 0,1)	0. Opportunity Analysis 1. Plausibility Study
2. DEVELOPMENT (phases 2,3,4)	2. Construction of demonstrator * 3. Development of KBS 4. Implementation, installation and release of KBS
3. OPERATION (phase 5)	5. Maintenance and Upgrade

* Also called a prototype; they use "prototype" to describe the KBS core (without software for user interaction).

Figure 5 The KBS development cycle according to Guida and Tasso, 1994

Many of the ideas in this plan are sensible and are discussed later as good practices to incorporate in a formal approach. It is a good source of ideas to consider during every

step of development. However, the plan and the required documentation is extremely complex, consisting of hundreds of formalized steps and dozens of documents. In addition, some of the underlying concepts are not accepted by the developers that were interviewed. This should not be a surprise, as the authors intended it to be used in teaching the development process, but not for use in the commercial development environment. As an example, they emphasize the need to prove concepts by building a demonstrator that should not utilize knowledge in the way the completed system will. In contrast, a developer at company F explained that "financial and time constraints usually require" them to build prototypes that are essentially the skeleton of the application being developed.

A less formal plan is described by Prerau (1990). He provides checklists for carrying out many development activities, and some ideas will be mentioned later. His ideas were drawn from experience developing systems at GTE. Because the book is based on expert systems, however, it does not discuss issues that pertain to the development of applications for design automation.

Another type of methodology is formulated around the knowledge modeling that is part of developing a knowledge based system. For example, KADS (Wielinga, 1992) and KAUS (Clibbon and Edmonds, 1996) provide frameworks for capturing and modeling expertise. Although this will be discussed briefly in chapter 6, the knowledge modeling process is beyond the scope of this research.

4. Formal Methods For Developing Design Automation Applications

The developers described problems they have encountered in following the development process and problems shown in Figure 3. These problems were studied to identify activities whose improvement will help the developers avoid or minimize these problems. Five development activities were identified. If developers used formal methods for these activities, they would improve their ability to produce applications that work, are used by the engineers, and provide large benefits to the company. The activities are described below.

1. **Opportunity Evaluation**: In most companies, ideas for automation come from many sources. There is no organized way of evaluating how the company should utilize limited resources for automation, and the technology is often not well understood. Like the market analysis and technology evaluation that occurs in the very early stages of product development, a formal method may help the developers understand the capability of the technology, the company's needs, and its resources.

2. **Task Review and Selection**: In some companies, developers consider many aspects of the project and the future application before selecting a task for automation. They

feel that assessing these issues helps them define and select a task that is well suited for automation. A formal selection method would enforce the review of criteria that determine the desirability of a task for automation. Developers will be better able identify tasks that provide greater benefit and will be able to ensure that they meet the objectives of the company.

3. **Knowledge Acquisition**: Currently most knowledge is gathered by interviewing experts and learning from them. However, while some companies are looking at ways to capture knowledge differently, little effort has been made to improve the current techniques. Developers agree that thorough and accurate knowledge capture facilitates faster and more efficient application development. Using formal techniques will ensure that design knowledge is captured as quickly and completely as possible.

4. **Development of the software**: Developers have noted that although new ideas are continually being researched, the technical expertise for representing knowledge as a model of the design task exists. However, the quality of the application and the efficiency of its development are compromised by ad hoc techniques for developing software. Applying standards and formal methods to the creation of the software will improve quality and efficiency and will also increase the reusability of code.

5. **Deployment**: In deploying applications the developers must introduce it into the design environment, and get the engineers to use the application. determine how the application will be maintained and modified, and evaluate the success of the project. However, despite the effort they put into validating the application with the users, the developers encounter resistance to the new application by the engineers. In addition, although applications are maintained, most companies do not have a method for ensuring that the technology in the application remains up to date. Developers do not think that they evaluate application success or the process for development very well. A formal method for deploying the application will ensure that difficulties introducing the application are minimized, application maintenance is controlled, and that the development process can be used as a learning experience for future development projects.

These ideas were discussed with developers, who concurred that these were all areas in which formal methods could improve the development process. The activities are linked to the steps of the development process as shown in Figure 6. Four were selected to be studied. In addition, the effects of the organizational differences between the companies on these methods were studied. Application development depends heavily on work in other fields that could not be researched effectively for this project. The development of the software is the subject of much research in the software industry (see for example Paulk et al., 1995 on the capability maturity model for software development), and some

research in artificial intelligence software (e.g., Holt, 1996). It is a subject better covered by research in software development, and is beyond the scope of this. The formal approach for the other activities is described in detail in subsequent chapters.

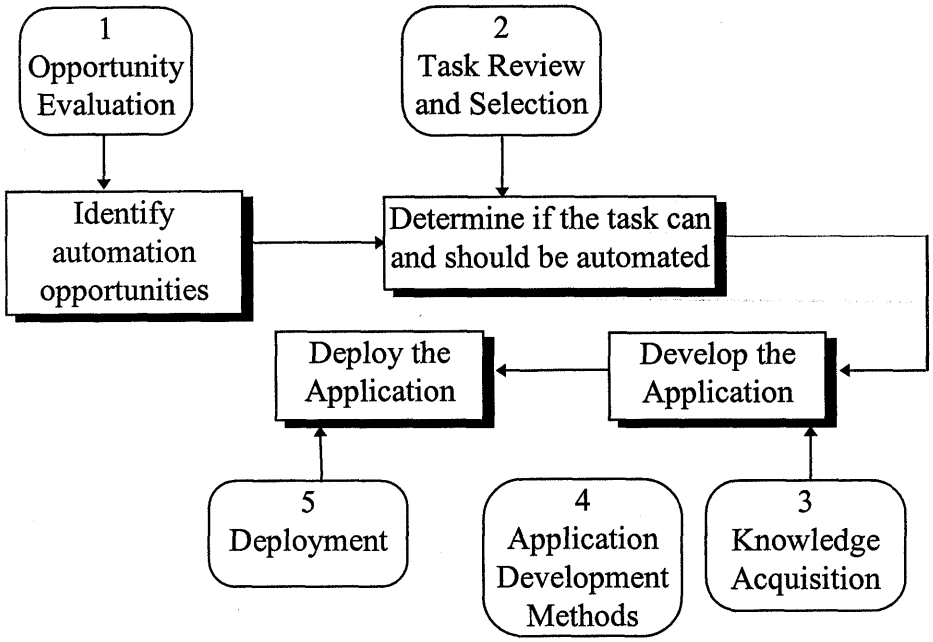

Figure 6 Development activities that benefit from formal methodologies

5. Conclusion

Developers have found that while they are often successful in developing useful applications, their development process is not consistent or controlled enough to ensure success. A review of the current development process revealed several activities that could be improved through the use of formal methodologies. In addition it became clear from discussions with developers that a company's organization for developing the applications has an effect on the use of formal methodologies. Formal approaches to the organization and four of the activities shown in Figure 4 are presented in the remainder of the thesis.

Chapter 3 Organizational Structures for Development

1. Introduction

The development plan outlined in Chapter 2 organizes activities that could be improved through the use of formal methods. However, developers carry out these activities differently because their companies have different organizations for implementing design automation technology. These differences affect the kinds of development problems they encounter, and the formal methods that can be used to alleviate these problems. However, with an awareness of these differences companies can assess what improvement ideas make sense, and where the current development organization should change.

Developers were asked to describe the people involved in different stages of the development process, and how they are organized. This is described in Section 2. Suggestions in the literature for who should be involved in the different development steps are described in Section 3. In Section 4 the development improvement options that depend on the choice of people for each step are presented. The analysis of current organizations is summarized in Section 5.

2. Current Activity

2.1 People who carry out the development activity

Developers identified several kinds of people and groups who are part of the development process. The backgrounds of these people are described here.

Consultants: There are consulting firms that specialize in developing the knowledge based design applications mentioned in Chapter 1. They may be hired to solve an already identified problem, as was described by the consultant that were interviewed. In addition, Guida and Tasso (1994) present several case studies in which consultants performed an opportunity analysis for a company interested in knowledge based technology. The consultant said that although software development skills are important, an engineering background enables them to understand more easily the design task to be automated.

Domain Expert: Experts are engineers who understand how the design task being automated is currently carried out. The domain is the set of design knowledge associated with the task and its automation. The expert is usually one of the most capable engineers at carrying out the design task, although developers at company B noted that they do not need such a high level of expertise as a source of knowledge. At some companies, such as C and E, domain experts also may be trained to use application development tools, and become what might be called developer-experts.

Knowledge Engineer: These are specialists in gathering, modeling and representing design knowledge as a computer model of the task. At some companies, such as companies B and D, they have training in software development. They may also be experts in computer science, as is the case at company F and in a research group at company E. Many developers avoid the term knowledge engineer, in favor of *application developer*, to avoid association with traditional artificial intelligence. At company D, for example, developers said that artificial intelligence is associated with deskilling jobs so that employees can be fired, and is not well received. In some companies, automation efforts originated from within the computer support departments. As a result, the infrastructure more readily supports development by knowledge engineers, with assistance from experts in the domain.

Multidisciplinary Development Team: At some companies, such as company E, engineers work together in concurrent engineering style to develop the application. The team consists of engineers whose expertise is being automated and experts in applications development and design automation technology. Developers explained that the teams are well organized and empowered to incorporate applications into the design process. These teams are well suited for developing their own applications.

2.2 How the current process is organized

The companies used people with different skills for carrying out the development activities, as summarized in Table 2 with the following initials to identify the people:

- KE = knowledge engineers
- DE = domain experts (or their management)
- MT = multidisciplinary development team (defined in Section 2.1)

(the companies in the survey do not use consultants.) The approach at each company is detailed in the case studies (Appendix A) and summarized below.

Table 2 Roles in the development process

Organization of Development Process	*A*	*B*	*C*	*D*	*E*	*F*
Identify automation opportunities	MT	KE	MT/KE	MT	MT	KE
Reviews and selects task for automation	KE	KE	MT/KE	MT	MT	KE
Captures design knowledge	KE	KE	KE/DE	KE/DE	DE	KE
Develops the software	KE	KE	KE	KE/DE	DE	KE
Deploys the completed application	KE	KE	KE/DE	KE	KE/DE	KE

Company A

Developers at company A explained that opportunities are primarily investigated by higher level teams, or suggested by engineering groups and investigated by application developers (they indicated that little work is knowledge based, so the term knowledge

engineer is not appropriate). If a problem is identified, a team comprised of potential customers and a developer is formed to review possible solutions. The review, proposal, and proof of concept are carried out by the developer, but the decision to go ahead with a project is made either by a manager with a research and development budget or a manager in the department that will benefit from the application. The design knowledge is captured by the application developers from the members of the team; they also develop and deploy the software. A projects is overseen by the engineering group that funds it.

Company B

A developer explained that knowledge engineers control the complete process, from evaluating automation opportunities through managing the automated process. However, the ideas for automation are generated by the engineers or management as well. Although the domain experts are part of a team to develop an application, they are there so that their knowledge of the design task can be captured. Developers noted that in their company, the knowledge engineers have an understanding of the technology, and the domain experts have a background in computer science. As a result, knowledge engineers understand the problems they are automating, and domain experts can assist in identifying opportunities.

Company C

More than one development approach is used at this company. Some developers described the use of multidisciplinary teams for improving the engineering processes. These teams evaluate the possibility of applying design automation technology, and select suitable tasks. Knowledge capture and application development by knowledge engineers. Other developers said that it is their job to evaluate opportunities and review prospective tasks. They then train experts to capture the knowledge and develop the application. The Developers said that the engineering groups maintain their own software, unless it is extremely complex or it is used by many groups in the company. In those cases, the application is deployed and maintained by computer support people.

Company D

Multidisciplinary teams follow a documented procedure for improving process (design and manufacturing) quality. This process can result in the decision to automate a task. Developers indicated that the engineers must be completely involved in any improvement activity, both to get their buy in and to ensure that the process to be improved is completely understood. This is also why final decisions on the application and its development are made by the engineering groups who will use the application. Once the team has selected a task for automation, knowledge capture and software development may be carried out by knowledge engineers or by developer-experts. The experts deploy the applications themselves. However maintenance of the application usually falls to the knowledge engineering group.

Company E
Ideas are identified and studied by multidisciplinary teams. In addition, a research and development group looks at prospective opportunities, although final decisions and the remaining project work will be handed over to multidisciplinary teams. To better control the large quantity of engineering automation work going on, "a new organization for knowledge based engineering now organizes and prioritizes opportunities with a strategic intent in mind." As part of the multidisciplinary team, developer-experts capture knowledge and develop the applications in most cases. They also will maintain the applications they developed in many cases. At times they are handed over to a centralized computer support group to be maintained. The engineering groups have control over the development process.

Company F
Knowledge engineers are members of a research group that investigates technology opportunities, while different engineering groups consider their own design process to find opportunities. A developer there noted, however, that coordinating the two has historically depended on the research knowledge engineers finding the right task for the technology. This can be difficult, because the research approach often means projects do not have a feasible application to the engineering processes of the company. The decision to move from prototyping to developing a usable application is based on reviews by the knowledge engineers, but is made by the engineering group who will fund the work. The researchers then become developers, capturing the knowledge and developing the software. Once the applications is completed, it is deployed, maintained, and modified by the researcher-developers.

3. Literature

There are case studies of the development of fully operational systems and of prototype or first phase systems in the literature. Generally, operational systems found in the literature were developed by the engineers who use them, or with the help of consultants (Proctor, 1995; Binder, 1996; Marra, 1997). In case studies of prototypes, the development process was carried out by knowledge engineers (Rehg et al, 1988; Rangan et al, 1994; the case studies in Guida and Tasso, 1994; Chin and Wong, 1996).

The underlying assumption in the literature on development methods is that the application development process should be carried out by knowledge engineers (Prerau, 1990, Edwards, 1991). Guida and Tasso (1994) suggest multidisciplinary teams for each step in their development methodology, but they assume that the application development is carried out by experts in knowledge engineering and software development. They also suggest that task review and selection, and possibly the opportunity analysis as well, should be carried out by external consultants. In addition, they suggest multidisciplinary teams for each step. Heatley and Spear (1992) conclude from their analysis of using

knowledge based design systems at Xerox that engineers with training in development systems (they used Concept Modeler from Wisdom Systems) made excellent developers of applications, and required much less infrastructure within the company than an entire knowledge based design group.

4. Formal Approach To Considering Organization Choices

People of different backgrounds may carry out similar development activity at different companies. Each personnel choice has some positive and negative aspects. However, developers cannot expect to weigh these aspects and determine the best person for the job, because there are organizational issues behind why a company selects one type of person to carry out a step in the development process. Rather, the formal approach suggested here is to review these positive and negative characteristics for the kinds of choices noted earlier in Table 2. By being aware of the positive and negative considerations of the personnel choices for each activity of the development process, developers can better identify the difficulties that that person or group will face.

Identify automation opportunities and Review and select tasks to be automated

- **Multidisciplinary Team**: The multidisciplinary teams described by the developers have the advantages of a mandate and the background to thoroughly review the current product development process and identify how they might be improved. In addition, as experts in application development are included in the team, they have knowledge of the technology to utilize in considering the possibilities of automation. However, independently operating teams can end up duplicating efforts.

- **Knowledge Engineers**: Being familiar with many knowledge representation issues allows knowledge engineers to better determine where the technology can be applied. However, as they are not familiar with the domains under consideration, they can underestimate the complexity of the task. The research teams (mentioned above at company E and company F) are comprised of knowledge engineers, and have funding to explore new ideas. However, as they are not working to understand the engineer's needs, they are not well suited for identifying possible automation opportunities.

Capture design knowledge

- **Domain Experts**: The experts may be working as part of a team, or by themselves. The team has the advantage that all of the knowledge that it will need spread among its members. User issues that must be captured are also understood well by the team. The team works together to capture the design task knowledge effectively and efficiently. However, developer-experts have to simultaneously learn the complexities of knowledge modeling and decompose their own design

knowledge into the steps and decisions that are used to build a model of the task. These problems are made worse when an expert works alone.

- **Knowledge Engineers**: Knowledge engineers are trained in understanding how to organize the pieces of design knowledge for the development of the application. Developers at companies A, B, C, and D noted that knowledge engineers depend on the experts as a primary source of knowledge. As will be discussed in Chapter 6, the relationship they have with the experts as well as some aspects of the expert himself affect the ability of the knowledge engineers to capture the needed knowledge effectively

Develop the software

- **Domain Experts**: Engineers understand the process to be automated. Developers at company E said that they can more easily catch errors in the way the application functions because of this. However, developer-experts can only work with development tools like I-CAD, and must be trained in their use. While the tools help them organize the knowledge and develop the application, a developer at company D noted that it is very difficult to get software standards and methods used. Developers at company E admit that expert developed applications usually cannot be upgraded or modified for other groups, and are written so that no code can be reused.

- **Knowledge Engineers**: Knowledge Engineers at most of the companies have expertise in knowledge representation and software development. They are in a position to utilize methodologies that ensure error free, efficiently written programs, as is the case at companies A and D. However, knowledge engineers are not directly in touch with user needs, and the way the engineers think about the task being automated. Extra effort is necessary to ensure that the needs and opinions of domain experts are incorporated in the development process.

Deploy and maintain the completed application

- **Domain Experts**: Engineers have an easier time getting fellow engineers to use the application than do outsiders, since they are part of the group who will use the applications. They also are in a position to identify changes in the design process or related disciplines that may necessitate modification of the application. However, unless the application has a method through which they can easily make the changes themselves, they will have trouble maintaining the application. In addition, as they are engineers with other jobs to do, applications are likely to be neglected. In fact at company E, the approach is often to use an application without making any changes, until it is obsolete.

