User Centred Rapid Application Development

Edward Lank[1,2], Ken Withee[2], Lisa Schile[3], and Tom Parker[3]

[1] David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada
lank@cs.uwaterloo.ca
[2] Computer Science Department, San Francisco State University, San Francisco, California, USA
withee@withee.com
[3] Biology Department, San Francisco State University, San Francisco, California, USA
lschile@sfsu.edu, parker@sfsu.edu

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.

1 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

N. Guelfi and A. Savidis (Eds.): RISE 2005, LNCS 3943, pp. 34–49, 2006.

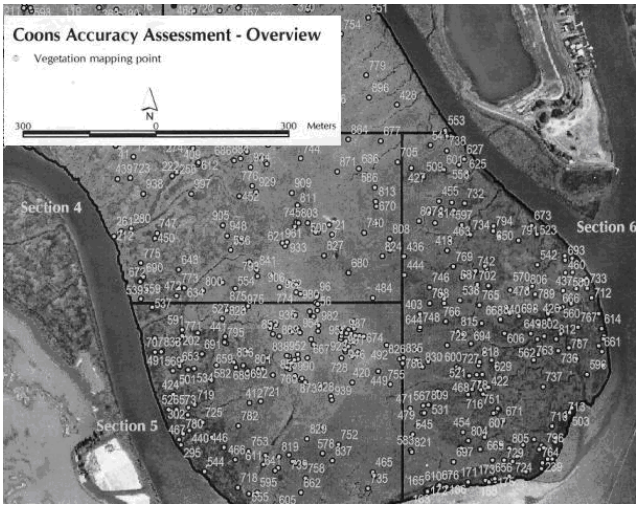

Fig. 1. Computer generated data collection points are displayed on a map, and the researchers collectively assign data points in the field

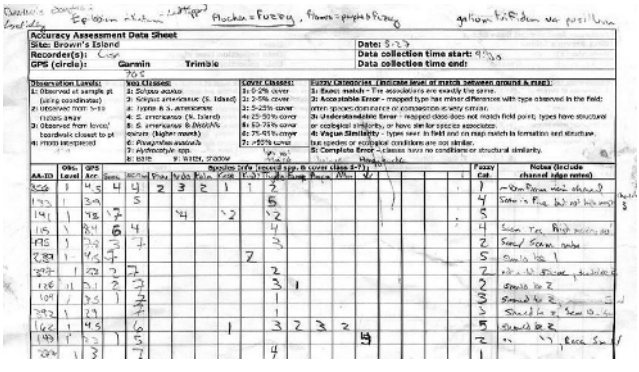

Fig. 2. Data sheet used to record field data. The data is captured using numerical scales.

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 (See Figure 2). 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 [1] [8].

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.

2 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 [4], or as guide systems for use on university campuses [6]. Finally, we note that many researchers are working on a complete understanding of context, in various fields, including scientific inquiry [5].

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 Fieldworker[1], 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

[1] http://www.fieldworker.com/

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.

3 User-Centred 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. Figure 3 depicts our RAD process for developing software for mobile 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

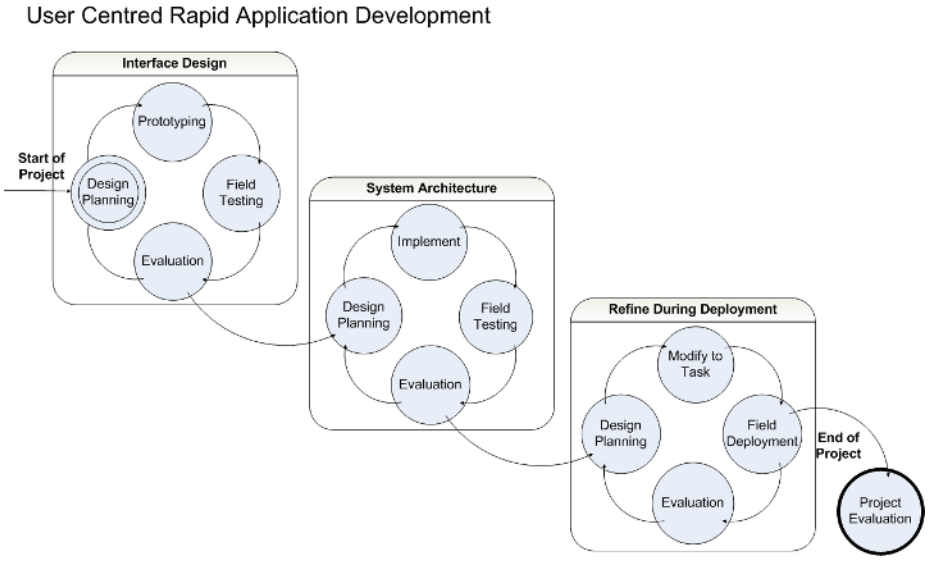

Fig. 3. Our software development process for field biology

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.

