DOCUMENT RESUME

ED 476 971 IR 021 699

AUTHOR	Battig, Michael E.
TITLE	Utilizing the PDA as a Vehicle for User Interface Design Pedagogy.
PUB DATE	2002-06-00
NOTE	7p.; In: ED-MEDIA 2002 World Conference on Educational Multimedia, Hypermedia & Telecommunications. Proceedings (14th, Denver, Colorado, June 24-29, 2002); see IR 021 687.
AVAILABLE FROM	Association for the Advancement of Computing in Education (AACE), P.O. Box 3728, Norfolk, VA 23514. Tel: 757-623-7588; e-mail: info@aace.org; Web site: http://www.aace.org/DL/.
PUB TYPE	Reports - Research (143) -- Speeches/Meeting Papers (150)
EDRS PRICE	EDRS Price MF01/PC01 Plus Postage.
DESCRIPTORS	*Computer Interfaces; *Computer Science Education; *Computer Software Development; *Computer System Design; Higher Education; Information Systems; Instructional Development; Instructional Materials; *Screen Design (Computers); *Student Developed Materials

ABSTRACT

As computing and embedded systems become ubiquitous in our world, the importance of user interface design knowledge increases in our curriculum. Students of undergraduate information systems or computer science programs should possess some competence in this computing sub-discipline. However, many programs do not have the curricular space to host a separate course in usability or user interface design. To address this concern, results and observations of incorporating user interface design pedagogy in the context of a software engineering project course are presented. The project centers around a data collection application to be hosted on a PDA (Personal Digital Assistant). The application has significant constraints concerning usability and human factors that provide a rich context for teaching and demonstrating user interface design concepts. An appendix highlights the evolution of the actual interface developed by one of the development teams. The user interface design changes were the result of feedback that students received from four sources: course materials on usability, direct instructor feedback, fellow classmate feedback, and outside-the-course student feedback. (Contains 10 references.) (Author/AEF)

Utilizing the PDA as a Vehicle for User Interface Design Pedagogy

Michael E. Battig – mbattig@smcvt.edu
Computer Science Department, Saint Michael's College
Colchester, VT 05439

Abstract

As computing and embedded systems become ubiquitous in our world, the importance of user interface design knowledge increases in our curriculum. Students of undergraduate information systems or computer science programs should possess some competence in this computing sub-discipline. However, many programs do not have the curricular space to host a separate course in usability or user interface design. To address this concern, results and observations of incorporating user interface design pedagogy in the context of a software engineering project course are presented. The project centers around a data collection application to be hosted on a PDA (Personal Digital Assistant). The application has significant constraints concerning usability and human factors that provide a rich context for teaching and demonstrating user interface design concepts.

Keywords: user interface design, usability, ergonomics, software engineering, laboratory course, PDA.

1. INTRODUCTION

While change in computing seems inevitable, educators continue to wrestle with the problem of packing more topics into a finite curricular space. Therefore, an approach that has gained momentum is to integrate some topics into the fabric of courses rather than creating individual courses for each topic [Johnson 1997]. In this work, we will investigate the goals and outcomes of integrating user interface design pedagogy into a traditional software engineering course. Specifically, we will look at the benefits and detriments of utilizing the Palm PDA platform as the focus of a semester-long student project.

The software engineering course for this project is a four-credit course taken during the senior year of a traditional undergraduate program in Computer Science. A significant component of this course is the semester long project. The project allows students to work in teams of no more than three students. The four phases of the project are: requirements specification, design specification, implementation, and test plan design and execution. Thus, the teams have the benefit of seeing a project through many typical phases of a software lifecycle.

Of particular interest is the utilization of the Palm OS® platform as a vehicle to reinforce fundamental user interface design and software engineering principles. This project concept has been employed for two consecutive academic years. The students have a background in Java/C++ programming, but no prior experience with the Palm OS® platform. The implementation and testing phases require the use of the Metrowerks CodeWarrior® IDE, which supports C++ for the Palm OS® [Rhodes 1999].