- **Knowledge Engineers**: Knowledge engineers have the necessary skills for quickly fixing bugs in the system, as well as making changes to keep the system up to date. However, they face a serious obstacle to deployment in that engineers do not trust software experts to give them a useful and reliable system.

5. Conclusion

The skills of the people who carry out the development process differ. These differences cannot usually be used to select more desirable people for different development steps. The formal approach suggested in this chapter can help developers to consider the strengths and weaknesses of their developers during each step of the development process. While the organization may choose one person over another for many reasons, understanding the pros and cons of this choice will help the company focus improvement efforts where they are needed.

Chapter 4 Opportunity Evaluation

1. Introduction

Most companies have many sources of ideas for where to implement design automation technology. However, they do not have an organized way of evaluating how limited resources for automation should be utilized. While this ad hoc approach to evaluating opportunities may not necessarily lead to failure, developers agree that a structured evaluation process would ensure that the company makes wise choices in determining where and how it will implement design automation.

A survey was conducted to determine current evaluation practices, what developers feel should be gained from the evaluation, and where the current practices are deficient. This is described in Section 2. Discussions in the literature that are relevant to evaluation are described in Section 3. A formal approach for evaluating the opportunity that describes what developers should understand before beginning, is presented in Section 4. The value of this approach in improving the development process is summarized in Section 5.

2. Current Activity

Developers describe four kinds evaluation that can be part of the evaluation process.

- Evaluation to determine whether design automation is the right solution for the problem being addressed.
- Evaluation of how design automation might be applied to their design process by considering objectives and understanding the design process.
- Evaluation of the technology of design automation, including the tools that can be used and the knowledge modeling approaches that may be applied.
- Evaluation of the physical and experiential resources that can be harnessed in the development of design automation applications.

They explained that the way the company carries out evaluation activity is related to the approach the company has to the implementation of automation technology. Evaluating the use of design automation as opposed to other possible solutions is beyond the scope of this research.

The developers were asked to rate the importance of the evaluations they described to the use of design automation. Their responses, shown in Table 3, are scaled from 0 (no effort or not important) to 5 (significant effort, or very important). They are explained below.

Table 3 Opportunity Evaluation Activity

Opportunity Evaluation	*A*	*B*	*C*	*D*	*E*	*F*
Important to understand engineering and design	4	4	5	5	5	4
How well are engineering and design issues understood?	4	5	3	5	5	2
Important to understand benefits of automation	3	4	5	4	5	4
How well are benefits understood?	4	4	3	3	3	3
Important to formalize the objectives	3	3	5	3	2	3
How formally are objectives stated?	4	2	4	3	2	2
Evaluation of tools and concepts of automation?	5	4	3	4	4	3
Consideration of existing competencies	3	3	3	3	3	3
Consideration of physical resources	4	4	4	4	4	2
Use of a technology driven approach?	1	0	1	1	1	3
Use of a problem driven approach?	4	5	4	4	4	3

Engineering Evaluation

Generally, the companies surveyed understand the domains in which design automation applications may be implemented. Company F has difficulty in that the developers are not able to work with the domain experts as a group. In addition, the fact that ideas for automation are often from the developers themselves means that limited engineering experience is available and utilized in evaluating how the technology can benefit the company. For this reason, companies including A and F utilize some developers whose backgrounds are in engineering, not software development. As one developer put it, "high design knowledge" is valued.

The developers all indicated that in evaluating how to utilize design automation, it is important to understand the engineering domains that will be affected, and what to realistically expect applications to do. For this reason, companies such as C, D, and E use a multidisciplinary team to evaluate whether product development processes can be improved through design automation. Experts in manufacturing, engineering design, and computer aided engineering work together to look at how the engineering processes can be improved. At company F, however, there is poor communication between the developers and experts, so that the developers do not have a picture of the product design process to use when they consider the kinds of applications are viable.

The developers identified numerous objectives, and they are described below in three groups that were used by Heatley and Spear (1992), in formalizing objectives at Xerox.

1. **Save Time**: Automating routine and repetitive design work frees the engineers to focus on the special cases, or to work on other projects. The savings to the organizations come from reducing design time, information flowtime (waiting for data, coordinated meetings, etc.), and errors that would need to be reworked. In

addition, time in the design. Tasks in the critical design path are viable candidates.

2. **Increase Quality**: Automation can ensure that constraints and guidelines for manufacturability, producibility, etc., are satisfied during the design process. This increases design consistency and reduces design errors that would normally only be discovered much later. Tasks that are often ignored due to time constraints or lack of expertise, such as manufacturability checks, become feasible. Design quality can also be increased by incorporating more analysis or by using some of the saved time to carry out multiple design iterations.

3. **Process Review**: Automating lets the company review the design process to be automated, make enhancements to ensure that the process being used is acceptable, and capture best practice knowledge. This is critical when the source of expertise will be lost, such as with the retirement of an employee.

Although companies consider what benefits they can expect from design automation, they do not usually make a formal statement of objectives. Rather, the company's objectives are defined when the deliverables of the project are determined. The developers did not feel that it was vital to formalize the objectives of the application. They felt that generally candidate tasks are design processes that need to be improved. The application's objectives are the improvement objectives, and need no further elaboration. However, some developers at companies C and E indicated that it was important to formalize the objectives. The opinions in the literature will be discussed in the next section.

The engineering evaluation is passive in that developers consider it important that this knowledge be there, but do not have methods in place for actively seeking it. In addition, although developers consider it important to understand what to expect from design automation, several expressed doubts that evaluating technology could be a formal preparatory step. Rather, companies beginning to automate their design process should make sure that they possess the knowledge of the engineering issues described above.

Technology Evaluation

Developers explained that in evaluating the technology they consider the kinds of tools that are available, and the knowledge modeling concepts that are feasible.

1. **Tool Evaluation:** Different kinds of development tools, such as programming languages and development environments, are evaluated to determine how best to automate a task. In this way, as a developer at company A explained, they can solve problems with "the most practical tool for the job." When considering expensive, possibly new development systems or new programming techniques, a benchmarking study may be conducted to test the capabilities of different systems

and approaches. This kind of study is sometimes conducted even after a tool was selected, as was seen at company C, as a way of continually comparing the capability of the selected tool to others.

2. **Concept Evaluation:** Applications development engineers (as opposed to the developer-experts described in Chapter 2) will also evaluate choices of concepts that are available for providing knowledge based solutions to groups within their companies. However, developers noted that in some cases the choice of how design knowledge may be modeled depends on the past experience with particular tools, and other, political factors, are also part of the evaluation of the tools that should be used. For example, developers at company D said once a costly development tool was selected and proven, the evaluation of possible projects revolved around whether it is suitable for development with the tool.

In all of the companies that were surveyed, there exist separate groups whose expertise is in developing these applications. Even where design engineers develop their own applications, it was noted in Chapter 2 that support teams with this expertise assist them. As a result, they all have a strong understanding of how different techniques for automating design work. Developers therefore felt that the knowledge of tools and concepts was available without formalizing the their evaluations.

They admitted, however, that formalizing the evaluation of concepts could help in two areas. First, where a new company or group is interested in design automation, a formal review of the available concepts will show the novice developers what they need to think about. Second, because the evaluation is not structured, it can be subjective. For example, at company D, when design engineers began to develop their own applications, they became especially interested in using a programming language for developing applications, based on advice from other engineering groups. Experts in knowledge engineering, however, understood that for their needs, the design engineers were better off with other tools. An evaluation process would have required the developer-experts to objectively look at possible development tools.

Resource Evaluation

Developers explained that two types of resources are evaluated: physical and experiential. Physical resources include project budgets, human resources for carrying out development projects, and the priorities of the company in improving its design processes. Experiential resources include the competency of the company in the tools and design automation techniques that can be used, and past experience in developing automation applications.

Developers said that during evaluation the different physical resources are kept in mind to give some definition to project size. Consideration of specific equipment and personnel

needs is done when a task is being reviewed for automation. At company F, where the developers can move ahead on a project without connecting it to any known engineering need, the evaluation of these resources is limited to choosing projects to be budgeted. The overall view of the developers was that little would be gained from requiring a structured evaluation of resources when no project has been selected.

Developers indicated that past experience within the company is an important resource in developing design automation applications. At some companies, such as D, developers had prior experience in expert systems. As a result, when the company began looking at automating the design process, knowledge engineering competency was already there. However, without this resource, attempting to develop knowledge based design systems entails more than just training in the tools.

None of the developers felt that they evaluated the existing experience particularly well. For example, developers at company E noted that there was a lot of duplicated effort because past experience (as well as computer code) was not being leveraged in new projects. The new, centralized organization for knowledge based engineering is intended to combat this problem.

Approaches to Automation

Developers described two general approaches to evaluating design automation technology.

1. **Technology Driven Approach**: In the first approach, a company that becomes aware of the capabilities of design automation investigates how it may be used. This can be called a technology driven (TD) approach, as the company decides to leverage automation technology before identifying design processes to improve. Developers explained that in the technology driven approach, the company has the opportunity to utilize design automation as a means for rethinking its approach to product design and development.

   The TD approach has the advantage that through it a company can view the entire design process, and identify how it can use design automation as a tool to redefine the design process, with individual applications being parts in a larger picture. This is the thrust of a program going on at company C. In this context, a company may determine the tools it will use, and the overall strategy for changing its current product development process, before considering which tasks should be automated. However, this approach also can result in the selection of projects because of the appeal of a particular automation techniques, without an organized approach to improving the product development process, and without a careful evaluation of other methods. This problem was described by a developer at company F. In the worst case, it can result in the selection of tasks without proper assessment of their suitability for automation.

2. **Problem Driven Approach**: In the second approach, a company that has already marked a specific process for improvement evaluates design automation as a way to make the needed improvements. This can be called a problem driven (PD) approach. Developers explained that with this approach, individual problems can be quickly addressed.

   Most of the automation activity described by the developers is carried out with the problem driven approach (see Table 3). In many cases, as was described by developers at company A and company B, the problems are specific design activities. Suggestions of problems that can be solved with automation also come in the form of a request for an application with specific capabilities. For example, engineers at company A wanted to simplify the detailed drawing process, and asked for an application to help. The consultants who were interviewed noted that they must approach companies primarily from the problem driven approach, so that they can convince potential clients of specific benefits that come from automation.

Developers noted that there are situations that mix the two approaches. In these cases, a high level problem, such as a product development cycle that is too long, needs to be solved. Design automation is determined to be the way to solve the problem, and then developers find the specific activities that could be automated to reduce the cycle time. This situation was described by developers at companies A and C.

3. Literature

3.1 Approaches to automation in the literature

Edwards (1991) raises concern that the technology driven approach is more likely to result in applications that fail or are of little value. He explains that historically, numerous toolkits and "shells" that facilitated development of knowledge based systems became available before companies understood what the technology could do for them. In expert systems development this approach had limited development to problems that were quickly and easily solved by using the selected approach. More complex problems, that required longer development times, were often dropped.

The problem driven approach does not have these shortcomings because two important questions must be answered before an expert system is considered. First, how is this kind of solution superior to other technologies? Second, can one determine if the solution's value exceeds its cost? In other words, the developers question the necessity and the desirability of building an expert system. Therefore Edwards (1991) concludes that the problem driven approach is the correct way to decide that design automation is the right approach.

3.2 Evaluating opportunities in the literature

Guida and Tasso (1994) present a methodology for an "opportunity analysis phase", in which a company defines a "strategic, long term plan for the correct and effective introduction of knowledge based technology in to an organization". The goals of this phase include identifying opportunities within the company, describing the technical aspects of these opportunities, ranking them, and creating a plan that will guide the entire effort to use knowledge based technology. The opportunity analysis process consists of four parts and fifteen subsections that include planning the analysis, analyzing the company organization, analyzing prospective problem domains, and synthesizing a master plan for implementing knowledge based technology.

Engineering Evaluation

Past research has emphasized the need to understand design and how it can be automated (Miles and Moore, 1994; Hopgood, 1992). Some case studies also emphasize the need for formal objectives (for example, Heatley and Spear, 1992), and this is one of Prerau's (1990) guidelines for beginning application development.

Technology Evaluation

Evaluations of tools and automation concepts can be found in past research. For example, many of the books on expert and knowledge based design systems development evaluate knowledge representation techniques and tools (Waterman, 1986; Edwards, 1991). They do not, however, recommend that developers carry out an evaluation of technology. More useful are the guidelines that Prerau (1990) gives for companies that are first implementing expert systems. He includes the careful selection of a development environment including hardware and knowledge engineering software. Guida and Tasso (1994) provide several case studies to emphasize the need for objective consideration of the choices of tools and concepts. In addition they emphasize that this research should not interfere with the development of applications. In one case that they describe, a research group was successful in demonstrating many concepts for applying knowledge based engineering, but could not develop any commercially used systems. Only when the team separated into a research group and a production group were they able to succeed in developing usable applications.

4. Formal Approach To Opportunity Evaluation

Evaluating the technology and available resources provides the developers with the background understanding of how they can apply automation to the current design process. The discussions with developers revealed that a formal evaluation process would help for three of the evaluation activities that were described above. These are:

1. Engineering evaluation, including identification of objectives and how automation can be applied to design.
2. Better evaluation of the companies competencies and experience.
3. Consideration of the approaches to automation that were described above.

4.1 Formalizing the engineering evaluation

Objectives in automation

Developers debated the need for an experienced company to formalize the objectives of automation. They agreed that in beginning to implement design automation, formalizing objectives provides a starting point in the evaluation of what kinds of design activities can be automated. With their objectives thought out, the company can also make sure that it understands how automation can help in the design process. The evaluation will prepare them to make informed decisions in identifying and selecting desirable tasks for automation. In cases where the problem driven approach is being used, it can provide the justification and guidelines for development. The description of objectives that was presented in the literature provides some ideas for the developers to consider.

Applying automation to design activities

The developers need to know how design can be automated to meet their objectives, so that they can have realistic expectations about the benefits and costs of meeting the objectives. In cases where the developers have little familiarity with design (as with developer-experts described), they will gain an initial understanding of how the technology can be applied to design. In cases where the developers have little familiarity with automation and knowledge based applications (as in the engineering development approach), they will gain an understanding of what the applications can do. The description of design activity that was presented in the Chapter 1 provides background and some resources for the developers.

4.2 Formalizing the evaluation of competency and experience

Every company has some way of evaluating the kind of budget that will be available for purchasing the equipment needed by the developers. However, evaluating the people and their value is difficult and often overlooked. As part of a structured evaluation of the company's ability to develop design automation applications, the company needs to review and consider existing competency and experience. It was noted earlier that some developers have previous experience with expert systems, or with specific tools. The company can leverage this past experience in determining how the company will begin to implement automation.

4.3 Consideration of approaches to automation

Each of the approaches described in Section 2 has advantages and caveats. Since the choice of approaches may be made by senior management, the best the developers can do is to be prepared to utilize either one. When the approach has not been chosen for them, the developers should consider the approach to automation that was described by developers at company A and C. They noted that since design automation is a set of approaches to improving design, the company is better off beginning with the problem driven approach at the high level of design problem described in Section 2. At this level of problem definition, the company can evaluate the kinds of solutions that are

appropriate. With these solution approaches in mind, they can look at the kinds of more detailed problems that can be solved with them. This eliminates the problems brought on by the assumption that design automation is suitable for solving all of the companies design and manufacturing process difficulties. It also allows the company to consider the use of automation in a global way, where it facilitates rethinking of how the engineering process are being carried out.

5. Conclusion

Developers intentionally approach some opportunity evaluation activities without a formal procedure, because they feel that these evaluation activities are better off not structured. However, evaluation of engineering issues, existing competency, and the approaches that may be used for considering automation were areas that developers thought could benefit from a more structured approach. The formal approach described in Section 4 was intended to outline a structured approach to evaluation and to identify areas of the evaluation process that should be part of this formalization. With a thought out evaluation behind them, developers are prepared to review the prospective projects and select one that will provide significant benefit to the company.

Chapter 5 Task Review And Selection

1. Introduction

Developers consider the task selection process to be an important step in the development process. If a task that is unsuited for automation is selected, the project is likely to fail. Many developers described reviews of technical and economic aspects of the potential project prior to selection. However, they admitted that developers assess the viability of a project in an ad hoc fashion, and this assessment is not always of good quality. Developers agreed that organizing the issues that should be assessed into a formal approach for reviewing and selecting tasks would enable to them to select projects with greater chances for success.

Developers were surveyed on the current review techniques. These are described and analyzed in Section 2. There are also methods for task selection in the literature, and these are discussed in Section 3. A formal approach that is based on current techniques, ideas in the literature, and on discussions with developers, is presented in Section 4. The value of a formal review and selection process is summarized in Section 5.

2. The Current Selection Process

Developers were surveyed to determine what techniques are currently in use for evaluating and selecting tasks for automation, and which of these were considered to be valuable. Although each developer listed several criteria that may be evaluated, the routine activity that they described varies greatly in formality and comprehensiveness. Their responses are summarized in Table 4 and are on a scale from 0 (no effort) to 5 (significant effort). The review processes are described in the case studies (Appendix A), and summarized below.