We separate application logic from interface development for two reasons. First, the application logic is a secondary concern to front-end system design in mobile fieldworker environments. The goal of the application is to replace paper as a data collection paradigm for field ecologists. Given that fieldwork is expensive and occurs in harsh conditions, the success of the application is determined by the user's ability to quickly and accurately record data. The ability of the application to archive that data once recorded is necessary but not sufficient for the success of the project. Where usability is the key determining factor in project failure, focusing on the user interface first allows us to manage risk. By focusing first on interface design and second on application logic, we maintain the

primacy of the user interface in successful application design. Second, beyond the management of risk, application logic naturally occurs at a different stage in development. We use high-fidelity user interfaces to both elicit and validate our understanding of requirements, while application logic, instead, instantiates the requirements in a functioning software application. By explicitly separating these stages, we maintain a focus on *what* the system should do (first stage), followed by a focus on *how* the system should accomplish its goals.

The final stage allows us to tune the application during deployment in the field. The projects we seek to design applications for are, as noted earlier, limited term fieldwork projects. As a result, deployment occurred early in the project's lifespan, typically after approximately three weeks. One result of early, rapid, deployment it that the application that is deployed is a work-in-progress. The application continues to be tailored to the data collection task our users performed.

4 UCRAD in Practice

In this section, we briefly describe the evolution of our target software system. We first describe the sequence of development tasks. We then describe in brief the development of our software system for field biology to show how we elicit, validate, and then implement the system requirements.

Figure 4 depicts the development process over the four week period resulting in a deployed software application. The target ecology project continued for approximately two months after the completion of development, and we continued

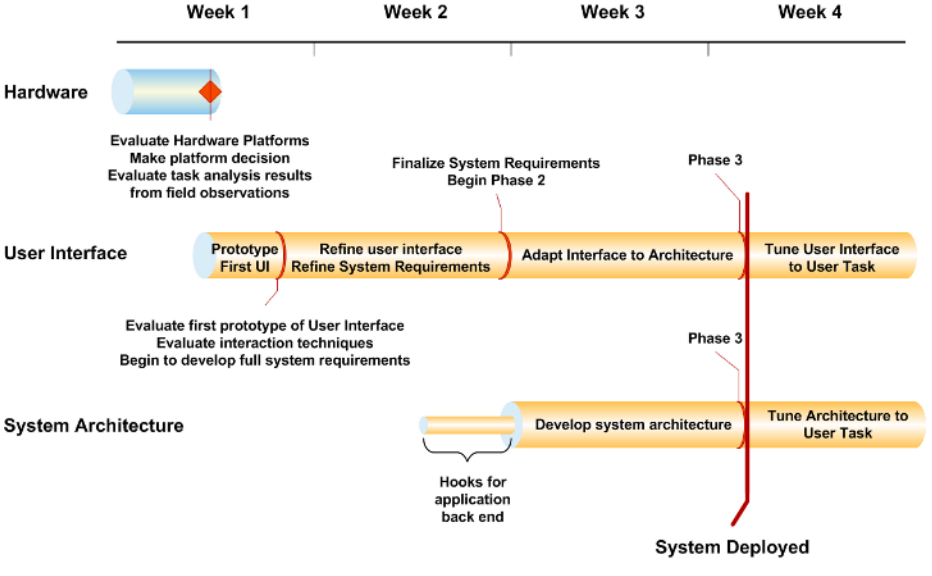

Fig. 4. In our methodology, software is designed from the front end to the back end

refining our application during deployment. As shown, we begin by selecting a platform, then design and implement the user interface. Back-end application logic is gradually developed and integrated into the application after validation of the user interface.

