Abstract

This paper presents a case study in the application of evaluation-guided software development methodologies to software systems to support engineering design. In particular, we consider a research prototype system for Computer-Aided Conceptual Design (CACD): CUP. CUP, which stands for Conceptual Understanding and Prototyping, is an interactive 3D modeling system that enables engineers to specify the semantic layout of an assembly and label major structural and functional relationships.

In our case study, CUP was subjected to a cycle of user studies to evaluate its usability, functionality and to provide feedback into the software development cycle. We selected a sample of potential student users, identified evaluation criteria, developed benchmark tasks and executed a user study using the “think out loud” protocol.

This study revealed numerous areas for improvement in CUP—changes based on these recommendations are being incorporated into the next version of the system. In addition, we believe that the evaluation methodology we present can be generalized to other systems for collaborative engineering design.

1 Introduction

When developing new software tools to support complex intellectual activities such as those involved in engineering design, it is important to maintain a human-centered focus on the task-related need of the target users. One procedure for ensuring a human-centered focus during software development is to utilize evaluation-guided development. Growing in part out of a generic strategy for evaluation of interactive systems proposed by Hewett [5], evaluation-guided development makes the assumption that software design and development of new tools is a process that is analogous to embarking on an ocean voyage under uncertain conditions.

The basic claim of the generic evaluation strategy proposed by Hewett [5] is that any design project should go through a series of evaluation-design cycles. Each cycle involves an analysis and evaluation of the specifications and requirements for the artifact being designed. These results are then incorporated into the design and development of a prototypical artifact, which may then be instantiated in any form, from vision statement to a fully functioning prototype. This prototypical artifact is then subjected to a re-analysis and reevaluation of the design specifications and requirements, leading to redesign. While it may be necessary to repeat the evaluation-design cycles several times, these processes result in a tangible artifact that accurately addresses the need of its users within the design constraints.

Developing a novel software artifact is similar to any voyage of discovery: while project goals and direction may be stated at the outset (i.e., find a passage to the Indies, water route via the Missouri to the Pacific, put a man on the moon, etc.), intermediate results are continually re-evaluated. In this process, improved understanding about fundamental problems may not come until the project is well underway and sometimes goals and evaluation criteria
have to be reevaluated and changed—and the destination reached might not be as expected (i.e., the Indies turn out to be Caribbean islands). While these are valid initial concepts, the more one knows about the goal and the criteria of determining success, the more likely one is to wind up at a destination worth visiting.

This paper presents an application of a general Evaluation Guided Development methodology to a software tool to support conceptual mechanical design, CUP [1]. CUP is virtual environment that allows designers to sketch out, in 3D, a conceptual layout for a new product model and specify major structure-behavior-function relationships among assembly components. The paper first presents the background and an overview of our Evaluation Guided Software Development methodology; second, it presents our application of these techniques to the CUP systems. We describe the user study we executed as well as its results. Our conclusions present ideas on how this research approach can be generalized to other engineering systems, such as those for collaborative and distributed design.

2 Evaluation Guided Software Development

2.1 Background

There are two kinds of evaluations that can be performed on software—summative and formative. Summative evaluation analyzes the software after the final release. Users can look at the finished product, examine how well the product fits their computing need, and determine whether or not to use it. On the other hand, formative evaluation should be an ongoing or ‘symbiotic’ process. Formative evaluation is ideally conducted at every level of development and help to shape the evolving form of a software artifact. Using formative evaluation procedures to gather feedback from target users while a piece of software is being developed is a very useful method for keeping the project on course and approaching the successful completion of its goals. Information gleaned from the analysis of feedback from a representative sample of the targeted users yields an artifact superior in quality and time/cost effectiveness. Taking into account such feedback while the software artifact is in beta form helps the software engineer(s) make course corrections so that the project is pointed in the right direction and will be well received by its intended audience. The early additional costs of doing formative evaluation are more than compensated for by avoidance of unbudgeted costs of rewriting/retooling a software artifact which turns out to be unwieldy or inappropriate. Finally, formative evaluation can sometimes lead to the reformation of the goal for software development.

2.2 Motivations

2.2.1 Support Human-Centered Design and Development

An important characteristic of evaluation-guided development is its ability to support a human-centered focus for the software being developed. With hundreds of computer applications being generated each year, it is important to ask how well these applications perform in the workplace for which they were designed. The creators of software systems are face with increasingly complex issues to address, not the least of which involves consideration of what functions to provide for the user.