Table 4 Current Selection Process

Task Review and Selection	*A*	*B*	*C*	*D*	*E*	*F*
Use of Methods for Selection						
How much effort to review and select?	4	3	4	4	4	1
How formally are selection criteria reviewed?	2	2	3	4	2	1
Technical Considerations						
Understanding of the current design process	3	4	4	4	4	1
Understanding of the automated process	4	4	2	4	4	1
Understanding of how to automate the task	4	4	2	4	4	4
Needs/Requirement analysis	3	3	3	2	3	1
Incorporation of users/engineers opinions	4	3	3	4	4	3
Consider application's role in design process	3	3	3	2	4	2
Metrics for assessing technical criteria	0	1	3	3	2	1
Economic Considerations						
Assessment of cost	4	3	4	2	4	1
Assessment of return on investment	4	3	4	2	4	1
Metrics for measuring return on investment	2	3	3	2	3	1

Use of Methods for Selection

While most of the companies have guidelines and put significant effort into selecting projects, this is not the case at company F. A developer explained that because applications development is primarily done by a research and development group, ideas are often pursued before a real business need of an engineering group is identified. Little review is carried out, although technical aspects of the project are determined in advance by the thrust of the research. The developer noted that historically big projects have failed, most likely because the project had proceeded too far without focus on a specific task to be of interest to the engineering groups. Company F is currently working on shifting the process so that the engineering groups push the developers for applications of interest, instead of the current method. The engineering groups will investigate the cost of development and the function of the application in order to develop a business case.

At the other companies, developers review projects and prepare some form of project proposal. Although the proposal generally includes a business case and a description of the proposed project, the quantity of information in the case and description varies. The review process is unstructured at most companies, but company D requires a documented analysis on which the decision to work on a project is based. Developers there admitted, however, that while the documentation requires an evaluation of the opportunity and a preliminary analysis of the project, it is subjective and varies in quality from developer to developer. Consultants also use a documented process for assessing the project. However, discussion with consultants revealed that although the estimates and

evaluations guide them, the need to acquire and please customers can skew, albeit unintentionally, the evaluation process.

Technical Considerations

Included as technical considerations are any considerations regarding the feasibility of the project and the likelihood of success in developing the application. The developers described considerations that are associated with three aspects of the project: understanding the current design task process, understanding the future design task process, and understanding how the task can be automated.

- **Understanding the current design task process**
  At company B, this includes looking at the effects of task complexity, such as the kind and number of errors that occur in the manual process. At companies D and E, the design task is analyzed to determine how well it is understood. For example, developers at company E look at how well the steps in carrying out the design task can be expressed in writing. In addition, developers tried to assess the involvement of expertise in different disciplines in the design task process (company D).

- **Understanding of the automated process**
  Developers at company E stressed that an automated task requires inputs and provides data for the next step in the product development process. Therefore, the availability of the input data, and the form of the output data need to be considered before selecting a task for automation. Also of importance is the definition of what the application is supposed to do. For example, a developer at company B noted that too much functionality not only wastes development time, but penalizes the system by requiring more computer power as well. As a result, ensuring that only what engineers will use is included is a priority there. Developers at company D stressed the need to identify the roles of the engineers in a design environment changed by the new design automation application, although exact details can wait until the project is underway. Another important consideration is the interest of the engineers in having the prospective design tool.

- **Understanding of how to automate the task**
  In considering how to automate the task, some developers consider whether it should be their job at all. For example, developers at company A look to see if their computer aided engineering vendor can supply the needed application. Company C has an ongoing effort to review the functionality that the engineers need from their computer tools, and submits requests for desired functionality to its software vendor.

When the application will be developed in house, most developers do not have ways to assess how complex the programming of the application will be. Developers at company D try to look at reusability of previous code, and metrics such as the number of objects in the code. At company B developers look for past experience developing similar applications. Other issues include determining what kind of hardware, software, and training will be necessary (companies D and E); the availability of qualified developers and experts (company E); and the possibility of short term solutions or phased development of the application (companies A and D).

Another issue to consider is the uncertainty of responsibility that arises when the complexity of an automated system requires expertise outside the background of the developers. For example, developers at company F described internal arguments over a system for creating two dimensional die geometry. Modifying it to handle three dimensional geometry depended on the development a specialized surface modeler with which the developers did not have experience, and was part of another research group's work.

The role of the future user of the design automation application in the development process is critical. The integration of the application into the design process ultimately rests on whether the user will comfortably use the application as part of the daily routine, or ignore it and continue as before. In addition, seeing how the user currently works will help ensure that the development team considers the constraints they face from the user. As was noted earlier, without the user's input, the team can develop an application that does not provide the user with the help needed, or provides much more than the user needs.

- **Metrics for technical assessment**
  Developers at all of the companies indicated that it is hard be objective because there are no good metrics for evaluating the tasks. For example, developers at companies B, D, and E noted that they have no way to measure the complexity of the design knowledge for a specific task. They do however, estimate how long it will take to develop the application. Several expressed doubts that generic metrics could be developed at all, but agreed that structuring the review of tasks would help ensure that appropriate selection criteria were considered.

Economic Considerations

Developers described several considerations that are associated with the cost and value of the development. In addition, all of the companies had some metrics they used for assessing the cost and benefits of the application.

- **Assessment of cost**
  The companies all try to estimate the cost of development by estimating equipment needs and developer time. A developer at company E said that traditionally, maintenance costs are much larger than development costs and estimating them is an important part of the cost assessment. Not all developers agreed, and one suggested that it depends on the complexity of the application. Knowledge based design applications that require expensive computers and support to ensure compatibility of data from one system to the next are likely to be expensive to maintain. Applications that assist the current CAD system, such as an application at company A to provide CAD geometry libraries of standard parts, fit into the current system and do not require expensive support.

  The assessment of cost may be used for comparing candidate tasks. At Company E the cost of the current process is assessed and compared with an estimate of the cost of the automated process. In addition, the priority of different projects is compared to decide which should be funded first.

- **Assessment of return on investment**
  The savings to the companies vary with the application, and they are often intangible. Some of the considerations include the reduction in product development costs, savings in manufacturing costs, the value of a reduced time to market, and the value of more consistent design.

  The companies have difficulty estimating the value of many savings because data is often not available. Developers at company D noted that a project may cost a lot for the design group and save a lot for manufacturing. Because of accounting procedures, it is very difficult to see the cost and savings together. It is also difficult to give a value for increased quality in a design, or to evaluate reduced design time. For example, in some cases the saved time is used for carrying out more design iterations, and is not visible. Similarly, it is hard to place a value on being able to carry out tasks that could not be carried out any other way. For example, Digital Equipment Corporation reported large savings through the use of the XCON computer configuration application, but this was based on the belief that without it, they would have stopped tailoring systems to customer requirements (Edwards, 1991). An approach that some consultants use is to identify many tasks that will be affected, and to ask engineers to assess possible savings. They then sum the savings and present the numbers to the clients. Developers did not find this approach acceptable, because the original estimates are not reliable. They admit that the assessment is subjective, and more standard ways of estimating the return on investment are needed.

3. Literature

Many discussions of the use of expert and knowledge based systems include the selection of appropriate problem domains. They emphasize that the success of the systems depends strongly on the appropriate selection of projects (Bobrow, 1988; Prerau, 1990). Poor task selection is considered to be the reason why expert systems often did not succeed in the late 1980's (Blount et al., 1995).

3.1 Review methods in the literature

Prerau (1990) list four general steps for selecting a problem domain and two of these provide structure to some of the review activity described above.

- Meet with experts to learn about the problem: use the engineers and others associated with the design task process to gather the design knowledge needed for assessing the as is and automated task processes.
- Investigate each candidate in enough detail to either eliminate it or determine it to be a good possibility: specify the technical and economic aspects of the project that need to be evaluated, and ensure that the evaluation accomplishes this objective.

He also notes that even when a task has been selected in advance, evaluation will ensure that the domain is suitable for automation.

Guida and Tasso (1994) describe a method for carrying out a "plausibility study" of potential knowledge based systems. The plan consists of five steps with twenty eight tasks to be carried out. Although developers stressed that overly complex and time consuming plans are not feasible, this plan includes several suggestions that help structure the review process. These are explained below, in terms of design automation application projects.

- **People to carry out plausibility study**
  Evaluating candidates for knowledge based systems may be carried out by independent consultants, especially if the company has little experience with developing the applications. Otherwise, the company should make sure representatives of management, engineering, and the knowledge engineers are involved in the review process.

- **Defining plausibility**
  Guida and Tasso (1994) explain that deciding in advance what aspects of a project must be evaluated ensures that nothing is skipped. They describe five components to plausibility. These contain many of the considerations described by developers, and a few other suggestions.
  1. *Technical Feasibility*: Assess whether design automation technology is the right way to solve the problem being addressed.

2. *Organizational Impact*: Assess how the implementation of the application will affect the design environment.
3. *Practical Realizability*: Assess the availability of resources such as personnel, hardware, software and sources of design knowledge.
4. *Economic Suitability*: Assess the project cost and estimate the expected benefits.
5. *Opportunities and Risks*: Assess factors such as corporate strategy and the competition, and analyze possible risks.[1]

3.2 Technical selection criteria in the literature

Ineffective selection criteria

Selection criteria in past research are often vague, or are based on limitations of artificial intelligence techniques that are used. As such, they are not usually applicable to design automation. For example Waterman (1986) lists feasibility, appropriateness of technology, and justification as three areas to be assessed. He does not however, indicate how one might make the evaluations necessary to deem the task appropriate, justifiable, and feasible.

Another suggestion was to tie the appropriateness of an automated solution to the time it takes to carry out the task manually. Some suggested time constraints of not less than several minutes, and not more than several hours (Waterman, 1986; Badiru, 1992) or approximately 15 minutes (Durkin, 1994). Others described the task as "neither too easy (taking a human expert less than a few minutes) nor too difficult (requiring more than a few hours for an expert)" (Prerau, 1985). Developers dismiss these attempts, as have other researchers. Many publicized systems have been developed that accomplish in a few hours what designers did in two to three weeks (Wagner, 1990; Davies and Anderson, 1991; Cochran et al, 1993; Proctor, 1995; Marra, 1997). There is no simple relation between task duration and the feasibility or worth of automating (Dym and Levitt, 1991). In addition, this kind of criterion is not related to any understanding of the task domain (Kidd, 1987; Blount et al., 1995).

Useful selection criteria

Some have suggested that developers should consider the extent to which the task depends on creative thought by the designer. Large proportions of creative, or original, thought by the design expert mean that the problem is likely not suited for automation (Dym, 1991).

Other considerations that developers agreed are necessary include acceptability of the solution to the engineers who will use the application and understanding of how the task is currently carried out (Hart, 1986). Greenwell (1988) also emphasized the importance

[1] Developers at company A indicated that risk analysis is part of their project proposal.

of talking with users and understanding their needs. There is also ongoing work in a method focused on design automation task selection that is described by Blount et al. (1995).

Developers indicated that risk identification is valuable where the projects are very ambitious. Risks are identified by Guida and Tasso (1994), who consider it important and direct the reader to methodologies. Developers noted however, that they are describing a company that is going to reengineer current processes using knowledge based technology. Design automation is usually not that dramatic. They agree however, that developers should consider the availability of people for the project, the support of managers, and the arbitrary constraints that the company may place on the development process, in determining if the project is too risky.

3.3 Economic selection criteria in the literature

Guida and Tasso (1994) provide some useful guidelines to consider when assessing project cost and the expected benefits.

Cost Assessment

Two logical and fairly easy to implement suggestions are made. First, in assessing the cost of the project, developers need to consider both the cost of producing the application, and the continued cost associated with operating the application. This includes maintenance costs, the cost of developing upgrades, and unanticipated operation costs. The second suggestion is that in assessing the cost of the project, developers should consider the way it is distributed over the life of this and other projects. Developers agreed and noted that the cost of development tools and the necessary hardware may not be justified by an individual project. In accordance with what developers described, Guida and Tasso (1994) recommend that the cost assessment be described by definitions of the costs, a value for each defined cost, and the distribution of the costs over time. In this way, reliance on the numbers alone is minimized.

Benefits Assessment

Guida and Tasso (1994) suggest that one way to ensure that benefits are considered is to compile a list and refine it through verification meetings with the experts. In addition, effort should be made to avoid duplication of the benefits. Developers often consider the increase in design quality and the reduction of errors separately, even though they are both improvements in the quality of the design. They also provide lists of benefits that may be used to help developers compile lists of their own benefits.

4. Formal Approach To Task Review And Selection

Developers described criteria that they use in determining the desirability of a task for automation. The objective of formalizing the task review and selection is to take these criteria, and others suggested in literature, and define a set of questions that must be

answered. The answers are an evaluation of the important criteria for assessing the task. Developers can determine how to document and carry out these considerations, but the structured questions ensure that the important issues are considered. These questions are subdivided as the criteria were, into two groups that are defined as follows:

1. *Technical Considerations*: In reviewing the current method of carrying out the task, the future method, and how the task will be automated, are the expectations for automating it reasonable?
2. *Economic Considerations*: Do the benefits of the design automation application compared to the cost of the development make it financially worthwhile to automate this task? In addition, can the benefits be estimated and then measured effectively?

4.1 Technical considerations

The developers need to assess if the current method for carrying out the task can and should be automated, what the application will do in the design process, and what automating the process will entail. To do this, they need to talk to the design engineers and gather basic knowledge about the task. With the questions described below in mind, the developers can identify where and to whom to turn for the answers.

As Is Task Process

The developers need to assess if the current method for carrying out the task can and should be automated. To do this developers will have to answer the following questions:

- **Is the method in use stable?** Developers at companies D and E defined a stable process as one that is not going to be replaced or significantly altered in the near future. For example, a developer described a project to develop a design automation application for the design of a fixture for making a part. During development, the fixture manufacturing process was changed so that the design procedure being automated was outdated; the application was unusable. As technology changes, many processes are continually modified, and the significance of these changes has to be considered.

- **Is this the accepted method for carrying out the task?** There can be strong disagreement among experts on how to carry out the task. The developers need to consider whether there can ever be agreement on a standard method of performing the task. Developers described a project to automate the design of injection molds that was discontinued at least partially because consensus on how to perform some of the design steps could not be reached. A developer at company D described how each tool design engineer has a different 'favorite' shape for part of the tool. Getting them all to accept one shape may be difficult.

- **Is the current process understood and consistent?** Even though the developers are not yet gathering complete knowledge of the task, they need to assess whether the task can be completely broken down into discrete steps that can be modeled. This is particularly relevant to automating the design of a part, where the process is dependent on the skills and behavior of the human designers. The design process is subjective and often inconsistent; at many companies it is not documented. In addition, to having their own methods, the designers may not be able to identify the steps of their personal techniques, as explained in Chapter 1, Section 3.

Automated Task Process

The developers assess the feasibility of creating an application by reviewing the automated task process with the engineers and experts on software development. They should answer these questions:

- **What is this design automation application supposed to do?** The developers should conduct a documented needs/requirements analysis to determine what the design automation application is supposed to do. In particular, the developers should consider who will use the application, and utilize their input in answering this question. The developers will ensure that what the engineers would and would not like to see in the application is taken into account. In addition, identifying the scope of the application will ensure that the development cost and time can be estimated correctly.

- **How does the automated task fit into the overall design process?** What data will be needed by this design automation application, and what will it pass on to the next step in the process? For example, if a design automation application is to make design decisions based on existing geometry, is the geometry data be available in a form that can be used by the application? Will the geometry generated by the application be compatible with the current computer aided design (CAD) systems? This issue was not considered in one case that was described. The time saved with the design automation application was lost when the data format could not be converted easily to be read by the company's CAD software.

What automating the task entails

The developers need to assess how they can expect to create the software, and what problems they might encounter. They should answer these questions:

- **What knowledge representation and implementation schema are feasible?** What will the development of the software entail? Are there technologies available to manage the knowledge required? Are there natural implementation

schema available to represent the knowledge? Included here is also the identification of suitable hardware and software needed for the approach to knowledge representation and software development.

- **What proof of concept is needed?** Before developing a complete system developers often must build a prototype that proves the concept of the automated design process. The prototype demonstrates that a particular approach to representing and implementing the knowledge will work to automate this task. Developers should consider whether a paper demonstration of the concept, a demonstration model, or a skeleton of the application itself is needed, so that they can assess the time it will take to develop.

- **How complex will the automation of the task be?** The developers should use software development experience (their own or of software experts) to assess if the desired functionality of the application can be developed with the given time and cost constraints. In addition, what options are there for meeting some of the goals for less cost (financial and time), for developing a less comprehensive application, or for dividing the project into phases?

- **Is the project too risky?** The developers should consider who needs to be involved in the development process, and whether they are available and interested in the project. In addition, the level of support by upper management should be considered. Lack of cooperation from engineering managers practically ensures that the engineers will not willingly assist in the project.

4.2 Economic considerations

Developers note that the final selection is often dependent on demonstrating an acceptable return on investment for the automation project. To do this accurately, the developers need to estimate the development costs of the design automation application and the tangible and intangible benefits the application will provide. This information is not readily available, and they will need to determine how to collect the data on which the estimates are based.

Project Cost

The developers need to consider several aspects of the project cost. These include:

- The approximate man - hours needed to develop the application
- The cost of hardware and software
- The cost of training in the use of development tools in time and money
- The effect of the development process on current design activity. A key consideration is that design engineers will be removed from their regular work to

assist in the knowledge capture stage or, as in some companies, to develop the design automation application.
- The cost of operating, maintaining and modifying the application should be carefully considered. As mentioned earlier, this can exceed development cost.

The cost should be documented as objectively as possible. An approach described by Guida and Tasso (1994) was suggested earlier.

