2004

Rationale for user oriented design technique selection

Susan M. Saeger

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
Rationale
for
User Oriented Design Technique Selection

By

Susan M. Saeger

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Technology

Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences
May 20, 2004
Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences

Master of Science in Information Technology

Thesis Approval Form

Student Name: Susan M. Saeger

Project Title: Rationale for User Oriented Design Technique Selection

Thesis Committee

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evelyn Rozanski</td>
<td>Evelyn Rozanski</td>
<td>5-28-04</td>
</tr>
<tr>
<td>Chair</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anne Haake PhD</td>
<td>Anne Haake</td>
<td>5-28-04</td>
</tr>
<tr>
<td>Committee Member</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deborah Coleman</td>
<td>Deborah Coleman</td>
<td>5-28-04</td>
</tr>
<tr>
<td>Committee Member</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Thesis Reproduction Permission Form

Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences

Master of Science in Information Technology

Rationale
for
User Oriented Design Technique Selection

I, Susan Saeger, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction must not be for commercial use or profit.

Date: June 18, 2004  Signature of Author: Susan M. Saeger
Acknowledgments

I would like to extended a thank you to my wonderful parents, Theresa and Joe Wanchissen, who have always encouraged me to “push the pencil harder” Also, thank you to brothers, Joe and Steve for listening at every family event about my educational status.

I would also like to thank Ed and Jackie Saeger not only for believing in me, but for allowing me to visit them when I needed to recharge and reestablish my perspective.

I am grateful for all of my friends and my husband’s friends who have become my friends. You have all been so understanding of my schedule when school had to come first. At the same time you all knew when I needed to take a break and live a little. You are the people who helped to keep me balanced. You are all a little responsible for the initial idea of this paper because it crystallized as Jeff and I were training for the half marathon you all organized in 2003.

My co-workers and managers also deserve a thank you for listening and encouraging me not only during the creation of this paper, but for the past 6 years. All of you who have been there from the start know the challenges I faced and your support has been greatly appreciated.

Thank you to all of my professors who have contributed to my educational experience. Most of all I would like to thank my committee members and especially Evelyn Rozanski PhD. She was the person who established and fostered my excitement about the topic of this paper. She is not only a wonderful professor, but also a mentor and a friend. I am going to miss our bi-monthly meetings that included topics far beyond what is presented in this paper.

Most of all, a very special thank goes to my husband who has been there for me through it all from the very beginning. Thank you for always believing in me and helping to keep my life appropriately balanced. Thank you for knowing when I needed to go for a run or when I simply needed a hug and a bowl of ice cream. You have truly defined what it means to be supportive, encouraging and to loving.
# Table of Contents

1. Introduction ......................................................................................................................... 9
2. Statement of the problem ........................................................................................................ 12
3. Hypothesis ............................................................................................................................. 13
4. Definitions ............................................................................................................................ 13
5. Background ........................................................................................................................... 14
6. Participatory Design (PD) ..................................................................................................... 15
   6.1 History ............................................................................................................................ 15
   6.2 Definition ....................................................................................................................... 16
   6.3 Technique and process summary ..................................................................................... 17
   6.4 Project team type ............................................................................................................ 20
   6.5 Application domain ........................................................................................................ 20
   6.6 Integration feasibility ...................................................................................................... 21
   6.7 Advantages .................................................................................................................... 21
   6.8 Expectations ................................................................................................................... 22
7. Scenario-Based Design (SBD) ............................................................................................. 23
   7.1 History ............................................................................................................................ 23
   7.2 Definition ....................................................................................................................... 24
   7.3 Technique and process summary ..................................................................................... 24
   7.4 Project team type ............................................................................................................ 26
   7.5 Application domain ........................................................................................................ 26
   7.6 Integration feasibility ...................................................................................................... 26
   7.7 Advantages .................................................................................................................... 26
   7.8 Expectations ................................................................................................................... 26
8. Contextual Design (CD) ........................................................................................................ 26
   8.1 History ............................................................................................................................ 26
   8.2 Definition ....................................................................................................................... 26
   8.3 Technique and process summary ..................................................................................... 26
   8.4 Project team type ............................................................................................................ 26
   8.5 Application domain ........................................................................................................ 26
   8.6 Integration feasibility ...................................................................................................... 26
   8.7 Advantages .................................................................................................................... 26
1. Introduction

In today’s competitive environment, companies cannot afford to produce unusable products. This is particularly true for companies that produce software, as their sole revenue generator. Consumers choose software products that make tasks easier. This means that usability and usefulness of a software product are no longer requirements that are “nice to have”, but must be core requirements of software applications. Achieving these requirements is a complex task that requires intense analysis and commitment from several areas of an organization. Part of the reason this task is so difficult is because of the wide range of perspectives that influence the creation of software products.

Typically, the creation of a product involves stakeholders from all areas of an organization including: Marketing, Software Engineers, Hardware Engineers, Quality Assurance Specialists, Implementation Specialists, Project Management, and Directors. When people with different perspectives come together to move a product from a concept to reality, it is inevitable that all parties must compromise to achieve success. The first step to achieving success is to understand the overall organizational goals; the second step is understanding the consumer or end user of the product.

As achieving a clear organizational goal is not the focus here, the assumption is that the organization already has a clearly identified goal. Rather, the focus is to understand how to select the appropriate user oriented design method and techniques to achieve the organization’s strategic objective and the specific project’s goals.

User oriented design methods are multidisciplinary because they consider the psychology, ergonomics, sociology, engineering, and visual design factors affecting product creation. A user oriented design perspective focuses on the end user and the interaction between end user and product. The goal of user oriented design methods is to add value by incorporating the human aspect into product design. Because of this human nature perspective, user oriented design methods can be applied to many different types of products and systems.

The variety of user oriented design methods available in the industry makes it simple to assume that selecting the appropriate method is as easy as selecting a flavor of ice cream. However, this is not the case. A study completed by Vredenburg and Butler in 1996 found that many of the user oriented design methods are not practical for a
variety of reasons. The questions they asked included:

- “Which user-centered design (UCD) methods are most widely used and why?
- What are the benefits and weaknesses of each method in the eyes of practitioners?
- What are the organizational impacts of UCD and what measures are in place to assess progress?” (Vredenburg, Mao, Smith, Carvey 2002 p. 471)

(Note: the use of the term user-centered design by the authors is equivalent to the term user oriented design methods that are referred to in this paper.)

This study revealed that UCD is not well-established in the industry.

Recently Rosenbaum completed another study.

“Rosenbaum et al. (2000) surveyed 134 CHI professionals with a focus on the contribution of organizational approaches and UCD methods to strategic usability. It was found that major obstacles to creating greater strategic impact include resource constraints, which was mentioned by 28.6% of the respondents. Resistance to user-centered design or usability, lack of knowledge about usability were also stated as obstacles. However, partnering with marketing was identified as a very effective approach” (Vredenburg, Mao, Smith, Carvey 2002 p. 472) to UCD.

Hudson conducted an e-mail survey in 2000 and 102 usability practitioners revealed that the most commonly used techniques are: informal usability testing, user analysis, user profiling, evaluating existing systems, low-fidelity prototyping, heuristic evaluation, task identification, navigation design, and scenario-based design (Vredenburg, Mao, Smith, Carvey 2002). A similar 10-question web survey, conducted in 2001 by Gunther, Janis, and Butler that involved 100 usability practitioners found that 39% of the respondents believed usability testing to be the most accurate in gauging success (Vredenburg, Mao, Smith, Carvey 2002). The statistics indicate that the selection and implementation of the appropriate user oriented design method is not simple and that there is a gap in knowledge about user oriented design methods.

Due to the current economic stress organizations are placing finer attention on cost benefit analysis and unfortunately the cost benefits of user oriented design methods are not realized. A survey conducted by Vredenburg, Mao, Smith, and Carey focused on overall organization impact of UCD and measures of UCD success, but had disappointing
results (Vredenburg, Mao, Smith, Carvey 2002). The results found that 32% of respondents were not sure if UCD methods had helped save product development costs; only 24% believed that UCD methods actually saved product development costs and; 44% thought that UCD increased the product development cost. Similar results were found when respondents were asked about the product development time. Vredenburg, Mao, Smith, and Carey argue that the results are inconclusive because the respondents may not have looked at the “big picture including service cost and redesign” (Vredenburg, Mao, Smith, Carvey 2002, p. 474). Nevertheless, the Vredenburg, Mao, Smith, and Carey study was consistent with Hudson’s informal e-mail survey taken in 2000. Both studies indicate that usability testing, prototyping, and heuristic evaluations are the most widely used low cost methods.

Vredenburg, Mao, Smith Carey concluded that:

- “UCD method adoption is quite uneven across different organizations.
- UCD staff in many organizations, 41% of [the] sample, is centralized, and only 15% of the organizations have completely decentralized UCD staff.
- The average spending on UCD constitutes about 19% of the total project budget.
- There is a lack of measurement of UCD effectiveness and any common evaluation criteria across the industry.
- A multidisciplinary approach to UCD appears to be closely related to perceived UCD effectiveness, although practitioners were not always clear about what constituted multidisciplinary.
- UCD was perceived to have higher impact when there were two or more UCD specialists on the project team compared with only one.
- Cost-benefit tradeoffs play a major role in the adoption of UCD methods” (Vredenburg, Mao, Smith, Carvey 2002, p. 477-78).

Productively implementing user oriented design methods requires awareness of the organizational structure. Vredenburg, Mao, Smith, and Carey suggested that the organizations used in the studies may not have properly aligned its staff (Vredenburg, Mao, Smith, Carvey 2002). “Respondents were asked to characterize the organization of UCD staff. In 41% of the companies, UCD staff were centralized in an organizational unit, 15% decentralized, 34% mixed, and 10% unclear” (Vredenburg, Mao, Smith,
Carvey p. 474). Understanding the methods and techniques provides insight about how to appropriately align staff members.

The information provided in this paper will begin to close the gap in the understanding of the variety of methods and their techniques that provide an organization the tools to move towards a user-centric design approach. The intent is to provide a comprehensive comparison of user oriented design methods, and guidelines for method and technique selections. It provides the reasoning behind the selection of techniques rather than random method selection. The focus of this paper will be on selecting a user oriented design method for software application development. By providing a central location for the information required when selecting user oriented design methods and techniques, this paper will close the gap that currently exists in the user oriented design industry.

2. Statement of the problem

User oriented design’s broad nature has provided the opportunity for practitioners to combine techniques in a unique fashion thereby creating several similar, yet different approaches to user-centric software product development. As practitioners have successfully combined techniques that have created perceived successful software application design, the practitioners have classified the selected techniques with methodology names. The practitioners that have moved into consulting roles align themselves with the method that created their initial success and then specialize in that method. The pervasive user oriented design methods that consultants have aligned themselves with include the following:

1. Participatory Design (PD)
2. Scenario Based Design (SBD)
3. Contextual Design (CD)
4. User Centered Design (UCD)
5. Performance Centered Design (PCD)
6. Learning Centered Design (LCD)
7. Usability Engineering (UE)

Since consultants align themselves with a method it is problematic for an organization interested in introducing user oriented design to select a method that is generic enough to be appropriate for more then one type of project. Selecting a user oriented design approach should be based on the method’s capacity to become a
sustaining process towards user-oriented design in the organization.

Speculation can be made that the selection of a user oriented design approach within an organization happens in conjunction with an immediate need of a high profile project. Further speculation can be made that for organizations new to a user oriented design approach look to consultants for support. Another assumption is that the initial selection of a user oriented design method only includes a review of the prominent consultant and the methods they offer. This type of selection may resolve the immediate understanding of the project objectives, but may not result in a sustaining user oriented design approach.

3. Hypothesis
If knowledge about proper user oriented design technique selection is available then an organization will be empowered to customize a user oriented design approach for successful integration into its unique software development project requirements resulting in a sustainable user oriented design approach. A sustainable user oriented design approach will have a positive strategic effect for the organization.

The result of the analysis contained in this paper is a set of guidelines for selecting a user oriented design technique. A fictional case study is used to illustrate the application of the guidelines. Included in the case study is an analysis of the methods and their techniques; whereby, the techniques have been logically grouped by the three major stages of a software application project those being: requirements, design, and testing. In addition, the deliverables provide by each method are discussed.

4. Definitions
Project types, cross-functional teams, and traditional software development are terms used throughout this discussion that requires a definition to ground the bases of the analysis.

The general reference to project types is based on the project objective and includes the following list (Olson 2004):

- New product
- Reengineering project
- Maintenance to an existing product
- Redesigning an existing product
- Integrating independent systems
- Redesign a legacy system
· Transition to a new platform

Also, throughout the discussion reference is made to cross-functional teams. Cross-functional teams are composed of people from various functional areas whose competencies are essential for achieving a successful outcome. These teams include employees from all levels of the organization and may include members from outside the organization including: key customers and consultants.

Lastly, Seffah and Metzker (2004) define the term traditional software development as technology and developer driven with the primary focus on functional requirements with little regard for end users workflow or end users environments.

5. Background

At first glance, the different user oriented design methodologies seem to share a common theme: keeping the user in mind when designing and creating software applications. However, not all methods are equal. The solution to the problem may not be as simple as the selection of a method, but rather the selection of the proper techniques from the different methods. Selecting the right method requires analysis of the techniques within the method. It requires consideration of each technique’s ability to facilitate the current project goal alongside the strategic direction of the organization.

This discussion of user oriented design methodologies follows the time line of the methods emerging into the industry. The discussion begins with the Participatory Design method. Some of the method’s names may have been coined earlier then the development of the actual method and may have roots that go further back in history however, this was not factored into the timeline. The chart in Figure 1 represents the time line of the user oriented design methods. Notice several of the user oriented design methods emerged in 1989. The order of the methods emerging in 1989 is based on the analysis of the references to techniques within a previously defined method. The analysis reveals that each user oriented design method borrows concepts and techniques from the previous methods and sometimes renames the techniques. Therefore, some of the overlapping techniques are merely unique by the referenced name. The chart in Appendix A provides a comparison of the methods and their techniques.
<table>
<thead>
<tr>
<th>User oriented design method (UOD)</th>
<th>Approximated initial availability</th>
<th>Person who originated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participatory design (PD)</td>
<td>1960</td>
<td>Christina Floyd, Kristen Nygaard, Olav Terje Bergo</td>
</tr>
<tr>
<td>Scenario based design (SBD)</td>
<td>1980</td>
<td>Ivar Jacobs, John Carroll, Mary Beth Rosson</td>
</tr>
<tr>
<td>Contextual design (CD)</td>
<td>1988</td>
<td>Hugh Beyer, Karen Holtzblatt</td>
</tr>
<tr>
<td>User centered design (UCD)</td>
<td>1989</td>
<td>Don Norman, Dr. Steven Draper</td>
</tr>
<tr>
<td>Performance centered design (PCD)</td>
<td>1989</td>
<td>Gloria Gery, Barry Raybould</td>
</tr>
<tr>
<td>Learning centered design (LCD)</td>
<td>1989</td>
<td>Elliot Soloway</td>
</tr>
<tr>
<td>Usability Engineering (UE)</td>
<td>1993</td>
<td>Jacob Nielson, Deborah Mayhew</td>
</tr>
</tbody>
</table>

Figure 1 – UOD Time Line

6. Participatory Design (PD)

6.1 History

Participatory Design (PD) began in Scandinavia around 1960 from collaboration between the Norwegian Employers’ Federation (NAF) and the Norwegian Federation of Trade Unions. One of the outcomes of this collaboration effort was “a revised Worker Protection and Working Environment Act [AML, 77; Sørensen, 92]”(Levinger 1998). This act required that end users participate in the design and be sufficiently trained to use new systems. Three major projects are credited for creating the foundation of PD. They are:

- The Norwegian Iron and Metal Workers’ Union (NJMF) conducted from 1970 to 1973. Their objective was to include workers in new technology plans.
- The Swedish DEMOS project (DEMOkratiske Styringssystemer) was conducted from 1975 to 1979. This project confirmed that the inequalities between workers and management were particularly unbalanced when introducing new technology. The outcome of this project was an institutionalized conflict management process (Levinger 1998).
- The Danish DUE project (Demokrati, Udvikling og Edb) was conducted from 1977 to 1980. The objective of this project was to encourage union membership so that these groups would have greater
influence on the use of computer systems. The project contributed to the initiation of a professional curriculum and research program in systems development (Levinger 1998).

The common theme of these three projects was to democratize the unity of labor and management when introducing new technologies. It is clear to see that “PD began in an explicitly political context, as part of the Scandinavian workplace democracy movement” (Muller 2002 p. 1). The outcome of these three projects was the bases for PD which is also referred to as the Collective Resource Approach.

The more recent developments in PD resulted from the first PD conference that took place in 1990 in Seattle, Washington. The participants included “prominent leaders from Scandinavia who had pioneered participatory approaches to computer systems development in the 70’s and 80’s.” After this conference future conferences were held every two years to further develop and define PD (Levinger 1998).

There is no strong evidence that one person is responsible for the development of the PD approach. Christine Floyd is the main European figure given credit for bringing PD out of Scandinavia (Bouvin 2004). However, the roots of the Collective Resource Approach are generally associated with the efforts in Norway by Kristen Nygaard and Olav Terje Bergo in the early 1970s (Levinger 1998).

6.2 Definition
Defining PD is not easy because, similar to other user oriented design methods, practitioners vary in their perspectives, backgrounds, and areas of concern. The more recent PD definitions are as follows:

- “Participatory Design (PD) is a set of diverse ways of thinking, planning, and acting through which people make their work, technologies, and social institutions more responsive to human needs” (Trigg and Clement 2000).
- Participatory Design (PD) is “an approach to the assessment, design, and development of technological and organizational systems that places a premium on the active involvement of workplace practitioners (usually potential or current users of the system) in design and decision-making processes” (Trigg and Clement 2000).
- “Participatory design (PD) is a set of theories, practices, and studies related to end-users as full participants in activities leading to software and hardware computer products and computer-based activities” (Muller 2002 p. 1).
• PD is not necessarily a “unified ideology or methodology but is a technique that can be embraced. It is an approach towards computer system design in which the people destined to use the system play a critical role in designing it” (Schuler and Namioka 1993 p. xi).

Although there is no single definition for PD, the common theme among practitioners who use PD is that the end user must participate throughout the entire product development process. This includes participation in preliminary discussions between stakeholders and managers (Trigg and Clement 2000). PD practitioners understand that the organizational culture and the setting in which work is done impacts design, and therefore, PD emphasizes the importance of spending time with end users who are doing the work in their own environment. PD practitioners also have learned that the end users are the prime source of innovation and can greatly contribute to generating design ideas. In addition, PD practitioners look to resolve problems that extend beyond technology and often include the process and arrangement of people.

6.3 Technique and process summary

According to Bødker, Brønbaek and Kyng, PD is a perspective and a technique not necessarily a design method (Schuler and Namioka 1993 p. 158). Schuler further explains PD as a perspective and not necessarily a blue print for a design process. The perspective of PD is to achieve the development of a usable and stress-free software design. The use of appropriate instructions along with the availability of the appropriate functions to support the workflow are two elements that contribute to the stress-free nature of a software design (Schuler and Namioka 1993 p. 6-7). PD provides the proper techniques to achieve this goal.

Although PD does not have a specific process, the techniques lend themselves to the analysis and design phase of the project lifecycle. During the analysis phase design team members, learn about the work through workplace visits where the end users participate in interviews or work demonstrations. Team members meet after the interviews and demonstrations to compile what they have learned. PD uses the scenario, contextual inquiry, stories and prototype techniques all of which have been borrowed by other user oriented design methods.

