An Interdisciplinary Approach to Emergency Responder Mobile Technology Design

Patricia Collins[1] and Sean Lanthier[2]

[1] Carnegie Mellon University Silicon Valley, Moffett Field, CA 94035
[2] IC[3], LLC, Los Altos, CA 94022
patricia.collins@sv.cmu.edu,seanlanthier@me.com

Abstract. Using an interdisciplinary approach to requirements engineering for an emergency responder support system ensures that system capabilities as well as such considerations as usability and utility are based on the clear needs of emergency responders and other stakeholders. The described methodology integrates brainstorming, competitive assessment, user interviews, scenario development, use case development, mobile device user interface mockups, and emergency responder validation of the resulting requirements. The approach was employed to consider diverse stakeholders in emergency response situations.

Keywords: first responder, emergency responder, paramedic, interdisciplinary design, requirements engineering, emergency response, disaster response.

1 Introduction

Getting requirements right for emergency responders mobile technology needs can save lives and reduce injuries. Therefore, it is worth the effort to take an interdisciplinary approach to gathering, validating, and analyzing candidate requirements for products and services that aid emergency responders.

Carnegie Mellon University Silicon Valley (CMUSV) has undertaken such an approach to requirements engineering for emergency responder mobile technology design, following an Agile software engineering methodology. The aspects of the methodology which meet the Agile guidelines [3] include ongoing customer collaboration, rapid prototyping, accommodation of change, and a focus on individuals and their interactions in a high-performing team.

1.1 Software Engineering

Laplante notes that software engineering is a "systematic approach to the analysis, design, assessment, implementation, test, maintenance and reengineering of software, that is, the application of engineering to software."[9] It differs from computer science as an academic discipline in that software engineering is focused on methodologies, processes, and techniques for each aspect of the lifecycle, while computer science is

K.S. Nikita et al. (Eds.): MobiHealth 2011, LNICST 83, pp. 284–291, 2012.

the study of theoretical concepts in computer-based computation and information management. The study of emergency responder mobile technology needs took place within the context of an Agile software engineering methodology. This paper describes only the requirements engineering aspect of that lifecycle.

1.2 Requirements Engineering

Software requirements engineering includes the acquisition, validation, analysis, selection, and managed evolution of software requirements for targeted software-based systems. In Agile approaches to software engineering, requirements engineering is an ongoing process, with refinements and modifications occurring just in time for implementation of the corresponding part of the system. However, most Agile approaches do not explicitly address the initial exploration of requirements, which is part of the product visioning process. [12] The early development of prioritized candidate requirements is, nevertheless, essential to understanding the user's needs. CMUSV focused its requirements engineering efforts on the identification of requirements for a product line that would support emergency responders with mobile technology. Towards this end, the team engaged in an interdisciplinary approach to requirements engineering. This approach enabled small requirements engineering teams to gain a well-rounded understanding of the domain and the potential product space, before launching into Agile development.

2 The Interdisciplinary Approach

An interdisciplinary approach is best matched with the goals of deep understanding of design requirements. User-centered design often focuses on interviews with and observations of the target users. [4, 8] This narrowly scoped approach does provide a clear understanding of the user's current work flow, task sequences, and work environment. However, the opportunity to discover creative and innovative solutions is often limited to addressing the identified "breakdowns" in the current way of doing things. To support more innovative solutions, one can combine an assortment of requirements engineering approaches. The following subsections describe the interdisciplinary nature of the various approaches used to discover and validate software requirements.

2.1 Competitive and Analogous System Analysis

Requirements engineers (REs) can begin their investigation by looking at competitive systems. Using an analytic approach, the RE reviews the brochures, technical specifications, and other information about each competitive system. In the case of mobile technology for emergency medical responders, the current market is mostly limited to isolated applications. Therefore, the analysis involved studying partial solutions and identifying candidate features for a more comprehensive solution. [10] In the early stages of the competitive analysis, the REs added all relevant features to

the candidate requirements list. Then, as the team's notion of the scope of the product and product line evolved, they marked these candidate requirements with a proposed priority.