Return on Investment

The development team also needs to measure the benefits expected from the application, and must devise suitable metrics for this. As described by Guida and Tasso (1994), they should compile a list of benefits and look for ways to quantify them. Some expected benefits include:

- Reduced design time by minimizing the errors that need to be corrected
- Reduced time to change design when specifications change
- Increased quality of the design by performing more iterations
- Increased quality by using more accurate computations and analysis
- Decreased manufacturing cost due to better designed parts or tools

To evaluate these benefits, developers suggested simple methods. One method is to compare the time the task takes with the current technique to the estimated time it will take once the design automation application is being used. In cases where the time benefit can be calculated, such as meeting a time to market goal, the developers can estimate whether the application will help meet this goal or not. For example, if the goal of automation was a reduction in the time it takes for the company to submit design bids, the developers can estimate how much faster bids will be submitted. However, because it is difficult to quantify many other expected benefits, the developers should identify and describe those in their assessment of the benefits (Guida and Tasso, 1994).

5. Conclusion

Some companies are already using many criteria in reviewing and selecting tasks for automation. However, the unstructured review process means that even at these companies, the reliability of data used to assess the tasks is suspect. The formal approach to review structures useful criteria described by developers and found in the literature into questions that developers should answer at this stage. To answer these questions the developers must evaluate the criteria that are important. By giving careful consideration to the technical and economic aspects of developing the application, the developers can select tasks for automation that will have strong positive impact on the design and manufacturing activity in the company.

Chapter 6 Knowledge Acquisition

1. Introduction

The development of an accurate model of the task and of the application in the application development stage depends on thorough knowledge capture. Some developers feel that this process differs too much from case to case to be organized into formal methods. Most, however, concede that simple techniques to optimize the acquisition process and to avoid mistakes can be of value. This chapter presents some of techniques and considerations that can help guide developers through a more efficient knowledge capture process.

To determine what techniques can be of assistance, developers were surveyed about their current practices for capturing and modeling design knowledge. These are described and analyzed in Section 2. In Section 3, ideas in the literature are reviewed, and their suitability for use in the commercial environment is discussed. In Section 4 a formal approach to the capturing knowledge is presented. Finally, in Section 5, the value of these practices and their use is summarized.

2. Current Activity

The developers described different sources of knowledge. To determine how these were used, questions were asked about the sources they used, about the techniques in use for capturing knowledge, and about methods that affect the capture process. Their responses are shown in Table 5, and are on a scale from 0 (not used) to 5 (significant use). The sources, capture techniques, and methods that affect the capture process are described below.

Table 5 Current knowledge capture activity

Knowledge Capture	*A*	*B*	*C*	*D*	*E*	*F*
Effectiveness of current techniques	3	3	3	3	3	3
Knowledge Sources						
Experts	3	3	3	5	5	1
Documentation, manuals, industry codes	4	4	3	5	3	2
Case studies (i.e. past designs)	2	2	4	4	3	5
Users (i.e. sources of information on user issues)	5	4	3	4	5	2
Capture Techniques						
Formal Process?	1	1	3	1	2	0
Interviews of experts by developers	3	4	3	5	3	1
Observation of experts carrying out task	2	1	3	4	2	1
Experts capture their own expertise	1	0	3	4	5	0
Deduce design knowledge from case studies	2	0	4	4	5	5
Computer techniques	0	0	4	1	1	1
Methods that affect the capture process						
Expert Selection	2	2	3	3	5	1
Methods for conducting interviews/task observation	2	2	2	4	3	2
Expert involvement in the process?	5	4	3	4	5	3
Training in knowledge acquisition issues	2	2	3	4	1	2
Reviews of process (to learn for next time)	1	1	3	1	2	1

Sources of knowledge

The developers gather their knowledge of the design process from experts in the domain of interest; from documentation such as company procedures, engineering codes, and manuals; and from case studies such as past designs. All of these resources are used together, as each contains some information that is either not available or difficult to obtain from the others. Some time is spent becoming familiar with the domain and the problem to be solved. This varies according to the background of the developers. As was described in Chapter 2, they may be engineers with working knowledge of the problem, or knowledge engineers with little or no familiarity with the problem.

All of the companies utilized experts as an important source of knowledge, except for company F. Both companies D and E utilize developer-experts a great deal. In addition to learning from themselves, they also utilize case studies to assist them in breaking down the design task into steps. At company A, where many applications enhance CAD activity, the primary source is the company procedure that is relevant to the task. However, they place great importance on keeping the applications in line with user needs, and talk to experts a lot for that information.

Approaches to knowledge capture

None of the companies has a formal approach to capturing knowledge. However, at company C and company D, there are ongoing efforts to create a knowledge capture methodology.

Two basic techniques for capturing knowledge are the gathering of knowledge from experts, and the gathering of knowledge from reviews of past designs, or case studies. Their use varies according to availability of the sources of the knowledge, and the way the company has organized the development process. Knowledge capture from sources such as written guidelines, catalogs, and industry codes is just a matter of gathering facts and explicit procedures. Developers said that it does not require any special approaches or techniques. At company A, for example, the requirements analyses will document the codes and guidelines that are to be incorporated in the application.

- **Expert knowledge approach:** The primary source of the rules and procedures for the design task process is the expert. Knowledge can be elicited from the expert in two ways. One approach is the interview, in which the developers asks questions and the expert answers. The second way, called protocol analysis, is for the expert to carry out the design task while the developers watch and record the activity. They then attempt to learn the steps of the task process from transcripts of his actions. Variations on these methods include having the engineer perform the task while prompting for information as each step is carried out, or having the engineer think through the task and introspectively break down the task into steps.

  The formality of the process depends on how the team is organized. For example, the development team may be a group of knowledge engineers, and experts are called in to provide their expertise. This is the case at company D. In other situations, the development team is comprised of experts as well, so that the interaction between them and the knowledge engineers is more inclusive of the engineers. This is the case at company A. However the experts and knowledge engineers interact, the developers use the interviews or analyses to build a model of the process, which is then tested for its accuracy and completeness through further discussion with the experts.

- **Case study approach:** In some cases the inability to get time with the experts pushes developers to look elsewhere for design knowledge. In several companies the developers are geographically and organizationally separated from the experts, so they cannot use experts as a primary source of knowledge. In the case study approach, used by knowledge engineers at company F and developer-experts at company E, the developers attempt to capture the reasons and rules used in the design task process from existing designs. When enough information has been collected to begin modeling the task, the developers will check with experts to

ensure that they have interpreted the past designs correctly and completely. This minimizes interruptions to ongoing work by the engineers and frees the developers from relying on finding time to work with the engineers. For example, the developers at one company develop working prototypes by reviewing case studies to learn the design knowledge. They rely on the experts to help test the prototypes they develop.

- **Other Approaches:** The usefulness of experts depends on the task being automated. There are times when neither experts (including expert prepared documentation) nor case studies can provide the design process knowledge that is needed. For example, a design automation application was developed to design the geometry of dies. Experts currently do this based on visual approximation. Since the automated method cannot work this way, the developers concentrated their efforts on determining how best to solve the problem computationally. Company F also does a lot of work that depends on analytical tools that had been developed previously. This work is used in conjunction with the case study approach.

Methods that affect the capture process

Developers identified aspects of working with experts and of understanding knowledge engineering that affect how well developers can capture knowledge.

- **Expert selection, interview, and involvement in development:** Developers described several constraints in the use of experts. First, experts are often busy with other projects, and cannot commit the time developers need. As a result, developers try to use written resources for more basic knowledge, and the experts for the rules of thumb that are used for making design decisions. Developers noted, however, that because of these time constraints, the available expert may be a manager who no longer has the most up to date knowledge of the design task. In addition, the developers may not have a choice of experts, and may be required to work with this person.

  Developers have identified the experts as the most commonly used source of knowledge. However, developers do not share a common set of questions for even a small part of the interviewing process. A developer at company D said that "it would be good to have a 'standard' document" of the set of questions that are normally asked.

  Methods in which the expertise is gathered by knowledge engineers face the difficulty of convincing the design engineers to use the application. There is a lot of mistrust, because the design engineers usually have difficulty accepting first that what they do can be automated, and second, that it can done by someone who does not know much

about design. This is why some companies have trained engineers to develop their own applications.

- **Training in knowledge acquisition issues**
  The developers said that although they train to utilize development tools, they do not regularly work on improving knowledge acquisition skills. However, there is ongoing work to improve knowledge acquisition by improving the way knowledge is modeled. The knowledge that is captured needs to be organized in a way that makes it easy to create the representation of the design task. This organization may be called a *knowledge model.* Although methods for modeling knowledge are beyond the scope of this research, some structured approaches also include the knowledge capture process in a framework for developing a knowledge based application. Developers at company D explained that as they allow developer-experts to develop the applications, tools are need to simplify the knowledge capture process. In addition, the knowledge that is part of the model of the design task changes as the design process changes. New knowledge has to be added to the model with regularity. Knowledge modeling software may help simplify the initial capture of knowledge, and the updating of the knowledge when necessary. Some companies, such as company D, are investigating the use of structured knowledge modeling with KADS, described by Wielinga et al. (1992).

- **Reviews of the process**
  The developers all recognized that they learn many new concepts while working on each project. However, with the exception of company C, none of the developing groups takes the time to review and share experiences with other developers. Developers at company F said that learning from past experience could be valuable. However, in their case, where developers take "different philosophical and analytical approaches for the sake of diversity and experimentation", they are certainly missing out on this opportunity. Both companies D and E admitted that they currently did not try to learn from past experience any more than the fact that "experience travels with the developers as they are involved in new projects." However, a developer at company E noted that if the development process becomes more organized with centralized development groups, then it may be advantageous to emphasize experience sharing.

3. Knowledge Acquisition In The Literature

In the literature, lack of management in the knowledge acquisition process is considered the cause of many problems during development. Some of the problems include increased development time and cost, wasted knowledge engineer time, and disgruntled domain experts (McGraw and Harbison-Briggs, 1989). To counter these problems, researchers have suggested ways to plan the acquisition process. They also have divided

the knowledge acquisition process into steps, and described several techniques for carrying them out.

3.1 Planning the process

The knowledge acquisition process has been researched extensively, but few frameworks for the process have been suggested. One structured approach that addresses many of the concerns described in Section 2 is given by Mcgraw and Harbison-Briggs (1989). They present phases in a structured knowledge acquisition process as follows:

1. *Domain familiarization*: gather reference information and research the problem.
2. *Facilities establishment*: determine what equipment will be necessary for the acquisition process.
3. *Set up the knowledge Acquisition program*: develop procedures for record keeping, select suitable acquisition techniques, and develop procedure for reviewing the process for its completeness.
4. *Orient participants*: establish relationship with experts, and train knowledge engineers.

3.2 Knowledge acquisition from users

Kidd (1987) and Greenwell (1988) emphasize that part of identifying the problem to be solved is identifying the types of users and their mode of use (albeit in terms of how to classify the problem) in the task selection process. Kidd (1987) considers an analysis of what knowledge the user brings to the problem solving process an important part of preparing for knowledge acquisition. This includes keeping the user's goals, constraints on the solution, and the acceptability of the system to the user as based on user's understanding of how to solve the problem, in mind throughout the knowledge acquisition stage.

3.3 Knowledge acquisition from experts

Hart (1986), McGraw and Harbison-Briggs (1989), and others suggest methods for expert selection and interviewing. These include interview techniques, methods for beginning and maintaining the relationship with the expert, and methods for recording and tracking the knowledge that is required.

Expert selection

Knowledge acquisition from experts depends on the ability of the expert to describe what he or she does, and the interaction the expert has with the developer (if the expert is not the developer). Lists of important expert attributes can be found in the literature (McGraw and Harbison-Briggs, 1989; Greenwell, 1988; and Miles and Moore, 1994). The concepts they present are summarized below under the headings of domain expertise and project interest.

- **Domain Expertise:** The expert should have much experience in carrying out the design task to be automated. In addition, it is useful to look for experts who have been called upon to train others in their methods or to develop a company standard procedure for carrying out the task. Further, the experts that are available are often those who are not currently carrying out the design task, such as managers of the engineers. It is worthwhile to try to engage experts who are actively engaged in the task or its review, as they are more likely to be able to articulate the heuristics that are behind steps that they take, and are more up to date on the process.

  An important part of the expert's mastery of the design task is the ability to communicate the way that the task is carried out to the developers. The expert must be patient and confident in his or her ability to carry out this activity.

- **Project Interest:** The expert must be willing and able to see the need for his input. One who cannot (or refuses) to accept that expertise can be modeled in the design automation application being developed will be unlikely to provide knowledge that is complete and accurate enough to use in a design automation application. In the real world, the best sources of knowledge also must continue with the day to day job of carrying out this and many other design tasks. It is critical, however, that the expert chosen not only be there for the initial sessions with or as part of the development team, but also be available throughout its development to review the design automation application for missing knowledge. The expert must be available and interested in actively pursuing the development of this application.

Multiple Experts

Early research in expert systems development found that it was easier to elicit knowledge from one expert because it minimized the already difficult job of translating transcripts of conversations or tapes of activity into rules. Some limit the use of multiple experts to cases where knowledge of many disciplines is being incorporated into the design task (Mittal et al. 1985). Others feel that using many experts can be an effective strategy for handling subjects over which there is much disagreement, provided that it is possible for a consensus to be reached by the experts (Hart, 1986). Similarly, Miles and Moore (1994) suggest the multiple experts are useful because they can give a more balanced view of the task activities, and possibly reduce knowledge acquisition time by ensuring that vital information was not missed. They do not, however, explain how to reconcile conflicting views of the process. Further discussion is found in Greenwell (1988), Mcgraw and Harbison-Briggs (1989), and Guida and Tasso (1994).

Interviewing Experts

Some of the important aspects of conducting the interviews include how structured they should be, the environment for conducting interviews, and how to gather the data from the interviews. Guida and Tasso (1994) suggest a group of ten interview techniques that can be used at different steps in the acquisition process to stimulate the expression of knowledge and expertise. Others add details of how to carry out the interviews, including recording techniques, details of room furnishings, and interacting with the experts (Greenwell, 1988; McGraw and Harbison-Briggs, 1989). They also discuss how to conduct interviews at various stages of development, including moving from an unstructured format -so that the knowledge engineer can learn about the domain - to a structured format more suited to addressing particular aspects of the design task process.

Using Protocol Analysis

Research in acquiring knowledge from experts shows the initial stages to be very important, as the tone of the relation with the expert, as well as the framework for subsequent knowledge acquisition, are set then. However, it is exactly then that the developers have more difficulty eliciting knowledge. Their own understanding of the domain, even after learning the terminology and basic principles, is too incomplete to guide the experts in the interview. In protocol analysis, the developers observe the expert in carrying out the design task, based on the idea that minimal interruption of the expert will provide a close view of what the expert actually does, not what he thinks he does. The expert's protocol is transcribed or recorded, and analyzed by the developers. The developers can base further interviewing on the broad structure of the expert's knowledge acquired this way (Edwards, 1991). This method minimizes interference with domain experts and relies heavily on their ability to break down their actions in a way that is understandable to the developers.

3.4 Other techniques

Hayes-Roth (1983) identified the process of extracting knowledge from experts as a bottleneck in the development of expert systems. This thinking was supported by research that showed the ability of experts to articulate the steps they follow to decrease with increasing task complexity (Michie, 1987). There was concern that this problem would hinder the development of commercial applications, where the quantity of knowledge to be gathered is much greater than in research models. As a result, methods of acquiring knowledge that either minimized reliance on experts, or made capturing expert knowledge easier were researched. Automated rule induction and repertory grids are two examples of these methods. They are described below with an explanation of why they have not been adopted by industry.

Automated Rule induction

Automated or machine rule induction utilizes a computer to analyzes large numbers of examples of the design task to identify rules that govern the task. Induction requires and depends on four components to produce useful results (Hart, 1986):

1. An inductive algorithm for inducing the rules from the examples
2. The examples that are analyzed by the inductive algorithm.
3. A set of characteristics to describe the examples so that comparisons can be made between them.
4. Classes that represent the decisions an expert would make based on the rules.

This method was advocated by Michie (1987) who saw it as a viable method for breaking the bottleneck. The main traffic of knowledge would pass through this alternate route, and would be augmented by the use of experts to check for conceptual bugs in the system.

Automated induction is not in use in industry today, although attempts have been made to implement it in the past. It is hardly mentioned in more recent literature (Miles and Moore, 1994; Guida and Tasso, 1994). Its failure to be accepted has been attributed to several implementation difficulties. These include the trial and error period needed to tune the induction program to the problem domain, and the fact that induction mechanisms only can support knowledge acquisition for relatively simple systems (Kidd, 1984). In addition, the savings in expert utilization is questionable because the expert must be involved throughout the knowledge capture process in order to be able to review the automatically generated description of how to carry out the design task. The most important difficulty is that the rules become excessively complex and less like what is actually done by the domain expert (Greenwell, 1988).

Repertory Grids

The repertory grid is a method to assist the expert in breaking the task down into its components. It is described in depth by Hart (1986). The repertory grid is based on the model of human thinking developed by Kelly called the personal construct theory. The theory is that people classify the objects in their world in a certain way, and use their own classifications to move about their world. In the same way, an expert develops his or her own way of perceiving the problem domain, and the knowledge sought for the design automation application is the expert's classifications and way of moving about the subjective (personal) model of the task. The person acquiring the knowledge assists the expert to articulate this model by identifying domain concepts that are shared by key examples of the task.

