User Centric Rapid Application Development

Sumit Kumar & Tanya Sharma
Computer Science and Engineering Department
Dronacharya College of Engineering, Gurgaon, Haryana, India
Email: skd2592@gmail.com

Abstract-
This paper describes our experiences modifying the Rapid Application Development methodology for rapid system development to design a data gathering system for mobile fieldworkers using handheld computers in harsh environmental conditions. In our development process, we integrated User-Centred Design as an explicit stage in the Rapid Application Development (RAD) software engineering methodology. We describe our design process in detail and present a case study of its use in the development of a working system. Finally, we use the design of the working system to highlight some of the lessons learned, and provide guidelines for the design of software systems for mobile data collection.

In pursuing this project, we worked with field ecologists monitoring the evolution of coastal wetlands in the San Francisco Bay Area. The overall goal of the ecology project was to provide accurate information on the impact development has on these wetland areas. While the architecture of our system is tuned to the specific needs of the ecologists with whom we worked, the design process and the lessons we learned during design are of interest to other software engineers designing for similar work practices.

Introduction and Background

Recently, much interest has been paid to supporting the information needs of biologists, specifically with respect to the collection, analysis, and management of large data sets. While much of the work is geared toward genomic data, other fields of biology also suffer from inadequate data collection and management processes. In this paper, we describe our development process that resulted in the design of a data collection application for ecologists studying plant species in the sensitive coastal wetlands area near our institution. The wetlands project measures species and frequency of vegetation at randomly generated data points in an area of ecological interest. Figure 1 depicts a region of interest with sampling points. Data points are loaded into a GPS system and ecologists travel to each of the data points, recording species and frequency data for the vegetation located there. To record data, each team used a clipboard with a sheet of paper attached.

Information is recorded using fixed numerical scales. The use of numerical scales speeds the recording process and minimizes transcription errors. To analyze the numerical data collected, the data is transcribed into a spreadsheet application. The transcription of one day of paper-based field observations typically takes two or three days of data entry time in the laboratory. Our goal in this project was to enable electronic data collection, thus eliminating transcription.

While a complete description of the data collection practices of ecologists is beyond the scope of this paper, one important question is whether the fieldwork techniques we observed in our target project can be generalized. Certain aspects of any project are unique, while others are characteristics of the general work practice across many or all projects.

The ecologists we work with perform fieldwork constantly. In follow-up interviews with our user group and other biologists, several themes that are common across current biological fieldwork practice came to light. Typical practices include:

1. The use of numerical scales or other shorthand symbols or shorthand notations to simplify data capture and to minimize transcription errors is common.
2. The data collected are predictable. Biologists know the species of vegetation (or animals, soil moisture content, etc.) that they expect to find at a given location, and how much variability is likely in measured values.
3. The need for data transcription to electronic format, and a desire for this process to happen quickly, are generally true of many projects.
4. The use of pen and paper is typical in the field, due to paper's tactile characteristics and to the persistence of data recorded on paper despite mishaps, i.e. "If it falls in the mud, I can still read it."

One other important characteristic seems to be common across biological domains. The most significant hassle associated with data collection is the transcription process in the lab. As one participant noted: "Transcribing is error prone knowing whether something is a 4 or a 9, lining up numbers with names. It takes a lot of time and no one likes [doing] it."

A constraint on system development for limited term biological fieldwork projects, where the duration of the project is measured in weeks or months, is the need for a rapid development process to design and deploy systems early in the fieldwork project. To support rapid development for mobile fieldwork, we present a modified form of rapid application development we used in the design and deployment of our system. Rapid application development (RAD) is an iterative software development methodology described, originally, by James Martin[11]. Since its inception, many authors have identified difficulties associated with software development using RAD methodology. To overcome problems with typical RAD methodologies, we introduce User Centred Rapid Application Development (UCRAD). UCRAD is a three-stage process. In the first stage, user interface design is combined with the elicitation of requirements. In the second stage, a high fidelity prototype is evolved into a functional system that is gradually deployed in the field. Finally, we maintain the deployed application through constant tailoring to the data collection process.

This paper is organized as follows. In Related Work, we outline some previous work in the design of data collection systems for ecologists, and some related work in the use of Pocket PC devices and PDAs for data collection in other fields. Next, we describe the User-Centred Rapid Application Development methodology we evolved during the course of the design of the system. We briefly describe the system architecture we designed and its success as a vehicle for data collection for field biologists. Finally, we conclude by outlining lessons learned during the evolution of our design process.

Related Systems

In this section, we focus on data capture or data recording systems designed for handheld computers. The use of handheld computers, such as Personal Digital Assistants (PDAs), as data collection devices has been studied in a number of fields as diverse as Emergency Response [16], Learning environments [6] [13], and even in Human-Computer Interaction during usability trials [7]. Of particular interest to us is the use of handhelds in the ecological sphere.