Analogous systems are systems that serve a somewhat different purpose than that of the proposed product, but they are instructive to consider for relevant features. [Eclipse 2010] Analyzing analogous systems involves a creative and even playful mindset. For example, mobile technology for emergency responders almost certainly involves communication and collaboration support. Analogous systems might include meeting management tools, such as Avaya's Flare [2] or Adobe's Acrobat Connect Pro [1]. These systems do not solve the emergency responders' problems, but they are used for other kinds of communication and collaboration. Each RE team looked at these systems (and others) as sources of inspiration, rather than as competitive solutions, identifying features that might translate into corresponding features in the imagined emergency medical services product line. In fact, while the analysis of competitive applications yielded only requirements for point solutions, the study of analogous systems identified the need for a unifying platform that could enable such point solutions to be integrated.

2.2 User Interviews and Observations

Using ethnographic techniques as defined in the contextual design methodology [4], we conducted interviews with fourteen emergency responders. In some cases, it was possible to observe emergency responders at the fire station, where we could gather additional information about the emergency responders' culture and environment. In all cases, the interviews relied on prepared sets of open-ended questions, improvised as necessary to get at the heart of issues that emergency responders currently face in their jobs. About half of each interview focused on the participant's current job-related tasks. The other half of each interview focused on the participant's thoughts about how technology might improve his ability to carry out his job safely and efficiently.

2.3 Brainstorming

Brainstorming uses yet another kind of intellectual skill. The REs conducted their brainstorming activities after completing the competitive systems analysis, analogous systems analysis, and user interviews and observations. By this time, the REs had enough understanding of the domain that they could use intuition and inspiration to come up with additional features for the emergency responder mobile technology.

2.4 Scenario Development

In order to validate the requirements with stakeholders, the REs next developed scenarios of use. [13] This was carried out on a creative writing exercise that relies on the RE to tell a story, in which personas (based on character sketches) accomplish a goal, using the envisioned emergency responder mobile technology. The scenario

describes the essential features or capabilities of the system in terms of how the user might interact with the system. In the process of developing the narrative, it's common for the RE to discover additional requirements. This generally occurs when the writer reaches a point in the narrative where the emergency responder needs to communicate or collaborate in a way that the existing requirements do not address. Nevertheless, the primary value of the scenarios is that they can be used with stakeholders to ensure that the requirements match the emergency responders' needs.

2.5 Use Case Development

Once exemplary scenarios have been vetted by stakeholders, requirements engineers may choose to analyze the software implications of each scenario in terms of use cases. [5] In this project, the REs identified critical and complex use cases to be "fully dressed." This practice involves selectively diving into the fine points of how the user and system interaction should take place. For more straightforward use cases, Cockburn recommends more "casual" or "brief" coverage of the use case—identifying the primary flow of interaction between the user and the system, without the need to document the more detailed paths of execution that the user and system might undertake. This is in keeping with the Agile philosophy in which the details of the requirements are not created early in the software engineering lifecycle, but rather are elaborated just in time for design and development. In this project, the requirements engineers were all experienced software engineers. Therefore, they were able to review each other's use cases to analyze the implications for software design. And while use cases are most commonly referenced by software engineers, they are readily understandable by target users.

2.6 User Interface Mockups

To complement the scenarios, the REs developed sketches of key user interface (UI) considerations. Today, there are many open source tools available to support rapid development of user interface mockups for mobile technology (e.g., Balsamiq). In the emergency responder application domain, however, UI mockups may involve more than capturing a graphical user interface. In some circumstances, like densely smoky fires, a visual interface may be unusable. Therefore, the REs needed to mock up audio dialogs that might be used when visual interaction is unfeasible. These mockups must be validated with target users. The REs discovered the need for significant refinement of their early mockups after running through a scenario with illustrative GUI mockups, reviewed by actual emergency responders. Practical issues, such as the use of thick gloves during some emergency response activities (e.g., extracting an injured passenger from a car), meant that the capacitive touchscreen might not be a usable part of the interface. Other user feedback identified the need to very simple screens with maximally differentiated "buttons" (e.g., for vacate, utilities on/off, and man-down alerts).