For collaborative engineering systems, designers are faced with an unprecedented number of new and complex tools—ranging from simulation and analysis systems to instant prototyping and fabrication services available over the Internet. It is never easy to combine all of the demands effectively into an artifact that is both powerful and simple to use. One of the important parts of software testing involves considering the types of experiences encountered and work processes performed by the system’s end users on a daily basis. The purpose of utilizing evaluation-guided development is to ensure that the system can serve the knowledge workers for which it was designed in a manner that is efficient and fosters productivity. After the artifact is released, summative evaluation will give developers a good idea how their creation will perform.

2.2.2 Eliminating the Productivity Paradox

Landauer [7] reports on work of economists who have discovered something dubbed the productivity paradox: the inability of economists to find or demonstrate any growth in worker productivity associated with the use of information technology in doing knowledge work. The paradox lies in the fact that this lack of growth exists despite strong expectations to the contrary. In short, there are many problems with existing computing systems. Many do not seem to assist the end user’s work productivity because the software is difficult to use or does not appropriately support the work that the individual needs to perform. Landauer argues that when developing these faulty systems, software engineers missed the mark of developing effective human-centered products. Landauer further argues that it is the past failures of computing systems that have led to the productivity paradox, and that these failures can be traced back to a lack of understanding about the intended task and user of the software. Incorporating human-centered feedback into the design process by means of evaluation-guided development (e.g., usability testing, structured interviewing, etc.) will strengthen the viability and power of future systems
2.3 What is Being Evaluated?

An additional question asks what is being evaluated. This paper describes one cycle in the evaluation-guided development of a 3-D modeling tool for the conceptual understanding and prototyping in engineering design. CUP enables a user to specify a spatial layout of components and sub-assemblies in a mechatronic artifact. A mechatronic artifact is some object begin design that has both electronic and mechanical components and/or subassemblies. CUP provides the means for a user to specify structural, behavioral, and functional (S-B-F) information about the various components and subassemblies. CUP also includes mechanisms for capturing textual information about the designer’s preferences and design intent. It is our belief that this case study with CUP can be generalized to other collaborative engineering design tools.

3 CUP: A Tool for Conceptual Understanding and Prototyping

There are two major goals for CUP project [1]. First it is intended to be part of an environment for computer-aided conceptual design (CADC)—enabling engineering designers to quickly capture structural information in a top-down fashion and record the behavior and function of the intended artifact and its subcomponents. Conceptual design is the initial phase of the product design process. This is when product development teams perform their “back-of-the-envelope” calculating, sketching, and planning—fleshing out of their initial product concept. This conceptual design process is a key component of the creative thought process, providing a clearer idea of what is being created. Drafting, CAD, and detailed analysis can then follow in a more goal directed manner. This design phase of an artifact development process has a tremendous impact on the life-cycle cost of a product. This phase of design has been estimated to have an impact that eventually accounts for as much as 75% of the eventual product life-cycle costs [2]. In addition, several other authors [3, 8] have emphasized the critical role of conceptual design and variant design in overall process of engineering design. While there are many tools for aiding detailed design in the marketplace today, there are few tools that support conceptual design.

The second major goal for CUP is to provide a query interface for design repositories. These repositories [11, 9] are effectively digital libraries of the specifications of existing artifacts that might serve as models for or as subcomponents of the new artifact being designed. A fully implemented design repository would enable access to a knowledge base containing legacy engineering design data. CUP allows engineers to engage in conceptual level design for mechanical, electrical, and mechatronic engineering tasks, thereby providing a tool that contributes to increased productivity. The focus of evaluation-guided development on CUP is to ensure that the software facilitates how engineers normally do their jobs in the workplace and that it provides new and usable functionalities without creating new problems with which to cope.

3.1 Evaluation Criteria

Part of the evaluation of CUP should be rooted in knowledge of the engineering design process and how engineers actually work in practice. We considered two evaluation criteria based on the task-related needs of CUP users. First, to evaluate CUP’s ability to enable designers create conceptual designs of artifacts. Second, as CUP allows the use of models originally designed using other CAD tools, we wished to evaluate its ability to import and annotate such models.

3.2 Methodology

There are a variety of techniques that can be used to gather information about the engineering design process so as to better understand how that process is proceeding (e.g., questionnaires, observational studies, etc.). The method chosen for use in this cycle of the evaluation-guided development of CUP involves “thinking out loud” and protocol analysis [4]. An extensive review of work on this procedure [4] has shown that protocol analysis of “thinking out loud” and of retrospective verbalizations by problem solvers has been shown to be a useful sources of information in gaining an understanding of the problem solving process. When applied to gathering the type of feedback needed to understand problems with the usability of software this technique is equally useful [6].