During the ini tial stages of user interface engineering, we worked with the biologists to understand exactly what the data collection process involved. In our development process, planning begins as a simple process of first interviewing the fieldworker about their data collection task and then observing that task in the field. Our initial planning involved a design meeting where the project goals were outlined, and hardware options were explored by the designers and users. Particularly during the first iteration, developers traveled to the field with the ecologists. This allowed the development team to evaluate different hardware platforms and to begin to develop a complete picture of the data collection task.

During the initial fieldwork with the ecologists, we spent two days working in the field. During this initial field outing, we validated the Pocket PC as an appropriate development platform. We considered other options, including Logitech electronic pens and tablet and laptop computers. Tablet and laptop computers are too expensive and to awkward for use by mobile fieldworkers, and electronic pens, while appropriate for fieldwork, work best for qualitative, not quantitative, data recording tasks. For quantitative data capture, handheld computers are the most effective application platform. During this initial fieldwork, observations of paper-based data collection tasks were also performed. We developed hierarchical task models, which we then validated with the ecologists during our evaluation meeting following our initial deployment. Being in the field watching the data collection process allowed us to develop a rich picture of the data collection task, the primary use case of our system. In validating this with the biologists, the basic requirements of the system were developed.

Using a drag and drop graphical user interface editor, we prototyped an initial user interface with no supporting back-end logic. The interface was designed to follow, as closely as possible, the specific actions involved in the ecologists' data collection task. During hardware evaluation, we also noted that, although the Pocket PC was an appropriate platform, stylus-based input was not appropriate as the stylus was frequently misplaced. The interface was designed to be used with fingers rather than with a Pocket PC stylus. The initial interface was developed in two days, and the biologists carried the interface with them when they went to the field and tested the interface for their data collection task.

We refined the interface over two additional iterations, and then began developing the application logic to support the interface components. During the deployment of a high-fidelity interface, several non-functional requirements came to light. For example, one of the ecologists noted during a design meeting, *"During our last field outing, someone fell and a GPS system got wet and stopped functioning. Many lessons were learned, [and we are even more nervous about technology in the field] ... Will that data persist even if we drop it in the water?"*

Fig. 5. Final user interface for our application

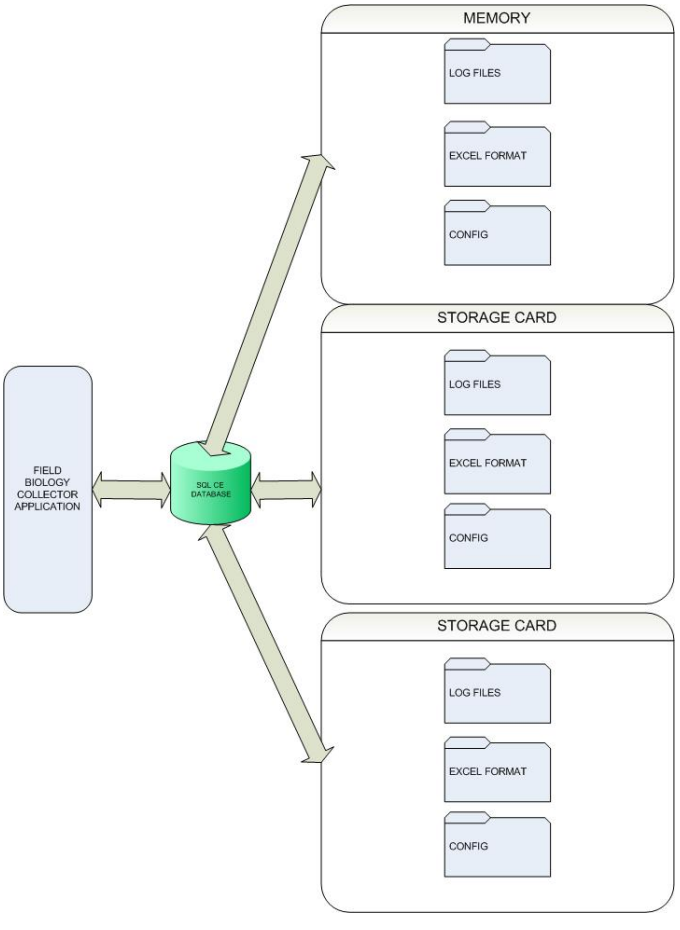

Fig. 6. Final architecture of our data collection application

In Figure 5, we show the final, functional application. The first two screens support data entry. Numerical entry is performed using a 10-key numeric keypad screen (not shown), while commonly found species are entered by selecting from a customizable screen that lists the species biologists expect to find at a given location. Another screen, not shown, lists over ninety-five species that are found in San Francisco wetlands. As we began to implement the back-end architecture, the need to support data collection came to light, and we added that functionality to our interface. When biologists select a previously entered data point from a list, they are presented with a screen that allows them to edit values associated with that data point.

