The Role of Blogging in Generating a Software Product Vision

Shelly Park, Frank Maurer
University of Calgary, Department of Computer Science, Calgary, Alberta, Canada
{sshpark, fmaurer}@ucalgary.ca

Abstract

The core problem with requirements engineering is that often even the customers have no clear idea what they need; they don't know how to express it; or even if they express it really well, what they thought they need wasn't what they really need. Despite having the technical skills and being able to speak the domain language, generating requirements for software developers by the developers is found to be quite a difficult task. We discuss four types of strategies for expressing one's desire for requirements. We analyze how stories turn into consensus.

1. Introduction

It is estimated that 85% of the defects in software originates from the requirements [1]. Part of the problem is vague requirements due to a customer's uncertainty about their needs, which account for 49% of the requirements problem [1]. The core problem is often even the customers have no idea what they need; they don't know how to express it; or even if they express it really well, what they thought they need wasn't what they really need. Rittel and Webber defined a "wicked problem" as figuring out what the problem is the problem. Wicked problems have no stopping rule; solutions to wicked problems are not true or false, but good or bad; there is no immediate and no ultimate test of a solution to a wicked problem [2].

Software developers get trapped in a wicked problem dilemma when they become a customer or a domain expert and need to generate requirements for other developers. Despite having the technical skills and being able to speak the domain language of the software development discipline, generating requirements is found to be quite a difficult task.

This is a short position paper that looks into the knowledge creation in product visioning process for a software tool. Particularly, we are looking at how stakeholders express their wishes for a new tool that no one has yet seen. What makes this particular tool visioning process interesting is that it is happening over the Internet through blogs and online forums.

We are starting to look at the role of blogging for software tool visioning process, the types of knowledge that people present in this media form, how people express their wishes and what kind of requirements are being generated for the tool through this communication form. Basically, what we are analyzing is a process of discussions and consensus building that is something between pure anecdotal evidence/single expert opinion and hard-core empirical studies as in a traditional scientific approach. In a sense, this Internet-based process can be seen as an alternative to traditional peer review in science. It may be faster but it is more error-prone process. This paper will not delve into the actual tool requirements that were generated so far, but discuss the *processes* and *behaviors* observed during the requirements generation through this communication form.

2. Motivation

The tool that we are studying is an executable acceptance testing tool or story testing tool. Recently, there is a strong desire in the requirements and testing communities to create a link between testing and requirements engineering, because they both have a lot to gain from each other [3]. The concept of creating an executable and testable requirements specification recently gained a lot of interest and the Agile software development community is organizing various workshops to inspire a group of volunteers to build a tool that can help write requirements in a testable and executable way. Andrea [4] recently published an article describing what her vision of such tool looks like.

Our interest in this tool building process is multi-fold. While we were ultimately interested in joining this group to help build such tool, we were also interested in observing the social phenomenon that was driving this loosely associated group of developers to come up with the requirements for such tool(s). Although they shared a very similar vision of what the success may look like, everyone had a very different view of what the implementation may look like. In essence, these loosely associated practitioners were stakeholders for a tool building project that is still trying to discover what its requirements should be.

The first generation of tools that mapped the requirements to the software implementation was developed already several years ago. For example, Fit [5] is one of the first generation of such tools. Some developers used unit tests to achieve the same goal. However, practicing this idea in real development projects was much harder, because these tests needed to be read and written by customers who have no software engineering background. There was also the question of how much is good enough.

Through trial and error, many practitioners gathered empirical knowledge or often just hunches of what works and what doesn't. What is fascinating about this process is that this loosely associated and globally distributed group of practitioners are continuously generating and adding more knowledge to the community without a centralized control or a centralized repository, but rather through many blog sites and message boards. A blog entry from more notable bloggers would trigger discussions. The blog readers would respond to the blog entry by creating another new blog entry. Such actions would propagate through various development communities.

Because the knowledge is scattered throughout the Internet, we started to collect and compare people's opinions and views. We want to find out what the requirements for this tool look like once we collect everyone's opinions and how the consensus was reached in this online community.

3. Methodology

The research began when we participated in the first functional testing tool workshop organized by Agile Alliance in October 2007 [6]. This particular tool-building community keeps track of each other's progress mainly through a message board [7]. We started our data collection by going through the entries in the message board and branching to blogging sites as they were recommended by the users in the message board. The blogging site may recommend another blogging sites, then we branched again to collect more data from the recommended blog sites. By doing so, we had a collection of social networks of people who read and had influences on the blogger and we can also gather what kind of topics their colleagues were interested in. What happened was a very large network of message threads consisted of message boards, blogs, tool websites and book/article/thesis recommendations as what this group of practitioners thought were relevant and knew about.