2. USER INTERFACE DESIGN GOALS

The software project in this instance is a basketball statistics data collection program for the Palm IIIc and Palm IIIxe PDAs. The program must allow a novice Palm user to track basic statistics (e.g., field goals attempted/made, personal fouls) on an individual player and team basis during the fast-paced action of a live NCAA basketball game. Therefore, the program's interface must be simp le, minimize the user's memory load, provide intuitive short cuts, allow for easy reversal of actions (e.g., undo), and prove to be reliable in the real-time setting of a basketball game. Participants are given the opportunity to test their product at an actual NCAA Division II basketball game during the final week of classes of the fall semester.

During lecture, students are given the opportunity to dialog on a number of outside resources related to user interface design. Students are exposed to a variety of literature, some from the popular press [Carlton 1994] that illustrate, often in a very humorous fashion, the broad range of expertise that is common among users. Students are also given supplemental material from the traditional user interface design discipline [Shneiderman 1998, Nielsen 1993]. The following concepts are presented in this segment of the course:

- Prototyping
- Parallel Design
- Know the User
- Heuristics for Interface Design
- Usability Testing.

The students are required to develop interface prototypes as part of their requirements and design specification documents. The value of this has many facets, not the least of which is the reinforcement of demonstrating an iterative design process. Thus, we see an effective interface developing over time rather than developed in its entirety the first time. Furthermore, since each team is developing a unique interface, students may then compare designs prior to implementation in order to see the benefits of participating in a parallel design. Related to this point is the need for the instructor to communicate a relaxing of the traditional standards of academic dishonesty (i.e., we encourage the participants of this course to share ideas and to incorporate the "best of class" into their own implementation).

Classical user interface design pedagogy invariably includes the novice/expert user continuum. To apply this knowledge, students are encouraged to consider the attributes of the typical user of their system. Basketball scouts are typically assistant coaches who are novice Palm users that will use the system on an intermittent basis. Thus, many of the heuristics discussed in the course are applicable.

Simplicity of the interface is essential. For starters, the typical PDA does not provide an abundance of display real estate. Secondly, all game time features need to be placed on a single screen (navigating multiple screens during the fast-paced action is too burdensome). Student-developers are also encouraged to provide informational feedback, minimize the user's memory load, and provide consistent short cuts where possible. Given that users will invariably make input mistakes, the system must allow for the easy reversal of actions as well. The system also provides the opportunity to emulate direct manipulation versus indirect manipulation as developers are encouraged to minimize user input via Graffiti® in favor of stylus input (potentially integrating an error prevention strategy into the interface design).

The course introduces students to the differences between utilizing heuristics for user interface design (as discussed above) and actual usability testing. Although students don't typically have the time typically (nor the inclination) to recruit human subjects for usability studies, they are encouraged to give their system to a non-CS student for review. The process of looking at a software system's interface through someone else's eyes is an enlightening experience for many.

3. SOFTWARE ENGINEERING GOALS

The software engineering course that is the focus of this research is in many ways traditional. We utilize a well-known text [Pressman 2001]. We teach a variety of process models, paradigms, and techniques with an eye toward balancing theory with practice. However, after ten years of teaching this course, we have found that some unique aspects of our Palm OS® project illuminate certain lessons that are historically difficult to reinforce beyond telling (i.e., our assumption is that telling is not teaching and that students will embrace that which they self-discover through practice). Those lessons are:

- Difficulties of software maintenance
- Learning new tools/platforms
- Differentiating essential from accidental
- Applicability of non-technical skills
- Working in a team environment
- Dealing with non-contrived constraints.

The Palm project in this course gives many students their first hands-on experience with software maintenance. For most of them, the project represents their first attempt at modifying a non-trivial program written by another developer. This is accomplished with the students being given a base Palm C++ project as a jump-start. The base program includes over 1400 lines of C++ spread over four classes. The original motivation was that a typical Palm application was deemed too large for undergraduate students to develop in the context of a one semester course, given that they must develop requirements and design specifications before commencing with implementation. Students have typically added another 500 to 1,500 lines of C++ to the base project. Post semester student assessments have shown that at the conclusion of the project, students say they would prefer to write their own application as opposed to maintaining code written by someone else. Thus, we have achieved a significant learning outcome by allowing students to self-discover the hardships of software maintenance.