After developing the core functionality in our interface, we designed and implemented a back-end architecture. The back-end architecture and hardware is designed to support data preservation in spite of submersion and other environmental hazards. In our system, we are using Dell Axim X50v Pocket PCs. These devices have random access memory (RAM) typical of Pocket PCs. They also have persistent internal storage in the form of 128MB of internal, writeable ROM. Finally, the Axims have both CompactFlash and Secure Digital (SD) Card slots. To protect the integrity of data, our system does several things. First, all data is stored in a SQL CE database. Second, each action by the user is added to a log file stored on persistent internal storage. Third, after each data point is added, the data is written to the database and a comma separated value (csv) file is written to the CompactFlash or SD cards if present or to persistent storage if these cards are not present. Typically, the ecologists we work with use SD

Fig. 7. Pocket PC application in use in the field in its waterproof case. In spite of the glare from the sun, the application can be seen faintly running through the waterproof membrane that permits interaction while protecting the Pocket PC device.

cards in all the Pocket PCs. Thus, even if the device falls out of its case and is submerged in water, the data collected by the ecologists is secure, and can be obtained either from the SD card or by reading the internal ROM in the Pocket PC. Figure 6 depicts the final architecture of our software system. To further protect the device, a waterproof case is used (Figure 7).

Modifications continued during the third phase. At one point, the data collection task changed slightly, requiring modifications to the interface. The storage format for data also changed several times, requiring a readjustment of the back-end architecture. By continually evaluating the software application, the system maintained a tight fit with the data collection task being performed.

5 Broader Lessons from Design

5.1 Successful Characteristics of Design

We continue designing field applications for use by biologists. From this design and early design of new applications, we note several characteristics of our design that assisted in its success. These include:

- Keeping features simple and data capture fluid.
- Deploying early.
- Rapid iteration.
- Over-engineering.

Simple features, as advocated by Extreme Programming, RAD, and other iterative methodologies, are an important consideration in design. Pen and paper are very successful in the field, and some biologists argue that it is because pen and paper only support the most basic interaction. As one user noted early in design: *"I can always transcribe in lab. I just need the data collected and I don't know how a computer will help."* Keeping features simple in the application allowed us to get an application into the field early, which is a second advantage. Many features that were planned prior to our design process were eliminated during our rapidly iterating cycle. As one example, early in design we proposed a set of web services to upload data. However, the use of a secure digital card in the Pocket PC proved sufficient. Our application currently saves two days of transcription time associated with each day spent collecting data in the field. Saving an additional ten minutes was a much lower priority. Plugging an SD card into a card reader connected to a PC and opening the CSV file in a spreadsheet program was sufficiently fast.

Early deployment, even of an imperfect interface, is not something to fear as long as the interface is solely for testing. To test our application in the field, we noted for biologists that it wasn't stable, that they should continue to use paper, and that all we wanted was what worked and what didn't for collecting data. Taking care in the design process to understand the task was important. The results of careful task analysis allowed us to develop and deploy a usable interface quickly. Small glitches in interaction were addressed through iterative refinement.

Rapid iteration through prototypes was another positive aspect of design in this domain. It is true that rapid prototyping results in initial applications that need significant improvement. However, in our case, as noted, getting software to the field quickly is a benefit in the limited term projects typical of field biology. As fixes are identified, correcting the software improves the effectiveness of technology in the field. Rapid iteration can continually improve prototypes, making technology even more useful over the course of the target project.

As well, early deployment, in our case, improved the transition of the ecologists we work with from informants to analysts and even to beginning designers. Once they understood how technology could be designed for the field, they were more able to engage actively in the design and evaluation process.