We pursued content analysis of the entries and categorized the contents into four main headings: existing tools, the writer's training background and jobs, principles or methodologies, and opinions. Based on the data provided, we looked at how people expressed their wishes or explained a concept in their own words. What we noticed is that there was a pattern to the types of requirements that people specified and also how they specified it.

While we are still in the data collection phase, we have browsed 11 sites with roughly 1158 pages so far. We are still around message #100 for the main discussion forum [7]. People mentioned 21 tools that they thought were important or relevant for this tool development. We discovered 12 principles or methodologies that people thought were relevant for this tool building project. We collected 18 different types of opinion categories. Most people who contributed articles to this data worked as a test automation engineer, although they had a wide variety of jobs in their career including developer, project manager, coach, process analyst, researcher and an entrepreneur. Almost everyone is capable of programming code. Most people were currently engaged in consulting roles and they worked on several development projects to draw their opinions from.

4. Requirements by Stories and Demo

The community is still in the requirements generation stage. People are still bouncing each other's ideas and figuring out what knowledge needs to be explicitly communicated. They are expressing their needs and wants based on their past experience. But the group achieved consensus on a few topics.

The general opinions so far from the community are as follows. Some projects succeeded, but not everyone saw the results as they originally envisioned using executable acceptance tests. Maintaining the requirements specification became a hassle. Customers couldn't write requirements in a testable way. Teaching customers how to write and use Fit specification was also a hassle. Developers didn't like Fit as much as xUnit tests, so executable specification didn't become popular among the developers as much as test-driven development. Requirements elicitation involved a lot of communication but the tool didn’t fully support or facilitate better communication with the customer.

The general opinion is that they saw an opportunity in this concept, but something seems to be missing when they try to practice it. The community is trying to express in words what that missing component might be. If they can figure out that missing component, they might be able to create a requirement specification and develop a tool to address the problem.

We tried to figure out how people express that mysterious missing component. The process is like blind men trying to express what an elephant looks like by touching only a part of an elephant in a dark room [8]. Nobody has the whole picture, but they are trying to express their experience in a way that can help the community reach a consensus on the requirements.

From these discussions, we noticed that there are two forms of communication for expressing their requirements wishes: story telling and demo. The stories are always told in three distinct ways. There are three types of stories: metaphor stories, war stories and editorials.

By “Metaphor stories”, we mean the contributor is specifying the requirements by alluding to similar objects or experiences. Often they convey the kinds of satisfaction that they want to experience through the new software tool instead of how it should be done. For example, one of the contributor said, “I’ve been learning piano this past year…and I’m completely in love with musical notation. It is aesthetically beautiful and very powerful. (Msg #9)”[7].

The second type of stories is war stories. The contributor tries to convey a situation where the idea being presented either worked or didn’t work. They may even describe how they implemented a tool. They add their view of the situation, but they do not get into a deep methodological debate or complex analysis behind what they saw or experienced. For example, here is an example that tells one’s experience of integrating requirements with testing. “My next exposure to agile was as the head of a larger test organization when the development team decided we needed to do a crash conversion to Scrum to meet some insane – I mean visionary – requirements and product deadlines. So in crash courses of reading and Scrum master training, the hardest part of that conversion was figuring out how to accommodate testing.(Msg #7)”[7].

The third type of stories is editorials. These entries have strong opinions and the contributor often tries all arsenals at hand to either reject or promote an idea. These articles try to emotionally appeal to others why the readers should also feel the same way. For example, here is an example contribution. “FIT’s reflectopornographic architecture introduces a great deal of overhead when writing and maintaining tests and diagnosing test failures. That overhead might be worth it if FIT helps you communicate with users and customers. However, if the FIT specifications are not read by anyone outside the development team, then the overhead is just not worth it”[9].

There is a lot of communication happening between the stakeholders and they do this through these three types of stories. Ultimately, all three forms of stories are meant to influence the requirements in some way and meant to convince others why their viewpoint is important.

Software developers are domain experts in software engineering. The uniqueness about this group of domain experts is that they can develop the tool by themselves without needing another group to make the tool for them. Therefore, if words do not express what they are thinking about, these developers would simply build the tool first, present to the community and see how people react. The tools are readily downloadable and any developers can readily try the tool. We categorized this as “requirements by prototyping”. If this was any other group of domain experts, this may involve drawing, playing or showing. The action of creating such tool results in an artifact that everyone can point at. Thus, we call this type of presentation as “requirements by demonstration” or “requirements by prototyping”.