The techniques that make PD unique are workshops and organizational games. PD uses workshops and organizational games to bridge the analysis phase to the design
phase. PD specifically uses a workshop called the future workshop. The future workshop technique was created by Robert Jung and Norbert Miller (1987) to include citizens in the decision making process in town planning and other political venues. This technique requires that the participants write short statements of their ideas on paper and then place them on a wall for the entire team to see. The outcome of the future workshop resembles an affinity diagram. The future workshop then goes through three phases, which are; critique phase, fantasy phase, and implementation phase. The purpose of the critique phase is to define specific issues about current work practices. The team then moves into the fantasy phase. The purpose of the fantasy phase is to define an ideal future model. The participants generate ‘what if’ statements that would solve current issues. The last phase is the implementation phase. The implementation phase aligns the ideas generated in the fantasy phase with reality. During this phase, the team begins to design a realistic solution. The use of the term implementation here is not the movement of the software application to the end users, but rather the implementation of ideas into a design.

The technique of ‘design-by-playing games’ also bridges the analysis work into design work. Ehn, Sjögren, Bødker, Grønbaek and Kyng developed the idea of ‘design-by-playing games’ because of the ability of these techniques to influence participatory design methods (Schuler and Namioka 1993 p. 168). The games support the creation of alternative work organizations by enacting the current routine and then discussing problems. It includes the discussion about the problems that a new alternative may create.

Using games creates a less emotional environment for discussion. Games keep the work entertaining and at the same time, the team focuses on the workflow. These techniques allow for members unfamiliar with the design discipline an easy way to contribute. Muller explains that the games remove “the conventional authority of the software professionals [and replaces it] with a shared interpretation based on contributions from multiple disciplines and perspectives” (Muller 2002 p. 18). This type of discussion allows the analysis and design process to move at a quicker pace and produces better end products because people listen to each other’s needs and the reasoning behind those needs. The game techniques are only one of the many PD
techniques that create a safe and productive atmosphere for people from different perspectives to discuss conflicting opinions. The chart in Figure 4 summarizes these games.

<table>
<thead>
<tr>
<th>Game Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organizational Game</td>
<td>A charades-like card game that facilitates the discussion of current workflow problems and to achieve a suitable solution.</td>
</tr>
<tr>
<td>Specification Game</td>
<td>A scenario-based game based on a set of 'situation cards', each of which describes a workplace situation. The design team takes turns drawing a card and leading the discussion of the workplace situation described on the card.</td>
</tr>
<tr>
<td>Layout Kit</td>
<td>A game of floor-plans and equipment layout. This illustrates the workers’ perspective of how the shop floor should be redesigned.</td>
</tr>
<tr>
<td>Desktop Publishing Game</td>
<td>A story-board game that illustrates and annotates on poster size paper the components of work.</td>
</tr>
</tbody>
</table>

**Figure 4 Design Game Summary** (Muller 2002 p. 17)

Stories are another technique PD embraces. Stories, just like workshops and game techniques, bring various perspectives together during the analysis phase. “Stories in participatory work may function in at least three ways: triggers for conversation, analysis, and feedback” (Muller 2002 p. 11). If an end user tells the story, it provides the team with background information to understand what functions and features the product must provide. Alternatively, if the story is told by a member of the design team, it is used to present their concept of what a designed product will do, how it will be used, and what changes will occur as a result.

Drama is another technique that supports the analysis phase in the PD method. “Drama provides another way to tell stories – in the form of theatre or of video” (Muller 2002 p. 14). The drama may represent a current situation or a future situation. However, it is sometimes difficult to define whether a given drama depicts the present or the future because the two are often mixed.

Once the team has done the analysis, they can move into the design phase with the use of prototypes. Prototypes are the most commonly shared technique across all user oriented design methodologies. However, Bødker and Grønbaek (1991) introduced the idea of cooperative prototyping. Cooperative prototyping is different from traditional
prototyping in that traditional prototyping approaches mainly take the perspective of the analyst or designers, who conduct investigations in the end user’s environment, develop prototypes on their own, and then share the results with the software engineers and eventually with the end users (Bødker, Grønbaek and Kyng p. 170). Cooperative prototyping means that the end users actually create the prototype along side the other team members.

6.4 Project team type
PD requires more than a cross-functional team. PD has been successful where unions are popular and when ‘Workers’ Bill of Rights’ have been established. The hierarchical structure that the majority of U.S. companies use is not conducive for PD. PD would require major restructuring for most U.S. companies. U.S. software development companies would need to integrate software engineers with the rest of the organization, and allow software engineers to interact directly with the end users.

6.5 Application domain
The PD method was primarily designed to fine tune existing systems. PD is successful on projects with well-defined scopes and unlimited accessibility to the end users. PD as a method does not work well for general use software creation or in situations where the end user group is large and diverse. Grudin and Pruitt explain that “cooperative design techniques can be effective in in-house or customer development contexts but are less effective in commercial product or package software development” (Grudin and Pruitt 2002 p. 1). The problem with PD for mass-market software development products is that it is not reasonable to have end user representatives for each type of user group participate in the analysis, design and evaluations. The time lines do not allow for this type of participation and the cost would not provide enough benefit. This is not to say that PD as a technique is worthless, just as an end-to-end method it is not reasonable for most types of software product development.

In addition, the ‘Workers’ Bill of Rights’, that are established by unions may contribute to the success of PD. This ‘Workers’ Bill of Rights’ sets the guidelines and keeps the designers and software engineers accountable to the end users. It is questionable if PD would be successful in an environment were such agreements were not already in place.
6.6 Integration feasibility

PD is an approach that requires a cultural change within an organization which to date has prevented this method from seamlessly integrating into the software development industry. This type of change requires changing an organization's culture. Grudin, a Usability Engineer at Microsoft, explains that, “participatory design is more difficult to accomplish outside of the Scandinavian countries because of differences in legislative environment, workplace unionization, and scale and fragmentation of software development organizational models” (Muller 2002 p. 229). Implementing a true PD approach would require an organization to allow end users to fully participate on project teams and be treated as equal team members and not just as peripheral team members who are involved in requirements gathering and then are not spoken to unless compromises to the requirements are required. Changing the way an organization thinks about end user participation does not happen overnight. The traditional one-directional approach to software product development would need to become a two-way discussion were the analysts, designers, software engineers and end users all participate in the discussion of the solution (Muller 2002 p. 6). This type of discussion requires IT to remove the walls and truly listen to the end users throughout the project lifecycle. These means that unless software development organizations are willing to modify the level of priority for end user involvement and are willing to truly listen to the end users needs the best the PD method can hope for is to be used as a technique and to continue to be integrated with other user oriented design methods.

6.7 Advantages

PD goes beyond other methods to include end users as a full-fledged team member. It strives for designers and end users to have a practical understanding of each other. Practical understanding differs from prepositional knowledge. Practical understanding “is understanding that comes from the practical experience of doing something and the recall to mind of earlier experiences” (Ehn 1993). Prepositional knowledge is being able to use definitions to explain something that has never actually been experienced (Stanford University). PD encourages design by doing thus minimizing the language misinterpretation which is common in traditional requirements gathering approaches.
Another advantage of PD is that it encourages decentralized decision making because end users participate in the entire design process. End users are not only observed and interviewed, but actually participate in the creation of the design.

The goal of PD is to understand organizational change that will occur and may include restructuring the end users workflow when introducing new or different technology in organizational structures (Grudin and Pruitt 2002 p. 99). PD minimizes the number of surprises that the new technology may create in the workflow. This is because PD encourages a mix of nonlinguistic and linguistic design artifacts. Using nonlinguistic techniques such as mockups and prototypes helps team members see a picture and minimizes the amount of interpretation. Using linguistic tools such as scenarios also allows for all team members to participate in and understand the end users process prior to designing. PD considers more than the end users in a silo, but rather the end users within the organizational structure.

In addition, PD encourages and supports early product adoption from the end user community. There is evidence that an increase in end user involvement through participation in design activities can lead to increased willingness to use the product when it is implemented (Muller 2002). Thus, PD can be valuable not only in shaping technology to the end user’s needs, but also in shaping the end user’s willingness to try a new technology (Muller 2002 p. 299).

The ultimate advantage of PD is that the new software product will not only be useable and useful but the end users will have a greater understanding of how the application actually works.

6.8 Expectations

Present and future software is expected to be more interactive and unobtrusive to the end user than earlier software applications. Therefore, new software design cannot exclude frequent participation from the end users. PD claims other user oriented design methods do not truly include end users. The reality is that the other methods include end users as post ad hoc members of a team that simply review and sign off on requirements.

However, embracing PD will be difficult in the United States primarily because of our management philosophies. Embracing PD requires end users full participation in the project as actual designers. “Participatory design implies that workers as users of
computer products should take part in the decisions that affect the system and the way it is designed and used” (Greenbaum 1993 p. 28). Although the U.S. culture is considered democratic, when it comes to software design and development, a few people dictate the outcome. In theory software engineers agree that end users should be involved, however, in practice software engineers often feel that end users make impossible requests that cannot be accommodated in the time frame and budget established by stakeholders. “Many countries have legislated that representatives of different interests be involved in projects; elsewhere, management and labor have agreed to consultation over any proposed change in working conditions. In most European countries formal technology agreements between trade unions and employer organizations establish requirements for development processes” (Grønbaek, Grudin, Bødker, and Bannon p. 80). Therefore, although PD works in other countries, the United States has adopted other design methods, which limit end user input in the software development process. “Although sustained user involvement seems desirable, its effect on commercial products is not clear” (Grudin and Pruitt 2002 p. 1). This uncertainty about the impact to the bottom line limits the ability of organizations to embrace PD.

7. Scenario-Based Design (SBD)

7.1 History

The origin of scenarios is in theatrical studies. Scenarios were and are used by many people such as military employees, economists and policy makers for long range planning and to weigh the consequences of actions (Carroll 2000a p. 320). However, Ivar Jacobsen promulgated the popularity of scenarios with his use case approach to software engineering (Carroll and Rosson 1992 p. 182).

In the early 1980s, SBD emerged from the HCI field to resolve the need to provide the right balance between modeling approaches and purely experimental approaches that relied heavily on usability testing (Jarke, Bui and Carroll 1998 p. 159). As HCI continued to develop through the 1980s, focus was placed on the concept of usability and its role in system development. It was becoming obvious that usability testing could be moved up and included in each stage of the software development process to minimize rework after the system was created. Usability testing became part
of every step of the software development lifecycle starting with analysis (Rosson and Carroll 2002 p. 13).

In the 1990s, scenario-based software development techniques became popular in software engineering because they described key situations in a narrative format that an audience representing various perspectives could easily understand (Rosson and Carroll 2002 p. 7). This eliminated the need to share complex models with the end users and stakeholders. Scenario techniques bridge the communication gap between the team members that provide requirements and the team members that produce and implement the design. Scenarios are a mechanism that can be used to streamline the expectations about the final product.

7.2 Definition

“Scenario-based design tries to identify the critical and typical things that people do or want to do and the tradeoffs associated with design features that enable these activities” (Rosson and Carroll 2002 p. 359). Scenarios focus on the situation including the people involved, the environment, the objectives and the goals. “In scenario-based design, descriptions of how people accomplish tasks are primary working design representations” (Carroll 2000b p. 45). Scenario-based design contains five key elements. The information in Figure 2 defines and describes these elements.

7.3 Technique and process summary

The goal of the SBD process is to focus the design team’s thinking on the necessary end user tasks (Rosson and Carroll 2002 p. 315). The SBD process does this by providing an iterative approach using different types of scenarios. The process provides the ability to refine the originally defined scenarios. The SBD process includes the following steps:

(Rosson and Carroll 2002 p. 346)

1. Development of requirement scenarios
2. Validation/refinement of scenarios with users and customers
3. Development of basic-level task scenarios
4. Refinement of design scenarios with development team members and customers
5. Development of information model
6. Review with team members
7. Development of paper prototypes
8. Walk-through of paper prototypes with users
9. Development of interaction model
10. Review with team members
11. Develop and refine a running prototypes
12. Formative evaluation
14. Detailed design and prototype driven iteration of previous three steps

Figure 2 – Five elements of Scenario Based Design

The scenarios are defined using a five-step method, which is: analyzing the organizational decision, identifying the key decision factors, analyzing environmental forces, defining the scenario logic and analyzing the implications for the final decisions.

Analyze organizational decisions is the first step and occurs prior to creating scenarios. This is because the team must understand the organization’s goal and vision for the future within the scope of the topic being analyzed. This analysis includes gathering information about the capital allocation, diversification, infrastructure
investments, and market strategies. The second step to the scenario building process is to identify the factors that influence an organization’s decisions. These factors provide insight into the organization’s values that can help the project team understand the strategic goals and provide a framework to make tactical decisions within the product development process. The third step is to analyze environmental forces. This analysis directly defines future business strategy. This includes considering potential competitors and general economic conditions. The background analysis provides the mechanism to create the initial scenarios. The fourth step is to define scenario logic. This step defines the scenario template that will be used throughout the process. “Scenario logic involves organizing themes, principles, hypothesis and assumptions that provide each scenario with a coherent, consistent and plausible logic underpinning” (Jarke, Bui and Carroll 1998 p. 164). This then moves the process to the fifth and final step, which is analyze implications for decisions and strategies. This step considers the implications of the scenario. The following questions should be answered in this step:

- What do the scenarios say about the design and timing of particular strategies?
- What threats and opportunities do the scenarios present for the future?
- What critical issues emerge from the scenarios?
- From the organization’s planning perspective, what kind of flexibility do the scenarios suggest are necessary?

SBD process is hierarchal and iterative in its approach to defining and refining the different types of scenarios. SBD defines five scenario categories: problem scenarios, activity scenarios, information scenarios, interactive scenarios, and edge case scenarios. The scenario categories provide the analysis focus for the team members.

A problem scenario focuses only on current tasks. These are created with the help of the stakeholders early in the project because they help define the scope of the project (Rosson and Carroll 2002 p. 25).

Activity scenarios (a.k.a. daily use scenarios) focus on the information the end user needs to complete the task. These scenarios attempt to resolve the issues defined in the problem scenarios. Activity scenarios are the initial step to creating a more efficient
workflow (Rosson and Carroll 2002 p. 142). These scenarios define what needs the process should be without regard to the design.

Information scenarios are an extension of activity scenarios. Information scenarios provide the detail about the information the system provides to the end user.

Interaction scenarios describe the details of the user action and feedback. More explicitly interaction scenarios explain: the information needs, the actions the user takes to interact with the task information, and the responses the system provides to the user’s action. These are very similar to use cases only in a more narrative format (Rosson and Carroll 2002 p. 16, 26).

The last type of scenario, edge case scenarios, should be used cautiously. Edge case scenarios include tasks that are uncommon, but analyzing these causes and effects can prevent an incomplete product design. In most cases, there is minimal value in creating edge case scenarios because they have a very narrow focus. As such, they should only be used when the time permits, or the level of complexity and impact of possible mistakes are so costly that these situations cannot be ignored (Cooper 1999 p. 181).

Once the scenarios have been drafted they are further refined with the use of claims analysis. Claims analysis is a form of participatory design where the design team and end users scan the scenario for the causes and effects and capture the design rationale for the future system. Claims analysis is defined by Chin, Rosson, and Carroll (1997) as “a refinement of participatory design in which the users directly analyze current and envisioned scenarios of use” (Carroll 2000b p. 272). Claims analysis provides a method for systematic questioning that is imperative in understanding the user’s needs. This technique brings to light issues, conflicts, and contradictions that are not obvious in the scenario. The problem with claims analysis is it is an iterative process that does not have a well defined starting or ending point (Carroll 2000b p. 135). This means that it is difficult to know when enough analysis has been done. Typically, this activity is time boxed so that when the time runs out the analysis is considered complete.

The diagram in Figure 3 provides an overview of the scenario development process. Notice that each type of scenario builds on the previous type. Although, at first
glance, it appears that scenario development is a linear process, it is actually an iterative process.

**Scenario Building Process**

![Diagram of Scenario Building Process](image)

**Figure 3 – Scenario building process**

### 7.4 Project team type

Cross-functional teams work best when using an SBD process. However, because time lines are shrinking due to market demand, it is increasingly more difficult to have all team members participate in every step of the SBD process. Each team member typically has their own area of expertise and it is rare to have them move out of that area even
when participating on a cross-functional team. Typically, design team members are assigned to build certain scenarios and then the team gathers to share knowledge.

7.5 Application domain

The two examples Carroll provides suggests that SBD is successful on well defined, small-scale projects, using existing technology and a centrally located teams. For example, Carroll describes a project delivering a multimedia system for educating engineers in a university. The project characteristics include; an enormous ill-defined scope, implementing complex technology, and geographically distributed team members. In the example of the multimedia project, Carroll explains that a vision scenario was created but only rarely referenced. The multimedia system project veered off the SBD approach and used a prototype driven process because of the enormous scope. The second example Carroll provides is a design of a library system that had a smaller, well-defined scope, and centrally located team members. Carroll explains that this project was also not completely successful. The SBD process used in the library system design did not accurately identify the root cause of the problem and this created rework. Carroll admits that the multimedia and library examples were not representative of best case practice for scenario-based design (Carroll 2000b p. 43). This calls into question the capability of this method.

The capability of this method is further questioned because Carroll explains that SBD has not used on projects that include safety-critical applications, embedded systems, or concurrency and, therefore, is not sure of its ability to support such projects (Carroll 2000b p. 327). “Scenario-based design may be differentially effective for different kinds of development projects” (Carroll 2000b p. 327). Carroll admits that “scenarios are often used ineffectively or inarticulately” (Carroll 2000b p. 43). Although there may be benefits of using scenarios as a technique, it is suspect that SBD will provide successful results.

7.6 Integration feasibility

Scenario-based activities can be integrated into a system development methodology. “McGraw and Harbison (1997) have described a scenario-based engineering process that integrates scenario based design with traditional structured development methods, and therefore, perhaps renders it more easily assimilated by organizations already practicing
these traditional methods” (Carroll 2000b p. 317). They explain that scenarios closely mirror how software engineers and software quality assurance team members reason within their functional tasks. The claims analysis will seem recognizable to organizations that have used traditional structured analysis methods to software application design. This is because software claims are analogous to usability claims used in SBD. The disparity is that software claims address implementation features that have consequences on software design quality. The usability claims articulate the rationale for designing with end users in mind.

7.7 Advantages
Scenarios are easy to produce and assist with cross-functional team participation and information sharing. The natural language of scenarios allows team members from various perspectives to participate. They also provide a mechanism to share knowledge within the team and to transfer knowledge to members outside the team.

Scenarios help plan the work. “The scenario-based process allows all the activities to communicate in a common vocabulary: orienting goals and visions, workplace needs analysis, design walkthroughs, envisionment, prototypes and rationales, evaluation of critical incidents and test cases, and potential generalizations and abstractions” (Carroll 2000b p. 309). The review of the scenarios helps the team members see the project goals in terms of the end user’s needs along with the organization’s goals. In addition, the scenarios facilitate the completion of each team member’s functional tasks. For example, usage scenarios can help to define interview questionnaires that the interaction designer may be required to create. They can also be used to help guide the implementation and deployment because scenarios explain the frequency of the tasks and potential peak times when it may not be appropriate to implement a new system. Scenarios expound the patterns and social mechanisms that are important to consider in the design. The patterns of task completion in a scenario identify the appropriate functionality layout.

Another advantage is that scenarios help designers reflect while giving them a sense of accomplishment. “A scenario is a concrete design proposal that a designer can evaluate and develop, but is also rough in that it can be easily altered and allows many details to be deferred” (Carroll John M 1999 p. 10). Creating scenarios allows designers
to move towards a more concrete design. "Scenarios are good at vividly capturing the consequences and trade-offs of designs at various level of analysis and with respect to various perspective" (Carroll 2000b p. 87). This leads to the next advantage.