3 Discussion

Each requirements engineering technique contributed uniquely to the acquisition or validation of requirements for emergency responder mobile technology needs. Competitive systems analysis led to the identification of features and capabilities such as situational awareness support and UI navigation support. In fact, the envisioned product line design was significantly influenced by the competitive analysis. With the wealth of emergency medical services (EMS) applications available, the REs recommended that the platform support integration of best-in-class EMS apps, rather than building proprietary versions of commercially available functionality. Medscape, for example, has produced a wealth of applications for EMS personnel, including a drug reference, drug interaction checker, disease and condition reference, and procedures and protocols reference. [11]

Analogous systems analysis clarified thinking about communication and collaboration technology. As an example, some online meeting collaboration tools support maintenance of a complete log of information exchanged during a meeting. The REs recognized the potential value of maintaining a rich log of incident information that could later be referenced for after-incident critiques and for training. This novel capability for EMS personnel translated into clear system requirements.

User interviews revealed details of the protocols the technology would need to support. For example, when paramedics have search and rescue responsibilities during a fire or hazardous materials (hazmat) incident, they must carry out a manual check-in protocol, the Personnel Accountability Report (PAR). Currently, this protocol involves the use of push-to-talk radios, with the incident commander (IC) initiating the PAR request to each team captain. The team captain then checks with each person on the team. Due to the bandwidth limitations of the radios, the report-back involves slow, sequential, verbal acknowledgement of status. The interviews, however, revealed that first responders and ICs would be well served by an automated PAR protocol, one that automatically generated the PAR requests at fixed intervals, tracked each first responder's acknowledgement, and reported the information to team captains and IC. The target users also provided guidance on the prioritization of capabilities like real-time alerts, multi-way real-time voice communications, and multimedia information sharing. Interviews with paramedics and emergency response medical doctors uncovered the potential benefits of real-time image sharing, where the paramedic could send a photo of a patient at the incident scene to the medical doctor for timely consultation and collaborative assessment of the situation.

Brainstorming, done in the context of what had already been learned about emergency responders' needs, enriched the set of candidate requirements with such feature ideas as searchable incident logs and the identification of nonfunctional (quality) requirements, such as software availability and reliability. The potential danger of RE brainstorming is that the engineers are not experts in EMS or overall emergency response. Therefore, each brainstormed requirement needs to be validated with target users. In this project, the entire set of nonfunctional requirements were reviewed and refined with the help of a veteran firefighter/paramedic. It will be essential to develop prototypes that meet these nonfunctional requirements, because first responders and ICs are often kinesthetic

learners—people who understand best when they have the opportunity to interact with a tangible implementation. (In fact, in further work that built on this project, a small team developed a rapid prototype of the alert and acknowledgement capabilities and validated what would be an acceptable real-time performance with the same veteran firefighter/paramedic.)

When it came time to validate the candidate requirements, the REs focused first on the development of personas. By creating characters that spanned the diversity of technology aptitudes and attitudes of real emergency responders, it was possible to test out the feasibility of adoption of the gathered requirements. For a technology-naïve, 55-year-old paramedic persona named John, the question in determining the usability of a feature became, "Would this feature be easy for John to use?" This proved to serve as a reality-check on some of the more grandiose and technologically complex candidate requirements. In numerous conversations with first responders, the REs heard repeatedly that it is essential to keep the UI as simple as possible. Furthermore, the first responders cannot be distracted by their mobile phones from carrying out their primary responsibilities.

REs developed scenarios for medical emergencies, as well as other types of emergency incidents. By writing a complete story of a realistic emergency incident, each RE was able to recognize when a requirement was really unnecessary for that type of situation. As well, the unfolding story identified missing requirements. For example, one RE team created a rich incident scenario that involved an explosion and fire at a pharmaceutical plant. As the scenario unfolded, it became clear that off-duty EMS responders might need to be called to respond to such a major incident. This resulted in a novel set of requirements for how EMS personnel get timely notification and give timely responses when they are off duty.

The development of use cases was probably the most technical of the requirements engineering tasks, because the use case developers had to think in terms of the software system requirements in greater depth. While this approach to modeling requirements has a certain appeal to some detail-oriented software developers, Agile methodologies encourage a lighter-weight documentation, such as the user story. [Cohn] In hindsight, significant effort could have been reduced by adopting user stories instead of use case documentation, without a reduction in understanding of the problem space. A follow-on prototyping effort for some of the highest priority requirements revealed that it was easy enough to prototype, evaluate, and revise the user-system interaction design without reference to a use case document.