The use of “thinking out loud” and retrospective procedures (i.e., post-study discussion of particular points) was employed in this study because of its ability to more immediately track the thought processes and intentions of the user while engaged in a given task. This was done while producing minimal interference with the actual thought processes of the participants. Asked to speak their minds while completing the task, the participants were able to provide the experimenters with useful information as to how one type of user goes about their work processes. Using test participants from differing backgrounds provides us with the added information about the various approaches that can be taken. Once this information had been collected, the results provide insight as to how well CUP performed when compared to the ideal work methodology of engineers. Therefore, whenever this study finds an important flaw in CUP’s
Figure 1. Screen shots of CAD model of a computer mouse in SDRC I-DEAS, and then in the CUP system (b), and with fully-authored SBF links (c).
design, it has followed the actual thought processes of users to conceive the reason behind the failure.

3.2.1 Participants

The general methodology of the study involves giving a task to a participant and asking them to “think out loud” or to verbalize what they were thinking as they work upon their task. Participants were recruited based upon educational background and experience. Although CUP is eventually expected to provide support for working professional engineers, at this stage of testing it is desirable to find and eliminate any basic usability faults with the system before examining the full range of functionalities being provided using the time of experienced engineering designers. Therefore it was desirable to have a range of test participants to allow for comparisons between relatively experienced and relatively inexperienced users of such systems before utilizing the time of working professionals.

Each of the 7 participants in this cycle of evaluation guided development of CUP were Drexel University students of a variety of educational backgrounds, including Engineering, Graphic Design, Business, and Computer Science. All had at least a novice proficiency in computing. Some had experience with engineering packages (2D and 3D) and others had experience with other graphical packages. One participant had prior experience with CUP.

3.2.2 Task Selection

Members of the evaluation team experimented with CUP to develop a set of design tasks that would be approachable by engineers and non-engineers alike. At this stage of evaluation guided development the goal was to evaluate the intuitiveness and functionality of the CUP interface. Once the task had been chosen, some example prototypes were developed to show the participants. For the users in this study, the simple design of a number 2 lead pencil (Figure 2) was selected because it requires the use of most of CUP’s main features: the creation of four separate component objects (lead, wood, eraser, metal ring) that are linked together. We expected each participant to create 4 separate objects to represent the lead (graphite), wood, eraser, and metal ring of the pencil as shown in Figure 2 (b). Furthermore we expected the objects to be linked together with some description of the relationship between the linked objects.

3.2.3 Study Procedures

First, each participant was given a brief overview of the project and the application. The participants were then given a brief, open-ended preliminary questionnaire to assess their general level of computer expertise and background. The investigator then read the instructions for “thinking out loud” to the participant, followed by 2 or 3 practice thinking out loud problems adapted from [4]. Each participant was then presented an example of a template object already designed with CUP (the 3.5 inch floppy disk shown in Figure 3). Next the participant was allotted three minutes to ‘play around’ with the software before being given the task of creating a conceptual design of a pencil to the best of his or her ability, while using the “think aloud procedure” (Figure 2 (b)).

During work on the design task, the participant was recorded (audio) and observed by one or more investigators (who were out of the participant’s line of sight). There was little or no social interaction between the investigators and the participants at this time (except for needed cues such as “talk” or “keep talking”). During the design task the investigators noted any problems that the participant seemed to have during the task (e.g., spending several minutes trying to label an object). Although social interaction was kept to a minimum, some coaching by the investigators was necessary at times. After an interaction problem had been identified and the participant was clearly ‘stuck’, the investigators would guide him or her through that part of the task. After receiving guidance the participant was asked to continue with the task and was not given further guidance unless once again stuck on another sub-problem.

Immediately upon completion of the design task, the investigators asked the participants to explain the source of their difficulties (regardless of whether they had been coached or not). During the thinking out loud portion of the
(a) Expected results of task.

(b) Events of typical evaluation.

Figure 2. Evaluation scenario and results.

study a few participants had to be constantly reminded to keep talking, but others freely shared their thoughts. While the need to remind some participants to keep talking somewhat inhibited the evaluators’ ability to assess the problems the participant might be having, the retrospective discussion of problems proved useful in clarification of dilemmas encountered during the thinking out loud portion of the study.

At the conclusion of the post design interview, each participant was asked to complete a second brief questionnaire. This questionnaire asked participants to give any additional feedback about the software (problems, suggestions, etc.) that might not have been discussed during the interview. Finally, each participant was thanked for his or her time and involvement in the study.

3.3 Evaluation Results

Most of the results of the study fall into two main categories, those associated with object manipulation and those associated with tool bar and menu functionality. Observations that do not fall into either of the former categories are clumped together as miscellaneous observations.

3.3.1 Object Manipulation

One of the major problems most of the participants encountered was with manipulating the objects they created. Problems encountered was as follows:

- *Object movement* with the mouse was difficult for the participant unaccustomed to the 3-button mouse for
several reasons.