SBD provides a systematic method to design and evaluate. SBD facilitates design rationale using claims analysis. The claims analysis technique, which is a large aspect of SBD, makes the relationship between functions and consequences of use explicit (Carroll 2000b p. 152 - 53). Claims analysis allows the team to analyze explicitly the pros and cons of a specific design as it relates to the scenario. The systematic method minimizes downstream design changes.

SBD and its techniques match the design and development team members’ cognitive model making it easy for them to learn this method. Carroll stipulates that object-oriented software requires developers to think in terms of functions and SBD complements object-oriented design reasoning because the various types of "scenarios are ideal structures for engineering requirements and usability, that is, for identifying the required functionality of systems and evaluating the effectiveness of their realization (Carroll 2000b p. 233).

7.8 Expectations
SBD techniques have been successful and will continue to be successful when integrated into other user oriented design methods. Scenarios as a technique have been borrowed by other user oriented design methods because of the value this technique provides to the initial analysis phase. However, SBD as a user oriented design method has not yet proven itself robust enough for any type of project. Carroll admits that "it is far from clear that scenario-based design will provide a completely general and comprehensive system development methodology" (Carroll 2000b p. 327). It is questionable if SBD will ever be perceived as a useable design approach because its reliance on the scenario as its main deliverable.

One of the challenges for SBD is that the project team must be able to tolerate the vagueness of the requirements. Scenarios allow much interpretation or misinterpretation by the reader. Software engineers customarily have a low tolerance for ambiguity within the requirements, which is the most probable reason for SBD not being widely accepted in the software development industry. Carroll explains, "most of the projects studied
used scenarios in the development and analysis of prototypes typically an iterative process of requirements development." (p. 317). The use of scenarios as an analysis tool works well; however, to use them as the primary deliverable to a software engineer has not been proven successful.

The informal nature of SBD is yet another obstacle for SBD. Carroll explains, "Scenarios are not formal; they are not technical in any fancy sense" (Carroll 2000b p. 17). SBD techniques produce words rather than pictures or diagrams, and most technical team members do not enjoy reading volumes of information. The technical team members cannot see the picture if they do not read the words.

In addition, the repeatable success of SBD has not yet been proven. Although Carroll's ultimate goal is to make SBD the key to moving design work into a science, this goal is unrealistic. The problem is that no two software development projects are exactly alike.

Lastly, SBD success depends on the training, support, and creation of scenario management tools. There is a critical need for tools that create, modify, organize and track the scenario creation process. Scenarios can become difficult to manage when there are too many iterations and variations. Although scenario creation on the surface seems easy, producing the right number with the right amount of detail is a challenge. In addition, designer and software engineers need to learn how to transform the scenario into a useful and usable software design.

8. **Contextual Design (CD)**

8.1 **History**

Contextual Design (CD) began to evolve in 1988. "CD was designed at Digital Equipment Corporation under the leadership of Dr. Karen Holtzblatt" (Marion 1997a). Holtzblatt created CD to prevent project teams from agonizing about what should be designed by involving end users upon the project initiation. In 1992, Karen Holtzblatt and Hugh Beyer founded InContext Enterprise, which is a consulting company that is aligned with the CD method. CD continues to develop as it is being used by more organizations striving to achieve user-centric software products.
8.2 Definition

Contextual Design (CD) is "an approach to defining software and hardware systems that collect multiple customer-centered techniques into an integrated design process (Beyer 1997 p. 3). The CD method provides techniques for gathering data, driving design, and managing teams. This method stresses the need to gather end user data in the form of data models that build on a structural analysis approach using techniques such as object modeling, enterprise modeling, and process reengineering. The techniques help to facilitate the decisions about how the end user will function in the future. “Contextual Design provides complete support for the design process, from the initial customer data gathering through the transition to object-oriented design or whatever other implementation model you favor” (Beyer 1997 p. 5). It provides a clear process and concrete actions. The central aspect of CD is that it promotes staying grounded in the understanding of how the end user works.

8.3 Technique and process summary

The steps for CD are as follows:

1. Contextual inquiry
2. Work modeling
3. Consolidation
4. Work redesign
5. User environment design (UED)
6. Mock-up and evaluation

The initial step in CD is performing contextual inquiries. Contextual inquiry is a technique that provides insight into the current workflow and includes uncovering the end user’s tacit knowledge. Contextual inquiry typically begins with one-on-one interviews in the end user’s work environment. An alternative approach to contextual inquiry may be needed if the work takes place over a long period or is difficult to observe. Alternative approaches can be identified in Appendix A.

The subsequent step in CD is creating work models. Work models consolidate the information gathered during the contextual inquiry and provide a tangible representation of end user’s work. CD provides several different modeling techniques that assist the project teams understanding the breadth of the end users requirements. By considering the end user’s work process from end-to-end, the project team can develop a
more flexible software application architecture that supports future expansion. In addition, these models are useful in managing the project and the marketing of the software application because they clearly display end user’s workflow and the high-level functionality. CD uses five types of work models: workflow models, sequence models, artifact models, cultural models and physical models.

Workflow models help teams to understand the end users’ communication patterns. The team identifies the roles and responsibilities of the people who do the work and how they interact. “The consolidated flow model is the best map to how work is done, showing the breadth of work and the details of how people interact. The flow model shows what roles people play, and if you have systems already in place, you can see what roles [are] supported” (Beyer 1997 p. 170). Typically, work roles are consistent across domains which can assist the team in providing a software application that is useful to a broad range of customers if that is the goal of the project.

Sequence models show the order of steps taken to complete the work, including the trigger that initiated work. “Sequence models show what the customer is trying to do and how they go about doing it” (Beyer 1997 p. 197). The sequence model illustrates communication breakdowns within the process being analyzed. These models are similar to task on arrow models and decision diagrams that are used in structured software analysis approaches. However, the goal of the sequence models is not only to show the order of the steps to complete the work, but also to illustrate the current communication breakdowns.

Artifact models include job aids, reference manuals, and guides. The artifacts illustrate an opportunity to enhance the current software application. “These [artifacts] reveal how people organize and structure work from day to day” (Beyer 1997 p. 178). The artifacts are analyzed for structure, usage, and intent. The goal of this analysis is to integrate information into the software application.

Culture models reveal work constraints caused both by an organization’s policy as a whole and values of individual employees. The way employees dress and the language they use defines an organization’s culture. The culture is further defined by the relationship between departments and the general regard to the customers. “The cultural models speak the words people think but don’t say” (Beyer 1997 p. 112). Attentiveness
to employees’ regard for each other and the customer assists the design team to understand the organization’s culture and provides insight for establishing the basic design standards. For example, if a software tool is created for a conservative company the look and feel should not include glitzy graphics.

Physical models represent the grouping of people and equipment, and the movement between objects in order to get the job done. The environment that a tool is used in will often reveal design constraints. The movement of artifacts, communication, tool placement, amount of space, and temperature all affect design. This model guides the design team’s creation of a suitable metaphor and the placement of functions within a software application.

Each model focuses on a unique perspective to help the team stay focused on the end users’ issues. The flow model identifies the roles and responsibilities for sequence model. The responsibilities are further analyzed to define the tasks that make up the sequences model. Artifact collection identifies the tools necessary to complete one or more of the steps in the sequence. The combination of these models is then used to understand the company culture, and the physical layout of the workplace in relation to the end user’s requirements.

CD provides a specific step that involves consolidating the information, making it manageable enough to move into a design approach. The consolidation effort uses interpretation sessions where affinity diagrams are developed. The interpretation session include participation of the entire team and builds a sense of commitment in the development of the appropriate solution. The interpretation sessions allow everyone to hear the same questions, answers, and discussions. Therefore, team members build on each others’ ideas. Beyer explains that:

“Interpretation is the chain of reasoning that turns a fact into an action relevant to the designer’s intent. From the fact, the observable event, the designer makes a hypothesis, an initial interpretation about what the fact means or the intent behind the fact. This hypothesis has an implication for the design, which can be realized as a particular design idea for the system” (Beyer 1997 p. 57).
Therefore, the interpretation sessions provide the appropriate forum to develop an affinity diagram and consolidated work models.

Affinity diagramming is one of the tools used to illustrate the consolidated work models. The affinity diagram typically has a hierarchical structure in which the affinity is an outline of the end user’s work. It illustrates the key elements of the problem that the design team must resolve. The consolidated work models including the affinity diagram represent seemingly trivial end user anecdotes as real system obstacles that require a solution.

The consolidated task sequence model can be done with the use of storyboards to sketch out how the system will work to accommodate the task. Storyboards are rapidly created and used to illustrate with minimal detail how the work will be done. This technique provides the team a visual aid for what has to happen in the design and when it has to happen.

Work redesign is the next appropriate step if it is within the scope of the project. Work redesign provides efficiencies to the work practice that may or may not include the use of technology. This step also takes advantage of the storyboard technique because it visually communicates the workflow.

Once the work models are consolidated, the user environment design (UED) model is created. CD uses a UED model to illustrate how each part of the system supports the end user’s work, what functions are available, and how they are interconnected. The UED is not concerned about the UI details, but rather the functions that the system will provide and how these functions relate to each other. The functions are then made available through menus, toolbars, and keyboard commands that can be considered later in the design process. If storyboards were created they can be leveraged to build the UED model. However, the UED model can be built without storyboards if a current system is in place and the project goal is to reengineer the current system. In this situation the UED is used to reveal the current system’s problems in meeting the end user’s workflow requirements. Similar to the storyboard, the UED is used to manage the structure of the system; the difference is the UED adds a level of detail.

The UED can be used for several purposes depending on the goal of the project. For any type of project, the UED provides a mechanism to plan a phased or incremental
release. This is because it illustrates the scope of functionality for the end-to-end work processes being analyzed. When the goal is to create a new software application, the UED provides the team a concrete and consistent understanding of what data needs to be available to complete each activity within a given process. The UED specifically assists the interaction designers in creating an appropriately layered interaction design because “the UED works against proliferation of dialog boxes” (Beyer 1997 p. 398). If the UED has allotted for a function to be part of the focus area, then that function must have allocated space in the design. The UED can also be used to leverage reusable components because it illustrates the functional needs of the software application prior to the design step. When the goal of the project is to enhance a current system or consolidate systems, the UED is used to reveal the problems with the current system and in this case the use of storyboards may not be necessary because the goal is to only fix system functionality problems. The UED supports the comparison product analysis by illustrating the functions within each product and reveals each system's current systems drawback.

Once the UED models have been developed and the design team has a solid understanding of the tasks and related function, the user interface design can begin. This step entails creating mock-ups also referred to as paper prototypes. The mock-ups are used in the initial evaluation by the end users. This stage is the most iterative.

8.4 Project team type

The CD method requires a joint effort from several different perspectives, and therefore requires a cross-functional team. This is because to have a successful product all team members must understand each other's abilities and limitations. For example, marketing people need to understand an organization’s technical limitations, and software engineers need to understand what makes a valuable product from a marketing perspective. Both groups need analysts to keep them focused on end user's needs and the customer's budget.

8.5 Application domain

The CD method is flexible and can be applied to any type of project, however it has been “optimized for large projects” (Beyer 1997 p. 21). The CD method is flexible enough to “incorporate additional techniques and processes as the need arises” (Beyer 1997 p. 21).
or to remove techniques that may not be appropriate for a particular project. CD can be successfully applied to the following project types:

- Maintenance of an existing product.
- Redesigning an existing product.
- Creating a brand new product
- Integrating independent systems. (Beyer 1997 p. 427).

The different types of projects may require focus on a different aspect of the design and therefore, the project type and project goals will dictate the data-gathering process. “The kind of data you look for will not only be driven by the work you plan to support, but also by the goals of the project” (Beyer 1997 p. 69). In addition, the number of available team members and end users can require alterations to the CD techniques. “What works for a two-person team won’t work for a fifteen-person team; what works to design a strategy for a new market venture won’t work for the next iteration of a 10-year old system” (Beyer 1997 p. 25). The CD techniques have been flexibly created to accommodate the various needs of software projects.

8.6 Integration feasibility

The CD techniques focus on the front end of the software development project lifecycle, and therefore can be seamlessly dovetailed with any software engineering process. The concentration of CD is on the analysis and design phase. CD is simple to integrate into traditional software development organizations because the deliverables that CD produces are similar to data diagrams and flowcharts that the software engineers will be familiar with and comfortable interpreting. Organizations striving to achieve ISO 9000 compliance are required to document in detail the system functionality and continually measure customer satisfaction. The techniques in CD provide the organization the necessary techniques to gather and measuring the end user’s satisfaction prior to implementing a solution. In addition, the various models produced within the CD method facilitate the necessary deliverables that describe a system’s functionality that are necessary when an organization is striving to be ISO 9000 compliant.

8.7 Advantages

CD is flexible. “You can alter or substitute steps that achieve the same intent, add new techniques to put more emphasis on a step, or remove steps you believe are irrelevant to
your particular problem” (Beyer 1997 p. 415). CD can be customized to fit the project type and project goals. CD is typically noted for its flexibility because it focuses on the front end of the design process, and this provides the opportunity for this method to be used in conjunction with many of the other user oriented design methods.

CD supports defining structure and concepts. Using various modeling techniques, CD supports the team in discovering and understanding the end user’s workflow.

CD provides structure to an unstructured design process. CD “balances the need of an engineering organization to produce a result in a given time frame against the need of the design team to really understand their [end users] and how they work. It provides a structure concrete enough that people know what to do when they come into work in the morning, but with freedom enough for people to be creative” (Beyer 1997 p. 437-38). It keeps the cross-functional team focused on their task, and minimizes conflicts that arise when people from various perspectives come together.

CD makes the design work visible in the early stages of the project. “Contextual Design takes part of the design conversation out of the designer’s brain and puts it on the wall as a model!” (Beyer 1997 p. 153). The design is a shared effort. “The design is owned not just by one person, but by the cross-functional design team” (Beyer 1997 p. 152). The advantage is CD produces the design that incorporates many different perspectives and ideas. It also produces illustrations that can be shared with management to measure the team’s progress.

CD incorporates prototyping, which helps communication between designers and end users. The combination of linguistic tools such as contextual interviews with non linguistic tools such as prototypes provides the tools to communicate between the various perspectives which can lead to a usable and useful design. The continued interaction with the end users provides valuable information to the design team and the marketing team. The side effect of end user involvement that often goes unnoticed is the early adoption by the end users who participated in the design process and their word of mouth marketing. Having end users continually involved in the design process builds trust, which is also integral to the perceived success of a software application.
8.8 Expectations
The success of CD depends on the ability of organizations to recognize the need for a vigorous front-end design process that includes the implementation of cross-functional teams. The traditional company cultures where work is done in silos must be modified for CD to be effective. This migration from silos to integrated work teams will require organizations to change the physical layout of software design areas. Organizations need to provide space for group meetings and long-term design sessions. In addition, since CD is no different from any of the other user oriented design methods it also needs additional tools to support storing and indexing for the information that is generated in each technique.

9. User Centered Design (UCD)
9.1 History
“User-Centered Design has its origins with the seminal work of Norman and Draper (1989). Others have further operationalized and optimized the basic approach” (Vredenburg, Mao, Smith, Carvey, 2002 p. 471). In the early 1990’s, IBM employees, Vredenburg, Isensee, and Righi created a more integrated version of UCD. These three individuals continued to be instrumental in the evolution of the integrated UCD approach (Vredenburg, Isensee and Righi 2001 p. xxiii).

9.2 Definition
The generic nature of UCD resulted in the creation of several different flavors of UCD along with several definitions associated with this approach. Vredenburg, Isensee, and Righi explain, “because the term UCD has been used to describe a generic approach to product development, many flavors of UCD exist in the industry” (Vredenburg, Isensee and Righi 2001 p. 20). The generic flavors they are referring to are the other user oriented design methods discussed in this paper. UCD is generically defined as

“a highly structured, comprehensive product development methodology driven by (1) clearly specified, task oriented business objectives, and (2) recognition of users needs, limitations and preferences. Information collected using UCD analysis is scientifically applied in the design, testing, and implementation of products and service. When rigorously applied, a UCD approach meets both user needs and the business objectives of the sponsoring organization.” (Keller et al. 2000).
UCD in its generic definition forms the bases of all other user oriented design methods. The generic definition of UCD can be applied to many of the user oriented methodologies discussed in this paper. Therefore, by establishing an understanding of the UCD method and UCD techniques the similarities between the other user oriented design methods and their techniques become obvious.

Vredenburg, Isensee, and Righi have defined an integrated approach to UCD that distinguishes UCD from other user oriented design methods. The definition of integrated UCD as defined by Vredenburg, Isensee, and Righi is:

“[designing] ease of use into the total customer experience with products and systems. It involves two fundamental elements: multidisciplinary teamwork and a set of specialized methods of acquiring user input and converting it into design.”

The integrated UCD approach encompasses more than the interaction between the end user and the software. It includes the “total customer experience from the first time a potential customer sees an ad about a product or service through the time he or she would like to upgrade to a new version” (Vredenburg, Isensee and Righi 2001 p. 20). The integrated UCD approach incorporates analysis of the entire software acquisition and maintenance process from the end users perspective. The process includes analysis of the end user locating the ad, ordering, receiving, unpacking, using the software application including the use of various help mechanism, and finally the upgrade of the software application.

The terms Katz-Hass uses to define UCD are different from the other practitioners, but the connotation of UCD aligns with Vredenburg, Isensee, Righi, and Keller.

9.3 Techniques and process summary

Although many people associate usability testing with UCD, usability testing is just one of the techniques in the UCD method. UCD contains several techniques that can be applied not only to the software design, but to the total customer experience. The various techniques are used throughout the software product development lifecycle to accomplish the project specific user-centric requirements. UCD applies the techniques to the overall development lifecycle with the use of multi-disciplinary teamwork (Vredenburg, Isensee
and Righi p. 19). Multi-disciplinary teamwork is discussed in a later section. The focus here is on the UCD process that converts the user’s voice into a successful UCD design.

The integrated approach that Vredenburg, Isensee, and Righi suggest using initiates with the product concept and ends with the validation that the product met end users expectations. The process allows parallel work activities to keep timelines reasonable. The integrated UCD approach is summarized in Figure 6.

Figure 6 illustrates that the initial ideas and drafts are created in the product concept stage. By the end of this phase, a high-level product concept is defined. Once the product concept is established, the analysis phase can begin. This phase is often refereed to as ‘requirements gathering.’ The requirements gathering technique approach contained in UCD is different from the traditional software engineering approach because in UCD the requirements gathering techniques include the end user’s perspective and are not merely functional requirements that have been traditionally captured. Furthermore, requirement gathering in the UCD method seeks to understand the end users needs, cognitive model, and environment. UCD requirement gathering phase also considers how the competition is meeting the end users needs. The requirement gathering phase in UCD utilizes one or more of the following techniques:

- Interviews
- Observations
- Focus groups
- Feedback from prior versions, and system comparisons

![Marketing Diagram]

**Figure 6 – Integrating UCD into the product development process**

(Vredenburg, Isensee Righi p. 71)
As the requirements begin to come together, the conceptual design phase launches, in which initial design ideas and paper prototypes are produced. This phase consists of several iterations and although, these iterations seem time consuming the iterations at this phase prevents design problems in later phases. During this phase, it is critical to gather feedback from the end users. When team members say, “I think the end user wants it this way or that way” and an argument ensues, it is a good indication that it is time to request end user feedback. One or more of the following techniques can be utilized in the conceptual design phase (Vredenburg, Isensee and Righi 2001 p.132):

- **User profiles** – a technique that produces “detailed descriptions of the relevant characteristics of each user category” (Vredenburg, Isensee and Righi 2001 p.132).
- **Competitive evaluation** –a technique that evaluates other software applications that support similar work.
- **Contextual inquiry** – an information gathering technique that requires visiting the end user in their own environment. That way, observers can ask questions about the tasks that need to be supported by the new software being created.
- **Task analysis** – a technique that requires end users to define the tasks necessary to complete the work being analyzed. Once each task is identified, it is decomposed into the steps necessary to perform the task. This technique provides an understanding of workflow. The new software product must provide efficiency in the workflow. Task analysis is most successful when done in a focus group setting after the completion of contextual inquiry, which provides a base level of knowledge of the work process.
- **Hierarchical Task Analysis (HTA)** – a hierarchical illustration based on the task analysis that defines the tasks and the decomposed sub tasks.
- **Use cases** – a technique that illustrates the interaction between the end user and the system. Use cases are scenarios that include a description of the end users actions and the system’s response to that action.
- **OVID (Object, View, and Interaction Design)** – is a technique “for designing the user interface by analyzing user goals and tasks and using these results to create an abstract diagram that describes the user model” (Vredenburg, Isensee and Righi 2001 p. 133).
- **Rapid prototyping** – is a technique that visual illustrates on paper the user interface, also known as a low fidelity prototype. A conceptual design is created using this technique. The conceptual design is then refined with the use of additional techniques.
- **Design walkthroughs** – a technique used as a checkpoint to ensure the design is moving in the right direction. This meeting includes all team members, end users, and stakeholders with the purpose of evaluating the design and gathering additional information to create a more detailed design.
Once the conceptual design feedback is provided and incorporated into the conceptual design, the detailed design and development phase begins. This phase is the combination of two sub phases, the detailed design phase and the development phase. The reason it is presented together is its iterative nature.