This technique has been useful for gathering data to use in structured interviews (Miles and Moore, 1994). However, it is very time consuming and requires that knowledge

engineers be trained in how to analyze human thinking according to this model. As a result, it is not in use commercially.

3.5 Summary

Many of the detailed techniques found in the literature for eliciting knowledge from experts, as well as other ways of gathering the knowledge of the design task, have been unsuccessfully applied to industry. The primary obstacle is that techniques that require special training cannot be used by knowledge engineers on tight financial and time budgets. On the other hand, ideas in the literature for interacting more efficiently with experts have already been received with some interest at companies, and should be considered in the creation of a formal approach to the acquisition process.

4. Formal Approach To Knowledge Acquisition

The objective of a formal approach to acquiring the design knowledge is the quick and complete capture of the knowledge needed for constructing the model of the task. Based on some of the more effective current activity, and ideas in the research, a formal approach that is comprised of three components was developed. The first is the orientation of the developers in the domain, in which the developers learn about the problem and organize for the step of actually gathering the knowledge. The second is actually applying the commonly used techniques of learning from experts and case studies in more efficient ways. The third is to improve the first two components, through reviews of the process. These are described in the sections that follow.

4.1 Domain orientation

Prior to acquiring the knowledge, the developers need to plan the acquisition process. Developers may have no prior experience with the design task, so they cannot identify what needs to be learned about the task until they familiarize themselves with the domain (Durkin, 1994). With this background the developers can review the sources of knowledge available to them, and determine how they will make use of the sources to their advantage. Finally, this orientation process should include ensuring that the experts, who have little or no experience with knowledge based systems, understand the objectives of the project and their role in it.

Learning the Domain

The developers who will be gathering the knowledge accomplish two things by learning about the domain. First, they learn the vocabulary of the domain, so that they can communicate well with the experts. Second, they build a relationship with the experts by making the effort to understand the way they operate. In addition to ensuring smooth knowledge acquisition, the atmosphere of trust and understanding will facilitate the application's acceptance by the engineers.

Different sources of knowledge are available, and they often overlap in what they can provide. In many cases, however, the only source of knowledge about the way the design task is actually carried out or of the related disciplines is a human expert. Heuristics developed over years of experience are rarely written down as amendments to an original task procedure plan, and are difficult or impossible to deduce by analyzing past designs. Because the engineers may not be readily available, efficient knowledge acquisition depends on the identification of which sources of knowledge can be used for acquiring different parts of the knowledge.

The most common sources of needed knowledge are::

- Experts in the domains of the problem
- Documentation such as manuals, company procedures, and industry codes
- Previous designs
- Users of the applications

Although the users are often the experts, they are listed separately to emphasize that they provide important insight into how the application should function. This knowledge can only be gathered from them.

Part of the orientation process should be the engineers becoming familiar with design automation. Often the concepts are completely new to them. If they understand the concepts behind automation, they will be more willing to assist. In addition, this will help dispel fears they may have about the system replacing them at their jobs. In situations where the experts are developers, this is also important because they are being empowered with the development tools. With some background in the concepts behind their use, they will be able to carry out the development process more efficiently.

4.2 Improving knowledge capture

Primary improvements to the capture techniques in use lie in implementing methods to improve knowledge capture from experts and case study review. The developer, by methodically approaching the task of knowledge acquisition from experts, should look to accomplish three objectives that help ensure efficient knowledge acquisition and application development. The developers should develop a good personal relationship with the experts that incorporates them in the development process. The developer should also ensure that the expert is willing and able to provide the needed knowledge. Finally, the developer should carry out sessions with the expert in an organized fashion.

Relationship with the expert

The personal relationship between developer and expert is important. Experts often distrust the capabilities of "computer experts", and may find it difficult to work with the developer. The guidelines for choosing experts based on their personality and interest in assisting (Hart, 1986, and McGraw and Harbison-Briggs 1989) do not seem applicable, since developers usually do not control which experts are available. To counter the

feelings of the experts, the developer must incorporate them in the development process so that continued interest and enthusiasm is ensured. It was mentioned earlier that companies incorporate experts into the process in different ways. For example, in some companies the experts are team members, in others they are just a knowledge source to be interviewed, and in others they are the developers. The evidence indicates that the experts need to be included in a way that will make them feel part of the project. Therefore, effort should be made to create development teams in which experts sit with equal footing to the knowledge engineers. This will help build the relationship between expert and knowledge engineer as fellow developers who will continue to work and refine the design automation application.

Experts ability to provide the knowledge

A second requirement is that the expert have a demonstrated proficiency in the domain, and the ability and desire to share it with the developers. Often an engineer who used to carry out the task but is now managing it is more available for assisting in application development. This expertise may dated, and will need to be supplemented by other experts. Similarly, it can happen that an expert is so good at the task that they cannot articulate any of the steps in the process. The developers should also work to have other experts available to fill in during the development process. Several developers noted that it is often not necessary to use the best expert in the knowledge acquisition process. This person is often too busy to provide enough time for development. Instead, these developer believe that someone who is experienced and understands the process well is good enough.

The debate over the use of many experts for knowledge capture was mentioned. In this research it was found that some advocate using many engineers because it is possible that the approach of one or two to the design task may not accepted by other designers. In addition, each expert is often proficient at only a part of the task. Together they can come up with an even better method for carrying out the task. Opponents counter that adding more experts increases the difficulty of reaching a consensus at all. Developers that were interviewed believe that realistically, using a few experts is desirable, and will not cause much trouble in getting a consensus. It is not usually possible to have many experts on the project anyway. However, provisions must be made for resolving conflicts between experts on the subject.

Organized interaction with experts

Finally the developers should consider how they can structure interviews or other techniques to gain the most from the limited time they have. Although the particulars of the interview techniques mentioned earlier (Guida and Tasso, 1994) may be time consuming, the ideas are valid. For example, at the beginning of the acquisition process, the developer may use an unstructured approach, allowing the experts to talk. Later, when the developer is more aware of the task and what knowledge is needed, he or she

may want to use a more structured format. The developers should work to avoid wasting their time and the time of the experts. For example, if several people will be interviewing the same experts, structured approaches will keep them from wasting the expert's time with overlapping questions.

4.3 Review of the process

Training in knowledge engineering is focused on learning how to model knowledge, and how to use the development tools for creating design automation applications. However, it seems logical that the effectiveness of the knowledge capture techniques can be improved by minimal training and by reviewing the process.

Training in knowledge capture techniques

Several approaches to capturing knowledge from the experts were identified above. By devoting some time to learning about the knowledge acquisition process, a developer can better choose methods for knowledge capture, as well as better carry out the methods chosen. The developers doubted that an in-depth process for knowledge acquisition would be usable because of the time it would require. However they admitted that in addition to learning how to choose the way they capture knowledge, they could benefit from learning more about the way knowledge can be stored.

Reviews of past experiences in knowledge capture

Most development teams do not review their knowledge acquisition process and track where one approach was successful while another was not. Although it has not been proved, those developers that review and share their knowledge acquisition experiences with the group appear to be sharing some of the experience they gained. In addition, as the developers work on more projects with the same group of people, keeping track of how people worked together, what techniques were effective, and similar data, can improve the effectiveness of the acquisition process.

5. Conclusion

The concepts that were described in the approach to knowledge acquisition and modeling are based on applying common sense and structure to the current process. It is not likely that complex techniques for carrying out knowledge acquisition will be used, unless they are implemented in software (an idea that is being pursued in research at several companies). However, by structuring the current process as describe, and emphasizing the preparations that should be made, the developers can effectively orient themselves for the task, and efficiently capture the knowledge needed to automate the task.

Chapter 7 Deployment

1. Introduction

When the developers complete the development and validation of the application, they are prepared to introduce it to the engineer. However, the resistance of the engineers to the new application can stop the application from being used. In addition, as the information on which design decisions are made changes with changing technology, the application needs maintenance to ensure that it remains useful. Finally, to determine whether the project was successful, developers need to evaluate the benefits of the application, and the effectiveness of the development process itself. By utilizing a formal approach to ensure that these issues are considered before and during deployment, developers can minimize problems with getting the application integrated into the design environment, and learn from the experience for future projects.

2. Current Activity

All of the companies that were surveyed follow a cycle of development and alpha (and sometimes beta) testing before the application is deployed. Developers and consultants stressed that the validation of the application must be complete for it to enter into service. They put significant effort into this activity. The validation process and different approaches to ensuring the reliability of the application are beyond the scope of this research.

The developers consider deployment to be the introduction of the application into the design environment, its maintenance and its modification (Figure 3). As part of this step, the benefits that the application is providing are evaluated. To identify current practices in application deployment, the developers were asked to describe what activities comprised the introduction of the application, the effort put into maintaining and modifying the application, and how they measure the benefits. In addition, the developers were asked to what extent they review the development process to learn from the experience. Their answers are shown in Table 6, and are on a scale from 0 (no effort or problem) to 5 (significant effort or problems). Their activities are described below.

Table 6 Current Deployment Activity

Deployment of the Application	*A*	*B*	*C*	*D*	*E*	*F*
Introducing the Application						
Convincing engineers to use the application	3	3	2	2	5	1
Problems in getting applications used?	1	2	4	3	3	1
Changes to the design environment	2	1	2/4	2	3	0
Plan to maintain and modify application?	4	3	2	2	3	4
Evaluating the Application						
Measure savings and benefits	3	2	3	3	3	1
Reviews of development process	1	1	2	2	2	1

Introducing the application

In addition to setting up the necessary equipment and managing the integration of the new software with the company's current systems, the developers include several activities in the introduction process. In many cases they must convince the engineers to use the application. Also, in many cases implementing the application effectively requires changes to the design process and to the roles of the engineers, as was discussed in Chapter 5. Once the application is in use, it is necessary to keep the application up to date and operational.

- **Convincing engineers to use the application**

  Developers at company D indicated that although they believe that through the use of software development methodologies the development process is carried out smoothly, applications do not meet their potential because developers must still sell the proven technology to the internal engineering organizations. A developer in one division of company C agreed that the fact that software developers are providing the engineers with the applications creates resistance to the applications. Both companies C and D are working to shift to the approach of company E, where many applications are developed by the engineers themselves. In addition a developer at company D suggested that "a strategic direction toward design automation might eliminate some of the 'sales' effort" that is currently required.

  Developers at companies A did not have significant problems in getting users to use the applications, partially because the applications are usually enhancements to the existing CAD system, and are readily integrated with the tools the engineers currently use. At company B, the entire process being automated falls gradually under control of the developers, and their control ensures the use of the application. Similarly, at company F, where an engineering group sponsors the project, its implementation is usually supported by champions in that group, and they ensure its use.

Developers also described misconceptions about what to expect from automating design in terms of capability and in terms of how the system will operate. Especially in cases where the developers are computer experts and the users are engineers (as in company D, E and F), very complex applications are often not understood. For example, consultants note that clients often complain that the computer program takes more than several minutes to run, despite the fact that the manual method took several days. They said that these issues, trivial as they may sound, can seriously erode the confidence of the engineers in the application

The current method for 'selling' the use of the application to the engineers is through formal and informal training. For example, at companies D and C, the developers, although they are knowledge engineers by training, work in an integrated team with engineers. As described in Chapter 6, while the knowledge engineers are learning from the experts, the experts are learning about the power of design automation. The engineers find the application more acceptable when presented by peers. In addition, formal training in the use of the applications emphasizes its capabilities.

- **Changes to the design environment**
  Developers said that since the design automation application is intended to replace the current process, there are naturally some changes in the design process. However, developers at companies B and F stressed that their approach is not to replace the current design process. Rather, "the idea is to 'augment' the existing system" by implementing changes in an evolutionary fashion. Revolutionary change is not desirable, and has failed in the past at company B. An engineer at company E said that this approach often means that initially the application adds more work because the process being replaced has not been phased out. One division at company C uses design automation to facilitate change in the design process, so changes there are significant. At others however, most applications do not require changes to the way the engineers work.

- **Plan to maintain and modify application**
  At company E, where the applications are often developed by engineers, they also usually maintain what they developed. Sometimes the application's maintenance is passed to a computer support group. Developers at company F consider maintenance to be part of a never ending knowledge acquisition process. The design processes being modeled are always changing, and a result, "the tool is never done." For this reason, long term support of the application is considered vital to its success. Other developers however, noted that while the support is important, large changes to the application an regular intervals may indicate that the design process was not stable enough to make it a suitable candidate for

automation in the first place. A developer at company A state that their experience has been that applications fall into disuse unless they are fully supported. The developers therefore plan to support all of their applications.

Evaluating the application

Developers explained that the benefits to be evaluated were identified when the project was selected initially. They described both how they evaluate applications and some of the metrics they use to do so.

- **Measure savings and benefits of application**

  A key consideration in attempting to evaluate how much the application is saving the company is how much the application is really being used. Company D employs a system for monitoring how often applications are used, which functions are used, what errors were encountered, and other issues. This information is used for estimating the value of savings through the use of the application, to compare with original estimates. A developer explained that they had tried to verify the savings with accounting methods, but no charging process in place permitted a fair comparison. As a result, one of the primary indicators of value remains the customer's satisfaction. "If they are happy - we're happy" explained one developer. At company A, the application use is not tracked, although a developer said that this is something that is being changed. The developer at company F said that since the engineering group decides for itself whether they are satisfied with the application, the developers do not have the ability to track its value. The developers rely on customer satisfaction when considering if the application was a success. In this case, the savings are usually the fact that the part is of higher quality, something that is very difficult to measure. For the developers, however, customer satisfaction means that the application is accepted and working as desired. At company B, the primary goal of automation may be to facilitate designs that are extremely complex and could not be completed without automation. The application is a success if it facilitates these more complex designs. Similarly, at company A reduction of lead time is often a key consideration. If the automation application succeeded in reducing lead time for the product, then it is considered a success.

  Company D has been using several metrics in its evaluation. To evaluate labor saving, the automated process is compared to the manual one. Developers note, however, that since no one will pay for controlled calculation of labor hours, they must often compare dissimilar work as best as possible. To evaluate increased design consistency, the company is not interested in economics and focuses on measuring part similarities using a neural net system and manual comparison. For benefits such as support for last minute design changes and integration of analysis into design, the concern is whether this is the case or not, as opposed to how much

this is worth to the company (this would have been considered in selecting the task initially). These are documented by developers reviewing the use of the applications. Similar metrics are used at company E as well.

- **Reviews of development process**
  In Chapter 6 it was noted that the companies do not review the knowledge capture process to learn from the experience. They said the same about the overall process. Developers at company D explained, however, that in cases where engineers are used to develop applications, the development expertise is not wasted. These engineer-developers become local experts in application development and their experience is utilized. Developers agreed that were it not for project time and budget constraints, sharing experience would be a useful idea.

3. Literature

3.1 Application introduction in the literature

None of the over twenty case studies that were reviewed even mention how the application was introduced into the design environment, except to note the hardware and software that were needed. Books on knowledge based system development emphasize validating that the application provides reliable results (Miles and Moore, 1994; Guida and Tasso, 1994; Edwards, 1991), but do not discuss meaningfully the realistic problems encountered during deployment. The exception is Prerau (1990), who lists three considerations in integrating the system into the environment. These are the integration of the computer hardware with existing hardware, the integration of the system with the systems that provide input and accept output, and the integration of the system into the working situation of the engineers. He discusses working with the engineers as a separate concern, as described below.

User Acceptance

Prerau (1990) considers the problems that developers may face in getting the users to accept the system as a serious one, and suggests two ideas for smoothing the process.

- User education: This will help them avoid unrealistic expectations from the application such as the time it takes to complete the task. A consultant mentioned that even when the system reduces the time it takes to complete the task dramatically, engineers complain about how slow the system is. In addition, if users understand the technology behind these systems, they will trust them more. Guida and Tasso (1994) add that indirect education through interaction during the development is more effective than a series of training sessions.

- Introduction without major upheavals: The developers plan in advance how to set up and put in place the system so as to minimize the amount of extra work imposed on

the engineers. A key to this is developing an application that with an interface that is understandable to the engineers, and operates with their terminology.

Prerau (1990) also suggests including many features that will help the users. In fact the developers surveyed indicated that this approach is more likely to lead to confusion over which features are needed and which are frills. In addition, it will add unnecessarily to the development time.

3.2 Evaluation in the literature

Measuring the success of knowledge based systems is discussed briefly by Edwards (1991). He recommends the use of operations research analyses for turning commercial needs into a set of metrics for evaluating the application. He does not suggest how data for these metrics can be gathered. Guida and Tasso (1994), in discussing the plausibility study described in Chapter 5, list and discuss criteria for estimating the worth of the application. These will also be used now to compare with initial estimates.

4. Formal Approach To Planning For Deployment

Developers agreed that the success of the application depends on smooth integration into the design process. In addition, they admit that reviewing the development process can help them in future endeavors. A formal approach to planning for deployment ensures that these activities are carried out efficiently.

4.1 Considerations for application introduction

Convincing the Users

Many of the difficulties that occur in introducing the new design automation application into the existing design environment are related to the way the designers use the application and their resistance to change. To minimize these difficulties, the developers should consider three issues.

1. *Preparing for deployment throughout development:* From the project description through development the developers can minimize surprises by ensuring that the engineers are included and their opinions counted. Developers admit that if user opinions are heard during the development, the developers can more readily count on their support and patience during introduction of the application.

2. *Training for the current and future users*: For their part, the developers cannot let the design automation application "free fall" into the design environment. The formal training that is necessary is easily supplemented by informally checking that the new technology is understood and acceptable.