While it is very helpful to go through a scenario with target users in order to confirm the vision of the emergency responder mobile technology, we found that UI mockups were also helpful in getting target users to provide specific feedback on the user-facing requirements for the system. Like early prototypes, the UI mockups give the target user something specific to react to. For those who say, "I'll know what I like when I see it," UI mockups are especially important. Mobile screen mockups of maps, for example, demonstrated the need for simple zoom capabilities. The small footprint of the mobile screen made it clear just how little information could be provided at once and still be visible and interactive.

The REs were fortunate to have a 31-year-veteran firefighter/paramedic as the "onsite customer" representative. This provided a sanity check on the evolving product vision, requirements, and prototype.

The primary challenge to implementing this interdisciplinary approach is finding requirements engineers with the aptitude and inclination to tackle the wide variety of tasks that are involved in generating a comprehensive, validated, prioritized set of system requirements. Nevertheless, this project, which involved a class of Carnegie Mellon University Software Engineering graduate students, demonstrated that these skills are readily learned and applied in a very short period of time (seven weeks from requirements engineering inception to completion of all deliverables).

4 Conclusions

A multi-pronged approach to requirements engineering results in a richer set of functional and nonfunctional requirements. With such an approach, it is much easier to identify the requirements of an assortment of stakeholders, to ensure a more innovative and powerful solution, and to validate that the requirements will meet the stakeholders' needs. The use of an *interdisciplinary* approach accommodates the variety of ways in which stakeholders may best understand requirements, whether that means reading narrative scenarios, viewing mockups, walking through use cases, or reviewing lists of requirements statements.

The methodology described in this paper has been demonstrated to support a variety of stakeholders in the requirements engineering efforts for mobile technology support for emergency medical services. The deliverables from this project have been successfully applied to the development of a first responder application, software system architecture, and technology roadmap.

Acknowledgements. Professor Reed Letsinger, co-designer of the Carnegie Mellon University Silicon Valley Requirements Engineering course; the CMUSV Software Engineering students of the Spring 2011 Requirements Engineering course; and the fourteen first responders, fire chiefs, and other emergency response stakeholders who participated in interviews and requirements validation activities.

References

1. Adobe Connect 8, `http://www.adobe.com/products/adobeconnect.html`
2. Avaya Flare Experience Guided Tour, `http://www.avaya.com/usa/campaign/avaya-flare-experience-guided-tour/`
3. Beck, K., et al.: Manifesto for Agile Software Development. Agile Alliance (2001), `http://agilemanifesto.org/`
4. Beyer, H., Holtzblatt, K.: Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann, San Francisco (1997)
5. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley Professional, Boston (2000)

6. Cohn, M.: User Stories Applied: For Agile Software Development. Addison-Wesley Professional, Boston (2004)
7. Guideline: Requirements Gathering Techniques. Eclipse (November 20, 2010), http://epf.eclipse.org/wikis/openup/core.tech.common.extend_supp/guidances/guidelines/req_gathering_techniques_8CB8E44C.html
8. Holtzblatt, K., Wendell, J.B., Wood, S.: Rapid Contextual Design: A How-to Guide to Key Techniques for User-Centered Design. Morgan Kaufmann, San Francisco (2004)
9. Laplante, P.: What Every Engineer Should Know about Software Engineering. CRC Press, Boca Raton (2007)
10. Leffingwell, D., Widrig, D.: Agile Software Requirements: Lean Requirements Practices for Team, Programs, and Enterprises. Pearson, Boston (2011)
11. Medscape App for Android. WebMD, http://www.medscape.com/public/android
12. Pilcher, R.: The Product Vision. The Scrum Alliance (January 9, 2009), http://www.scrumalliance.org/articles/115-the-product-vision
13. Sutcliffe, A.: Scenario-based Requirements Engineering. In: Proceedings of the 11th IEEE International Conference on Requirements Engineering, pp. 320–329. IEEE, Washington