The Challenges in Creating Tools for Improving the Software Development Lifecycle

Susan L. Spraragen
IBM Watson Research Center
Hawthorne, NY 10532
+1 914 784 7194
sprara@us.ibm.com

ABSTRACT
Creating successful software systems for end user applications is a complex task. It is often proposed that tools can be built for development teams to help them do their job more efficiently and to help them communicate with their team members. The success of these tools relies on how well the technical community that builds software tools understands the needs of the technical community that uses these tools. How can we effectively apply a user centered design approach to building these tools?

General Terms
Design, Human Factors

Keywords
User Centered Design, Communication, Software Development

1. INTRODUCTION
When we develop applications with the end user in mind, we consider the tasks they need to perform. Some end user tasks are rather straightforward: making an online hotel reservation, logging onto a system to check mail, conducting a preflight checklist, preparing a monthly revenue report. These tasks have an order associated with them; they have a start point, a middle, and end point. Their outcomes are rather well defined. When we develop software tools for developers, their tasks are more fluid: modeling the data terms that flow across multiple systems, building pervasive applications, creating a web service. These tasks are not so orderly nor are their end results so predictable. Enabling and improving the productivity for these complex tasks requires a thoughtful design practice.

2. AN APPROACH
My position on just how to create useful tools for improving the software development lifecycle and for improving the end product begins with considering users as participants. It is imperative to engage in discussions, interviews, and observations with our partnering technical community, in order to gain a better perspective about their work, where they spend the most time, and how the costliest mistakes are made. Then once we have an understanding of their domain, we can address how to improve their productivity. Much has been written about participatory design practices [4, 7], and here I site an actual case where these methodologies proved to be successful.

To achieve success in these engagements requires gaining an appreciation for how the teams work and interact amongst themselves and across their organization. There are many steps in the software lifecycle process, often designated to various team members across various geographies who may not share a common mindset about the project [3]. If we create tools to improve the development lifecycle, then one feature of the tool should aid in the coordination of activities amongst different functional areas of the project.

At the same time, as we, the tool providers, interact with a similar but different technical community, we need to encourage developers to communicate with us effectively. It is wise to consider possible sources of hesitancy one may encounter. It is valuable to provide some guidance or outline to these discussions so real problems can be revealed. One way to engage in such discussions is to have developers focus on the areas that they own [1]. Alternatively we may take a topic such as data flow and learn from each team member how they need to deal with that piece. As such, during this walk-through, usability practitioners have an opportunity to interject questions about how to make improvements and to ask developers how their end user population will also be interacting with the data.

These communications are important to continue throughout the development of the tool, to ensure that we translated their needs into features that they find to be useful. It also serves as a terrific means for prioritizing the many features that could be added to the tool but that may not actually be needed. This gives us some valuable, concrete validation of the approach and premise for the tool.

3. AN EXAMPLE
Considering an example of the importance of clear communication, we developed a data modeling tool that was designed to capture the knowledge of subject matter experts in a manner that would be useful for developers to immediately grasp. Often IT architects or database administrators collect the data requirements and hand them off to the developer in any number of formats. We wanted to provide them with a method for creating an XML Schema from their data dictionaries, without having the concern of writing the syntactically correct schema themselves. We also wanted to see whether this tool could be an effective

Human and Social Factors of Software Engineering (HSSE) May 16, 2005, St. Louis, Missouri, USA

means of communication during the requirements gathering stage [6].

Data managers and IT architects need to communicate with their colleagues during the course of a project. The usefulness of a data modeling tool goes beyond what we provide just for them. Our tool provides a basic image that describes the data relationships, and in addition to XML schema, we supply HTML output describing the data that can easily be viewed in any browser. With this variety of outputs, their work becomes more accessible to others. Thus the team members can achieve more of a common ground for determining and pursuing shared goals of the project, even when they don't physically work side by side.

Once we had a prototype, we needed better focus and validation, so I attended a conference for data management experts. I began to understand how they gather requirements, interview subject matter experts, and how they work within the constraints of their organization when trying to communicate with various stakeholders. It was there where I found our participant, and where I heard that the most time consuming aspect of gathering requirements is in gaining consensus on the data. So as we were trying to advance the usefulness of our data modeling tool, I quickly realized that this was a real issue our tool could address. Many other issues arose as we began working with our participant, which mapped into the features of our modeling tool. This work requires the developers of the tool, to step back from their technology, and think about how to present useful features to a different technical community.

Along with engaging with a participant, we established some basic design principles:

1. We proposed a lightweight tool, with minimal training required, that could be used immediately to provide some spark to the development process.
2. We made concerted efforts to lower the barrier of entry by using simple interaction techniques, and terms that were generally familiar. With a basic table layout and context centered graphics, our participant could see the data dictionary under construction and how the terms related to each other.
3. We made efforts not to restrict or interrupt the flow of thought. Simple considerations, such as limiting unnecessary mouse movements kept the user's focus in one area.