3. *How will the application be maintained*: The practice of supplying full support for the application is useful because it ensure that the developers keep watch over

the application. This does not have to be the developers' job, however. For example, as at company F, the developers and engineers who will use the tool are not geographically collocated, and responsibility is given to a closer support group. The key here is that the application is supported and modified in an organized fashion that maintains the reliability and usefulness of the application.

What changes to the design environment are expected

Companies generally do not expect to alter the entire design process with the advent of the new application. However, management, with the advisement of the developers, can take into consideration how people's jobs and work load will change both during and after the application is developed.

- *Including the Design Engineers During Development*: Whether the application will automate the complete design of a product, act as an assistant, or provide information, the role of the engineers in the development needs to be defined. The importance of engineer buy-in to the application makes it vital that they be part of the development process. This involvement will take resources from current design activity. The company must develop plans to cover the current work, so that development of the application will not interfere with the company's product development process.

- *Changes To The Design Environment When The Application Is Introduced*: The new role of the design engineers needs to be defined for two reasons. First, success of the application depends on convincing the engineers to use it. To allay their concern for their jobs they must be convinced that their expertise is needed even more with the introduction of the tool. Second, the roles of the engineers must be redefined to take advantage of the benefits the application will provide. If an application was designed to free the engineers to improve design quality, this new responsibility must be made clear to them and to management. Their new responsibilities should be agreed upon before the application is developed. Developers have described situations where, out of concern for their own bottom line, managers have cut positions when applications became operational. This undermines the entire automation endeavor.

4.2 Evaluating the design automation application

The developers indicated that very few benefits can be measured quantitatively. Even where they can, such as with the neural-net system at companies D and E, it may be too costly and time consuming. As a result, developers should not expect to formulate an exact set of metrics, and this should be clear to management as well. Instead, the benefits that were identified in the task review and selection should be assessed now.

4.3 Development process review

The development process used should also be reviewed and documented for future use. As with any development process, the developers should review the development of the design automation application to identify where they can improve, and to identify lessons learned. For example, the experience gained in the knowledge acquisition process can be shared with other developers who were not involved in knowledge acquisition, providing a case study of what works and what does not.

5. Conclusion

In deploying the application the developers can encounter obstacles to the success of even well developed and validated applications. The goal of a formal method for deploying applications is to ensure that these problems are considered well in advance and plans laid to minimize the difficulties. In addition, as part of deployment the developers can review the entire process and learn from their mistakes and successes for future applications.

Chapter 8 Conclusion

Summary of Findings

Much past research has focused on the feasibility of automating design tasks, and the development of the technology needed for this. At this point, when the technology is mature enough, it is the development process that must be refined (Guida and Tasso, 1994). For this reason, companies today are expressing interest in learning about their development process, and finding ways to improve it. The feeling among developers is that a significant factor behind the lack of consistent success in developing these applications is the ad hoc development processes currently in use. Developers made it clear, however, that require major changes to the current organization in order to improve the development process are not likely to be accepted by their companies.

In this research the current development process was studied to find where formal methods could help improve the development process. Five development activities were identified, and four of these were addressed. These were:

1. Opportunity Evaluation
2. Task Review and Selection
3. Knowledge Acquisition
4. Deployment

Each of these activities was studied, and useful current practices identified. These and other ideas found in the literature and discussed with developers were synthesized into a formal approach to utilize in carrying out each of these four activities. In addition, the effects of a company's organization for developing design automation application were noted. Recommendations were given for identifying, based on the company's choice of people for carrying out development activity, the positive and negative aspects of using a person with that background.

These formal methods do not provide the developer with instructions on what to do, or on documents that should be written. Rather, the developer should utilize this thesis as a set of formal approaches for thinking through these steps. In addition, not all of the considerations apply to everyone. Developers should look at these issues to be considered, and note which are relevant to the organization in which they work. The intent of the formal approach is not to constrict the developers and require documents that may not be of value. Rather, the formal approach reminds the developer what are the key issues that should be considered during these activities.

Future Work

Future work in developing formal methods for application development will be in several directions. These include suggesting formal methods for other parts of the development process, developing metrics, and developing techniques through which these formal methods can be taught to developers. Each is described below.

- **Formal methods for other parts of the development process**: In this research software development was identified as an area that can be improved. As the subject of methodologies for software development is extensively treated by researchers in the development of conventional software, it was beyond the scope of this research. However, there are many differences between design automation applications and conventional software, and their development processes (Wagner, 1990; Guida and Tasso, 1994; Miles and Moore, 1994). Understanding how the software methodologies and standards can be utilized to improve the design automation application development process is one area to be studied further. Another area that was not identified in this research is project planning.

- **Developing metrics for the development process**: Several development activities that were formalized in this research could benefit from the availability of metrics. For example, in reviewing a task for selection, metrics for determining the complexity of the task to be automated and for the complexity of the knowledge representation problem that is being posed would help developers make objective comparisons between different candidate tasks. Similarly, metrics that can be used to evaluate the success of the application would help the developers better determine the worth of the applications they have developed or are considering. Determining which metrics are of value and how they should be measured is a subject for further research.

- **Developing techniques for teaching formal methods**: Unlike the complete methodology of Guida and Tasso (1994), the formal methods presented here are only intended to supplement the current development process. A major obstacle faced in presenting this research was finding a way to make the formal approaches usable. An area for further study is how to better coordinate these ideas using such techniques as Internet based tools and test development projects.

Bibliography

1. Badiru, Adedeji B., *Expert Systems Applications in Engineering and Manufacturing*, Prentice Hall, Englewood Cliffs, NJ 1992.

2. Binder, John, "Knowledge-Based Engineering: Automating the Process," *Aerospace America*, pp. 14-16, March, 1996.

3. Blount, G., S. Kneebone, and M. R. Kingston, "Selection of Knowledge Based Engineering Design Applications," *Journal of Engineering Design*, Vol. 6 (1), pp. 31-38, 1995.

4. Brown, D. C. and B. Chandrasekaran, "An Approach to Expert Systems for Mechanical Design", *Trends and Applications, 1983*, IEEE Computer Society Press, Silver Spring, MD, pp. 173-180, 1983.

5. Chin, Kwai-Sang and T. N. Wong, "Knowledge-based Evaluation for the Conceptual Design Development of Injection Molding Parts," *Engineering Applications of Artificial Intelligence*, Vol. 9(4), pp. 359-375, 1996.

6. Clibbon, Kelvin and Ernest Edmonds, "Representing Strategic Design Knowledge," *Engineering Applications of Artificial Intelligence*, Vol. 9(4), pp. 349-357, 1996.

7. Cochran, W. Joseph, Don Hainley, and Loay Khartabi, "Knowledge-Based System for the Design of Heat Exchangers", *Applications of AI 1993: Knowledge-Based Systems in Aerospace and Industry* edited by Fayyad, Usama and Ramasamy Uthurusamy), International Society for Optical Engineering, Bellingham, WA, pp. 138-148, 1993.

8. Davies, P. T. and J. D. A. Anderson, "Knowledge Based Engineering Systems in Practice", *Effective CADCAM*, Mechanical Engineering Publications Ltd., Suffolk, England, pp. 121-127, 1991.

9. Dixon, John R., "Artificial Intelligence and Design: A Mechanical Engineering View," *Proceedings AAAI-86*, AAAI, Philadelphia, pp. 872-877, 1986.

10. Durkin, John, *Expert Systems Design and Development*, Macmillan Publishing Company, New York, 1994.

11. Dym, Clive L. and Raymond E. Levitt, *Knowledge Based Systems in Engineering,* McGraw-Hill Inc., New York, 1991.

12. Edwards, John S., *Building Knowledge Based Systems,* Pitman Publishing, London, 1991.

13. Gammons, Richard A., "ESKIMO: An Expert System for KODAK Injection Molding Operations," *Artificial Intelligence in Design* (Edited by C. Tong and D. Sriram), Academic Press, San Diego, pp. 81-104, 1992.

14. Greenwell, Mike, *Knowledge Engineering for Expert Systems*, Ellis Horwood Ltd., Chichester, England, 1988.

15. Guida, Giovanni and Carlo Tasso, *Design and Development of Knowledge Based Systems,*" John Wiley & Sons, 1994.

16. Harmon, B., and P. Sawyer, *Creating Expert Systems for Business and Industry*, John Wiley & Sons, New York, 1990.

17. Hart, Anna, *Knowledge Acquisition for Expert Systems,* Kogan Page Ltd., London, 1986.

18. Hayes-Roth, F., "Knowledge-Based Expert Systems - The State of the Art in the U.S.", *Experts Systems State of the Art Report* (edited by J. Fox), Pergamon Infotech Ltd., Maidenhead, England, pp. 49-62, 1983.

19. Heatley, Lynn and William Spear, "Knowledge-Based Engineering Design at Xerox," *Artificial Intelligence in Design* (Edited by C. Tong and D. Sriram), Academic Press, San Diego, pp. 179-195, 1992.

20. Holt, Jon D., "Practical Issues in the Design of AI Software," *Engineering Applications of Artificial Intelligence*, Vol. 9(4), pp. 429-437, 1996.

21. Hopgood, Adrian, *Knowledge-Based Systems for Engineers and Scientists*, CRC Press, Inc., Boca Raton, FL, 1992.

22. Kidd, Alison and Margaret Welbank, "Knowledge Acquisition", *Expert Systems State of the Art Report* (edited by J. Fox), Pergamon Infotech Ltd., Maidenhead, England, 71-80, 1984.

23. Kidd, Alison, "Knowledge Acquisition - An Introductory Framework," *Knowledge Acquisition for Expert System* (edited by Alison Kidd), Plenum Press, New York, pp. 1-15, 1987.

24. Marra, John, "An Expert System Eases Rotor Design," *Mechanical Engineering*, Vol. 119 (4), pp. 70-72, 1997.

25. McGraw, Karen L. and Karan Harbison-Briggs, *Knowledge Acquisition: Principles and Guidelines,* Prentice Hall Inc., Englewood Cliffs, NJ, 1989.

26. Michie, Donald, "Current developments in Expert Systems," *Applications of Experts Systems* (edited by J. Ross Quinlan), Addison-Wesley Publishing Co., Sydney, pp. 137-156, 1987.

27. Miles, John and Carolynne Moore, *Knowledge Based Systems in Conceptual Design*, Springer-Verlag Ltd., London, 1994.

28. Mittal, Sanjay and Clive Dym, "Knowledge Acquisition from Multiple Experts," *AI Magazine*, Vol. 6 (2), American Association for AI, Menlo Park, CA, pp. 32-36, 1985.

29. Pahl, G. And W. Beitz, *Engineering Design* (translated by Arnold Pomerans and Ken Wallace; edited by Ken Wallace), Springer-Verlag, New York, 1988.

30. Paulk, Mark, Charles Weber, Bill Curtis and Mary Beth Chrissis, ed., *The Capability Maturity Model: Guidelines for Improving the Software Process,* Addison Wesley Publishing Company, Reading, MA, 1995.

31. Prerau, David S., "Selection of an Appropriate Domain for an Expert System," *AI Magazine*, Vol. 6 (2), American Association for AI, Menlo Park, CA, pp. 26-30, 1985.

32. Prerau, David S., *Developing and Managing Expert Systems,* Addison-Wesley Publishing Co., Reading, MA, 1990.

33. Proctor, Paul, "Boeing Adopts 'Expert' Design System", *Aviation Week & Space Technology*, p. 27April 24, 1995.

34. Rangan, Ravi M., Kenneth Maddux, and William Duwe, "A Practical Methodology to Automate Routine and Data Intensive Design Activities," *Engineering Data Management: Integrating the Engineering Enterprise* (edited by P. Bocks and B. Prasad), New York, pp. 119-129, 1994.

35. Rehg, Jim, Alberto Elfes, Sarosh Talukdar, Rob Woodbury, Moshe Eisenberger, and Richard H. Edahl, "Design Systems Integration in CASE", *Expert Systems for Engineering Design* (edited by Michael D. Rychener), Academic Press, Inc., San Diego, pp. 279-301, 1988.

36. Suh, Nam P., George Chryssolouris, Timothy Gutowski, Emanuel Sachs, and Nathan Cook, "Manufacturing Engineering," MIT, Cambridge, MA, 1990.

37. Wagner, Martin R., *Understanding the ICAD System,* Concentra Corporation, Cambridge, MA, 1990.

38. Waterman, Donald, *A Guide to Expert Systems*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

39. Wielinga, B. J., A. T. Shreiber, and J. A. Breuker, "KADS: A Modeling Approach To Knowledge Engineering," *Journal of Knowledge Acquisition*, Vol. 4, pp. 5-53, 1992.

Appendix A Case Studies

1. Introduction

Developers at several companies were interviewed to gather data. Most responded to the survey that is in section 2. In addition, phone interviews, electronic mail correspondence, and site visits were used in order to obtain a complete picture of development activity at the companies.

2. Sample Questionnaire

DESIGN AUTOMATION:
APPLICATION DEVELOPMENT QUESTIONNAIRE

These questions are designed to walk through the respondent's company's current practices for developing applications for automating portions of its design processes. All the information in the completed questionnaires will be confidential, with the results of the questionnaires made available. While the results and conclusions will be used in publication, company names and activities will not be published without permission from the company.

COMPANY: _____
NAME: _____
TITLE: _____

OVERVIEW

I. What organizations are typically involved with design automation, and how are they related? What is your own involvement in design automation? *Some application development groups are self-contained consultants to various units of a company; others may be more closely integrated with the people using the applications.*

II. What kinds of design automation applications are being developed (e.g., to provide design analysis, CAD data, design guidelines, etc.)? Please use examples.

III. How large are the design automation projects being undertaken? *How long does it take to develop the application? How significant is the task to the design process? Is the development cost significant to the company?*

IV. What is the educational background of the people involved in developing design automation applications (AI, mechanical engineering, software engineering etc.)?

IDENTIFICATION AND SELECTION OF DESIGN TASKS FOR AUTOMATION

I. What are the objectives of the design automation effort (e.g., reduced time to market, lower design and manufacturing costs, improved design quality, demonstrate automation technology, etc.)?

II. How are candidate design tasks and processes identified (e.g. research, suggestions from design teams or managers, a search by a team devoted to identifying processes to automate)?

III. Who researches a candidate task and decides if it should be automated?

IV. How does the company determine whether to automate a suggested design task or process? Please describe any formal methods used in this step. *In some cases, design automation in the company began because a task was selected for automation. If so, why was the task chosen?*

   A. What economic issues are considered, and what metrics are used?

   B. What organizational issues (resources, changes resulting from automation, widespread use of the application, etc.) are considered?

   C. What are the technical considerations in deciding if efficient automation is feasible, and what metrics are used?

V. What proof of concept (prototype, etc.) is developed before embarking on a full scale project? What is the purpose of this proof (e.g., demonstration of a representation method, a skeleton of the application)?

PROJECT PLANNING

I. When and how are the requirements of the application to be developed determined?

II. How are design automation projects planned?

   A. Are applications implemented in stages?

   B. Are temporary solutions implemented and replaced later by a more developed application?

III. How is the application development process organized?

   A. Who captures the design knowledge?

   B. Who develops the software?

   C. Who maintains the design automation application?

IV. What kinds of development tools are being used in your company?

V. What plans are made for changes to the design environment during and after the development of the application? *The design process is changing and the responsibilities of the design engineers must be redefined.*

KNOWLEDGE ACQUISITION AND MODELING

Describe how knowledge is captured in your company:

I. What are the sources of design knowledge and knowledge needed for the application?

II. What knowledge acquisition techniques are used for gathering design knowledge? (e.g., interviews, case studies, protocol analysis, etc.)
III. Are there formalized procedures for these techniques? (e.g., interview evaluation techniques, interview formats formalized for use by application developers, case study review to ensure representation of the general problem)?
IV. How are the users of the application being considered in defining the functionality and usability of the application?
V. How do you determine that enough of the rules and process information have been elicited to move to developing the application? *Although there will usually be knowledge that was missed, there is a point when the amount of knowledge gathered is not worth the time it takes.*
VI. How is the design task modeled (e.g., paper diagrams, using a development tool, etc.)?
VII. Is the knowledge acquisition task separated from the representation and software development? How?

KNOWLEDGE REPRESENTATION AND IMPLEMENTATION

Describe how the software is developed.

I. Are any software development methodologies and standards used in the application development process?
II. How is the application's ability to carry out the design task validated?
III. How is the usability of the application validated?

DEPLOYMENT

I. Who maintains and supports the application?
II. Is the use of the application studied? How?
III. How are modifications to the knowledge base and to applications made?
IV. How is the success or failure of an application determined?
V. How does the company track and evaluate success/failure and who is responsible for this? If the company has no direct methods, how do you evaluate if a design automation application was a success?
VI. What reviews of the application and its development are made, and how are they used to improve the development process?

SUMMARY QUESTIONS

I. Based on your definition of success, how successful has your company (group) been at producing design automation applications?
II. What weaknesses do you see in the current method of creating these applications?
III. What should be the next step for improving design automation work?
IV. What changes can the company implement to make the application development process more efficient?

3. Company A

Team members

The interviewee is a group leader for team of people that develop mechanical CAD/CAM/CAE tools. They also support and maintain the software.

Educational background of group members: Mechanical technologists (2 year degree), Ph.D. and MS (mechanical engineering). Help is also solicited from software engineers and a variety of other people with and without degrees. These are engineers whose problem solving capabilities are their primary asset. Programming and writing code is how they provide the customer with the solution.