As a final comment, as designers in the field we noted that we underestimated how hostile the environment is. During one outing, a member of the design team was trapped in thick mud. On returning to the lab, he noted that he *"almost didn't make it back. I was sinking in mud, trapped, and I couldn't move..."* Data redundancy, protective cases, and other aspects of fault tolerant design are especially essential in biological fieldwork. Losing hardware is not critical. Losing time sensitive data is much more costly. Our rapid iterations and early deployment of semi-functional prototypes allowed us to develop an appreciation of the non-functional requirements of the system prior to full deployment.

5.2 Less Successful Design Processes

Paper prototyping, wireframes, or other low fidelity prototyping processes are one of the primary vehicles for cooperative and participatory design. It is a typical stage in many design processes, and some researchers have presented the results of low fidelity prototype testing as the validation for design guidelines in various domains [9].

The challenge with paper prototyping is that feedback on a design is suspect for highly mobile, context sensitive environments such as ecology. This is not a novel observation; Rudd et al. note that low-fidelity prototyping is not appropriate for evaluation, but is more appropriate for requirements elicitation [14]. Typical wireframe walkthroughs occur far from the context of use of software artefacts. While we did use wireframes in support of design meetings, we were careful to focus our evaluation and design planning on experiences with higher fidelity prototypes with functional graphical user interfaces that could execute in limited ways on the specific hardware platforms in the field.

Another common tool in interactive application design is participatory design, but it is a tool that must be used with caution. Our design process evolved out of a failure in participatory design. Researchers note that using participatory design is not straightforward. For example, Taxen notes that, in introducing participatory design into a museum environment there is a need to constantly validate products with respect to current work practice [17]. Carroll et al., in their work with schoolteachers, note a transition from Practitioner-Informants through Analysts to Designers over a five year period [3]. To participate in design, users need some experience with the technology in its use context. Rapid

prototyping allowed our users to experiment with technology in the field and this improved their ability to identify positives and negatives in requirements and early designs.

6 Broader Applicability of UCRAD

In analyzing the problems with RAD, Howard notes two different forms of RAD: Rapid program development, where a detailed software specification is quickly implemented in code; and Rapid System Development, where a kernel of an idea for a project is evolved into a functional piece of software [8]. Our goal, in introducing our three-stage RAD model, has been to support Rapid System Development. However, as Howard notes, Rapid System Development requires extensive user involvement to describe what the system needs to do, to react to proposals, and to validate prototypes. He also comments on its "chaotic" nature [8].

Much of this paper focuses around our work in supporting the capture of quantitative data by mobile fieldworkers, specifically, in this case, ecologists using handheld computers. The process described in this paper was designed for the development of handheld computing applications, and handheld computers are particularly suited to quantitative data capture in mobile environments. Handheld computers are less suited to the capture of qualitative, observational data.

Although this development process has only been used to design systems for data capture for field biologists, it seems to hold promise for Rapid System Development tasks where user involvement is needed to simultaneously specify and validate both requirements and implementation. In this way, it is particularly suited to domains where the users have no prior experience using technology. In such domains, users find it challenging to work with designers to specify requirements as they have no pre-existing frame of reference. Participatory design breaks down, and even the specification of requirements with the users is challenging because they are unconvinced of the merits of technology. By first prototyping and implementing the interface, we can capture user requirements more effectively. Simultaneously, users become more informed about what the technology can accomplish and can see directly the benefits of using the technology. The result of using and experimenting with the technology aids users' transition from informants to co-designers.

The use of evolutionary prototyping in the process, coupled with continual evaluation of the prototype, allows the risk inherent in any development process to be managed. Systems evolve that suit user needs. If a suitable system cannot be implemented, this becomes obvious early in evaluation as the user interface will fail to support users' tasks. We noted this failure when working with qualitative data capture. The interface for handheld computers was not as efficient as pen and paper for capturing qualitative data in another ecology project we explored.

One advantage we had in our system design is that the back-end application logic was relatively simple. The requirements were that it captures specific

data when entered in the interface and that it protects that data against environmental mishaps that would destroy the Pocket PC device. As the software architecture evolved, additional features were added to the user interface (e.g. a data grid that allowed direct manipulation of the database fields, a data correction screen that allowed users to select data points entered and correct values associated with them), but the core use-case could be fixed, validated, and supported immediately through user interface prototyping.

