Using a Wiki to Build the Software Testing Skills of Communication and Collaboration

Marlena Compton
Atlassian Software Systems
173-185 Sussex Street
Sydney NSW 2000, Australia
61 2 9262 1443
mcompton@atlassian.com

ABSTRACT

Marlena Compton wants to share her experiences of using wikis for software testing and she also wants to hear yours. This session will focus on using wikis in widely differing software environments to bring conversation and collaboration into the software development process. Marlena began using wikis in a waterfall environment and has since moved to using and testing wikis in an agile environment. She will share:

- how wikis can help to gently introduce agile principles and values to non-agile teams
- how she used James Bach's Low-Tech testing dashboard to delight her boss and improve her exploratory testing behind the waterfall
- how she currently uses a wiki as part of an agile testing process
- what to watch out for when using wikis for testing

1. INTRODUCTION

Agile is a practice that is frequently advocated as an "all-or-nothing" approach. It is possible, in a non-agile environment, to get a gentle introduction to agile principles through the use of a wiki. Although the wiki was originally proposed with a very specific work environment in mind, they have proven flexible enough to be suitable for teams working in many different types of environments. In this paper we will examine how wikis can be useful for software testing in a waterfall environment and in an agile environment. We will begin by examining the original motivations for wikis. We will then show how wikis can be beneficial for teams entrenched with Test Case Management (TCM) software. Finally we will look at how effective teams make extremely high usage of wikis in an agile environment.

2. ENTER THE WIKI

When Ward Cunningham developed the first wiki in 1995 [1], it was initially created to draw the story of software development out of developers who were more comfortable with communicating through source code. At their most basic, wikis are sets of pages viewable and editable by anyone. Pages be edited and comments can be added about the content of the page. Perhaps the most ubiquitous example of a wiki is Wikipedia. At the time that this article was written Wikipedia contained over 15,000,000 pages. Cunningham describes the wiki as, "the simplest online database that could possibly work" [2].

Wikis were created as a way to empower teams by placing their work in a format intended to change over time as the team itself made changes. The main idea is that anyone can view or edit any page. While this requires an element of trust and respect within the team, it has the benefit of allowing team members to work together more quickly and easily. There is no need to ship documents around in an email, find them at an obscure location on a shared drive or wait for someone to make updates. Many wikis such as dokuwiki and twiki are free and trivial to set up, giving them a low cost of entry. They include the capability of linking and searching throughout the wiki.

In contrast, the design of Test Case Management (or TCM) testing tools such as quality center suggests that they are designed for a top-down approach to creating a testing process where much is assumed about how a test team will function. Systems such as Quality Center are delivered to testers with a structure already in place. If the test team is allowed enough of a budget, the TCM may have been through some customization before being delivered to the team. While this allows large test teams to start using the system quickly, it can become a dense forest of red tape and overhead requiring access to an administrator who may or may not be present. If a team requires customization that the TCM isn't set up to handle, or if the test team needs visibility with a business team unwilling to learn or even log into the TCM, wikis can provide a flexible, accessible back-channel. Instead of providing a pre-defined systems for testing, wikis are formed through a grass-roots process that is well suited to smaller teams who are tired of fighting with TCM systems for the customizations they want.

For small test teams and solo testers, wikis provide a workaround when the customization of TCM has left too little room for adjustment. Wikis can be used for the collaborative creation of test objectives, requirements traceability of those objectives and links between tests and the issues that arise during testing. Once test objectives have been set, it is simple to

create a testing dashboard visible to the entire software team.

3. COLLABORATIVELY FORMING TEST OBJECTIVES

Currently, I test software in an agile environment for Atlassian maker of the Confluence wiki. Confluence is heavily used for specifications and test objectives. In their book, *Agile Testing*[3], Lisa Crispin and Janet Gregory write about how agile tests rely on collaboration between the testers and the stakeholders making them less of a domain that is focused on solely by testers. In this case, test objectives, “flesh out the requirements.” Test objectives posted on a wiki can be directly linked to requirements for requirements traceability. By placing the test objectives on a wiki, it is also possible for stakeholders to comment on test objectives. Because wikis are versioned, the test objectives and their comments can be changed any time or reverted, if necessary. This collaboration between testers and other stakeholders on the open pages of the wiki ensure that requirements stay in plain view, and are not locked away. This allows testers to adjust their plans for testing whenever it’s needed.