The Palm project also provides students with the opportunity to apply their knowledge of object-oriented software development to a novel target platform. For all of the students to date, this represents their first exposure to the Palm OS® Metrowerks CodeWarrior® development environment. Although the learning curve for a new tool causes a certain level of anxiety in most of us, the experience creates a nice backdrop to discuss the differences between the essential and accidental activities for a software project [Brooks 1986]. Thus, we are able to highlight the essential nature of many non-technical skills such as writing and speaking. Since the project includes written documents (requirements, design, test plan) in addition to source code, students are able to see first hand the essential nature of effective communication. The potential pitfalls of communication are further highlighted, some might say exacerbated, by virtue of the fact that the students work in teams.

Perhaps the most beneficial attribute of a learning experience is creating an enthusiastic atmosphere. We have found that the combination of a novel platform and a believable application provides students with enthusiasm in that they see the project as more "believable" than others that may have been contrived in the past. At the start of the term, students are shown a commercial Palm OS® application for tracking football statistics. The real-time differences between football and basketball game situations provide a nice scenario for discussing usability issues. Center stage in this discussion is the fact that football provides frequent, short breaks in the action that basketball does not permit. Fortunately, college students are almost universally aware of the differences between the games, although the presence of international students provides a ready reminder of cultural issues in software design. Thus, the elaborate multi-screen interface in the football scouting application is insufficient in this case.

4. DIFFICULTIES & DISTRACTIONS

Without question this project provides some hurdles that every team mentioned in their post-mortem survey:

- Limitations of the Palm screen
- Lack of implementation tools
- Frustrations with maintenance.

Due to a combination of the usability demands of the application and the limited screen size on the Palm III, all development teams concluded that the challenges in user interface design were more significant than they had faced previously. Specifically, participants noted that buttons needed to be large enough to tap accurately, yet not so large as to chew up too much screen real estate. On a related note, teams also mentioned that they were challenged to come up with meaningful abbreviations for each button (e.g., "FreeTAtm" or "FTA" as an abbreviation for "free throw attempt"). For these reasons and others, the use of the PDA in this course provides a pedagogical platform that allows students to deal with interface design issues that are commonplace among practitioners.

Every team lamented the fact that they were limited to the C++ programming language and the CodeWarrior® IDE. Participants are accustomed to a richer set of interface development tools such as JBuilder® or Visual Basic®. Previous work has considered user interface design in the context of the PC [Battig 2000]. However, the PDA target environment is significantly different from the PC. For starters, the resources of memory (both primary and secondary) and processor are significantly less on the PDA. Secondly, the PDA development platform has not been around for over two decades like the PC and thus has not been the recipient of the sophisticated tools that accompany such tenure. The result is that students are given the opportunity to learn a new development platform during development in much the same way as practitioners do routinely.

On the post-mortem surveys, a majority of teams indicated that they would have preferred to develop their own application from "scratch" instead of maintaining an existing one. For most, this is the first exposure to the tyranny of software maintenance. Professional developers experience this frustration more than some care to admit. However, the constraints of the marketplace do not usually permit developers to take the time and money to build new systems, even when it seems justified from their perspective. Therefore, the project has the benefit of demonstrating to participants the difficulties inherent in software maintenance.

5. CONCLUSION

The results and observations of using the PDA platform to teach user interface design and implementation to undergraduate computing students include many benefits. Students are confronted with usability obstacles that are more challenging than most they have faced to date. For example, several project teams labeled their player buttons in sequence (from 1 to 5) instead of using the player jersey number. This example provides a poignant application of reducing the user's memory load. Although usability heuristics like this are commonly taught, one final attribute of this project merits discussion. Because the PDA platform is fairly novel (i.e., most students own a PC, but none of the participants own a PDA), it provides a "gee-whiz" factor that increases student motivation. In other words, students are more enthusiastic participants and learners in a project that they perceive is using cutting-edge technology.

The project also addresses many software engineering issues that present problems for pedagogues. Because this project is larger than the typical student project (several thousand lines of code), it presents issues of software engineering that are hard to expose in the classroom and yet are common in practice [Dawson 1997]. Chief among the software engineering issues is the difficulty associated with software maintenance.