Overview

Design automation is carried out by internal consultant teams. In addition to the work done by this group, other groups provide more general computer support and analysis tools. The group has been functional, almost since the introduction of CAD at the company.

Design Automation applications that have been developed since 1995 fall into three categories (a partial list of examples):

1. Automation of CAD data creation and CAD activities. These applications also support the use of data standards:
   - Symbols are available to be included in electronic drawings
   - Automated creation of geometry. In addition to data on the geometry, manufacturing requirements are also incorporated so that the model can be used to select the appropriate tools to make the desired geometry.
   - Application to plot CAD models in batch mode, avoiding the time consuming part of opening the CAD system and waiting for each of a large set of drawings to load interactively.
   - Checker to find differences between two solid models
   - Automated completion of a drawing format border, including a prompter for information which is then placed in the appropriate places on a drawing
   - Aid for the transition of CAD models from the old version of the CAD system's method of managing assemblies to the newer version.
   - Libraries of commodity part geometries (CAD models) that are automatically available to design engineers.
2. Automation of design checking for the geometric robustness of CAD models relative to standards:
   - Checker for interference within a whole assembly (not just two parts)
   - Checker of a part's conformance to CAD standards, such as noting if dimensioning is on the correct layers of the drawing (this cannot tell if a dimension is missing).

3. Automation of the design process for a class of product
   - Automation of design and manufacture process for component assembly

     The current process
     1) Use commercial software to find assembly with desired properties.
     2) Make engineering calculations for design and manufacturing process
     3) Document in CAD system
     4) Decide on manufacturing process
     5) Decide whether to outsource, begin prototype production, or begin mass production.

     The automated process
     1) Set up prompting system for transfer of data from commercial software to CAD and automate whole transfer of data
     2) Automated engineering calculations for design and manufacture process (separate application was developed).
     3) Automated CAD documentation -- model and specifications.
     4) Decision on manufacturing is still manual, but is now based on this design process
   - Design assistance for the designers :
     - Automated selection of desirable machine components
     - Tolerance analysis software

Most day to day work is on projects of relatively small size, lasting for a few months, and involving one group member. Some took more than a year's effort by several people.

Identification and Selection of Design Tasks for Automation

The company's primary objectives in implementing design automation are to improve data management and save time in the use of CAD. Not very much work is being done on applications that incorporate and automate design knowledge for whole "product models".

There is no formal process for reviewing tasks to look for automation ideas. There are two sources of project ideas:

1. Response to a request from internal customers (design and manufacturing groups). Work is funded by the customer group.
2. Strategic work, i.e., design tools that may meet the needs of many users. Funding is from corporate research and development money.

The ideas are studied and turned into formal proposals or rejected by a member of this group. The potential application is described verbally, and questionable ideas may be demonstrated with a prototype. Sometimes a customer develops a basic application and sends it to the development group who will make it more usable (better user interface for example) and more useful (increased functionality) application. For example, a customer

created a CAD programming language script for managing CAD files, and the development group then created a standard database of CAD files for everyone to access. However, there is no formal way that these partial projects are directed to the development group for completion.

There are no formal methods for reviewing and documenting the appropriateness of automation for a design process. However, there are rules of thumb to use in selecting candidate tasks.

Economic issues

- A qualitative business case is developed informally by the developer for that project. The development cost is estimated to assess whether a project is worthwhile. The estimate is based on the time it will take. The business case also helps to assess validity of implementing short term solution.

Technical considerations

- The application will have breadth of usage (since most tools are generic for getting a job done, not designing a particular product) -- most tools are directed toward most or all of the company's CAD users
- The application will increase the productivity of designers, as measured by a decreased lead time for a project
- Can an existing solution be purchased from the software vendor?

Ideas for solutions come from a team comprised of customers, different types of users, and a member of the development group. Options are developed and a proof of concept is presented to the customers to consider.

Project Definition

Project requirements are determined with a needs analysis carried out by someone in this group, who will be in charge of this project. The needs analysis is used to determine if a short term solution that can be implemented in one or two months can be used while a long term solution is considered. The project leaders hears from the customer of the application and develops a concept and a proposal.

The group member leads a team that includes the customers for the application. This person will develop and maintain the software (with assistance from the overall group).

Knowledge Acquisition and Modeling

Design knowledge comes from either company standards and requirements that are documented, or from engineers.

Knowledge acquisition is an ad hoc process. Since the customers are part of the development team, their input is gathered through interviewing.

Flow charts and similar tools are used to turn the design knowledge into a model of the design task, and the development team is expected to document and manage requirements and specifications. There is no control of the format of this documentation, and no generally accepted appropriate level of detail.

Knowledge Representation and Implementation

Most of the projects do not entail complex modeling of an engineer's activity. As a result, the primary concerns in implementing the knowledge as software are the needs analysis, and software project management through models like the Capability Maturity Model (CMM).

Experience has shown that demonstrating the validity of the application is critical for its acceptance. However, the validation process differs for each application.

Project Deployment

Experience has shown that without the developers supporting their own application, it soon falls into disuse. Therefore the group supports all of its applications.

As mentioned, most applications are developed for all or most of the engineers to use. This makes them easier to maintain, but they often cannot meet all of each individual customer's needs. These needs are met with customization of the application at the customer level, by this development group.

Success is based on developing an application that does what it is supposed to do, and is subsequently accepted by the customers. However, there are no mechanisms for tracking application use and for formally determining the savings obtained with the application.

Notes on Project Success

1. The team has had success in developing useful data management and CAD assistance applications. However, the team has not developed knowledge based design systems for designing complete products.
2. Mechanical CAD/CAM knowledge is very important, for good requirements gathering. This is as opposed to software development background.
3. Success of project may depend on people's confidence in the validity of the tool, especially if it is performing calculations or providing expertise.

4. Company B

Team members

The interviewee is a project manager for the development of CAD tools. The interviewee manages a group that develops applications for customers within the company, such as design and manufacturing. Some work is done that spans several company sites, depending on the scope of the product.

Educational background of group members: electrical engineers, computer science, (not specifically with a background in artificial intelligence). The interviewee describes the team as having a general knowledge of the domains being automated, but that is not their expertise.

Overview

Design automation applications have been developed for simulation, testing, verification of productivity, and data mining activity. The applications are developed using many kinds of development technologies -- from simple spreadsheet applications to expert systems. Sometimes the applications are developed from university research; otherwise the developers come up with the right approach for modeling and representing the task on their own.

Example application: simulate product before it is built using knowledge of the electrical characteristics, to determine how well the product performs.

Applications are usually developed in less than six months.

Identification and Selection of Tasks for Automation

The company does not have a formal process for reviewing tasks to look for automation ideas. Ideas come in various levels of clarity as to how to best automate the task, usually with only a basic description of the core needs. These ideas come from three primary sources:

1. Members of the group, while working on projects for customers, identify other areas of interest. They then suggest a process that should be automated.
2. A customer approaches with a specific problem to be solved, or with a request for an application with specific capabilities.
3. A corporate need is presented to the group by upper management.

There are no formal procedures for reviewing tasks for selection. However there are rules of thumb that are considered in selecting candidates.

Technical Issues

1. Consider whether a particular automation method has been used before, or the kind of application being considered has been developed before.
2. Consider size and complexity of project; larger projects require more resources but are good candidates for automation because of their payoff.
3. Consider the nature of the task. Some tasks have an inherent need to be carried out by humans. For example, a specification has parts that must be written in plain language, and read and implemented by humans.
4. Consider how the knowledge will be captured. For example, knowledge capture for a design advisor application is very difficult at this company. Instead of a developing an application to assist with design, an application may be developed to be used as a checker that compares the manually completed design to established rules to verify whether the design is valid.

Economic Issues

1. Consider how to keep the customer happy – even if the need is economically unsound, try to implement individual pieces of what was wanted.
2. Consider savings in terms of product development money saved.
3. Look at time to market savings.
4. Look at accuracy -- money saved through minimized errors in design.

Other Issues

1. Geographical distance between design teams (teams in different states need to communicate).
2. The amount of data being handled is larger than the comprehension of one human.
3. The desire to develop more complex products. These products cannot be developed at all without the use of automation applications.

The developers do not have ways to measure the complexity of a task, but they are trying to look at the effects of task complexity. For example, issues like errors, bugs, problems, rework, high maintenance cost, time to market are all related to the task complexity and are reviewed. In addition rough estimates of return on investment are made.

Sometimes a mock-up is used to prove that an idea will work, but it is not the first stage in the application. It is discarded when work on the application begins.

Project Definition

When a project has been selected, a development team develops the application. The team consists of five to twenty people, some from the CAD tool group, and one or two engineers sufficiently experienced in the domain being automated (not necessarily the best). If users of the application are also people other than the engineers (experts), they will be involved in the knowledge capture, but will not be on the team.

This team researches how to meet the project goals, and develops the design automation application. Many situations call for not creating a solution from scratch and implementing it as a whole; rather, the idea is to integrate the new applications into the current process as they become ready.

Knowledge Acquisition:
Sources of knowledge are the domain engineers, feedback on the application as it replaces the current system piece by piece, and any documentation that is relevant. A larger project is usually broken down into components so that one or two people work to learn what they need from the domain engineer(s) in many sessions.

Formal methods for requirements gathering are used, but there are none for understanding and defining the core design rules associated with a task. Mostly the knowledge is gathered in an ad hoc manner, and getting a complete view from the domain expert takes a long time.

Domain experts are used both as the main source of rules used in the system, and also in some cases, they provide a library to be called by the design automation application (the general programming experience of most employees makes this feasible). Occasionally, an expert is maintained to consult and provide guidance (beyond interviewing and knowledge extraction).

Knowledge Representation and Implementation
The set of design rules is verified through validation testing. The company employs very rigorous reviews to validate the application.

Project Deployment
This CAD application group supports, maintains and upgrades applications.
The developers evaluate success in terms of whether the application meets the customer's requirements (although whether this results in a tool that is truly of value to the company will not necessarily be known).

Notes on the success of development projects

1. Projects depend on complete data to be successful; it is likely that projects have failed because the of lack of data
2. Claims of capabilities that can be put into an application must be realistic from a technical point of view. Applications that are overly complicated will have long development times, and will be difficult to support. This may be described as failing because of a 'revolutionary' as opposed to 'evolutionary' approach.
3. It is important to know what the core needs are, and what 'needs' of a customer are extra, so that a solution to the main problem can be quickly found and reached. Other concerns can be met afterwards.

5. Company C

Team members

Two people were interviewed, from different groups within the company. They are designated subject A, and subject B. Subject A is an engineer involved in a project for reengineering the company's product development process. Subject B works in developing applications for CAX (computer aided design, engineering, manufacturing, etc.). This group works with mechanical and electrical design groups.

Subject A is part of a small group with members from product development, manufacturing, and software development backgrounds. Subject B works with a large group (more than twenty five people) who are primarily experts in computer science, artificial intelligence, and expert systems development.

Overview

Subject A is part of a group that has the goal of letting one engineer complete an entire engineering task for a new product. They have developed and launched six or seven applications that work with the company's CAD system and incorporate manufacturing rules and information, quality methods, and design checking, into the development of a part geometry. The applications are also able to incorporate changes to the design of manufacturing tooling based on changes in the development. Applications that have been deployed have yielded dramatic reductions in design time.

Subject B is working on numerous projects that are enhancements to existing CAD tools. An example application is a design advisor to provide design guideline information to the CAD system. These applications take one to three months to develop.

Most applications are developed in three or four months. More complex systems can take a year.

Identification and Selection of Tasks for Automation

Subject A explained that the program's current focus is to create applications that will bring greater support for its methods within the organization. Management buy in and acceptance are needed. Therefore, they are looking for highly visible tasks on the critical path. These tasks are generally the design of components. General criteria are:

- The component should be one that undergoes frequent design changes during the development process. For example: a component that undergoes many changes throughout and late in the program due to packaging constraints. Redesign that used to take 2 weeks is down to 2 hours.
- Components for selection should be challenging enough to make the point about the methodology. For example, after doing application for component of one geometry, the team looked for applications needed for different geometry.

They do not currently estimate the return on investment, because the primary goal currently is proof of the methodology through applications where improved quality is highly visible, such as with subsystems that are on the critical path.

Subject B explained that engineers from design and manufacturing come with requests for an application to carry out a task. These request reach the developers only after having been screened internally by the groups that send the requests

Project Definition
Subject A indicated that a method for defining application functionality and organizing development should be part of the team's approach to developing applications. Subject B said that the requests for applications come with complete, documented requirements analyses.

Knowledge Acquisition
Sources of design knowledge are experts, end users (who are not experts in the design process), and outside consultants to the industry.

Subject A said that their approach includes a technique for gathering information, delivering it, and keeping it current.

Subject B explained that they follow basic steps in capturing the knowledge. First, the process is documented. An emphasis is placed on the end user to ensure that the application being developed is one that the end user will need and like. Knowledge is gathered from experts through observation of the experts at work.

Knowledge Representation and Implementation
Subject A stated that the group utilizes a method for creating the application code. Subject B indicated that they have no methods for developing the software

Deployment
Subject B noted that software engineers still own the applications, getting the groups of users to use it depends on their acceptance of it. Subject A said that that group supports and maintains applications it develops in most cases. When the application is very big, and support would drain team resources, support responsibility is passed to technical support within the group that asked for the application.

To evaluate success, subject B said that they track application use by counting hits to a database. They view the success of the projects in terms of the obvious improvement the application can make to the process. Since these are not fully implemented such that people are required to use them, their full potential is not yet realized.

6. Company D

Team members

The interviewee is a senior engineer in a group that develops knowledge based engineering applications, not necessarily for design. The group is part of an overall group for designing and implementing computing systems to support engineering and manufacturing groups. The group members may be called "systems personnel".

Educational background of group members: software engineering background, most with an AI background, and some with strong engineering backgrounds. These are highly skilled and trained personnel who are rated high in their job class group.

Overview

Design Automation applications are primarily used to produce CAD data (curves, lines, surfaces, solids, 2D drawings, etc.). They are currently being developed to automate the design of parts, data management, manufacturing processes, and tooling for fabrication. Generally the applications are for automating the following design activities:

- detailed part design
- tooling design
- checking of engineering drawings for conformance with standards and data management requirements
- the design of dies
- flat pattern generation
- surface combination and offset

Current projects include automated design of numerous components and tools. With changes in design, the applications are still being updated, with increased capability driving the evolution.

Generally the first phase of a project, at which point a usable application can be deployed, takes 3 months. Continuing phases produce additional benefits and are pursued until the project is complete. Project completion in this sense depends on the return on investment associated with continued modification.

Identification and Selection of Design Tasks for Automation

The overall objective in using design automation is to produce products more efficiently. This includes reducing cost and time of development. More specifically, the interviewee's group identifies these goals:

- reduced design (and therefore part) variability
- the capture and use of 'best practice' design knowledge
- reduced design and information flow time
- reduced manufacturing flow time

- the ability to perform multiple design iterations rapidly
- the application of engineers to more creative activity by freeing them from routine, repetitive tasks
- the integration of knowledge in areas relevant to the design, into the design process
- a reduction of errors due to design
- the support and assurance of manufacturability in the design

Selection of a task for automation follows a 3 step process:

1. **Quality Report:** There is an ongoing activity in the engineering groups to improve quality. Teams are formed in response to perceived problems with a process or product. Execution of a report on the problems is a structured process (including training in leading and being a member of these teams) which results in a *process definition* which can then be used as a basis for automation. Without the definition provided by the report, there are difficulties in automating that are associated with ill defined processes.

   Projects have been selected without this quality initiative, although the group then attempts to get a team to review the task. Also, where a process is easily identifiable as fitting well with the automation technology, it may be automated without going through the quality report process.

2. **Opportunity Evaluation**: This is the initial evaluation phase, in which general business and technical case issues are investigated. The extent of evaluation is up to the group member responsible, but the evaluation is not intended to be a detailed study. In this phase, the current situation is described and documented, and its problems are defined. This is used to evaluate the opportunity for automation and for possible ways to incrementally tackle the problems. A general project proposal is then developed that includes an rough cost/benefit analysis, and an evaluation report is generated. A favorable reaction to the report (managers sign off on it) moves the project on to the next phase.

3. **Analysis Phase**: A more in depth review of the current process, the problem definition, and issues is conducted, and management deliverables are included (e.g., project plan, release plans, etc.). In this phase management deliverables such as a project plan and a release plan begin to appear, as the design task is reviewed and the project better defined. The shift from selection to development is not rigid, and occurs at this phase.

Overall, the selection process is structured, and the team reviewing a candidate design task looks at economic, organizational, and technical issues for making judgments.

Economic issues: Benefits which are noted and estimated include:

- Decreased Design Labor
- Decreased Design Flowtime
- Increased Design Consistency
- Support of last minute design changes
- Incorporation of 'best design practices' into all design

Organizational issues

- Purchasing of Hardware/Software
- Training
- Alteration of traditional design process
- Reassignment of engineer roles (e.g., an engineer may be required to work on design-intent rule capture instead of producing a deliverable design)
- Involvement of upstream and downstream process expertise in the design automation effort

Technical Issues

- Many of these were listed in the description of the evaluation and analysis reports, above.
- In addition, a project's similarity to previous work is noted. Attempts are also being made from a programming aspect, to look at metrics such as number of objects, attributes, their complexity, number or object interactions, etc. Also, how to divide the project into manageable phases is an important consideration.

The above phases are carried out primarily by the systems personnel, with assistance from engineering personnel. The evaluation phase is a cooperative effort, where a systems person handles description of the application, while operations and engineering people can provide the information on the current process, and on benefits of the application and its impact on the organization.