4. USING WIKIS IN A NON-AGILE ENVIRONMENT

Prior to working for Atlassian, I used a wiki for at least some aspects of software development in varying phases. My experience began with my first job as a software tester for Equifax, a financial services company. I worked on a small team with 1 senior architect, 2 developers and 2 operations analysts. I tested a distributed system used for analyzing financial consumer data. All of our customers were internal. We worked as a very cohesive team built on trust and respect. Despite having this advantage, we were told by the senior leadership team that we were to use the waterfall method of software development.

We had a technical writer who installed a wiki as a way for us to manage all of the technical documentation about the new system we were building. When I started, I was not excited about putting my documents in a place where everyone could see them. I quickly observed, however, that the wiki exposed many holes in my team's software development process. As a tester, being able to see most, if not all of the documentation leading up to the requirements I was testing explained why some requirements were more "fleshed out" than others. Using the versioning capability of the wiki allowed me to see how changes had been applied in the documentation giving me more insight into the design process. This helped me to ask better questions of the requirements I was testing.

5. Obstacles of Test Case Management Systems

Looking over the requirements on our group wiki was in stark contrast to what I was experiencing with our test case management system (TCM). I initially requested the TCM because, as an inexperienced tester, I was unsure of the structure I needed to user for managing my tests. After all, this is not something taught in any computer science class. Things fell down, however, when I had to communicate with others about what I was testing and about the results of my tests. I found that it was extremely difficult to get any meaningful information out of the TCM. I could show a list of "tests" in a report, but this list gave no indication of the complexity of my testing nor was it possible to accurately show cases where the software had passed with a minor defect or two. I could not understand why the TCM insisted that these features had “failed.”

My experience is an illustration of how software testing has, in its brief history, been mired in a reliance on expensive, heavyweight TCM tools. These tools are specialized versions of knowledge management (KM) systems created for software testing teams. In his book, *Wikipatterns*, Stewart Mader[4] writes that KM systems, “were designed to structure the knowledge process and allow people to maintain the hierarchical control they were accustomed to. Furthermore, they were also built to meet the needs of one type of user." Because the structure of testing is pre-defined by these tools, the TCM, to some extent pre-defines the process that testing will follow. In these tools the design of tests is separated from the actual testing making them more advantageous for scripted testing.

The problems with using the pre-defined structure manifested itself to me when I began to also notice how the system's structure, upon which I had originally relied, had turned into an obstacle for exploratory testing. In exploratory testing, one test frequently leads to related tests that may have received a minimal amount of planning before the testing session. The TCM I was using had been structured for tests to be designed before a test session commenced. There was also no way to link families of tests together aside from having these tests in the same directory or to get external links for the tests. What the directory structure did not account for is that tests can be related in multiple ways and that these relationships are also important test data.

In the world of the TCM, tests are broken out into steps. Testers perform test steps with the rhythm and pace of a machine. Each area of the software has its own set of tests with no opportunity for collaboration from testers through the TCM. The ultimate problem created by TCM tools is that there is no deference or recognition of the role played by the creative human thought process which is required for good software testing. These creative thought processes require ways of linking ideas together and collaborating with others. Because they support a pre-defined structure for

testing, TCM tools are also unable to change and modify the testing process as software testing teams change and modify their processes over time

When I tried an exploratory testing approach while I was using my TCM system, I began to feel that there was too much on the screen telling me what to do. In his book, Getting Things Done, David Allen[5] writes about the importance of white space for creative thinking. I have found that wikis are capable of providing more of a "blank page" feel which can later be integrated with templates to hint at structure in a less intrusive way

6. TESTING IN THE WIKI

Upon realizing that my TCM system was holding me back and seeing how beloved our wiki had become by the team, I made the decision to try testing a smaller release through our wiki instead of the TCM. I would later update the TCM with my test results. This could have resulted in a disorganized mess, but I found my way to James Bach's low-tech testing dashboard[6] through Eric Jacobson's blog post, "Build Day Low Tech Dashboard" [7]. Eric employed the dashboard as a way to stay informed of builds at his company. Although Eric used it on a physical whiteboard, I thought it might translate well to a wiki, especially considering the employment of links for improving traceability.