One significant use of PDAs in ecological fieldwork environments is the work by Pascoe et al. for use in observing giraffe behaviour in Kenya and in support of archaeologists [12]. In their project, many of the characteristics of field biologist users were identified, including:

– Dynamic user configuration, specifically the fact that data capture occurs in hostile environments while walking, crawling, or running.

– Limited attention capacity, due to the need to record data while making observations.

– High-speed interaction, due to the fact that giraffes move and an observer may need to record a lot of data rapidly to capture a complete picture of giraffe behaviour.

– Context dependency, specifically the need to know location and timing information.

While the work of Pascoe et al. does identify many characteristics of users in fieldwork environments, their focus on ecologists observing animals has an effect on user characteristics. As well, while an understanding of users is important, there is a need to understand how these characteristics play out in design, and to develop methodologies for successful design.

Other related work that merits mention includes the use of PDAs as location aware guides in indoor environments, or as guide systems for use on university campuses. Finally, we note that many researchers are working on a complete understanding of context, in various fields, including scientific inquiry.

In these research systems, the focus of research is on how best to use handheld computers such as PDAs to support data collection tasks. The design process is not the focus of this work. Beyond the research domain, several companies design solutions for mobile data collection. One well-known company is Fieldworker1, which builds integrated solutions involving GPS, data servers, and PDAs to collect field data.

Fieldworker distributes a development environment to aid in the deployment of sophisticated applications. While reviewers have liked the advanced features Fieldworker Pro supports [15], there is a clear delineation to be made between Integrated Development Environments like Fieldworker Pro and software development processes which is our focus here. Fieldworker is a tool to support software development methodologies. The focus of this paper is a software development methodology which we have developed to design applications for mobile fieldworkers.

User-Centric Rapid Application Development

The goal of our project is the design of data collection applications for use in limited term fieldwork projects that are common in ecology and other biological fields. The limited term nature of these projects, ranging in duration from weeks to months, requires an agile software development process.

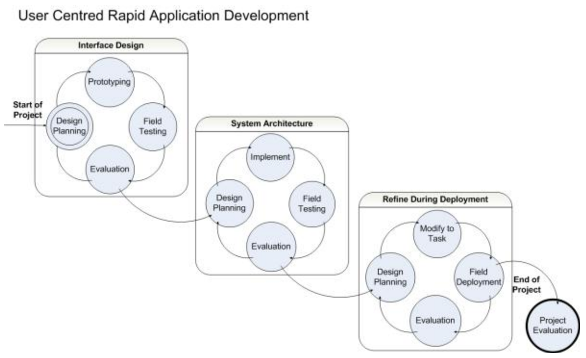
Agile development methodologies, including Extreme Programming, are difficult to manage. For example, in

Extreme Programming (XP), Bellotti et al. note that teams "must have a good grip on customer requirements ... by the time you engage in XP in order to prioritize engineering effectively" [2]. While XP does allow rapid development, the "Customer is King" aspect of design requires an ability, on the part of the customer or fieldworker, to prioritize features. This was absent in our development, as the ecologists had no experience with technology.

Working with them to prioritize engineering required providing them with some experience with the technology prior to requirements specification. Researchers also note that iterative development methodologies often result in poor quality software due to the need to specify system architecture and system logic early, in conjunction with evolving requirements [8]. This need to specify architecture first is also a characteristic of standard software development processes such as the Universal Software Development Process [10]. If the goal is to develop functional software in a short timeframe, eliciting requirements is often too time consuming a process to separate from development, but the need to specify a target for development still exists. When looking at RAD methodology, our goal was to combine software development with the elicitation of requirements and allow the prioritization of engineering to evolve with the project.

To compensate for a lack of requirements, we modified traditional RAD methodology to allow the early stages of development to focus on eliciting requirements fieldworkers. The process has three stages. In the first stage, high-fidelity prototypes are developed to design and evaluate the user interface and to manage project risk. In the second stage, the prototype is evolved into a functioning system by creating the back-end architecture. Finally, we deploy the full-featured application and continually tune the application to the evolving requirements of the fieldworker. Each stage is iterative in nature, and follows a basic RAD style deployment, where a prototype is developed, tested in the field, and evaluated via a joint meeting between clients and developers.

We designed this process for the express purpose of deploying a highly usable application in two to three weeks. We try to accelerate the development process.



Our software development process for field biology by giving biologists some experience using technology in the field early and to give developers, here a Master's student in Computer Science, some understanding of the requirements of the biological process early.

Early iterations occur in approximately four days. This works well in conjunction with many ecological projects. The typical data collection process we have observed in field ecology involves biologists spending one or two days collecting data in the field, followed by two or three days transcribing the data recorded on paper in the field into electronic format and doing some early analysis of the data. By matching our iteration to the biologists' data collection cycle, we prototyped a high quality user interface in approximately four iterations over a two week period and then added the back end application logic over a one to two week period. During the last week of the development cycle, the biologists used our application for data collection, but continued to maintain paper data collection as a back-up until the application development cycle entered the third stage.