Although UCD requires an iterative approach to be successful, planned iterations must expand the functionality defined in the conceptual model. The purpose of iteration is not to redesign, but rather to add functionality incrementally. Successfully iterating through the design and development phase requires a stable conceptual design. According to Vredenburg, Isensee and Righi the “prime cause of wandering iteration is not having a good conceptual model of the system” (p. 149) A stable conceptual design allows for the detail design to be segmented into logical functional components. The defined iteration allows the team to focus on the detail designs of the functional component and upon completion, hand it over to the software engineers for development; meanwhile design for the next functional component is set in motion. Some or all of the following techniques may be utilized during the detailed design and development phase:

- **Prototyping** – a technique that involves an iterative process of designing a software solution. The design moves from lower-fidelity to higher fidelity with each iteration allowing for evaluations that are more rigorous.
- **Usability Metrics** – a set of criteria used to evaluate the design. “Examples of usability metrics include completion time for a specified task, number of errors per task, and time to complete each task. An example of a measurable usability requirement would be: The top five user tasks will be performed with 10% higher satisfaction than the primary competitor, as measured in a usability lab test” (Vredenburg, Isensee and Righi 2001 p. 113).
- **Heuristic evaluation** – this technique evaluates the user interface design against a set of criteria. “Heuristic evaluations entail multiple evaluators inspecting a user interface with regard to its compliance with a set of recognized usability principles, known as heuristics” (Vredenburg, Isensee and Righi 2001 p. 151-52). Heuristic evaluation is considered a discount usability technique. The trade-off is that heuristic evaluations are not in the end user’s voice (Mayhew 1999).
- **Usability walkthrough** – a technique that involves “an informal verification of the interface” (Vredenburg, Isensee and Righi 2001 p. 153). Evaluators will step through a set of tasks where the focus is on the interface. These events can be conducted with a low, medium, or high fidelity prototype. The trade-off is that the usability walkthrough does not reveal if the end users understand the software application design.
- **Usability test** – a technique that provides statistics regarding the systems ability to interact with the end user. The usability metrics provide the criteria
for the evaluation. “Formal usability testing is the most rigorous form of
evaluation. It typically employs experimental design methodology to measure
user performance in a manner that allows statistical comparison to be made
and conclusions to be drawn” (Vredenburg, Isensee and Righi 2001 p. 154). The trade off is that usability testing requires time to plan.

- Design Guideline Development Documents – a document that defines the
details of the design and serves as the blueprint of the product. It is very
detailed and according to Vredenburg, Isensee, and Righi it includes:
  - Visual specifications (e.g., font type and size)
  - Interaction specifications (e.g., what will occur at
    the click of a control)
  - Error conditions
  - Associated messages
  - Selection states (e.g., selected, rollover, not
    selected) and their visual appearance) (p. 163)

- Early-ship survey – a technique that provides feedback about the completed
design. “An early ship evaluation, also known as a beta evaluation for
software products, occurs just prior to a product’s release” (Vredenburg,
Isensee and Righi 2001 p. 163). From this feedback minor changes may be
made. In addition, this information can assist marketing to create the
appropriate marketing materials.

As the detailed design and development phase concludes planning of product
launch activities can begin. Once the product is launched, the design team’s tasks are not
complete. Although many other more recent user oriented design methods cease at the
launch phase, UCD supports and encourages the use of benchmark assessments and
postmortem evaluations. Benchmark assessments compare performance of the recently
released product to the competitor’s products or the previous version of the product. This
assists the organization in determining what changes need to be made to remain
competitive or to continue to enhance and existing product.

The postmortem evaluations occur because of the project management process.
However, the UCD also encourages the completion of postmortem evaluation and
specifically the evaluation of the design process. These evaluations provide feedback to
the team about achievements and areas for improvements. This is particularly important
for organizations just starting out in UCD.

The following summarizes the integrated UCD process as defined by Vredenburg,
Isensee, and Righi (p. 28):
1. Set business goals. Determining the target market, intended users, and primary competition is central to all design and user participation.

2. Understand users. An understanding of the users is the driving force behind all design.

3. Design the total customer experience. Everything a customer sees, hears, and touches is considered in the design process.

4. Evaluate designs. User feedback is gathered with rigor and speed that drives product design.

5. Assess competitiveness. Competitive design requires relentless focus on the ways users currently carry out tasks and an analysis of how to improve their process.

6. Post implementation evaluation. Soliciting user feedback after the release provides valuable information for future software designs.

Figure 7 visually represents the key components of the UCD process.

9.4 Project team type

The UCD process requires the selection of a cross-functional team. “UCD involves design specialists from several disciplines, such as marketing, human-computer interaction, visual/industrial design, user assistance design, technology architecture engineering, service/support, and user research (often called human factors engineering)” (Vredenburg, Isensee and Righi 2001 p. 19). Unfortunately, it is difficult for team

Figure 7 – Multidisciplinary Design of the Total Customer Experience (Vredenburg, Isensee and Righi 2001 p. 48)
members from different perspectives to blend smoothly. Working in a cross-functional team requires that each team member has an explicit primary role, and at the same time, is receptive to crossing over roles.

The primary benefit of cross-functional teams within a UCD approach is that the team has a unified understanding of the project goal and has a unified understanding of the end users. In addition, when marketing and software designers work together the transfer of knowledge about each others perspectives naturally occurs. This knowledge transfer provides the opportunity for team members to work more efficiently. Marketing may be able to gather additional detail about the end users if they understand the value it provides to the team. Marketing gathers characteristics about the individuals who will be purchasing the product and those individuals may not be the actual end users. Therefore, it is essential to identify the characteristics of the actual end users. This is not to say that marketing has no interest in the end user, but the end user may not be the same as the target market as defined by marketing.

9.5 Application domain

Although the focus within this paper is on applying UCD to a software solution, UCD can be applied to many different types of projects. "UCD can apply to the design of any products, from toasters to nuclear power plant control rooms and on to computer systems" (Vredenburg, Isensee and Righi 2001 p. 24). Since the focus here is on software development it should be mentioned that UCD can be applied to any type of software development project type (Vredenburg, Isensee and Righi 2001 p. 9).

9.6 Integration feasibility

Using cross-functional teams is one part of the UCD integration solution, but it is not enough because not all cross-functional teams are created equal. The traditional approach to software development is that each team member has an area of expertise. It is common for the team to meet, share knowledge and go back at work on their functional task in silos. The consequence of working in a silo is a loosely integrated software application does not seamlessly flow together. The gap between traditional software development and UCD software development process is too large to bridge with merely the use of cross-functional teams. However, a cross-functional team using the
appropriate UCD techniques produces the foundation for transitioning to a UCD software development approach. The information in Figure 8 summarizes the differences between a traditional software development approach and a UCD software development approach. Notice the descriptions are virtually converse. The transition from traditional technology-centered design to user-centered design must be meticulously planned and sensibly implemented to ensure sustained success.

Once an organization agrees to utilize a UCD approach within a project, the project plan must incorporate UCD activities and not manage them separately. “If UCD activities are not an official part of a project, they are usually reduced to ‘too little too late’” (Vredenburg, Isensee and Righi 2001 p. 5). UCD activities must be given the same priority as other tasks within the plan. If timeline slips, removing a UCD task prevents the possibility of optimum successes. “UCD can be modular and flexible consequently, it can be tailored to the needs and realities of a particular project” (Vredenburg, Isensee and Righi 2001 p. 5). Therefore, the ideal alternative when a timeline is lagging is to apply a shortcut for a selected UCD task or tasks. The shortcuts should only be applied to the stages that will have the least impact for achieving the usability goals. If the stage is critical to the usability goals and the time is of the essence it is always better to apply a short-cut than to eliminate the task altogether. Additional discussion about shortcuts can be found in the usability engineering section of this document and in Deborah Mayhew’s book Usability Engineering Lifecycle.

<table>
<thead>
<tr>
<th>Traditional Software Development Approach</th>
<th>UCD Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology/developer-driven</td>
<td>User-driven</td>
</tr>
<tr>
<td>Component focus</td>
<td>Solution focus</td>
</tr>
<tr>
<td>Individual contribution</td>
<td>Multidisciplinary teamwork including users, customer, human factor experts</td>
</tr>
<tr>
<td>Focus on internal architecture</td>
<td>Focus on external attributes (look and feel, interaction)</td>
</tr>
<tr>
<td>Quality measured by product defects</td>
<td>Quality defined by user satisfaction, performance (quality in use)</td>
</tr>
<tr>
<td>Implementation prior to human validation</td>
<td>Implementation based on user-validated feedback</td>
</tr>
<tr>
<td>Solutions are directed by functional requirements</td>
<td>Understanding the context of use (user, task, work environment)</td>
</tr>
</tbody>
</table>
Figure 8 – Comparison of traditional software development to UCD (Seffah and Metzker p. 4)

9.7 Advantages

The foremost advantage of UCD is it is a scalable process, which means that with limited time and resources it can be utilized. “UCD is like exercise: You may never do as much as you should but every little bit helps” (Vredenburg, Isensee and Righi 2001 p. 16). The same holds true for UCD. It may not be possible to apply the level of vigor that produces the perfect statistical results, but it can be applied to the vital functions of the application to create a useful and useable design. Vredenburg, Isensee, and Righi explain that for UCD to be embraced, it ‘must fit into a company’s business strategy and demonstrate added financial value” (Vredenburg, Isensee and Righi 2001 p. 28). The scalability of UCD provides this opportunity.

Another advantage is that UCD, unlike some of the other user oriented design methods supports “a full-cycle from product inception, to product design, to development and deployment” (Vredenburg, Isensee and Righi 2001 p. 163). The all encompassing nature of the UCD method provides a variety of techniques for each stage of the product lifecycle making it a robust method. The encompassing nature of UCD also provides the advantage of continuous end user feedback which minimizes the risk of identifying issues late in the product development cycle; and therefore minimizes costly mistakes.

UCD also provides cost saving measures. The consideration to the end users at the inception of the product minimizes the possibility of developing a software application that does not provide the necessary functionality or provides unnecessary functionality. In addition, the UCD techniques help to reduce support costs because consideration to the end users cognitive load is considered throughout the design process and appropriate design actions are taken.

9.8 Expectations

The continuing demand within software development organizations to compete by producing software that meets expectations through the development of simple, powerful, and esthetically pleasing designs will promote the continued evolution of UCD.
However, if UCD techniques are going to be embraced by the software development industry work in several areas needs to be done.

First, a consistent definition of UCD must be agreed upon. Seffah and Metzker agree that, although there are a variety of UCD techniques and tools, they are not widely used. They believe this discrepancy exists because UCD techniques have been created in a silo independently of the traditional software engineering approach also known as the technology driven or system-driven philosophy. They also state, “certain UCD tools and methods are better suited in certain development context rather than others” (Seffah and Metzker). Educating the project team about the various user oriented design techniques empowers them in the selection of the appropriate technique to achieve the project goals efficiently. Seffah and Metzker suggest becoming familiar with Human Computer Interaction and Software Engineering cultures so that a bridge between the two perspectives can be built.

Second, additional training must be developed to support the UCD process. Seffah and Metzker suggest “cross-pollinating disciplines” to help with the transition process. They point out that “training developers and usability engineers on how to work together and understand each other is fundamental” (Seffah and Metzker p. 6). Working in cross-functional teams is a skill that must be learned.

Lastly, additional tools need to be developed to manage the UCD method more efficiently. The ability to store and sort the information gathered will provide additional opportunities for reusing information within and across projects. Such tools will add orderliness to the massive amount of information that UCD provides. This additional orderliness will allow management to see the benefits of the UCD method.

10. Performance Centered Design (PCD)

10.1 History

The two most noted practitioners associated with PCD are Gloria Gery and Barry Raybould. These two individuals are often credited with being the pioneers of performance support and PCD. “In mid-1989 Gloria Gery was working with a group from AT&T on the concept of “knowledge support systems” and envisioning tools that could be added to workstations to give end users timely information to enable them to finish their work in a more efficient manner. The term “electronic performance support
"systems" emerged from this project to describe the integration of these tools (Marion 1997a).

PCD is a method that Gery developed to create easy-to-use Electronic Performance Support Systems (EPSS). Early EPSS’s required many add-ons that lacked integration making them cumbersome to use. PCD is a method that assisted design teams in the creation of integrated systems that are not only useful but usable.

Raybould extended the PCD method to create Performance Support Engineering. Raybould introduced Performance Support Engineering out of a need to “describe the emerging professional discipline and associated methodologies for those who designed and built these [Electronic Performance Support Systems] systems” (Raybould 2001 p. 1-2). PCD has evolve over the past several decades through continuous research about what renders computer systems simple to use versus complex to use, and how system design affects the people who interact with it. (Marion 1997b). PCD will continue to evolve as practitioners continue apply the techniques to emerging end user’s needs.

10.2 Definition

PCD provides a methodology for software development that not only includes but integrates the perspective of the end user into the design philosophy. “PCD infuses tools with knowledge, structures tasks, and enables performers to achieve the required level of performance as quickly as possible – at the very most, within a day – with minimum support from other people” (Marion 1997b). PCD facilitates the design of software that is intuitive without the need for training or support manuals. The various PCD analysis techniques facilitate the understanding of the end users conceptual work model and appropriately incorporate the end user’s metaphors and language into the design. A commonly cited well design PCD software program is Intuit’s Quicken. This is because Quicken supports the entire range of end user’s experience level by building on the knowledge the end user obtains by using the various functions within the application.

The goal of the PCD method is to design a software product that can be successfully used by all end user skill levels on day one. PCD facilitates a design that provides proactive and integrated support. Proactive and integrated support is not merely creating elaborate system help files, but rather the delicate balance of providing useful information without requiring the end user to explicitly invoke a help function and
without annoying the end user with unnecessary help. Extensive analysis is dedicated to the label names and wording of the instructions that are provided directly on the user interface to achieve the appropriate design.

10.3 Technique and process summary
The PCD method is an iterative cycle of rapid prototyping that includes end user participation. The techniques contained within the PCD process supports the design of a robust interface that supports the desired business objectives through direct support of the end user (Ariel). PCD similar to UCD includes consideration to the organizations mission and business strategies (McGraw 1997 p. 25).

Many of the PCD techniques have been borrowed from other user oriented design methodologies. The chart in Appendix A provides a summary of the overlapping techniques. The PCD analysis techniques are slightly unique because of the perspective and focus. PCD analysis focal point is on the tasks that are identified as complex by the end users.

The PCD approach can be broken down into phases. Raybould suggests that there are four phases to PCD which are: Look and Listen, Understand the Work, Design the Work, and Design the Interface (Raybould 2001). Although, Craig Marion explains PCD in seven phases which include: preliminary, requirements, definition, design, testing, coding, and delivery (Marion 1997a). Marion’s explanation appears to be more comprehensive; however, Raybould’s explanation is more focused. The main difference between Raybould’s and Marion’s explanation of the process is that Raybould focuses on the phases in terms of PCD where Marion takes a more generic approach and considers the PCD techniques that apply to each stage of a project. For example, the preliminary phase that Marion identifies is more of a project management concern that needs to be addressed when implementing any user oriented design method.

Raybould defines the first phase as the Look and Listen phase, which focuses on gathering data regarding the work task under analysis. This phase is equivalent to Marion’s definition of the requirements phase. Raybould explains that during this phase, work observations are conducted, discussions with management about goals and issues driving the business are completed, surveys are created and sent to gather statistical
information, and focus group meetings are scheduled and conducted. Marion explains that this phase allows the team to investigate the work in the current environment.

Look and Listen phase techniques suggested by Raybould include:

- Workplace observation and event logging
- Contextual inquiry
- Structured interviews
- Focus group
- Cognitive survey
- Process tracking

Techniques suggested by Marion:

- User profiles
- Contextual inquiry
- Work flow models
- Affinity diagrams
- Scenarios
- Storyboards
- Scenario flowcharts

Deliverables produced in the Look and Listen phase: (Raybould 2001 p. 2-11)

- Physical map – This is a description and diagram of the physical layout of the workspace showing potential barriers to performance.
- Organization and process metrics – Specific measurable performance targets relating to an organization’s goals and objectives.
- Personnel metrics – Specific measurable performance targets relating to individual goals and objectives.
- Event log – A form for recording key aspects of a job performer’s daily routine, such as customers, questions, work-related problems, and the different types of work performed.
- Pareto Chart - A graph showing the cumulative impact of performance problems.

Raybould defines the second phase as Understanding the Work (Raybould 2001). This phase focuses on information consolidation. The information gathered in phase one is reviewed and consolidated. This means that workflow models are consolidated to present the current environment. In addition, key barriers and business goals are defined. During this phase, the design team builds an understanding of the range of the end users experience levels. The main deliverable from this phase is a Performance Support Map which identifies “the key work tasks, the decisions that impact the business, the key barriers that get in the way of the work, the knowledge, information data, tools and
communications needed to support the work” (Raybould 2001 p. 2-40). Marion explains this phase as the definition phase and is when the platforms, data structures, and data flow are analyzed (Marion 1997a).

Understanding the Work techniques suggested by Raybould include:

- Affinity diagramming
- Alignment or barrier analysis
- Cognitive task analysis

Techniques suggested by Marion (Marion 1997a):
- Flowcharting task
- Storyboards
- UI Mockups
- UI design marathon – also known as participatory design
- UI walkthroughs

Deliverables that may be produced in Understanding the Work phase: (Raybould 2001 p. 2-40)

- Sorted observation record – A listing of observations such as performance barriers, insights, cultural influences, questions, actions, and design ideas broken down into separate numbered notes and sorted into categories.
- Knowledge dictionary – A dictionary of acronyms, common terminology, and system terminology accessible to the layperson.
- Knowledge flow – A diagram showing the knowledge flow internally and externally for an organization.
- Organization process map – A diagram or flow chart showing how an organization does work.
- Performer profile – A character description of employees that will have an impact on the performance support design structures.
- Performer process map – A flow chart illustrating the exact flow of an employee’s work tasks and key decisions.
- Work objects – A description of employees’ job aids.
- Performance support map – A table or chart that identifies work tasks and decisions that influence the business and that need supporting. These are combined with the key barriers that get in the way of the work.
- Affinity diagram – A hierarchical diagram showing the relationship and grouping of different concepts.
- Fishbone diagram – A hierarchical diagram of an organization’s goals and performance improvement needs.
- Interrelationship diagram – An illustration highlighting factors that impact on business goals.
The deliverable that Marion suggests that Raybould does not refer to is the style guide. The style guide deliverable has been borrowed from the user centered design method and is used to ensure consistency of the interface design. (Marion 1997a).