7 Conclusion

In this paper, we describe a modified form of Rapid Application Development called User Centred Rapid Application Development (UCRAD). Our development process is a three-stage process. We first focus on high-fidelity, semi-functional user interface prototyping with a drag and drop GUI editor to simultaneously elicit and validate requirements. We then evolve our limited prototype with its high quality user interface into a functional application. Finally, we deploy and continue to modify the application to tightly integrate with the appropriate task. Each stage in our process follows a Rapid Application Development methodology where implementations are tested in the field and evaluation is performed via joint meetings between developers and users. Our methodology has proved useful in the design of mobile, quantitative data collection applications for limited term biological fieldwork projects.

Acknowledgements

The work described in this paper is supported by National Science Foundation Award #0448540.

References

1. Ritu Agarwal, Jayesh Prasad, Mohan Tanniru, and John Lynch, “Risks of Rapid Application Development”, CACM 43:11, November 2000, pp. 177–188.
2. V. Bellotti, N. Ducheneaut, M. Howard, I. Smith, and C. Neuwirth, “Innovation in extremis: evolving an application for the critical work of email and information management”, Symposium on Designing Interactive Systems, London, June 2002, pp. 181–192.
3. J. Carroll, G. Chin, M. Rose and E. Neal, “The Development of Cooperation: Five Years of Participatory Design in the Virtual School”, Proceedings of the Conference on Designing Interactive Systems 2000, New York, August 2000, pp. 239–251.
4. Ciavarella, C. and Paterno, F. The Design of a Handheld, Location-Aware Guide for Indoor Environments. Personal and Ubiquitous Computing 8 (2004) 82–91.
5. Chin, G. and Lansing, C. Capturing and Supporting Contexts for Scientific Data Sharing via the Biological Sciences Collaboratory. In Proceedings of the ACM Conferences on Computer Supported Cooperative Work, CSCW 2004, ACM Press (2004), pp. 409–418.

6. Griswold, W., et al. ActiveCampus: Experiments in Community-Oriented Ubiquitous Computing. IEEE Computer 37, 10 (2004), 73–81.
7. Hammontree, M., Weiler P. and Hendrich, B. PDA-Based Observation Logging. in Proceedings of the ACM Conference on Human Factors in Computer Systems, CHI 2004, ACM Press (1995), 25–26.
8. Alan Howard, "Rapid Application Development: Rough and Dirty or Value-for-Money Engineering?", CACM 45:10, October 2002, pp. 27–29.
9. Jiang, X., Hong, J., Takayama, L., and Landay, J. "Ubiquitous Computing for Firefighters: Field Studies and Prototypes of Large Displays for Incident Command", in Proceedings of the ACM Conference on Human Factors in Computer Systems, CHI 2004, Vienna, pp. 679–686.
10. P. Kruchten, The Rational Unified Process – an Introduction, Addison-Wesley, Reading, MA, 1998.
11. James Martin, *Rapid Application Development*. Macmillan, New York, 1991.
12. Pascoe, J., Ryan, N. and Morse, D. Using While Moving: HCI Issues in Fieldwork Environments. ACM TOCHI 7, 3 (2000), 417–437.
13. Rogers, Y., Price, S., Fitzpatrick, G., Fleck, R., Harris, E., Smith, H., Randell, C. Muller, H., O'Malley, C., Stanton, D., Thompson, M., and Weal, M. "Ambient Wood: Designing New Forms of Digital Augmentation for Learning Outdoors." in Proceeding of the Third Interation Conference for Interaction Design and Children, IDC 2004, ACM Press (2004), pp. 3–10.
14. Jim Rudd, Ken Stern, Scott Isensee, "Low vs. high-fidelity prototyping debate", *Interactions*, 3:1, January 1996, pp. 76–85.
15. Nick Ryan, Jason Pascoe and David Morse, "FieldWorker Advanced 2.3.5 and FieldWorker Pro 0.91", *Internet Archaelogy*, Issue 3, Autumn 1997.
16. Sawyer, S., Tapia, A., Pesheck, L. and Davenport, J. Mobility and the First Responder. CACM 47, 3 (2004), 62–65.
17. Taxen, G., "Cases and experiences: Introducing participatory design in museums", Proceedings of the eighth conference on Participatory design, Toronto, July 2004, pp. 204–213.