There are two main goals to the first stage of development. First, functional and non-functional requirements must be developed for the overall project. Second, we wish to manage risk and determine whether the project will result in a worthwhile software artefact. To do this, we focus specifically on the design and implementation of a high-fidelity user interface. Focusing on the user interface allows us to accomplish our goals in a number of ways. Eliciting requirements in any domain is a challenging task, particularly when the users have no frame of reference. In our case, working with field ecologists, the most significant hurdle we faced was the lack of experience on the part of the ecologists with technology in the field. While we had significant experience designing applications, we lacked experience with ecological fieldwork environments. A common language for expressing requirements was missing, as was any experience on the part of the user with what technology would be useful in the field. Focusing on the development of a high-fidelity user interface allowed us to work with the ecologists to concurrently specify and validate requirements. Second, and equal to eliciting requirements is the need to manage user expectations and user buy-in. With high fidelity prototypes, we can represent accurately to our users the functionality of the final application, and allow our users to determine whether the application will be an effective tool for data collection. Third, in any user-centred design task, work context plays an important role. There is a need to specify both the tasks performed by the system and the constraints on that task that result from the surrounding environment. However, the introduction of technology has an impact on both the task and the environment, and we wanted to measure not only how data was currently collected, but how an application could alter the paradigm of data collection, and what were the liabilities associated with use of technology in the environment. We worked to understand the impact of our technology due to two factors that together determine successful design for mobile fieldworkers. The first is rapid data input. Fieldwork is the most costly component in any data collection task in the ecological sphere, and we need to ensure that electronic data capture is not significantly slower than paper-based data collection. The second is data integrity. Work in coastal wetlands involves the need to engineer against mishaps. We need to balance these two factors in our design, and evaluating technology in the field allows us to determine whether we have successfully engineered for rapid input and against mishaps. A final benefit in early focus on the interface is that it allows us to begin to prioritize features in the application more effectively. We see how the application will be used, and evaluate the expected benefit of each feature in the interface.

The goal of the second stage is the evolution of a non-functional prototype to a functional application. With a user interface to design toward, the process of adding back-end data capture is relatively straightforward in mobile data collection tasks. However, care must be taken to preserve the performance of the application and to ensure data redundancy. While a high quality user interface has been designed in the first stage, the interface must evolve to match the application logic.

References

[1] Ritu Agarwal, Jayesh Prasad, Mohan Tanniru, and John Lynch, "Risks of Rapid

[2] Application Development", CACM 43:11, November 2000, pp. 177–188.

[3] V. Bellotti, N. Ducheneaut, M. Howard, I. Smith, and C. Neuwirth, "Innovation

[4] in extremis: evolving an application for the critical work of email and information

[5] management", Symposium on Designing Interactive Systems, London, June 2002,

[6] pp. 181–192.

[7] J. Carroll, G. Chin, M. Rose and E. Neal, "The Development of Cooperation: Five

[8] Years of Participatory Design in the Virtual School", Proceedings of the Conference

[9] on Designing Interactive Systems 2000, New York, August 2000, pp. 239–251.

[10] Ciavarella, C. and Paterno, F. The Design of a Handheld, Location-Aware Guide

[11] for Indoor Environments. Personal and Ubiquitous Computing 8 (2004) 82–91.

[12] Chin, G. and Lansing, C. Capturing and Supporting Contexts for Scientific Data

[13] Sharing via the Biological Sciences Collaboratory. In Proceedings of the ACM

[14] Conferences on Computer Supported Cooperative Work, CSCW 2004, ACM Press

[15] (2004), pp. 409–418.

[16] Griswold, W., et al. ActiveCampus: Experiments in Community-Oriented Ubiquitous

[17] Computing. IEEE Computer 37, 10 (2004), 73–81.

[18] Hammontree, M., Weiler P. and Hendrich, B. PDA-Based Observation Logging.

[19] in Proceedings of the ACM Conference on Human Factors in Computer Systems,

[20] CHI 2004, ACM Press (1995), 25–26.

[21] Alan Howard, "Rapid Application Development: Rough and Dirty or Value-for-

[22] Money Engineering?", CACM 45:10, October 2002, pp. 27–29.

[23] Jiang, X., Hong, J., Takayama, L., and Landay, J. "Ubiquitous Computing for Firefighters:

[24] Field Studies and Prototypes of Large Displays for Incident Command",

[25] in Proceedings of the ACM Conference on Human Factors in Computer Systems,

[26] CHI 2004, Vienna, pp. 679–686.

[27] P. Kruchten, The Rational Unified Process – an Introduction, Addison-Wesley,

[28] Reading, MA, 1998.