Raybould defines the third phase as Designing the Work (Raybould 2001). This phase focuses on workflow reengineering. The new workflow and the system that will support the workflow is conceptualized and designed. This then leads to the visionary prototype. The vision does not take into account what will and will not be done within the scope of the project. It is a vision of what the system would provide to the end user without regard to time or budget constraints. The vision allows the team to view the system’s breadth but not the depth. The functionality can then be prioritized. Storyboard scenarios are produced and used to design the necessary performance support structures within each functional component.

Designing the Work techniques suggested by Raybould include:

- Prototype the entire vision
- Storyboard scenarios
- Work focus map

Techniques suggested by Marion:

- Continue task modeling
- UI mockups
- UI

Deliverables produced in the Designing the Work phase (Raybould 20012-141):

- Vision – A single vision consolidated from several ideas that is generated during brainstorming session. It is then represented succinctly as a single drawing on a flip chart. It shows what the new workflow looks like, but does not show how it will happen.
- Performance support map – A completed performance support map showing how knowledge, information, and tools are integrated into performance support structures. This document now displays the detail that was unavailable in the previous phase.
- Storyboard scenario – An illustration of a scenario that shows steps in a performer process map.
- Work focus map – A diagram that shows the main navigation flow of the new system. It highlights the focus area of each screen and differentiates between what the end users do and what they see.
The deliverable that Marion suggests that Raybould does not refer to is the functional specification and high level technical specification. The functional specification defines the tasks that the software application will support. The technical specification defines the architecture of the entire system and contains the detail about the hardware and database structure (Marion 1997a).

The forth phase Raybould defines is the Design the Interface phase. This phase is iterative and produces a prototype that undergoes several evolutions. PCD approach to iterative design it to refine the design in the prototyping tool and not in the system intended to be placed in production. The benefit of this is that the coding in the prototyping tool does not need to be as clean as the code in the actual system. The disadvantage is the amount of time it takes to develop a high fidelity prototype that is being thrown away and the perceived waste of time. PCD differs from user-centered design in the iteration philosophy. PCD does not view design and development as the best practice iteration process, but rather design and redesign as the best practice iterative process.

The PCD process stops at the design phase. This is an obvious shortcoming for the PCD process. The PCD process in theory does not provide support through the actual coding stage. However, that is not to say techniques from other methods could not be integrated with this approach to provide a complete and customized user oriented design process. Marion defines this phase as the design phase. According to Marion it is the phase in which the detailed product specifications are established (Marion 1997a).

Designing the Interface phase techniques suggested by Raybould include:

- Scenarios
- Prototypes – low to high fidelity
- PCD heuristic evaluation
- Usability testing

Techniques suggested by Marion (Marion 1997a):

- Prototyping
- Usability test using prototypes

Deliverables produced in phase four: (Raybould 2001-178)
- Tabular scenario – A table format of a scenario that uses columns to represent how end users interact with the system and with other factors.
- Low fidelity prototype – A paper mockup of the new system showing performance support structures and user interface controls (widgets).
- Usability test record – A form to capture the results of a usability test. It includes an evaluation summary of an end users reaction to the proposed design.
- High fidelity prototype – An interactive mockup of a system interface or performance support structure.

10.4 Project team type
The team members that form a PCD cross-functional team include: the end user also known as performers, subject matter experts (SME), software engineer, project manager, and interaction designer. The team should also include members with perspectives in the following work areas:

- Learning technologies
- Human factors engineering
- Interface design
- System engineering
- Knowledge management (Huber et al. 1999 p. 11)

The cross-functional team is only one aspect of successfully implementing the PCD process. The team members must have clearly defined roles. The team members must be receptive to crossing rolls and filling multiple roles when only limited resources are available. The team must understand the project goal which requires the project goals to be clearly defined. The team members must understand the strategic goals of the organization(s) that will be using this product. The understanding of the strategic goals allows the entire team, including the end users, to participate in establishing the appropriate project goals. When team members are part of the project goal setting, they gain a sense of ownership for the outcome and this minimizes accountability issues. PCD also requires access to the end users for participation and feedback. Last, PCD requires a skilled project manager (Huber et al. 1999).

10.5 Application domain
PCD is most appropriate when the product design goals include ease of learning. This is because of its ability to support intrinsic, extrinsic, and external mechanisms that
integrate an end user's needs with the appropriate technology. PCD supports three design concepts:

1. Intrinsic – these design mechanism or interface controls are embedded directly in the interface. This means that the performer using the interface does not have to break the work context to use this control.
2. Extrinsic – these mechanisms support the performer doing the work, but require a context break. An example is cue cards because even though they support the work they are not part of the actual work.
3. Externals – these mechanisms completely change the performers work context. An example is a traditional help system (Lovgren).

The design concepts that are supported by PCD provide the ability for the method to be slightly less dependent on the project type but slightly more dependent on the project goal. This means that any type of project that requires a software product to be easy to learn can take advantage of PCD.

10.6 Integration feasibility

The PCD method can be integrated into a company's traditional approach to software development processes with just-in-time training. The difference between PCD and traditional software development process is the extensive involvement of the end users. Traditional design methods are deficient in involving the end user. The chart in Figure 9 compares typical development activities to PCD activities. The comparison is based on Marion's definition of the PCD process. Figure 9 illustrates the difference between the development activities and the PCD activities. However, according to Marion these activities should not compete but rather coexist. Notice that within the PCD activities the focus is placed on understanding the work that needs to be performed where as, the development activities take a system oriented view towards the design. Also, notice that the PCD activities continue to analyze and test the design throughout all seven phases.

10.7 Advantages

The PCD advantage is that it focuses on designing software products that provide the end user with day one achievement. PCD places priority on achieving "high levels of integration of knowledge, tasks structuring, data, and tools" (Gery 1997a p. 58). PCD accomplishes this by extending the user-centered design approach. PCD includes the analysis of the end users' cognitive model in terms of workflow that provides the ability to design the appropriate scaffolding architecture. PCD focuses on analyzing the end
users tasks, and on creating a performance centered system. “In a performance-centered design the emphasis is on providing support for the structure of the work as well as the information needed to accomplish it” (Lovgren p. 2). PCD places emphasis on the appropriate location of the information within the system. It strives to produce systems that are more interactive and integrated than traditional software.

PCD can also shorten the project life cycles. This is because PCD integrates activities that typically occur independently. For example, the consideration for technology is done in relation to the end user instead of considering technology in a silo. This consideration prevents the wrong technology from being implemented as part of the solution. In addition, PCD techniques focus on supporting the end user without the need for training and documentation. The overall results are that less training and documentation are required for PCD solutions and saves time and money for the end user and the participating organizations (Ariel).

PCD creates Performance Centered Systems that provide the following benefits: (Gery 1997b p. 2, 58)

- reduces performance cycle times (associated with tasks, processes, customer interactions, deliverables, creation etc.)
- reduces implementation costs (for a system, product, new process, etc.)
- reduced support costs
- increased customer satisfaction
- reduced transaction costs by providing the ability to shift work to less experienced employees or customers
- gap reduction between less experienced and star performers
- competitive differentiation as reported by customers
- increased performer confidence
- establishes and maintains work context
- aids in goal establishment
- structures progression through tasks and logic and institutionalizes best practice
- contains embedded and accessible integrated knowledge

<table>
<thead>
<tr>
<th>Phase 1: Preliminary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development Activities</strong></td>
</tr>
<tr>
<td>Begin project planning</td>
</tr>
<tr>
<td>Begin to organize teams</td>
</tr>
<tr>
<td>-</td>
</tr>
</tbody>
</table>
### Phase 2: Requirements

<table>
<thead>
<tr>
<th>Development Activities</th>
<th>PCD Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participate in investigation activities (led by UI team)</td>
<td>Investigate work</td>
</tr>
<tr>
<td>Gather technical/platform information</td>
<td>Envision work</td>
</tr>
<tr>
<td>Gather data source/flow information</td>
<td>Begin to differentiate base interface and performance support elements</td>
</tr>
<tr>
<td>A business analyst may work with the UI Design team in task analysis and creating use cases.</td>
<td>Set performance goals</td>
</tr>
<tr>
<td>Research documentation requirements.</td>
<td></td>
</tr>
</tbody>
</table>

### Phase 3: Definition

<table>
<thead>
<tr>
<th>Development Activities</th>
<th>PCD Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope project</td>
<td>Discuss/determine product objectives with Development</td>
</tr>
<tr>
<td>Discuss/determine design objectives with UI Design Team</td>
<td>Refine task modeling</td>
</tr>
<tr>
<td>Continue to develop project plan/schedule</td>
<td>Create UI mockups</td>
</tr>
<tr>
<td>Establish product objectives</td>
<td>Optional: Hold a UI design marathon</td>
</tr>
<tr>
<td>Analyze platform, data structures, and data flows</td>
<td>Conduct UI walkthroughs</td>
</tr>
<tr>
<td>Create functional spec</td>
<td>Create a style guide</td>
</tr>
<tr>
<td>Create high-level technical spec</td>
<td></td>
</tr>
<tr>
<td>Plan information deliverables (documentation &amp; training)</td>
<td></td>
</tr>
</tbody>
</table>

### Phase 4: Design

<table>
<thead>
<tr>
<th>Development Activities</th>
<th>PCD Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create detailed project schedule</td>
<td>Create prototypes</td>
</tr>
<tr>
<td>Create detailed technical specifications</td>
<td>Usability test prototypes</td>
</tr>
<tr>
<td>Begin creation of documentation from functional and technical specs</td>
<td>Create detailed UI Design</td>
</tr>
<tr>
<td>Create testing plan</td>
<td>Update the style guide</td>
</tr>
</tbody>
</table>
### Phase 5: Coding

<table>
<thead>
<tr>
<th>Development Activities</th>
<th>PCD Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code project</td>
<td>Give UI Interface specs to development</td>
</tr>
<tr>
<td></td>
<td>(including documentation and training)</td>
</tr>
<tr>
<td>Create first draft of documentation</td>
<td>Continue iterative usability testing</td>
</tr>
</tbody>
</table>

### Phase 6: Testing

<table>
<thead>
<tr>
<th>Development Activities</th>
<th>PCD Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit and function test</td>
<td>Continue usability testing</td>
</tr>
<tr>
<td>Component and system test</td>
<td></td>
</tr>
<tr>
<td>Create final documentation</td>
<td></td>
</tr>
<tr>
<td>Create installation materials</td>
<td></td>
</tr>
</tbody>
</table>

### Phase 7: Delivery

<table>
<thead>
<tr>
<th>Development Activities</th>
<th>PCD Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliver product</td>
<td>Continue usability studies at customer site</td>
</tr>
</tbody>
</table>

http://www.chesco.com/~cmarion/PCD/ImplementingPCD.html

**Figure 9 – Comparison of traditional software development to PCD method**

### 10.8 Expectations

The future of PCD looks bright. The current economic environment is providing the right elements for management to see the value of PCD methods. As industries continue to downsize there is a need for people to take on tasks outside their area of expertise. This creates the need for systems to provide embedded knowledge. The PCD method supports a design process to achieve embedded knowledge systems.

In addition, as more organizations strive to provide self-service applications, organizations will have to move away from traditional application design approaches and embrace PCD in order to produce applications that requiring little to no training.

However, the transition to PCD requires organizational changes for software organizations to embrace this approach. The addition of interaction designers will be required and will add to the dynamic of the organization’s culture. The interaction design of the software application is no longer the sole responsibility of the software engineers. The inclusion of interaction designers needs to be managed so that the appropriate PCD techniques are used within each project. In addition, since PCD is no different from any
of the other user oriented design methods; it also needs additional tools to support storing and indexing for the information that each technique generates.

11. Learning Centered Design (LCD)

11.1 History

LCD began to emerge as a method in the mid 1990’s. However, it gained worthy recognition in 1998, when a group of young researchers and LCD members of the CHI community combined efforts to hold the first LCD workshop (Hsi and Soloway 1998). This workshop kicked off the LCD method.

The creation of LCD was to some extent the result of the changing teaching strategies that occurred in the 1990’s. Educational strategies began to shift and open-ended teaching strategies become more popular. These strategies had the greatest impact on science and mathematics, where students began to engage in project-based learning. Project-based learning requires investigation into real problems. While these new teaching and learning strategies were emerging, technology advances in voice and video streaming also began to advance. This advance in technology provided the opportunity to create computer systems and interactive learning systems. The original systems were created with the use of the user-centered design techniques; however, it became obvious that user-centered design techniques fell short. Thus, using computers to ‘make people smarter’ requires a different design perspective such as LCD.

Elliot Soloway is recognized as a significant contributor to the development of LCD. “Soloway has proposed a Learner-Centered Design approach to developing software environments to address the special needs of users by reconceptualizing them as learners: growth, or the need for change in skills and knowledge; motivation, or the need for support in maintaining focus on work; and diversity, or the need to support a wide range of abilities and styles” (Loh et al. 1998 p. 628). The different perspective about a learner versus a user is what makes the LCD method abstractedly unique in comparison to the user-centered design method.

11.2 Definition

LCD has been defined as “a design approach aimed at developing software to support learners via scaffolding as they try to work in and learn the given work domain” (Quintana, Krajcik and Soloway p. 254). The goal for LCD is to achieve the appropriate
balance between direct support and empowerment resulting in the learner learning a new concept (Hsi and Soloway 1998). LCD must encompass a learner’s actions, visual feedback, and mental representation. LCD not only considers ease-of-use issues, but also the learner’s need for comprehension and building expertise (Soloway, Guzdia and Hay 1994). Achieving the balance requires the appropriate use of scaffolding. Scaffolding is a design strategy to support the learner in developing a new work practice. Scaffolding provides support for adaptive interfaces that can accommodate a diversity of learners.

LCD is unique because the target audience is learners as opposed to end users. “In descriptions of user-centered design, there is an implicit assumption that the end user of a computer tool already poses some measure of expertise about the work activity using the tool to engage in (Quintana, Krajcik and Soloway 2000 p. 257). Designing for learners is different from designing for end users because learners do not have domain knowledge, they are not motivated the same way, and they are more diverse. Designing for learners is in some ways more complex then designing for end users because learners do not share a work domain and often have less motivation than end users who are financially rewarded for completing tasks.

LCD further separates itself from other user oriented design methods because of differences in terms of execution, evaluation and expertise. “The gulf of execution is the difference between the goals and the intentions of the user and the permissible action on the tool. The gulf of evaluation reflects the amount of effort the user must exert to interpret the physical tool state” (Quintana, Krajcik and Soloway 2000 p. 258). The goals of most user oriented design methods and specifically user-centered design are to minimize the gap between gulf of execution and gulf of evaluation that supports the Theory of Action. Supporting the Theory of Action requires the design to focus on providing the information to assisting in the completion of the task. These types of application design leverage the end user’s knowledge and are not concerned with teaching the end user, but merely getting the job done. Alternatively, LCD focuses on gulf of expertise and supports the Constructivist Theory. Supporting the Constructivist Theory requires the design to focus on developing an application to build knowledge. These types of applications must teach the end user how to do the task.
LCD’s unique perspective and goals has generated a perceived unique set of heuristics. LCD heuristics must evaluate the attainment of “showing real world representations, supporting metacognition and reflection, providing multiple representations of data and information and designing task-oriented interfaces” (Quintana et al. 2002 p. 614). However, these heuristics are not very different from the user-centered heuristics. According to Nielsen and Molich, there are four main design goals which are: match the system and the real world as close as possible, provide recognition rather than recall, and provide user control and freedom. (For a comprehensive list of usability heuristics see www.useit.com/papers/heuristic/heuristic_list.html.) Although the practitioners supporting the two different methods use different words the underlying evaluation goals are equivalent.

11.3 Technique and process summary

The LCD does not have a well-established documented process and the proposed process is not all that different from the user-centered approach. Quintana, Krajcik and Soloway loosely define the following steps that create the LCD process (Quintana, Krajcik and Soloway 2000):

1. Define the learning problem
2. Define the learners
3. Define the underlying focus
4. Establish the goals
5. Visualize the process

The step of defining the learning problem requires brainstorming and researching the selected topic. Once the learning problem is understood, consideration to the types of students that would be interested in the problem can be identified. This step also incorporates the use of observations of learners and provides insight into how the learners learn. The technique is referred to as learner needs analysis and is similar to the work analysis that is done in many of the other user oriented design techniques. Understanding the characteristics of the specific learners and their learning challenges assists in identifying the goals. The goals are further established in focus groups. The goals are then used to design and evaluate the appropriate scaffolding architecture. This is done with the use of storyboards and iterative prototypes that undergo expert reviews.
(Quintana, Krajcik and Soloway 2002 p. 83). The LCD method does not seem to go beyond the analysis and design phase.

11.4 Project team type
LCD teams are different from other user oriented design teams. Instead of using cross-functional teams, it uses a work expert team. LCD teams are made of “work experts (such as work professionals) to analyze and articulate the work practice and create a good work model. The LCD team also needs educational experts (such as teachers and educational researchers) to articulate ways of communicating with learners and guide them in making the conceptual shift from learner to expert” (Quintana et al. 2002 p. 615).

11.5 Application domain
LCD can be applied to projects where the goal is not merely to produce a system that accommodates an end user’s workflow, but rather to teach the learner work concepts. This will enable the learner to build the necessary domain knowledge.

11.6 Integration feasibility
LCD does not integrate easily into these traditional software design approaches. LCD requires a slightly different perspective because learners do not have work domain knowledge.

11.7 Advantages
LCD allows a design team to focus on the specific perspective of the learner. LCD goes beyond usability needs by exploring approaches to develop systems that support people who need to build knowledge in certain work domain. The distinction it makes between end users and learners is subtle but valuable for producing LCD tools. LCD produces systems that allow learners and teachers to take advantage of the computerization that supports a new approach to learning. Teachers can develop inquiry-driven, project-based, and technology-pervasive curriculums that will better prepare students for future careers.

11.8 Expectations
LCD currently follows the user-centered design processes, but it is not working well. Although learners are also end users and some of the principles of user-centered design may be borrowed, they must be altered in order to accommodate a LCD system design. “User-centered design guidelines are not sufficient to address certain unique needs of
learners, such as intellectual growth, diversity of learning styles, and motivational needs” (Krajcik et al. 1998 p. 4). LCD requires designers to focus on the unique needs of the learner such as growth, diversity, and motivation. These focus areas make the analysis of LCD more complex that requires design flexibility. Therefore, the design process must be much more flexible than other user oriented design methods. Although scenarios, contextual analysis, process analysis, workflow models, and GOMS have been used to assist teams design LCD systems, one must consider the right combination of these techniques in order to produce systems in a timely manner.

"Future research directions include: defining analysis techniques to observer work practices and develop domain models for design, formalizing methods for determining learner needs within a given work domain, categorizing and cataloging scaffolding strategies and LCD principles to share with the design community, and determining structured methods for assessing the effectiveness for learner-centered software” (Quintana, Krajcik and Soloway 2000 p. 263).

As more tools are being created for educational environments designers continue to modify user oriented design principles and approaches to address new learning situations. "In order to continue on the path toward meeting the goals of learning-centered design, more work is needed from the HCI community to develop better design methods, techniques, and examples that can be used by software designers to build learning centered software” (Quintana et al. 2002 p. 606). LCD techniques will evolve as the needs of the educational environment become more demanding.