1. **Translation** of an object required the use of the right-most button where the users wanted to use the left-most button. Most users did not utilize the middle button which translates an object on the z-plane.

2. **Rotation** of an object with the mouse (which requires the use of the left-most button) was felt to be inaccurate and uncontrollable by the participants.

   - *Re-sizing* of an object was difficult for some of the participants because there was no reference as to the scale of the world. The users had trouble identifying what dimension \((x, y, z)\) they needed to manipulate and by how much.
   - The *position* of an object was difficult to determine. There was no way to determine the location of an object within the world coordinate space other than through then “Move object” function.

3.3.2 **Toolbar & Menu functionality and functions**

Use of the tool bar was generally preferred over use of the menu bar. The majority of the icons (e.g. the primitive, delete, and manipulation buttons) were easily understood.

**Functionality:**

- The *Group* and *Ungroup icons* are hard to distinguish from each other and difficult to identify as “group” and “ungroup”.
- *Pop-up identification* for each button exists but because of the quick movements many users employed over the tool bar they did not appear for most users.
- *Menus* were employed after the user felt that he/she exhausted all possibilities of the mouse and tool bar and helped identify buttons on the tool bar.
- The *Help Menu* was not found by most participants.

**Functions:**

- *Primitive creation* was successful by all participants but the window that appeared when creating them was generally found to be confusing.
- Some of the participants expected the pointer to change to a representation of the tool in use after clicking one of the object manipulation buttons instead of a window requesting numeric values.
- The procedure for linking objects was difficult to determine for the participants.
- *Undo* does not provide the outcome participants expected - it deletes the selected object.

3.3.3 **Miscellaneous Observations**

Many of the problems encountered with CUP are rooted in the participants expectations based on their previous experiences with graphical programs. The fact that some of the participants were used to working on Macintosh computers affected their ability to effectively utilize all of the mouse buttons.

- Windows with tabs were unclear as to which were selected.
- The participants who used the color chooser were only able to customize the colors due to the problem mentioned above.
- The default use of a black background to display made dark colors difficult to see
- It was hard to determine a change of color in CUP because the object retains the hue of its previous color until the user de-selects the object.
- Lighting of the scene casts some sides of objects in glaring light while others almost disappear into darkness.

3.4 **Discussion**

In the evaluation-guided development of CUP we have learned some useful lessons not only for the further development of CUP itself, but that should also be helpful to developers of other systems for engineering design. For example, it is clear that our users think of the mouse cursor as the point of contact that enables them to manipulate and track the objects with which they work. They expected to be able to manipulate objects on the screen in a relatively direct way and had a fair amount of difficulty with indirect manipulation such as establishing a correspondence between entering numbers and the effects of those numbers on sizing and scaling of objects. Similarly, the participants had expectations about directly linking objects. They expected the ability to just select multiple objects and then click the link button to establish a link.

Indeed, it seems that the principle of direct manipulation [10] has become so deeply ingrained in computer users that the participants displayed a distinct preference for using icon commands in the tool bar rather than the more indirect action of pulling down the menus of the menu bar. This seemed to be the case even though some icons were confusing and the same commands existed in both the tool bar and menu bar. Similarly, some of the participants expected the pointer to change to represent the tool after clicking one of the object manipulation buttons.
It is also clear that there were some problems with model of visual design embedded in CUP. The use of a black background makes dark colors hard to distinguish. Scene lighting in CUP casts some sides of objects in glaring light while others almost disappear in darkness. A more balanced composition of colors and lights would make the visual scene more natural and easier to grasp. While these problems are easily fixed they suggest that the development of CUP would have benefited from the early involvement of someone with graphic design expertise. Given the results of the study reported here we anticipate that future human centered evaluation studies will continue to provide useful feedback for the ongoing evaluation guided development of CUP.

4 Conclusions

This paper presented a case study in the application of evaluation-guided software development methodologies to software systems to support engineering design. The study considered the CUP system, a research prototype for Computer-Aided Conceptual Design (CACD). For this engineering design domain, we documented unique ways of applying generic evaluation-guided development methodologies to improve the human-computer interface for CUP.

The results of our study revealed numerous areas for improvement for the next version of CUP. It is our hope, in documenting our results and our procedures, that the evaluation methodology we present can be generalized to other systems for collaborative engineering design.

Acknowledgements. This work was supported in part by National Science Foundation (NSF), Knowledge and Distributed Intelligence in the Information Age (KDI) Initiative Grant CISE/IIS-9873005 and Grants ENG/DMI-9713718 and CISE/CDA-9729827. Additional support was provided by The National Institute of Standards and Technology (NIST) Grant 60NANB7D0092.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the other supporting government and corporate organizations.

References