Often the analysis requires the construction of a proof of concept prototype. This prototype serves as a demonstration of the technology, an idea generator, and typically as a skeleton of the true application. It evolves into the production released system, and requires more design knowledge so that it may be the basis for knowledge capture for the application.

Project Definition

In the analysis phase described above, expert interviews together with test cases of experts carrying out the design task are used to drive requirements generation. Often the expert will also point the team to company standards or design procedures. The results of the quality report described above may also be used. In some cases, a process "focal" assists the process through the identification of specific experts, the gathering of support data (drawings, procedures, manuals, etc.), cost/benefit analysis, project scoping, etc.

Commercial software is used to document the project's requirements and to define roles and duties for members of the development team. Roles and duties during development include project manager, system manager, developer, end user, system quality assurance manager, configuration manager, and others.

As in the initial development effort, the application development is carried out primarily by the interviewee's group or by similar groups, although people from engineering are part of the development team. Currently support, modification and upgrade of the applications is handled by the computer systems personnel. There are a few cases where experts in the process being automated are developing the application. This may be changing, as will be described in the section on deployment issues.

The interviewee's group develops knowledge based applications using KBE tools such as ICAD and inference engine tools such as that from Aion Data Systems (ADS). ADS is used in conjunction with the STONErule product from Stone and Webster (Prescient Technologies).

The company is using a commercially available application development process methodology. This methodology promotes/dictates deliverables in traditional areas of software construction and project management. It is easily tailored to the individual project however. Conformance to the software development methodology as embodied by the software should enable the interviewee's group to achieve level 3 of the Capability Maturity Model as defined by the Software Engineering Institute (SEI/CMM).

As described, plans for the deployment of the application are part of the preliminary analysis planning.

Knowledge Acquisition and Modeling

Sources of knowledge are the experts in the process to be automated, design manuals, and standards. The quality report usually contains very explicit documentation of the expertise. It is preferable to use this as much as possible, as this expertise reflects the opinions of several experts, not just one.

For the application, the developer gathers knowledge from the process experts. This method is often applied by introducing a representative case and using it as a focus of discussion. In this sense, the interview resembles a protocol analysis, where the expert walks through the design but may be sidetracked to provide background information. For recording, usually a 2D drawing is used, and the developer will mark it with descriptions, notes and reminder flags for later. In the past, tape recordings were used at times, and the interview were recorded and transcribed as a requirements analysis. Some feel this method may be useful with the availability of the equipment. Using two interviewers is useful in the same way.

As mentioned above, there are cases where the developers are the experts. The way they gather knowledge of their own expertise is being studied currently, but is not organized at all.

As part of the interview process, the developers focus on their interaction with the application, not just their expertise in the domain. These issues would probably be discussed in the document of phase deliverables. User issues include:

- Attitude - poor user attitude would emphasize the need for better user education and a strong user interface design effort
- Knowledge of the process - the level of expertise of the users is important; experienced people will not be impressed with, nor will they use, an application constructed for novices.
- Background of users - users without a computer background will not adapt well to a user interface that expects computer interaction familiarity.
- Work environment - if the user uses CAD tools to perform his work, the application should be implemented within the framework of that CAD system, if possible.

The phased development approach means that there are limits to the amount of knowledge of the task needed for a given phase. The techniques described are used until the developer feels that a prototype can be written to illustrate some of the capabilities of the required and documented in the analysis document. The prototype/interview process is repeated until the first phase of the design automation application has been released to production.

Despite all of the specifics given by the subject, there are no standard methods or, for example, interview formats that are used by the developers.

Knowledge Representation and Implementation

While a design automation application is being developed, it undergoes alpha and beta testing by the developer and the users. There are quality assurance programs in place for testing software, and the developers rely on feedback from the users as well. The configuration manager, on the advice of the developer, has the power to move the application from the beta environment to production.

Deployment

Use of design automation applications is monitored via logs that keep track of how often the application was used, for which parts, how many iterations on that same part, number of errors encountered, etc. This is used to compare with estimates and make approximation of savings and improvements resulting from use of the application.

It is felt that design automation application development is accomplished smoothly, though they are working on moving to higher levels of the SEI/CMM. However applications are not meeting their potential because developers must still sell the proven technology to internal engineering organizations.

Without strategic direction toward design automation, it is difficult to put in place tracking mechanisms that enable the identification of savings and improvements that result from these applications. As a result, better metrics are needed for project sizing and estimating. Also, ways to support reuse of code would make the development process more efficient.

Failures and disappointments are primarily the result of a lack of customer/user buy-in to the use of the application. Sometimes the user interface is poor. Also, design/manufacture offload and unexpected process changes occur, and these reduce the ROI of the application.

7. Company E

Team members

The interviewee is a research scientist with a group whose work benefits the activities of much of the company. The group does not develop working design automation applications. Its role is to examine technologies related to design automation, such as languages, evaluation methods, knowledge representation schemes, geometric modeling, conflict resolution, rationale and assumption management, and system integration. This group is also exploring many different paths for future development, some of which are very abstract.

Educational Background of group members: Ph.D. and Masters levels in computer science, and engineering fields. Engineers who develop applications are engineers in their respective fields, who take training courses in using the development tools.

Overview

Development of the applications is carried out by multidisciplinary teams with follow up work carried out by central teams that examine and rewrite, and in some ways recast, past applications for more general use. A lot of effort is spent on developing an effective relationship with people who can provide the needed input to the project.

Design Automation applications are being developed to automate component design, product configuration, and tooling design. Individual projects are many and hard to enumerate. Some are duplicates of ones being developed in other divisions of the company.

The applications that are described as design automation applications are computer aids for generating, checking, or querying product data during the design process. Product data includes CAD geometry, layouts, schematics, and bills of materials. For example, design checkers are included, as are systems that provide advice on materials and part selection.

The development process described by the interviewee has changed recently; some of the methods in this case study are no longer in use. Currently there is a central knowledge based engineering organization that organizes and prioritizes tasks with a strategic intent in mind. Projects range in duration from a few months to a year, and have often affected the product development process in critical ways.

Identification and Selection of Design Tasks for Automation

The stated goals of utilizing knowledge based engineering software are to dramatically reduce design costs, and lower critical cycle and flow times.

Ideas for automation come from managers of engineering teams, R&D groups, project engineers, and project managers. There is no single, formal process for reviewing activity in search of candidate design tasks. The engineering groups who will use the application are always centrally involved in researching a project. Separately corporate engineering, information systems, and R&D groups are investigating how tasks may be automated, and may also be given requests for assessing a particular task.

Until recently the decision to automate was not made at a high level. Since applications can be developed by the engineers themselves, engineers in different disciplines may build them if given the permissions. For example a product engineer may build a design automation application for configuration, after making arrangements for necessary computer hardware and software. Or a group with more general expertise may build one after receiving a requirement order from an engineering group.

While the decision to automate is not necessarily formal, there are general considerations, and some metrics that are used. A group of managers and technical advisors usually makes decisions considering technical and business issues together. Sometimes a development group will make an analysis and present it to a product manager for approval.

Economic issues

- The cost of the current process, based on the number of designs and the cost per design
- The cost of developing the design automation application.

Technical considerations

- How well is the design task understood?
  - A. Can what is done be expressed in writing
  - B. Is the design task performed pretty much the same way each time (with a set of understandable variants)
  - C. Are many errors made in carrying out the current process?
- What design parameters provide input for this task? Can the variation in the task be described and characterized as changes in the design parameters?
- What outputs are required? What forms do the outputs take, e.g., solid geometry construction, process plans, cost estimates, etc.
- How long will it take to develop the application, and what set of products will benefit from the application?
- Are qualified people available?
- A comparison to other possible projects to determine priority is made

A prototype may be built to prove how a task may be automated. Alternatively, a prototype may be used to introduce a candidate task for assessment. Often a rough prototype, taking one to two weeks, is developed, with a detailed prototype being rare. In many cases this is a part of what will be the design automation application.

Project Definition

Some requirements are already known in the initial identification. Most detailed requirements are identified in the course of development by the multidisciplinary team supporting development. The team also ensures that the process is understood before automating. The intended users are always an important part of the development team. Either as developers themselves, or as a customer with frequent input to the development process. They directly specify functionality, usability, and interface issues. Usually not all of the desired functionality can be created in a timely manner or at an acceptable cost. In selecting the task for automation the team will have made compromises that were acceptable to the users, lead engineers, and managers. These compromises are part of the project definition

The applications are often implemented in stages, and partially complete design automation applications can be tested while further functionality is developed. Sometimes the design automation application may be complete, but other work must be done. For example, a CAD geometry translator may need to be upgraded to handle a new data type between CAD systems.

The development of the tool is carried out primarily by the engineers who will use it. The same possibilities exist for the support of the tool. The project itself might be run by a development team with the oversight of the user group, or by the user group itself. Maintenance of the design automation application will either be by the user-engineers, or by central teams described earlier.

ICAD (Concentra Corp.) and STONErule (Prescient Technologies) are among several tools used in developing applications.

As the users themselves develop the application (in most cases), they are acutely aware of how their work will change during and after development. However, a more general approach for redefining their role, and for identifying how the overall process can best be modified is not in place.

Knowledge Acquisition and Modeling

The multidisciplinary team that develops an application gathers the knowledge needed, supplementing it with knowledge of how to represent the task learned in training or from knowledge engineers. The process experts are a key source of expertise either as the application developers or as customers for an application being developed. If the developer is someone else, then the developer may bring additional expertise, such as a knowledge of CAD geometry or a familiarity with manufacturing computer systems.

Knowledge is gathered primarily through the multidisciplinary team, as opposed to interviews with individual experts. Care in creating the team ensures that the necessary disciplines are properly represented. The teams may use sample designs as a catalyst for review, working off the design to determine how decisions are made. The closeness of the team allows it to leverage incremental implementation of the application, so that there is no to gather all of the knowledge before beginning the representation.

Knowledge Representation and Implementation

As the developers are not software engineers, development methodologies are not in use. In many cases, the maintenance of the application is passed to a central group, and this group may try to force the application into a more standard form.

The use of development environments like ICAD minimizes the effects of poor style at the level of the application users. It does make it difficult to reuse parts of previous projects.

There are no validation procedures that require a comparison of the application's results with other methods. The nature of most of these applications is that resulting designs are evaluated on a case by case basis, as a manual design would be. If errors are found in the way the design was developed, the application is then fixed.

Deployment

Success of an application is measured in several ways:

- By whether the application was completed so that the part it designed met product schedule
- By whether it was used by the users on the product program
- By how much the design cost to produce with the design automation application versus without it, based on accounting cost data or based on estimates of hours, made by lead engineers familiar with the process.

The company considers itself successful in the development of these applications. It is believed that mistakes in development resulting from bad selection, interdisciplinary conflicts, and misunderstanding of the process are avoided by the heavy up-front effort that is placed in making the multidisciplinary teams. Developers feel that a weakness in the current development process is that there is little reuse of what has been done before. From a technical viewpoint, the next step for improving automation work is the development of libraries of reusable components. Explicit, perhaps formal descriptions of design processes and standard languages of design steps. Finally, better integration of design automation across multiple product platforms will save a lot of duplicated effort.

The company has acted on the idea that a more centralized development process, with close support and teaming with product divisions would be beneficial.

8. Company F

Team members

The interviewee is a computer scientist in a group for knowledge based applications. This group is part of an overall computer technology group that is devoted to computational and software development tasks.

Educational background of group members: software engineering (interviewee), mechanical engineering, metallurgy, physics, mathematics. The common denominator is a wide background and experience in solving engineering problems.

Overview

This group is part of the company's research and development center and does work for other research and development groups and for internal customers around the world. In most cases, contracts for applications are initiated by this group in cooperation with technical contacts at a prospective business unit.

Design automation applications are being developed for using computational geometry in the transformation/creation of geometry, and in the analysis of geometry (e.g., FEA). These applications are for two main purposes:

- To automate a design task for a large percentage of the cases within the domain of interest.
- To enhance the designer's ability to perform a design task by providing assistance, usually in the form of analysis.

Design automation application development has included integrating existing software tools/analysis packages into a more accessible group of tools, as well as developing complete automated design packages.

Examples of work carried out in the past several years include:

1. Tooling design for various kinds of metal forming tasks
2. More "designer accessible" front ends to engineering analysis tools
3. Intelligent cad systems for specific kinds of solid modeling problems
4. Highly flexible and intelligent data analysis tools for process modeling and discrete manufacturing trend analysis

Projects are years (and several prototype generations) in length.

Identification and Selection of Design Tasks for Automation

Currently the goals of automation are improved quality and consistency of designs, and fewer problems once the tools designed with this assistance are made and tested in

manufacturing trials. The length of the projects (more than a year) has meant that speeding the production of a particular design is not the goal, although it does happen sometimes.

There is no formal process for identifying tasks that would benefit from automation. Ideas come from within this group, and occasionally from an internal customer where a need has been identified. There is no company wide initiative to work on knowledge-based automation, and internal businesses are not involved in any projects.

Automation concepts are being explored and applications are being developed by this research and development oriented group. The ideas are formalized into proposals by individuals in the group, with projects usually being of a scale where one person works on it.

The development group is usually not linked to the design activities of any particular business, so ideas are introduced for applications that are not based on specific design tasks. These ideas are explored, and prototypes are built, before a real need has been identified. Then the developer must try to get the customer interested in the project, as a continued source of funding.

Sometimes an internal customer approaches with a request for the group to solve a problem. In this case, the customer will use a business case to determine whether to accept the proposal for project funding. It is unclear how the customers determine project worth. If no customer is immediately involved, a project may proceed for some time under the funding of general research and development.

There are no formal methods for reviewing tasks and determining the value of automation. Recently the direction has been to require greater oversight, which means that a gated system of checks on the process will put in place. The developers will be required to justify continued work at various stages; this should lead to a more formal approach to task selection and development of applications.

Prototypes are used to demonstrate the capabilities to a customer both in initial demonstration of an idea, and in reviews of progress. These prototypes are built with the goal of evolution into a production system. All members of this group function as project leaders and supporters of their own projects.

Project Definition

The goal of these applications is to augment the existing system. Change happens slowly, and is driven by other forces such as marketplace and manufacturing problems. The development environment means that the requirements are not spelled out until a

particular business unit has expressed specific interest. The application's development will then incorporate those needs.

There is very little formal project planning, although this is changing, as mentioned above. Once a customer is interested, they will impose some constraints on the process.

The entire process is carried out by these group members. They capture design knowledge, develop the software, and maintain the software. As they are completely separate from a specific business unit, they are not involved in planning the future of the design environment. It appears that some internal customers do not see the applications as catalysts for dramatic changes, and do not plan for it.

Knowledge Acquisition and Modeling

Sources of knowledge include the experts in the process, and in the case of complex computations, expertise in geometric modeling and computation is needed.

The attitude of these developers is that a deep understanding of the design task is needed in order to automate it. This is probably because of their separation from the engineers carrying out the tasks. As a result of this approach and constraints of computing structures, the developer may find himself solving the problem differently than the designer, as will be explained.

The developers talk to the customer (if there is one, or at a later point, when taking a concept and applying it) to learn the process and why the design task is carried out that way. Note that these 'experts' are generally people with minimal analytical training, but through experience and some analytical skill they can use visual intuition to design, based on what they see on a screen. The application must work computationally, and the appropriate methods have to be researched separately. In determining how they will model the domain, the developers test their knowledge by attempting to carry out the design task, even though they will not use the same method as the designers would (since these are people who can apply more analysis even without an automation application.).

Although there are no formal techniques for capturing knowledge, the developers begin by accumulating example problems and their solutions for the purpose of developing a taxonomy and the terminology that describes the set of problems. Then, the developer tries to work out algorithms and heuristics that will bring about the same solution as the designer, incorporating here their own analytical tools (in particular where the task is algorithmic in nature, not rule based). This method will create solutions that can be shown to designers for feedback on their quality. This iterative process is coupled with the formalizing of the knowledge and the development of suitable computing structures, along with an interface that the designers can/will use.

At this point, the designers can be used to test the tool with new cases, looking for where it fails, and helping to include solutions to this case in the tool's capabilities. These exceptions are usually included as exceptions to the rules. Direct discussion with the designers is also used to maintain the connection with the end user, and to help ensure that the application and the user interface are acceptable.

A major drawback with being so separated from the design environment is that the process constraints may change, and the designers accordingly change the design process. This can render the application obsolete or even remove the need for its assistance entirely, making the project a waste.

Knowledge Representation and Implementation

It does not appear that standard methods are used for developing these applications. Applications are validated by using them for test cases. The usability of the application is tested during development, but there is no system for ensuring that it works in a desirable way.

Deployment

Currently success is measured only in terms of customer satisfaction, i.e., the satisfaction of the business unit. Applications that have not been picked up by an internal customer are not measured. It is not known what makes an application satisfactory to the customer, except that it meets the requirements and was developed on schedule. This is changing as more metrics for determining project worth are put in place, and the success will depend on whether the project lived up to the expectation. Recently there have been more formal techniques attempted for measuring a projects expected payback.

This group has deliberately allowed its developers to perform with different approaches and methods to the development process, rather than trying to use a standard consistent procedure. This is because of a desire to experiment with different techniques.

In general, 'big' projects have historically failed, while 'little' ones have been picked up by the business units, possibly because the larger ones have become too defined and problem specific to be of use to customers. Little ones are either close enough to the start or small enough overall to be controlled by the business unit. Large projects are more difficult to keep focused and for prospective customers to judge their payback.