Most computing curricula [ACM 1991, AITP 1997] recognize the importance of user interface design. However, many undergraduate programs do not have the freedom in the curriculum to include a separate course in user interface design. Therefore, the benefits of teaching usability issues in a related course (such as software engineering) prove compelling. As an added benefit, project participants begin to see the

important ways in which many usability heuristics are integrated into the fabric of user interface design, software implementation, and human relationships.

6. REFERENCES

ACM/IEEE-CS Joint Curriculum Task Force, Computing Curricula 1991, 1991 http://www.computer.org/education/cc1991.

AITP, IS'97 Curriculum Model for 4 Year Undergraduate Programs in Information Systems, 1997, http://www.is-97.org. Battig, Michael E and Ron

Sobol., "Migrating a Traditional Network and Data Communication Laboratory Course to an Information Systems-Friendly Environment," 2000, Proceedings of the Information Systems Education Confernece.

Brooks, Frederick P., "No Silver Bullet-Essence and Accident in Software Engineering," 1986, Proceedings of the IFIP Tenth World Computing Conference, pg. 1069-76.

Carlton, Jim, "Befuddled by PCs, Users Call for Help," 1994, The Wall Street Journal.

Dawson, Ray and Ron Newsham, "Introducing Software Engineers to the Real World," 1997, IEEE Software, vol 14, no. 6, pg. 37-43.

Johnson, Hubert A., "Integrating Software Engineering into the Traditional Computer Science Curriculum," 1997, SIGCSE Bulletin, vol. 29, no. 2, pg. 39-53.

Nielsen, Jakob, Usability Engineering, 1993, Morgan Kaufmann.

Pressman, Roger S., Software Engineering: A Practitioner's Approach, 5th Ed., 2001, McGraw-Hill, Inc.

Rhodes, Neil and Julie McKeehan, Palm Programming: The Developer's Guide, 1999, O'Reilly & Associates, Inc.

Shneiderman, Ben, Designing the User Interface: Strategies for Effective Human-Computer Interaction, 3rd Ed., 1998, Addison Wesley Longman, Inc.

APPENDIX

To give the reader a better understanding of the data collection application and user interface employed in this project, this section will highlight the evolution of the actual interface developed by one of the development teams. The user interface design changes were the result of feedback that students received from four sources: course materials on usability, direct instructor feedback, fellow classmate feedback, and outside-the-course student feedback.

Figure 1. First Prototype User Interface

Figure 1 shows the first interface implemented. This interface has numerous deficiencies. The most glaring is the memory load placed on the user to associate players on the floor with the numbers 1 through 5. Figure 2 shows the improved interface with jersey numbers providing a more natural approach. Note also that the improved version provides the ability to reverse previous actions via the "Undo" button. The latter interface includes features for two teams instead of just one. The developers of this interface also incorporated some feedback on usability to change the abbreviations used on the buttons. Notice that the abbreviation for a three-point shot attempt changed from "3pa" to "3ptAtm."

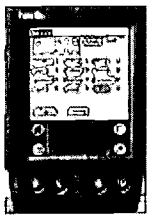

Figure 2. Revised User Interface

The revised interface in Figure 2 also provides buttons for substitutes ("Subs") and summary of statistics ("Report"). Screen shots for these features are shown in Figures 3 and 4 respectively. To make a substitution, the user clicks a player currently on the floor and a player currently on the bench and then clicks "Sub." Figure 3 shows that Johnson will be coming off the bench as a substitution for Strickland. Lastly, Figure 4 shows the summary statistics collected for Team 2. These statistics may also be viewed on an individual player basis.

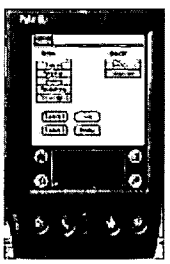

Figure 3. Substitution Screen Shot

Interested readers my access the syllabus for this course on the author's web site: http://personalweb.smcvt.edu/mbattig/CS407%20Syllabus.htm

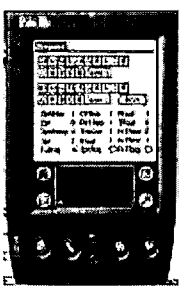

Figure 4. Summary Statistics Screen Shot

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

Reproduction Basis

X	This document is covered by a signed "Reproduction Release (Blanket)" form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.
	This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").