Beyond these issues there remains a fundamental question: Is the “distinction between the notion of users and learners merely rhetoric or is there truly a substantive distinction being made” (Soloway, Guzdia and Hay 1994 p. 47)? If there is a difference, as Soloway, Guzdial, and Hay suggest, then additional research needs to be done to identify new ways to analyze learners so that systems supporting learning can be successfully implemented.

12. Usability Engineering (UE)

12.1 History

The first signs of UE can be tracked back to the human performance and design techniques applied during the Second World War. These UE techniques led to the
creation of more complex equipment that began to proliferate during the Industrial Era. These design techniques during World War II were primarily applied to airplanes.

John Bennett is credited with coining the term ‘Usability Engineering,’ in the mid 1980’s (Carter 2004). Jacobson further defines the UE method in his book The IBM Guide to User Interface Design (Butler 1996 p. 60). More recent authors that have contributed to the development and education of UE include: Deborah Mayhew and Jacob Nielsen.

12.2 Definition
UE draws from many different disciplines. UE is defined as a “multidisciplinary field that developed out of human factors engineering (HFE) and into the design of computing systems. Its science base is largely in the experimental psychology of human information processing, but UE is most often applied to software engineering” (Butler 1996 p. 61). UE is also defined as a “discipline that provides structured methods for achieving usability in user interface design during product development” (Mayhew 1999 p. 2). UE extends the User-Centered Design method by applying a structured process to follow.

UE should not be confused with usability testing. “Usability engineering is not a one-shot affair where the user interface is fixed up before the release of a product. Rather usability engineering is a set of activities that ideally take place throughout the [project] lifecycle of the product, with significant activities happening at the early stages before the user interface has even been designed” (Nielsen 1993 p. 71). UE provides techniques to support the entire software product develop cycle.

12.3 Technique and process summary
Mayhew and Nielson both define the UE process similarly but each of them place focus on different tasks within the process. Mayhew believes that the tasks of UE fall under three major phases of a software project which are: requirements and analysis phase, design, testing, and development phase, and installation phase. She then provides the details about the deliverables within each of the phases. Nielsen, on the other hand, defines the UE process with specific steps. The UE process discussed here is based on Nielsen’s definition and incorporates the deliverables that Mayhew defines. Nielsen has established the following steps of the usability engineering lifecycle model

1. Know the user
a. Individual user characteristics
b. The user's current and desired tasks
c. Functional analysis
d. The evolution of the user and the job

2. Competitive analysis
3. Setting usability goals
   a. Financial impact analysis
4. Parallel design
5. Participatory design
6. Coordinated design of the total interface
7. Apply guidelines and heuristic analysis
8. Prototyping
9. Empirical testing
10. Iterative design
    a. Capture design rationale
11. Collect feedback from field users (Nielsen 1993 p. 72)

The steps are presented here linearly for the sake of discussion. However, many of these steps can happen in parallel. Nielsen explains that not every UE step needs to be completed in order to be successful in UE. Prioritizing UE activities can provide the desired benefits. “The least expensive way for usability activities to influence a product is to do as much as possible before design is started, since it will then not be necessary to change the design to comply with the usability recommendation” (Nielsen 1993 p. 72).

Nielsen explains that marketing groups through market research perform many of the pre-design activities. If usability practitioner collaborates with marketing this effort can serve both groups. However, Nielsen believes that the marketing definition of the end user group needs to be extended by the design team. The end users must include the people who actual use the system and the people who receive output from the system. Knowing the audience helps the design teams understand how end-users work and learn. With this knowledge, the design team can build a system that fits the end user’s mental model. Nielsen advocates interviews and observations to learn about end users. He stresses the importance of having end users not only tell, but also show the design team the tasks necessary to complete the work. It is also important to find out why tasks are performed in a certain way. Many times the answer is “Because that is the way it has always been done.” Comments similar to this indicate that there is an opportunity to provide a more efficient process. A new system should not propagate the way things
have always been done, but provide an easier, quicker and more intuitive way of completing a task that has customarily been complicated, slow, and convoluted.

One of the techniques used to capture information about the end user is called a user profile. The user profile describes the intended user group’s characteristics. In order to complete the user profile the team must know who will use the product. Often the marketing department has information that can be leveraged for the initiation of this task. The user profile includes the attitude, motivation, skill set, frequency of use, age, gender, and other physical characteristics about the end user group. If several end user groups intend to use the product, a user profile for each category should be created (e.g., teachers, students, parents). These user profiles can be reused for products intended for the same user group, but user profiles that are more than two years old should be revalidated because of the dynamic nature of end users (Mayhew 1999).

After the user profiles are created, task analysis can begin. Task analysis is a technique that helps team members identify the workflow and understand the ‘pain points’ within the current workflow.

“A typical outcome of task analysis is a list of all the things users want to accomplish with the system (the goals), all of the information they will need to achieve these goals (the preconditions), the steps that need to be performed and the interdependencies between these steps, all the various outcomes and reports that need to be produced, the criteria used to determine the quality and acceptability of these results, and finally the communication needs of the users as they exchange information with others while performing the task or preparing to do so” (Nielsen 1993 p. 75-76).

Task analysis often leads to work reengineering. Work reengineering is typically done by creating a workflow model that defines the current workflow and is then analyzed so that a new more efficient workflow can be created.

Another step in Nielsen’s lifecycle is competitive analysis. Competitive analysis determines how established functions are implemented and then improves upon those functions. “It is desirable to analyze existing products heuristically according to established usability guidelines and to perform empirical user tests with these products” (Nielsen 1993 p. 79). Taking advantage of an already implemented product provides much of the same benefits as post-deployment feedback. It is easy to see where things
can be improved when there is an existing tool. Nielsen asserts that he is not suggesting the use of competitive analysis to steal copyrighted user interface designs. Competitive analysis is used to see what functions may be missing because the new system may have discreetly changed the work process.

Usability goals are often a natural outcome during the user profiling, task analysis and competitive analysis. These goals should be captured as the analysis is going, on however an explicit task to fully define the goals and their priorities is required. Usability goals are measurable expectations that define success criteria. Attaching usability goals to time and cost savings will help managers see its benefits of designing a user-centered application. Nielsen explains that, because usability is multi-dimensional and sometimes includes conflicting goals, not all usability goals can be given equal priority on a given design project. This makes it important for teams to identify usability goals and assign them a priority.

After completing the initial analysis stage, the design stage can being. Nielsen suggests initiating the design stage with a technique called parallel design. Parallel design is a technique where each team member including the end user creates a design independently. The team members then join and review the different designs. This task allows for the brainstorming of alternative design ideas and lets the team combine ideas, which leads to better overall design. Parallel design prevents interaction designers from feeling too much ownership of the design, which hinders their objectivity later in the UE process. However, parallel design may not work as well for people lacking a design background or creativity and therefore, an alternative technique that can be used is participatory design. In fact, parallel design may not provide the necessary value if there are competitive products available.

Participatory design usually follows parallel design, but it may also be an alternative task to parallel design. However if the product is new, completing both activities provides a solid conceptual design. The advantage of participatory design is that it gives end users the opportunity to influence the software application design. Similar to parallel design, this task may not be successful where team members are not open to trying innovative techniques. This technique may be difficult to implement in organizations new to user oriented design techniques because the end users are not
familiar with this type of participation. The end users are now partially responsible for the design and that can be an intimidating responsibility. A common reaction for first time end users participating in this task is: “I thought this was a software engineer’s job, I’m not a designer.” This is in a true statement to some extent. One of Nielsen’s design rules is that end user’s are not designers (Nielsen 1993 p. 89). Therefore, it is not reasonable to expect them to come up with ideas from scratch. An initial approach may be to ask end users to describe what they would like the application to do while the interaction designer does the drawing. There are different levels of implementing participatory design, and this is where knowing and understanding the organizational culture and the end users will aid in implementing the right degree of participatory design.

The next step is to consolidate the designs created during the parallel and participatory design phases into what is often referred to as a conceptual design. Initial conceptual model mock ups are paper and pencil prototypes of the screen layout. These prototypes are intended to facilitate discussion for alternative design approaches. During the conceptual design and detail design step it is important to capture the design rationale. The design rationale is the reasoning behind the selection of one design alternative over another because as detail design moves forward the design rationale keeps the design from moving outside the scope of the intended solution. The design rationale becomes obvious when scenarios are applied to the alternative designs. The combination of scenarios and design rationales can help the design team understand why one design approach is preferred over a different design approach. As consensus is established and the design iterations progress, the conceptual model becomes more detailed and eventually defines the software applications foundational design. “The purpose of a conceptual model design is to define a coherent, rule–based framework that will provide a unifying foundation for all the detailed interface design decisions to come” (Mayhew 1999 p. 188). The conceptual design sets the scope boundaries in terms of the software application design

Once a conceptual design is created, the detail design effort must be carefully planned. Coordinating the design effort can be challenging and, if it is not planned well, the risk of end product inconsistencies and integration problems increases. Establishing
guidelines and heuristics are two techniques that can be implemented prior to the design phase to prevent inconsistencies. The chart in Figure 10 provides the description of the deliverables that should be in place prior to the detail design stage. In addition, during the detail design stage end users evaluate the design at logical points. Empirical testing is used to guide the iterative design process. Establishing severity ratings to help prioritize feedback from the evaluations is necessary to prevent over testing. Nielsen suggests that most usability problems are found after conducting a single usability test. Therefore, testing should not continue to the point that the value added becomes negative. By the end of this stage, the detail user interface design should be completed and ready for initial coding. The deliverables are described in the chart in Figure 11.

<table>
<thead>
<tr>
<th>Deliverable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform capabilities and constraints</td>
<td>This document explains what technology will be used and the functionality limitations and opportunities.</td>
</tr>
<tr>
<td>General design principles</td>
<td>This document defines software application interface rules. For example, it defines if the interaction style is touch screen or mouse driven.</td>
</tr>
<tr>
<td>Screen design standards</td>
<td>This document defines what widgets will be used for different types of information, when dialog boxes should be used, and how these dialog boxes will be respond to end users interaction. These standards define the general look and feel of the new software product. Industry and corporate standards must be given strong consideration.</td>
</tr>
<tr>
<td>Style guide development</td>
<td>This document is the consolidation of the analysis, conceptual design and screen standards for a specific software application.</td>
</tr>
</tbody>
</table>

**Figure 10 – Summary of Pre-Detail Design Deliverables** (Mayhew 1999)

<table>
<thead>
<tr>
<th>Deliverable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail user interface design</td>
<td>This document provides the user interface design specifications.</td>
</tr>
<tr>
<td>Usability feedback</td>
<td>The feedback is collected through the use of formal usability testing, questionnaires, interviews, focus groups, and usage studies. It is then summarized in a single document.</td>
</tr>
</tbody>
</table>

**Figure 11   Summary of Detail Design Deliverables** (Mayhew 1999)
Nielsen and Mayhew both advocate the use of follow-up studies after system installation. This process is inexpensive, yet valuable. Follow-up studies use the same techniques as end user feedback during the analysis phase. Follow-up studies can help a team prepare for the next product release even if the release has yet to be scheduled.

12.4 Project team types

UE works best with cross-functional teams. Ideally, in a cross-functional team all members work together on all tasks. However, the reality is that tasks are split up among team members with specific areas of expertise for more rapid results. Furthermore, Mayhew explains that the UE method can be modified based on available skill sets. For example, if software developers have UE experience, they may be able to complete some user interface designer tasks.

12.5 Application domain

Although the UE method is typically used when developing office software applications and works exceptionally well when reworking existing software products, the method can also be applied to more specialized software application development. UE can also be applied to an organization's internal use software application or for the creation of commercially used products. Mayhew explains that UE can be used on software for factory equipment, medical technology, and scientific data-gathering instruments (Mayhew 1999 p. 5).

UE is a very flexible method that can be modified for small or large projects for all ranges of complexity. UE can be applied to both contract development projects where teams are decentralized or in an environment where the team is centralized. UE is not so much dependent on the project type but rather on having a defined scope and an established time line. The project scope and time line provides insight into the selection of appropriate techniques. UE can be modified to fit into any time frame and any sized budget. Mayhew explains that the full UE method is only appropriate for complex projects that have high visibility and high potential of usability payoff. Complex, long-term projects where team members are geographically dispersed require documenting tasks and results; therefore, short-cut techniques may not be permissible. However, she
suggests for small simple projects that use minimal resources to apply short-cut techniques to meet the specific project needs.

There is no one approach to design that works for every project. UE is no exception. UE is not good for a reengineering process where the goal is to identify opportunities for new software development. Mayhew admits that Contextual Design is a better approach for those types of projects. She also believes that Contextual Design is a better approach when the project goal is to create new products in which goals are not defined prior to the analysis. Mayhew further stipulates that when the scope of a project is unknown, combining the Contextual Design and UE methods may provide the best results because the Contextual Design method leaves off where the UE method picks up.

“If you do have the opportunity or need to do major business reengineering before the development of specific automated products, and you have used Beyer and Holtzblatt’s Contextual Design technique to do this, I might point out that their technique leaves off just where actual user interface design begins in the Usability Engineering Lifecycle” (Mayhew 1999 p. 67).

This means that Contextual Design could be used for the front end of the process followed by UE starting with the conceptual design model.

12.6 Integration feasibility

Integrating the UE process can begin with subset of techniques that provide organizations with a starting point for implementing UE. Nielsen suggests starting with just a few of the easier UE techniques which are: visits to user sites, prototyping through scenarios, evaluating low fidelity prototypes by asking the end user to think aloud, and conduct heuristic evaluations (Nielsen 1993 p. 112). These techniques will provide the most visible benefits when initially implementing UE and are considered ‘discount usability’. This incremental approach to UE makes it easier for an organization to integrate at a comfortable pace.

According to Mayhew, when introducing UE it is easier to learn basic design principles and conduct simple usability testing prior to expanding the organization’s knowledge of the more rigorous UE techniques that include user profiles and contextual task analysis. Mayhew also suggests introducing UE techniques from the backend of the process starting with usability testing because this technique can provide statistical and
visible results to management and software engineers. This information can be used as a selling point for one design approach over another. Conducting a “usability test on a high-visibility project that clearly demonstrates dramatic usability problems can also be a powerful motivator” (Mayhew 1999 p. 423), which sets the stage for future projects. This is not to say that other UE techniques are not as important; however, implementing a new process such as UE requires office politicking. Winning the political battle with the right tools will insure selection of the entire process.

UE integrates effectively within a software development organization because UE is an “engineering-like process” that produces statistics. UE also integrates well with the Object Oriented Software Engineering approach to software development because “Usability Engineering and OOSE are not competing approaches, but rather are potentially complementary approaches to software development” (Nielsen 1993 p. 30). The structured, analytical, and statistical aspects of the UE method provides a convincing approach that includes techniques that can easily be measured in terms of cost benefit.

12.7 Advantages
UE is more than a design method; it is a well-defined end to end process for developing software. It bridges knowledge gaps between the analysis, design, development and end user’s. UE provides a technique that can be applied at every stage of the software development lifecycle. The techniques and alternative short-cut options make UE flexible and scalable. According to Mayhew, “The key to the general applicability and flexibility of the Usability Engineering Lifecycle lies in the choice of which techniques to apply to each task, not in the choice of which tasks to carry out” (Mayhew 1999 p. 20). She explains that all tasks should be carried out to achieve the best possible opportunity for creating a usable product; however, she also understands real world constraints and therefore provides suggestions for short-cuts that will accomplish similar results.

Mayhew categorizes the specific benefits of UE for each group in the software development organization. The chart in Figure 12 summarizes these benefits.

12.8 Expectations
UE has opportunity for additional growth and development. UE lacks the design specification tools that make it easy to document design ideas in different formats (e.g., text, graphics, flowcharts, table) and in a linked manner so that they can be easily updated
during iterative design (Mayhew 1999 p. 280). Many of the UE deliverables evolve simultaneously with iteration of a task, thus, on large projects, the documentation can be difficult to manage (Mayhew 1999 p. 318).

The UE process will continue to evolve as it is used for different types of projects. The hope is that additional short-cuts will be developed for the requirements analysis phase because this step is time consuming and resource intensive. In addition, as the educational curriculums educate and advocate UE as a process it will become easier for organizations to implement because employees will have the necessary educational background.

<table>
<thead>
<tr>
<th>Organizations Group</th>
<th>Benefit</th>
</tr>
</thead>
</table>
| Benefits of UE to high level management: | Usability data can improve marketing literature, influence early adopters, and convince potential users that training costs will be low. |}
|                                   | Usability Engineering can reduce development costs and minimize distribution and support costs.                                                                 |
|                                   | Usability Engineering can reduce project risk.                                                                                                           |
|                                   | Usability Engineering can reduce time wasted in redesign.                                                                                               |
| Benefits of UE to project managers | Usability Engineering can reduce overall development time.                                                                                             |
|                                   | Usability Engineering can provide an objective way to identify and prioritize problems so they can be fixed early at minimum cost.                         |
|                                   | Greater profits due to more competitive products/services.                                                                                               |
|                                   | Decrease overall development and maintenance costs.                                                                                                      |
| Benefits of UE to software engineers | Decreased customer support costs.                                                                                                                         |
|                                   | More follow-on business due to satisfied customers.                                                                                                       |
|                                   | Usability Engineering is objective and unbiased.                                                                                                         |
|                                   | Usability can be measured just like other engineering attributes.                                                                                         |
|                                   | Usability Engineering provides for creativity in the design process and provides a framework to structure design and allows a team to design quickly.       |

**Figure 11 – UE Advantages** (Mayhew 1999 p. 2, 421-22)

13. **Fictional Case Study Purpose**

The purpose of this case study is to illustrate what aspects of a project need to be considered when selecting a UOD method or combination of methods. The case study is based on a fictitious company, SMS Software Inc., with fictitious team members. The team members do not directly correlate to any particular individual. However, the
foundation for this case study is based on real life experiences from a variety of projects. The characteristics of this case study have been carefully put together to illustrate the rationale for selecting a user oriented design approach.

13.1 Formulation of Problem Statement

Historically, organizations have not explicitly stated that the defined initiative would produce a usable and useful product for the marketplace. Typically, marketing has dictated what the product would do and the software engineers have decided how it would do it. Furthermore, companies often view the consistent involvement of end users as costly and time consuming and therefore, have tried to minimize end user involvement. However, this shortsightedness has cost companies from staying competitive in today’s rapidly changing technological environment. That is, the introduction of new or modified software tools changes the end user’s environment and, unless there is constant and consistent end user feedback, opportunities will go unrealized. Thus, selecting the most appropriate user oriented design approach can make the difference between product development being successful or unsuccessful.

However, user oriented design selection is challenging because of the number of methods from which to select. Not only do project managers need to consider what project lifecycle methodology to use (i.e. waterfall, modified waterfall, spiral, incremental, etc), but also select the appropriate user oriented design techniques. In most companies there is limited knowledge about user oriented design methods and, therefore, it is common for an organization to apply the most publicized user oriented design technique. For example, usability testing has recently become very popular with organizations.

Unfortunately, when companies decide to implement a user oriented design method they typically select one method and expect that method to work for all projects. Since user oriented design is not a one size fits all approach, it is difficult to implement a sustainable user oriented design approach. A user oriented design approach is like many other tools in Information Technology (IT) in that it must be customized to meet project needs. The case study and analysis presented in the following sections attempts to bridge the knowledge gap about user oriented design technique selection.
13.2 Background Scenario

SMS Software Inc. is a publicly held medium-size software application development company with a focus on offering a complete suite of software and services that support online teaching. SMS offers several collaborative software tools including email, content management, and course management tools to meet the needs of each of the following market segments:

- Higher Education
- K-12
- Corporate/Government

SMS Software Inc. is located in Rochester, NY, and was founded in September, 1990 by Susan Saeger, who serves as President and CEO. SMS has approximately 100 employees. SMS continues to expand operations at the Rochester, NY headquarters.