The tool needs to support the developer's work style without over imposing on the work they need to do. Placing the fulcrum for balancing between providing helpful guidelines or wizards, and cramping the open style and fluidity of how someone actually works, and needs to communicate about their work, takes finesse.

Software architecture has been compared to building architecture in terms of how these fields may share participatory design practices [2, 5]. Here too, for this data management project, we became building architects, in a sense. We had to survey the property, see how to position our structure, size and place the doors and windows, while providing a strong roof and foundation that would hold all the pieces together. Each step of this process includes the participation with the inhabitants of this structure. We were fortunate to observe how well one person lived within this environment we created with our data modeling tool, but we have yet to see, more completely, how well it can accommodate others on the team.

3.1 Building the next tool

How can the experiences from this project extend to a larger team that wants to build a tool for developers, not data managers? Here we have a group of researchers creating something that should be useful for another technical community, this time a group of developers. This set of users have the complex job of building pervasive applications that need to work on different types of devices and that need to communicate with different types of servers. The requirements, protocols, device dependency issues, and data issues make for a complex development scenario.

The research team developing the tool already has leanings towards how developers work, as they perform similar tasks themselves, as programmers. So it is likely that a programmer, who is developing the tool, could offer any of the following remarks: "Well, this is how *I* would do it!" or "I can *imagine* someone doing it this way…" or "We can ask our *friends* what they think about this approach." The common piece missing from these falsehoods is the connection with the actual user of the tool. This is where the need for a usability focus becomes clear.

One role of a usability expert is to forge an ongoing relationship between these two communities. For the initial phase of creating a useful tool, we can validate research notions through interviews, user studies, and observations made with the targeted community. However, translating those findings in a timely, credible and useful manner is tricky. Later on when there is a prototype, the value of these kinds of exercises becomes more apparent. But in the early stages, the usability role becomes more of an ambassador role where we present the two communities with opportunities to get to know each other. Through interviews and joint discussions, we may find a person who currently develops pervasive applications and would serve as a participant throughout the course of the project. As these communities have geographic, stylistic, and work requirements that are divergent, making this match is a formidable challenge – but a necessary one to meet.

Care must be taken here while working with a prospective user who provides guidance for improving the tool. The research team needs to balance what may be identified as an important feature from the user perspective with features that will also serve a broader community of developers. As researchers, we are driven to be forward thinking and as such we need to do more than improve the developer's existing work practices. So by combining efforts made by quizzical researchers, with the thoughtful considerations provided by usability experts, with the practical suggestions supplied by our end users – we may very well have a productive and proactive merging of the minds - all uniquely contributing to the creation of a useful and useable tool.

4. CONCLUSIONS

Understanding how and when to apply usability techniques and findings to a project is not always obvious. We, in the human factors community, want to be a valuable and positive part of the development process and we need to understand and be aware of a variety of technical communities and concerns.

In these examples, the users in both these tooling projects are neither novices nor beginners. They are rather sophisticated in

their work, and what we provide them has to respect that sophistication. This is true not only for what we deliver as the end product but for how we progress throughout the evolution of the tool.

5. ACKNOWLEDGEMENTS

Thanks to Douglas Lovell who implemented the data modeling tool sited in this paper and who expressed a clear understanding of the role of a usability expert on a development team. In addition, I extend my appreciation to Danny Soroker for asking questions that motivate my position and for reviewing this paper.

I have been designing and implementing software systems in the forms of kiosks, web applications, and data modeling tools for fifteen years at the IBM TJ Watson Research Center. Here I have combined notions of psychology, programming techniques, user interface techniques with ethnographic observations in order to create useful and useable systems.

6. REFERENCES

[1] Beyer, Hugh and Holtzblatt, Karen. *Contextual Design; Defining Customer-Centered Systems*, Morgan Kaufman Publishers, San Francisco CA, 1998.

[2] Brooks, F.P., *The Mythical Man-Month*, Addison-Wesley, New York, 1995.

[3] Cohen, Cynthia F., Birkin, Stanley J., Garfield, Monica, and Webb, Harold W., Managing Conflict In Software Testing, *Communications of the ACM, 47, 1,* (Jan. 2004), 76-81.

[4] O'Neill, E., Johnson, P. Participatory Task Modelling: users and developers modeling users' tasks and domains, *TAMODIA '04*, (Nov. 2004), 67-74.

[5] Rank, S., O'Coill, C., Boldyreff, C., Doughty, M. Software, Architecture, and Participatory Design, *WISER '04*, (Nov. 2004), 45-48.

[6] Spraragen, Susan L. and Lovell, Douglas. Document models and XML Vocabulary Building for Business Users, *Proceedings of the 2004 XML Conference*, IDEAlliance, 2004.

[7] Winograd, Terry. From Programming Environments to Environments for Designing, *Communications of the ACM, 38,6,* (June 1995), 65-74.