In his book, The Art of Agile Software Development, James Shore writes about the importance of using big visible charts. The goal of these charts, Shore writes, " is to display information so simply and unambiguously that it communicates even from across the room" [8]. The purpose of being able to communicate from across the room carries the assumption that all team members are in the same room. The reality of software is that teams are increasingly more distributed. In my case, we were distributed throughout cubicles. This wasn't a factor over which we had any control. In order to make the best of it, I used a wiki as a way to take the walls down. In these cases where the team cannot sit in one room, wikis provide an "across the room" visibility that can otherwise be lacking.

For software testing, James Bach's low-tech testing dashboard provides a starting point for testers to create a wiki-fied version of the big visible chart. This dashboard is a simple table used to show progress and blockages in the testing efforts of up to 20 or 30 testing components. While Bach has set up his own attributes of importance for measuring progress with testing, it can be valuable to assess the worthiness of each of these for a individual test teams. Using a wiki for this board has the advantage of allowing for links to test objectives in each of the test components listed as well as allowing for links to issues and other important documentation. These links form a path of traceability from test objectives to defects.

With approval from my boss, I gave wiki-fied testing a try over a 3 week test cycle. I set up the dashboard in a space on the wiki with areas assigned for each feature I was meant to test. As I tested, I added links on the dashboard to pages I created for my tests. I updated the dashboard as I went along.

My conclusion about testing this way was that it was definitely helpful for "getting [the software] out the door" [9]. Creating tests in a format that allowed my mind to wander as I tested helped me think more creatively as I tested. The lightweight dashboard required minimal updating even though it made the progress of testing obvious. I was no longer hassled about the progress of testing. No one on the business team had every logged into my TCM, but they looked at my testing dashboard frequently during my test cycle. If there were questions beyond what was provided by the dashboard, the links were there for people to follow.

Although my then-manager told me he liked the way that I used the wiki, his approval was also apparent from what I was *not* asked during the test cycle. I was not asked: how the testing was going, was I going to be finished on time or how were the new features looking. Instead of having to prepare and explain a report to our business side at their convenience, I was able to update the dashboard during times when I was not focused on testing.

7. Wikis in the near-future

Aside from fostering collaboration, wikis form a back channel that will assist in the software testing of the near-future. Wikis and TCMs both have access to their information through function calls. Where they will diverge is that testers gathering data from a TCM will only have access to test suitable for metrics. Testers gathering data from the wiki will also have the ability to gather information about how testers have interacted through the back channel. They will be able to answer questions such as how many revisions has a test been through or which testers colloborate together more successfully.

8. Conclusion

Wikis can provide a flexible and easily customizable environment for software testing. They are a collaborative means of forming test objectives, and an easy way for distributed teams to manage through a big visible chart. For smaller software testing teams, wikis can provide a way to form their own testing structure for exploratory testing.

9. References

[1] Cunningham, Ward. 1995. *History of the Wiki.* http://c2.com/cgi/wiki?WikiHistory

[2]Cunningham, Ward. 2002. *What is Wiki*. http://www.wiki.org/wiki.cgi?WhatIsWiki

[3] Crispin, Lisa and Janet Gregory. 2009. *Agile Testing: A Practical Guide for Testers and Agile Teams.* Addison-Wesley, New York.

[4] Mader, Stewart. 2007. *Wikipatterns: A Practical Guide to Improving Productivity and Collaboration in Your Organization*. Wiley, New York.

[5] Allen, David. 2002. *Getting Things Done*. Penguin, New York.

[6] Bach, James. 1999. *Low-Tech Testing Dashboard*. www.satisfice.com/presentations/dashboard.pdf

[7] Jacobson, Eric. 2007. *Build Day Low-Tech Dashboard.* http://www.testthisblog.com/2007/12/build-day-low-tech-dashboard.html

[8] Shore, James. 2007. *The Art of Agile Development*. O'Reilly Media, Sebastopol, California.

[9] Venners, Bill. 2004. *Exploring with Wiki: A Conversation with Ward Cunningham*. http://www.artima.com/intv/wiki.html