The IT staff includes: process analysts, software engineers, quality assurance specialists, and implementation specialists. The IT staff historically used a structured analysis method and is familiar with tasks on arrow models, data flow diagrams, state transition diagrams, and logical data modeling. However, there is an effort underway to add a new focus area to the IT staff that would provide a usability perspective.

SMS Software Inc. has acquired WhiteBoard Inc. and Educational Tools Inc. within the last year allowing the company to increase its customer base and expand the company’s software product offering. However, due to these acquisitions, the company has faced increasing costs in software development, maintenance, and training. The acquisitions have also required additional employees to create documentation materials. These cost increases are making it difficult for SMS to stay competitive in the current economically stressed environment. Due to this situation, the directors of SMS Software Inc. are looking for ways to cut costs and maintain profits.

After several brainstorming sessions with all levels of management, the directors determined the best way to cut costs is to consolidate the two acquired teaching software applications and the company’s legacy collaborative teaching application into one modular software product. This would enable the IT staff to maintain the application software on one platform. Thus, the consolidated product would allow the company to continue to participate in its existing target markets while decreasing support costs. The
directors determined that the consolidation effort would need to be completed within the next eight months to maintain profits.

The directors made the request that received mixed responses. The software development manager was excited because the product would need to be written in Java giving the software engineering staff an opportunity to upgrade their skills. However, he was concerned that an eight-month period would not accommodate appropriate training need for the software engineers to be successful.

The marketing department was concerned about the potential to lose clients if the consolidation product functionality failed to encompass the functionalities already existing in the three separate products. The training and support departments worried that consolidation would lead to job cutbacks.

After listening to the company’s various constituents, the directors selected a project manager to address and manage all the concerns except potential job dislocation. The directors agreed to work with the impacted management team to create a job relocation plan outside of the scope of the project.

While consolidation discussions were underway, the IT division was working on creating a new department called the Usability Design Department. The manager of the new department campaigned to convince the project manager and directors that a user oriented design approach for this effort would incorporate the voice of the end users, thereby allowing the company to stay competitive. The IT manager also explained that the new group would create design specifications, a job traditionally reserved for software engineers. This would benefit the software engineers because they could then focus on their core competency: system coding. They would also have more time to learn Java techniques. The project sponsor agreed to use a user oriented design approach, as long as it would not elongate his forecasted timeline.

The Usability Department was then staffed with people from inside the company who had knowledge of the current systems, but had little to no knowledge of user oriented design methods. Because this was a high-profile strategic project, a consulting group was contracted to coach the employees through their first project. Deciding on a consulting group took place in parallel with the creation of the Consolidated Collaboration Teaching Application (CCTA) Project Definition and was not part of the
project timeline. It was decided that the consultants would assist in the analysis and design efforts, but would not perform any of the actual software development.

In parallel with the user oriented design consultant selection, the project manager made resource request to the necessary functional managers. The team included a usability engineer, a software engineer, a quality assurance specialist and an implementation specialist. As team leaders, these individuals would perform most of the work, but they would also have the authority to allocate work to members within their functional areas if additional resources were required to complete tasks on time.

The project team would be centrally located in the Rochester office, and included the following team leaders:

- Penny – Project Manager
- Unice – Usability Engineer
- Conrad – Usability Engineer – Consultant
- Savannah – Software Engineer
- Quinn – Quality Assurance Specialist
- Irene – Implementation Specialist

The following provides a brief description about each of the team members:

- Penny, the Project Manager has been with the company since it started in 1990. She is a senior project manager and led five large-scale company projects as well as several smaller projects. Prior to joining SMS, she worked as a project manager for a home construction company. Penny is well-respected by all levels of management. People do not often question Penny’s judgment.
- Unice, the Usability Engineer has been with the company since June 1995. He started as a telephone support representative and in August 1996, he became a Process Analyst. He was recently asked to be part of the newly created Usability Department. Unice can be very opinionated.
- Conrad, the Usability Engineer who works for UOD Consultant Inc. has been with his company for 15 years. Conrad has never worked with any of the
other project team members. However, Conrad is very easy-going and flexible.

- Savannah, the Software Engineer has been with the SMS Software Inc. since it started in February of 1990. She is a senior software engineer and participated in eight large-scale projects and several smaller efforts. Prior to joining SMS Software Inc., she worked for a large Fortune 500 software engineering company. Savannah is well respected by all levels of management, and her opinions are not usually challenged. Savannah is known for getting the job done on time and within budget.

- Quinn, the Quality Assurance (QA) Specialist has been with the company since June, 1990. She was hired into the training department. In May, 2000 she transferred to the QA department. Quinn is very detail-oriented.

- Irene, the Implementation Engineer has been with the company since June, 1990. She was originally hired as a QA. In June of 2000 she transferred to the Implementation Department. Irene is known to be easy-going.

These team members were selected because they had proven their ability to embrace new tools and ideas on previous projects along with a dedication to getting the job done. These team members are considered 'star' employees.

Prior to the kickoff meeting, Penny, the project manager, and the marketing manager worked on the Project Definition Document that included project goals, and the project's basic features, assumptions, constraints, capabilities, inclusions, exclusions, and methodologies.

During a friendly lunch, Penny and Savannah discussed the appropriate project lifecycle. They both agreed that using a Spiral Lifecycle Project Methodology would keep the work visible to both the project sponsor and the board of directors. In addition, they agreed that, because this is a high-profile and strategic project with a short timeline and limited resources, it would be best to break the work down into reasonable components. Components could be assigned to defined iterations to ensure project visibility.
At the kickoff meeting, all of the team members attended. The team reviewed the Project Definition. Penny explained the Spiral Lifecycle Project Methodology would be used in tandem with a user oriented design approach that was not yet completely defined. Penny further explained that the intention of each spiral was to add functionality, not to redesign the previous iteration. The completion of an individual iteration would represent a major milestone, and within a single iteration, the team would define miniature milestones.

Penny explained she would be meeting with each team member individually after completion of the conceptual design to define miniature milestones as well as the supporting tasks needed to create the initial project plan. She further explained that miniature milestones are defined as tasks that can be completed in less than five business days. Defining the miniature milestones would allow the project manager to remove roadblocks.

Penny then explained that the initial iteration would focus on the analysis and design and evaluation of the conceptual design based on prototypes created by the new Usability Design Department members (see Appendix B for usability task plan). Unice then expounded on the conceptual design technique. He stated: “The conceptual design will define the breadth, but not the depth of the application, thus illustrating the project’s anticipated functionality. However, the actual detail about the functionality will be defined within the analysis of the iteration for that function.” Penny noted that the conceptual design would be completed while the software engineers went to Java training to minimize disturbances to the project schedule.

After the kickoff meeting Unice and Conrad, discussed the user oriented design approach details. The following section explains the reasoning and final recommendation for a UOD approach.

13.3 Criteria for technique selection
As mentioned, it is unlikely that one method will completely fit the needs of a given project. However, because companies typically look to consultants when implementing a new technique and the consultants typically align themselves with one method, consultant selection often dictates method selection. However, Conrad is an independent usability specialist, and therefore not aligned with any one method. Many organizations fall into
the trap of selecting a method, rather than selecting techniques that facilitate the goals of the project. Selecting the appropriate UOD techniques requires careful consideration to specific project characteristics. These include the project:

- goal (purpose) and scope.
- timeline.
- team size.
- team experience with the tools being used.
- team type (cross-functional versus job specific).
- team location (centralized versus non-centralized).
- general team dynamics (individuals experience working with each other, use of contracted resources, and openness to trying new things).
- organizational culture (general attitude about: the work environment, the process and procedures to get the job done, the customers, each other, and management).

Articulating the goals and scope of a project assists in identifying the appropriate techniques because each user oriented design method and individual technique has been created to facilitate a solution. Therefore, by applying the right tool for the defined problem will enhance the probability of a successful design.

The next few project characteristics to consider are the project timeline, team resources, and the experience level of each individual. These characteristics must be considered together because each inherently impacts the other and ultimately the sum of these three characteristics dictate the actual timeline. Although most user oriented design methods are flexible enough to use with even the shortest amount of time, it is important to evaluate which techniques are going to provide the most value when timelines are stringent. In addition, the individual team members experience and background knowledge of user oriented design techniques will affect the amount of time needed to complete a task using a given technique. If a user oriented design technique is complicated, but provides the necessary value, then it may be worth the effort to learn and use the technique. However, if the user oriented design technique cannot be learned quickly and interferes with information gathering there is no value in using the technique.
In general, good project management techniques consider the number of new tools to be used in a project and either adjusts the timeline accordingly or discourages the introduction of the new technique within the given project. Even when ‘just in-time’ training is provided for the use of new tools or techniques, additional time must be factored into the project timeline. Furthermore, introducing too many new tools or techniques on a project can also have a negative impact on team dynamics. For example, if the software engineers are struggling to learn Java and expected to deal with a new form of analysis that produces different deliverables, tension can arise between the analysis team and developers. It is important to understand the impact of the new tool on the entire project team and not just on the team members directly responsible for learning the tool or technique. Knowledgeable consultants and contractors can help to minimize the tension that is common when team members are learning new tools and techniques.

However, the use of consultants and contractors brings about additional risks. For example, because consultants may not be familiar with the company culture, they may suggest inappropriate techniques or tools, thus creating unnecessary tension amongst team members. Furthermore, when timelines slip, consultants or contractors may switch roles and become full time team members as opposed to peripheral coaches. This may help the immediate needs of the project team, but, when the engagement is over the team may be left with a knowledge gap. However, drafting a mitigation plan can significantly lessen this risk.

Team type is the next logical team characteristic to consider. Most organizations will claim to be using cross-functional teams. Although this may be true, there are varying degrees of cross-functional team implementation. The reality is that typical software application project team members are from the IT organization and all others act as tangential team members who are not always included in regular team meetings. In addition, in true cross-functional organizations, IT team members are not the sole decision makers; when a decision needs to be made, tangential team members must be included.

The next project characteristic to consider when selecting a user oriented design method and techniques is the project team location. Although, the team location does not eliminate any of the user oriented design methods, it needs to be considered so that the
appropriate tools can be put into place to manage project deliverables. For example, when considering usability testing, a geographically dispersed team may provide access to wider and more representative end user spectrum. This is viewed as a usability advantage because a diverse end user population will provide more statistically accurate usability testing results.

The last project characteristic to consider is general team dynamics. Team dynamics are the often unnoticed 'natural forces' impacting a team’s interaction style. For example, if a strong friendship exists between two team members on a six person team, their 'natural force' may positively or negatively influence the rest of the team. The positive affects of a friendship within a team is that it creates an enjoyable work environment and seamless communication. However, such a friendship might exclude other team members, who will be less willing to participate in discussions. This will result in two factions within the team, causing disjointed communication and poor team performance. Thus, a team that enjoys working together is more likely to embrace the challenges of new tools and techniques.

13.4 Analysis of the CCTA Project

This section applies the criteria discussed in the previous section to the Consolidated Collaborative Teaching Application (CCTA) Project. Specific project characteristic analyses provided the basis of the final UOD recommendation.

13.5 CCTA Project Goal and Scope

The case study project goals are functionality and platform consolidation. As stated, the company currently supports three different collaborative teaching applications. The goal is to combine them into one flexible, but useable application. When functions overlap, the team must determine the best approach to incorporating that function. However, no additional functionality will be provided within the scope of the project.

13.6 CCTA Project Timeline

According to the directors, the project must be completed in eight months. This timeline allows for approximately 12 weeks to be spent on each applications redesign (36 weeks/3 applications). At first glance, the timeline seems reasonable; however, the addition of a new programming language and design approach makes this timeline aggressive.

Because the software engineers will be in training during the first month of the project,
the analysis and design team members will use that month to gather information. The goal is to have the entire breath of the new system defined at a high level.

13.7 CCTA Project Team Size and Experience

Although the project team appears small, it is important to remember that the case study is only focusing on the team leaders and that additional people resources may be used to get the job done. Therefore, the team size is flexible and does not really need to factor into the evaluation of the selected technique.

Team leaders and the adjunct team members’ experience with the user oriented design techniques must be carefully considered. Recall that the project team members have no experience with user oriented design techniques. However, they are familiar with structured methods for analysis. All of the team members are familiar with task on arrow models, data flow diagrams, state transition diagrams, and logical data modeling. Therefore, it may be helpful to select some user oriented design techniques that use similar deliverables. A consultant was brought in to keep analysis and design moving quickly. The use of a consultant as a coach on an actual project allows team members to learn the practical application of the technique in their own environment, and keeps the training in the context of the team member’s work. However, the use of consultants also entails a measure of risk.

Additionally, one must consider the lead software engineer’s limited experience with Java. Although the adjunct software engineers may have some experience with Java, there are no Java experts within the organization.

13.8 CCTA Project Team Type

A team such as the one in the case study is a modified cross-functional team. As evidenced by this scenario, cross-functional teams are given broad as opposed to specific objectives. Decision-making within a team is usually based on consensus. In this case, the project sponsor has already decided the scope of the project; thus, much of the decision-making capabilities have been taken away from the project team. Also, each person on this team will conduct his or her individual functional task. For example, Unice will only lead the user oriented design functions and not be expected to code and the Savannah will code and participate only minimally in the interface design. There is a distinction that needs to be made here: the software engineer designs the code to provide
the most efficient utilization of system resources. The software engineer is not responsible for the user interface layout because that is the responsibility of the usability engineer. However, for the optimum design and most efficient work practice the usability engineer should be reviewing design ideas with the software engineer.

13.9 CICTA Project Team Location
The team members in the case study are centrally located which will simplify communication. However, this also means that only the end users within a one-hour radius of the Rochester area will be able to participate. There is no money in the budget for traveling and, although there are tools available for remote communication, there is no time or money to introduce another new technique or tool.

13.10 CICTA Project Team Dynamics and organizational culture
The CICTA team’s major advantage is that they have all worked together on previous projects and are familiar with the current software applications. The case study also alludes to the fact that the organization culture is somewhat conservative when trying new processes. However, it points out that these team members have been selected because of their willingness to try new things and because they have already built a rapport with each other, which should help the project progress with minimal personality conflicts that can slow things down. The case study also suggests that Penny and Savannah are friends. This friendship must be kept in check to avoid the formation of potentially divisive subgroups.

13.11 User Oriented Design approach recommendation
The recommendation was made to move forward with a combination user oriented design method approach because of the project scope, timeline, and experience level. The combination approach will include techniques from the following methods:
  - Contextual Design (CD)
  - User Centered Design (UCD)
  - Usability Engineering

The following few sections explain why the other methods have been eliminated:

13.12 Participatory Design (PD) elimination explanation
The primary reason for the elimination of PD is because the organizational culture is not right for a true PD approach. PD is a method that requires a unionized organization that
is not present within SMS Inc. or within the educational environment. PD requires that
the end user fully participate in the project, which is not a practical expectation for this
endeavor. The project sponsor agreed to allow end users to participate under the
condition that their involvement did not decelerate progress. In addition, if the project
team has to wait for the end users availability to make decision, valuable time could be
lost.

Although PD has been eliminated as a method, it has been recommended as a
technique within Usability Engineering. As a technique, the PD session can be carefully
planned and facilitated to produce valuable results in a reasonable amount of time.

13.13 Scenario Based Design (SBD) elimination explanation
SBD techniques have been eliminated for two reasons: 1) it is an unstructured method
with high risks and would not be well received by an organization that has traditionally
used structured analysis techniques for software application design. 2) SBD techniques
focus on defining the tasks that need to be supported within a new product design. In this
project, defining the end users’ tasks within this application can be shortcut because
existing systems already meet the majority of the end users needs.

13.14 Performance Centered Design (PCD) elimination explanation
The primary reason this method has been eliminated is that all of its techniques are
included in User-Centered Design and Usability Engineering. The secondary reason is
that the goal of PD is to ensure day one performance for end users knowledgeable about
specific work domains. This philosophy did not seem to match the CCTA’s project goals
of consolidating applications. Although day one performance is important, learnability is
more reasonable usability goal for this project.

13.15 Learning Centered Design (LCD) elimination explanation
LCD has been eliminated because, similar to Performance Centered Design, it is no
appropriate for the goal of this project. Furthermore, many of the techniques in LCD
have been borrowed from other user oriented design methods. Finally, because the
project must build a collaboration tool, the scaffolding design that LCD techniques
facilitate will not be an integral part of this product.
13.16 User Oriented Design (UOD) technique recommendation analysis

Because the various methods share similar techniques, they will generate similar information. When techniques provide overlapping information, priority is given to the technique that is easier to learn for first time users. However, if there is familiarity with both techniques, then priority is given to the technique that will be more readily embraced by the organization’s culture or by the personal preferences and work style of the Usability Engineer.

The chart in Figure 12 summarizes the recommended user oriented design techniques for each stage of the CCTA project. The following sections will explain the reasoning for techniques selected and eliminated.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Technique</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Techniques</td>
<td>Functional analysis</td>
<td>UE</td>
</tr>
<tr>
<td></td>
<td>Focus groups</td>
<td>UCD</td>
</tr>
<tr>
<td></td>
<td>UED</td>
<td>Contextual Design</td>
</tr>
<tr>
<td>Design Techniques</td>
<td>Conceptual design</td>
<td>UCD</td>
</tr>
<tr>
<td></td>
<td>Low–medium fidelity prototypes</td>
<td>UCD</td>
</tr>
<tr>
<td></td>
<td>Design walkthroughs</td>
<td>UCD</td>
</tr>
<tr>
<td></td>
<td>Usability walkthrough</td>
<td>UCD</td>
</tr>
<tr>
<td>Test &amp; Measure Techniques</td>
<td>Usability test</td>
<td>UCD</td>
</tr>
<tr>
<td></td>
<td>Heuristic evaluation</td>
<td>UCD</td>
</tr>
<tr>
<td>New Deliverables</td>
<td>Platform capabilities and</td>
<td>UE</td>
</tr>
<tr>
<td></td>
<td>constraints</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Design Documents</td>
<td>UCD</td>
</tr>
</tbody>
</table>

Figure 12 – UOD Technique Recommendation for the CCTA Project

13.17 Analysis techniques included

Functional analysis is appropriate for this project because there are several existing systems that must be combined. This will expedite the understanding of the current system’s functionality and help prioritize its importance. Furthermore, if applications have overlapping functionality, it conveys to the project team that that a particular function may take priority because of the number of customers using it. If there is functionality in just one of the three applications, it may be worth asking the focus group
to determine the priority of the function. The deliverable of the functional analysis will be the user environment design model (UED).

The recommendation for focus groups instead of the contextual inquiry method suggested by the CD method is because focus group sessions provide information in a shorter span of time. In addition, all of the team members are familiar with collaborative teaching applications through either direct experience or observation. This means spending the time analyzing a familiar environment will not provided insight that is necessary to complete the design.

The user environment design (UED) model is a useful technique when the project goal is to consolidate existing software applications. The UED model provides a deliverable that assists the project manager plan a phased or incremental release. The UED allows the team to think about the functions that the system will provide and how these functions relate to each other without being bogged down in user interface design detail. The functions are then made available through menus, toolbars, and keyboard commands that can be considered later in the process. Because SMS must integrate three different systems, the UED model fits the required selection criteria.

13.18 Analysis techniques excluded
Some techniques do not overlap, but have been eliminated because the technique is not appropriate for the project characteristics. Two such techniques are the financial impact analysis and the competitive evaluation analysis. The project team will not complete the financial impact analysis because the directors have already identified this project as a cost saving effort. The competitive evaluation technique is out of scope for this project. Since the company has already acquired its competitors’ software, there is no need and no time for competitive evaluation.

The task analysis, hierarchical task analysis, use cases and OVID (object, view, and interaction design) techniques have been eliminated because there are already similar systems in place. The more appropriate technique is the (UED) combined with low fidelity prototypes. The UED is an appropriate technique when reverse engineering is being used to consolidate systems.