The effects of these four types of communication method are still unknown. Do certain types of communication methods work better than others in generating requirements? For example, could metaphor stories help direct the community to simpler, more abstract solutions? Does the act of comparing and contrasting between different metaphors help create requirements better? How useful are war stories? How do others perceive one man’s experience and integrate that information into the requirements? Could editorials help shape the direction for the community? Could tools without requirements help or hinder the process? Another important aspect is the different impact factors that people have. The same statements coming from more notable and respected person in the community may have more impact than a novice person. These are questions that are still unanswered and we are still seeking more answers.

5. Interpreting the Stories

As mentioned in section 2, our analysis shows that people mentioned 21 tools, 12 principles or methodologies and 18 different types of opinions. The authors suspect that these are not an exhaustive list. The product visioning or the problem structuring process is much closer to creative brainstorming sessions than a process that simply collects software requirements features.

People come into the brainstorming session with certain opinions or beliefs. Thus, people tend to respond to topics that they can mostly agree on, rather than topics that they do not necessarily have any interest in or have any knowledge about. People do not necessarily disagree with another person's opinion. Even if they do, they will simply start another thread rather than respond directly to the existing thread. Our observation is that people try to express their opinion in a way that produces a least amount of conflict with other contributors. Therefore, the contrasting views are not easy to pick up in this online medium. The contributors need to have the necessary knowledge to actually appreciate what the other contributors are saying. There are often a lot of unspoken gaps between the readers and, thus, in what they can appreciate.

What eventually dominates the requirements discussion is the one with the most amounts of interested contributors or the topics that most number of people knows about. While that observation is obvious, it has some fundamental implications. For example, out of 21 tools mentioned, only 6 tools were discussed more than once. Out of the 12 principles, 5 of them were discussed by more than one person. Out of the 18 opinion types, 5 opinions are discussed by multiple threads. Generally, only a few opinions or tools get talked about more than once. Unfamiliarity with certain topic is what causes a communication rift between different stakeholder groups.

There are three main categories of contributors: developers, testers and business people. What becomes very obvious is the interpretation of the definitions. Some of the equivalent names for executable specification include functional tests [10], customer tests [11], specification by example [12] and scenario tests [13] among many. Although they mean more or less the same thing, their word choice shows different preferences on how they choose to view this concept.

Testers interpreted 'testing' to mean an automated testing framework much the same way other automation testing tool or testing scripts work. They cared about finding defects and performing regression tests. Developers interpreted 'testing' to mean test-driven development. They worried about the overhead of maintaining another set of tests in addition to unit tests and they cared a great deal about test refactoring. People from more business oriented view interpreted them as in knowledge transferring artifact. Therefore, this group talked about tacit knowledge, information visualization and user centered design.

The details of the social networks and how the consensus was reached will be discussed more in our future works. The summary of the actual requirements will also be discussed further in our future works.

5. Conclusion

Blogging is an interesting form of communication medium and an interesting venue for collecting and discussing empirical data for generating a product vision for a new software tool. The difficult part of gathering information through blogs is that the context information for different topics is often hidden to most contributors and conflicting information is not readily available to detect. Therefore, the contributors make decisions without fully appreciating the implication of the other sides of the issues.

10. References

[1] Hooks, I., Farry, K. Customer-Centered Products: Creating Successful Products through Smart Requirements Management. American Management Association, New York, NY, 2001

[2] Rittel, H., Webber, M. "Dilemmas in a General Theory of Planning", *Policy Sciences*, 4, 155-169, 1973

[3] Graham, D. "Requirements and Testing: Seven Missing-
Link Myths," *IEEE Software,* vol. 19, pp. 15-17, 2002

[4] Andrea, J., "Envisioning the Next Generation Testing Tools", *IEEE Software,* May 2007

[5] Fit, http://fit.c2.com

[6] Agile Alliance Functional Testing Tools Visioning Workshop, Oct 2007, http://www.agilealliance.org/show/1938

[7] Agile Alliance Functional Testing Tools Discussion Forum: http://tech.groups.yahoo.com/group/aa-ftt

[8] Saxe, J., "The Blindmen and the Elephant", http://en.wikisource.org/wiki/The_Blindmen_and_the_Elephant

[9] Pryce, N., http://nat.truemesh.com/archives/000702.html

[10] Beck, K. Extreme Programming Explained: Embrace Change, 1/e. Addison-Wesley, Boston, MA, 1999

[11] Jeffries, R. "What is XP?" Online: http://xprogramming.com/xpmag/acsFirstAcceptanceTest.htm

[12] Fowler, M. "Specification by Example". Online: http://www.martinfowler.com/bliki/SpecificationByExample.html

[13] Kaner, C. "Cem Kaner on Scenario Testing: The Power of 'What-If…' and Nine Ways to Fuel Your Imagination", *Better Software, 5(5)*:16–22, 2003