Contextual inquiry was eliminated as a technique because it provides the greatest benefit when the designers, software engineers and other technical team members are not
familiar with the work process and the end user’s environment. In addition, work modeling, consolidation, and work redesign will not be necessary because the goal of this product is not to reengineer a business process, but rather to consolidate applications.

The user profile technique has also been eliminated because of the use of focus groups. The user profile technique can be eliminated with little risk because the actual users will be participating in the new design and will be evaluating the design at every stage. Access to end users will be easy, however, due to the timeline, some of the end users feedback may not be implemented and the project sponsor is concerned that too much end user involvement will cause scope expansion and missed deadlines. These concerns will be abated with the use of strong project management techniques and continual reference to the UED and conceptual design.

Also, the formal collection of feedback from field users via surveys will not be necessary. This is because of the use of focus groups, participatory design, and usability testing that occurs during the analysis and design phase and will reveal similar information.

The final technique for consideration is the early ship survey. An early ship survey will be eliminated to allow more time to be spent on user evaluations.

13.19 Design techniques included
The conceptual design technique will leverage the information contained in the UED that was created during the analysis phase. The conceptual design will assist the project manager categorize the work. It will also help keep the project team on task and in scope. The conceptual model provides the analysis and design team members with a starting point to gather the detail design. The detail design is captured using focus groups and participatory design.

Participatory design will be used to further detail the design features and to redesign functionalities similar across products. This technique will be used in place of parallel design during the conceptual design stage because of the limited number of team resources available. Since the software engineers will be in training and because there is only a single usability engineer available for creating the conceptual design, a participatory design session provides the necessary brainstorming to quickly move the design in the right direction. In addition, since this project has the luxury of a consultant,
he can provide an excellent resource for teaching proper facilitating skills of a participatory design session. Learning to use this technique allows an organization to creatively use resources by asking end users to provide design ideas. The end users provide valuable information and actually assist the usability engineer do her job. However, as stated earlier these sessions must be carefully facilitated so that the end users do not become intimidated. It is important to realize when implementing this technique that historically end users have not even been solicited for input let alone be requested to participate in the actual design. Therefore, if this session is not facilitated properly, the end users may react poorly to the request to participate. In addition, there is a risk that the end user’s will lose confidence in the project team’s ability to design, especially when the user oriented design techniques are initially introduced if the participatory design session is not well facilitated. Participatory design is not appropriate for all projects because it requires a certain personality type. On this particular project, the end users are teachers and students who are, in most cases, used to open participation. The benefit of this task is that it facilitates the sense of ownership for the entire team, which builds excitement about the final product early in the project lifecycle.

Usability walkthroughs dovetail nicely with the use of focus groups. The usability walkthroughs require directed focus groups, where the groups review low or medium fidelity prototypes. The benefit of the usability walkthrough technique is that it provides early validation of design decisions. It is an inexpensive technique because it can be completed with paper prototypes and does not require a usability lab. However, this technique requires some amount of preparation it takes less time then formal usability tests.

Usability walkthroughs are also an appropriate technique to use if usability testing cannot be accommodated in the project schedule. However, if formal usability testing cannot be conducted a medium-fidelity prototype should be created for the design walkthrough so that the end users can better visualize the interaction techniques. The usability walkthroughs often provide more robust information than usability tests because end users are more comfortable within a group setting. The group situation removes the concern about being tested as an individual. In addition, one person’s comment can
trigger another person’s comment providing a wealth of user feedback. The trade off is that usability statistics are not generated and the benefit may be difficult to prove.

Design walkthroughs will be conducted for each component of the new system. This technique allows all members of the team, including system engineers, quality assurance specialist, training, support, end users and stakeholders, to review the design and provide feedback. It gives each team member the opportunity to peruse the design and builds an understanding of what will be provided in the final deliverable. This technique is best done with medium fidelity prototypes. In some situations, it may be better to conduct two different design walkthroughs sessions because stakeholders and end users do not always want the level of detail about a design that the technical staff requests.

13.20 Design techniques excluded
The parallel design technique is not appropriate for this project because there are already similar systems in place. Additionally, parallel design is typically used when a new design is being created and team members need to generate concepts for the new design. Since there are systems already in place, it is beneficial to use the focus group to evaluate what is and is not working and use that information as the starting point. In addition, since participatory design has been recommended, results similar to parallel design will have already been achieved, because both techniques facilitate design ideas.

The main difference between parallel design and participatory design is that team member’s complete parallel design independently. The end users may participate, however typically the end user would be given a rough design as a starting point. Once each team member has an idea, they meet to share their designs and determine what combination of ideas will produce the most useable and useful design. Alternatively, participatory design is done together as a team and includes the end users. Since most of the technical members on this project team are learning other new tools, it is not likely that this technique would be successful.

13.21 Testing techniques included
Usability metrics will define the criteria for testing the usability of the prototypes and will be used again to test the actual system. Examples of such metrics include time to complete a task and number of errors. Having criteria to measure success provides value
to the design team and to the stakeholders because cost benefit analysis can be done to measure the number of ‘fixes’ that were identified prior to coding.

Informal usability test is being recommended when the low-fidelity prototypes are available. Formal usability test is being recommended with the use the medium-fidelity prototypes that can be created using HTML or a familiar prototyping tool. Usability testing allows the team to observe if real users can easily complete real tasks. Usability tests are a forum of end user feedback. However, it is important the test users understand that not all of the feedback from the usability test will be incorporated into the final design. In addition, a natural but often unexpected outcome is that usability testing provides a level of training. The end users that test the system are often more comfortable on day one of implementation because they know what to expect. In addition, if the system functions were easy to remember there is a chance that some of the end users who participated in the test will remember how to use the system.

Heuristic evaluations will be done prior to the usability test so that the end users are not providing feedback on consistency issues that can cause the usability test to slow down. In addition, heuristic evaluations may be used as an alternative for the system components that are deemed low priority. Heuristic evaluations are a systematic inspection of a user interface design. It is also considered discount usability because it is inexpensive to perform if a knowledgeable usability engineer is on staff. It does not require any preparation time and it does not require a usability lab with end users. It simply requires familiarity with usability principles and the ability to evaluate a design against the established design principles. It is best to use more than one person when doing heuristic evaluation because one person will not find all the problems. This provides an opportunity for the user oriented design consultant to coach and complete a task at the same time. The goal of heuristic evaluation is to find the usability problems in the design so that they can be fixed. The tradeoff for a heuristic evaluation is that the end users are not providing the feedback.

13.22 Testing techniques excluded

None of the testing techniques that exist within the suggested user oriented design methods are being excluded, because the various testing techniques indicate the usability of the design. The selection of the testing method is typically based on the frequency of
The goal is to perform some type of test for each component to prevent surprises upon implementation.

13.23 Additional deliverables

The recommendation includes that in parallel with the techniques previously described, that a peripheral usability team member and software engineer work together to create platform capabilities and constraints. In addition, the general design principles, screen design standards, and style guide development deliverables if already created would need to be updated; otherwise, those deliverables would also need to be created.

13.24 Expected project outcome

Since this case study is fictional the outcome can merely be construed. This is not to say this analysis is complete conjecture because it is grounded in the research, reasoning, and experience of the author.

There is a strong likelihood that the CCTA would need additional resources because of the aggressive timeline, unfamiliarity with Java, and novelty of user oriented design techniques. Typically, additional resources temporarily slow down the project team, and therefore, it is probable that the product release will be late. The additional resources and time will inevitably cause the project to be slightly over budget.

There is also potential for slight deviations from the recommended user oriented design process. The participatory design task and usability walkthroughs are dependent on the end users availability, and therefore may be suspended or replaced with an alternative technique.

The task of updating the platform capabilities and constraints along with the design standards will take significant time due to the new software language being used. Having a dedicated team responsible for the updates and enforcement of the standards seems like the logical approach to minimize the impact on the time line. Nevertheless, the CCTA project team will inevitably be dependent on the decisions about the new standards and the risk of time delays remains because the tasks associated with standards are not being managed under the same project time line. By having overlapping team members this risk can be minimized however, it is not eliminated. The reality is that reaching consensus about new design standards is tedious and time consuming.
13.25 Other considerations that were outside the scope of this case study

The project used in this case study was medium-sized with centralized resources. Therefore, the organization did not need to consider the needs of decentralized teams or internationalization of a software application. Decentralized teams require additional coordination of events and strong project management. The challenge is greater when there are team members in different time zone or speak different languages.

Although there are many additional project management challenges when a team is decentralized, it benefits the user oriented design process. The decentralized nature of a project team may provide access to a broader range of end users, which could lead to a stronger design. Nevertheless, a software product that is being created for international use requires much longer analysis time. Many companies think that once a U.S. version is created using the English language, it is a simple matter of converting the labels to the various languages of the target end users. However, there is much more to international software development. For example, one must consider different word connotations and work styles. Therefore, a software application created for the US may not be appropriate for end users in another country. If the tool created within this case study needed to accommodate a European country such as Germany, the obvious consideration to the user interface layout would be necessary because German words are often longer than English words. However, the greater challenge is the analysis required for the different educational environments. U.S. and German school systems are very different. German students have a larger variety of subjects to select from and these subjects may require modifying the tool created within the case study.

14. Conclusion

This comprehensive comparison of the user oriented design methods reveals that there is no silver bullet for software application design. None of the methods discussed in this paper will guarantee a successful software application design. However, implementing the right set of user oriented design techniques may provide a more useful and usable software application. Selecting the right user oriented design techniques requires more then a cursory evaluation of the user oriented design methods.

Method names are like packing boxes at the end of the winter season. For example, I place all of my sweaters in one box and all my wool pants in another.
However, my husband mixes his pants and shirts together in one box. The end result for both of us is the same. The clothes are stored away and each of us knows how to access the necessary article of clothing when the seasons turn. The same holds true for the techniques that are stored in a method name. The method name is one individual’s opinion about what belongs together. What works for one project team may not work for another project team.

The reality is that, if an organization focuses on integrating a single user oriented design methodology, it is unlikely that the method will meet the needs of all the projects within the organization. Companies need to consider user oriented design at a more granular level for a sustainable user oriented design approach. User oriented design is not a one size fits all solution. Selecting the right technique is like selecting the right tool for the job. A hammer will do no good if the required task is to insert a screw. The same is true for a technique such as contextual inquiry; if the task does not happen frequently or happens over a long period, a focus group may be a more suitable alternative.

This paper reveals that the selection of the right techniques is more likely to lead to a successful user oriented design approach with long lasting results. However, because the techniques are marketed as a package within a method and consultants align themselves with those methods, it is challenging for a company to define a customized user oriented design approach. In addition, the practitioners of user oriented design methods have done a fine job in borrowing ideas from each other and therefore, have created many different technique names that, when evaluated, provide the same information. Thus, the multitude of user oriented design methods and associated techniques have become available because the market will withstand the competition and because the U.S. is a culture that is driven by choice. This means that many of the user oriented design methods have been developed as marketing ploys as opposed to the real necessity of an alternative design approach.

The bottom line is that in order for an organization to be successful in selecting the appropriate mix of user oriented design techniques they must be familiarity with as many of the user oriented design techniques as possible. For a company just starting out, selecting a user oriented design approach the analysis of these methods and associated techniques is overwhelming. However, the information in this paper provides a staring
point for navigating through majority of user oriented design choices that will lead to a sustainable user oriented design approach with long lasting and far reaching positive results.
### 15. Appendix A – Method comparison by technique

<table>
<thead>
<tr>
<th>Stages/Techniques</th>
<th>PD</th>
<th>SBD</th>
<th>CD</th>
<th>UCD</th>
<th>PCD</th>
<th>LCD</th>
<th>UE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requirements</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Profiles</td>
<td></td>
<td>Personas</td>
<td>Personas</td>
<td>X</td>
<td>Performer Profile</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Interviews &amp; Observation</td>
<td>Ethnographic field studies</td>
<td>Contextual inquiry</td>
<td>X</td>
<td>Structured interviews</td>
<td>Contextual analysis Observations</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Focus Groups</td>
<td></td>
<td>Organizational games</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Work Reengineering</td>
<td>Layout kit</td>
<td>Problem scenarios Activity scenarios</td>
<td>Work flow models Sequence Models Artifact models Cultural Models Physical models Work redesign Storyboards</td>
<td>X</td>
<td>Process tracking Performer process map Work objects Performance support map Work modeling Physical map Personnel metrics Event log Organization</td>
<td>Learner needs analysis GOMS Scenarios</td>
<td>Contextual task analysis Task organization model Affinity diagramming Multi-dimensional scaling Trajectory mapping</td>
</tr>
<tr>
<td>Methods</td>
<td>PD</td>
<td>SBD</td>
<td>CD</td>
<td>UCD</td>
<td>PCD</td>
<td>LCD</td>
<td>UE</td>
</tr>
<tr>
<td>---------</td>
<td>----</td>
<td>-----</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>----</td>
</tr>
<tr>
<td><strong>Stages/Techniques</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 and process metrics Interrelationship Diagram Storyboards</td>
</tr>
<tr>
<td>Requirements</td>
<td>Scenarios</td>
<td>Information scenarios</td>
<td>Scenarios UED Affinity diagrams</td>
<td>Affinity diagrams Scenarios Task Analysis OVID (Object View, and Interaction Design)</td>
<td>Work focus map</td>
<td>Scaffolding design</td>
<td>Scenarios Task scenarios Functional analysis</td>
</tr>
<tr>
<td>Hierarchical Task Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prior Version Feedback</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>Follow up studies</td>
</tr>
<tr>
<td>Stages/Techniques</td>
<td>PD</td>
<td>SBD</td>
<td>CD</td>
<td>UCD</td>
<td>PCD</td>
<td>LCD</td>
<td>UE</td>
</tr>
<tr>
<td>-------------------</td>
<td>----</td>
<td>-----</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>----</td>
</tr>
<tr>
<td>Competitive Evaluation</td>
<td></td>
<td></td>
<td>Market research</td>
<td>X</td>
<td>Market research</td>
<td></td>
<td>Market research Competitive analysis</td>
</tr>
<tr>
<td>Use Cases</td>
<td></td>
<td>Interaction scenarios Edge cases</td>
<td>X</td>
<td>X</td>
<td>Fish bone diagrams</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
### Method comparison by technique

<table>
<thead>
<tr>
<th>Stages/Techniques</th>
<th>PD</th>
<th>SBD</th>
<th>CD</th>
<th>UCD</th>
<th>PCD</th>
<th>LCD</th>
<th>UE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design</strong></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Parallel Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Participatory Design</td>
<td></td>
<td>Cooperative prototyping</td>
<td>Claims analysis</td>
<td>Co-design</td>
<td>X</td>
<td>Collaborative design</td>
<td>X</td>
</tr>
<tr>
<td>Conceptual Design</td>
<td></td>
<td>Desktop publishing game Stories Dramas</td>
<td>Vision Interpretation sessions</td>
<td>X</td>
<td>Vision</td>
<td>Conceptual Model</td>
<td></td>
</tr>
<tr>
<td>Prototypes</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Mock ups</td>
<td>X</td>
<td>UI mock ups</td>
<td>X</td>
</tr>
<tr>
<td>Design Walkthroughs</td>
<td>Walk through paper prototypes</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Detail UI design evaluations</td>
<td></td>
</tr>
<tr>
<td>Stages/Techniques</td>
<td>PD</td>
<td>SBD</td>
<td>CD</td>
<td>UC D</td>
<td>PCD</td>
<td>LCD</td>
<td>UE</td>
</tr>
<tr>
<td>-------------------</td>
<td>----</td>
<td>-----</td>
<td>----</td>
<td>------</td>
<td>-----</td>
<td>-----</td>
<td>----</td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability Metrics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heuristic Evaluations</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability Walkthrough</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Usability specifications
- Quantitative goals
- Performance goals
- Usability goals
- PCD Heuristic evolutions
- Consistency inspections
- Guideline reviews
- Consistency inspections
- Standards inspections
- Cognitive walkthroughs
- Scaffoldin
g evaluation
- Pluralistic
c walkthroughs
- Empirical Testing
## Method comparison by technique

<table>
<thead>
<tr>
<th>Stages/Techniques</th>
<th>PD</th>
<th>SBD</th>
<th>CD</th>
<th>UCD</th>
<th>PCD</th>
<th>LCD</th>
<th>UE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform Capabilities and Constraints</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>General Design Principles</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Screen Design Standards</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>UI specifications</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Style Guide Development</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
### 16. Appendix B – Usability Task Plan

<table>
<thead>
<tr>
<th>Task #</th>
<th>Task Name</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Iteration 1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Requirements &amp; Analysis Task</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>Functional analysis</td>
<td>15 Days</td>
</tr>
<tr>
<td>2.</td>
<td>Conduct focus group session to validate functional analysis</td>
<td>5 Days</td>
</tr>
<tr>
<td>3.</td>
<td>Create UED Model</td>
<td>3 Days</td>
</tr>
<tr>
<td><strong>Design Tasks</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Conduct participatory design sessions</td>
<td>3 Days</td>
</tr>
<tr>
<td>5.</td>
<td>Create conceptual prototype</td>
<td>5 Days</td>
</tr>
<tr>
<td>6.</td>
<td>Create usability matrix</td>
<td>2 Days</td>
</tr>
<tr>
<td>7.</td>
<td>Perform heuristic evaluation</td>
<td>1 Day</td>
</tr>
<tr>
<td>8.</td>
<td>Conduct design walkthrough</td>
<td>½ Day</td>
</tr>
<tr>
<td>9.</td>
<td>Make updates to models and prototypes</td>
<td>2 Days</td>
</tr>
<tr>
<td>10.</td>
<td>Conduct usability walkthrough</td>
<td>1/2 Day</td>
</tr>
<tr>
<td>11.</td>
<td>Make updates to models and prototypes</td>
<td>2 Days</td>
</tr>
<tr>
<td><strong>Iteration 2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Requirements &amp; Analysis Task</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>Functional analysis</td>
<td>5 Days</td>
</tr>
<tr>
<td>13.</td>
<td>Conduct focus group session to define detail workflow for the assigned component</td>
<td>2 Days</td>
</tr>
<tr>
<td>14.</td>
<td>Update UED Model</td>
<td>1 Day</td>
</tr>
<tr>
<td><strong>Design Task</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>Conduct participatory design sessions</td>
<td>3 Days</td>
</tr>
<tr>
<td>16.</td>
<td>Create low fidelity prototype</td>
<td>2 Days</td>
</tr>
<tr>
<td>18.</td>
<td>Update low fidelity prototype</td>
<td>1 Day</td>
</tr>
<tr>
<td>20.</td>
<td>Make updates to models and prototypes</td>
<td>2 Days</td>
</tr>
<tr>
<td>23.</td>
<td>Make updates to models and prototypes</td>
<td>2 Days</td>
</tr>
<tr>
<td><strong>Test &amp; Measure Task</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.</td>
<td>Perform heuristic evaluation</td>
<td>1 Day</td>
</tr>
<tr>
<td>19.</td>
<td>Conduct design walkthrough</td>
<td>3 Days</td>
</tr>
<tr>
<td>21.</td>
<td>Conduct usability testing</td>
<td>3 Days</td>
</tr>
<tr>
<td>22.</td>
<td>Prioritize changes</td>
<td>½ Day</td>
</tr>
<tr>
<td><strong>Finalize Deliverables</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.</td>
<td>Detail design document</td>
<td>3 Days</td>
</tr>
</tbody>
</table>
Note: This plan does not illustrate overlapping tasks. The number order is showing earliest start of a task and illustrates the design and test phases are iterative. Iteration 2 would repeat for each major component of the system.
17. Bibliography

Ariel. *Ariel's Answers to Frequent Questions.* Available: 


