An empirical investigation on the use of customer requirements and acceptance testing in agile software development

- XP 2009 Doctoral Consortium submission

Author: Børge Haugset, The Norwegian University of Science and Technology (NTNU), Norway / SINTEF ICT, Norway
Address: (SINTEF) S.P. Andersens vei 15, NO-7465 Trondheim, Norway
Contact: email: borge.haugset@sintef.no, mobile: +47 92492454, fax: +47 73 59 29
Supervisors: Tor Stålhane, tor.stalhane@idi.ntnu.no (main)
Guttorm Sindre, guttorm.sindre@idi.ntnu.no (co)
Torbjørn Skramstad, torbjorn.skramstad@idi.ntnu.no (co)
Start of PhD: August 2008
PhD length: 4 years
Updated: 2009-05-15
Abstract—A natural first step in developing software is to know exactly what the customer wants. If this process does not work properly, the result could be an unhappy customer and an over-worked developer. By using acceptance tests the customer can assess the produced code, and hence it's value; does it perform according to the requirements or not. The problem is, customers do not necessarily know what kind of software they want. How can developers discover these true needs of the customer? This thesis looks at what happens when you approach this communication and its hindrances in various settings. Agile development focuses on developing small pieces of the software in iterations, and have customers evaluating and directing the process. In behavior-driven development (BDD) and acceptance-test-driven development (ATDD), customer requirements are described in a format that can be used by developers as rules for implementing automated acceptance tests. This relatively new approach has gained a lot of attention, yet little is known empirically about its validity. The candidate will, by conducting case studies, look at how different methods for requirements gathering and software development impacts the understanding shared between customers and developers.

Index Terms—customer requirements, agile development, acceptance testing

I. INTRODUCTION AND MOTIVATION

A software system is inevitably best created if, in the end, the resulting system fulfills the customers’ needs. These needs can be explicitly described in several ways, such as use case diagrams, user stories, automated acceptance tests, or as plain text. Different descriptions are not necessarily equally suited for expressing all kinds of user requirements, nor are they equally easy for customers to help describe and develop. One thing is certain, the more often these requirements are to be documented and changed, the easier this task has to be.

Once the requirements (or first parts of them, if you are using an agile method) are established, developers start coding with the (in some way broken down) requirements in mind. The requirements are also delivered to the test personnel, if that doesn’t happen to be the developers themselves. Tests are written, run, and errors are fixed. If you use test-driven development, it works the other way around: You write the tests according to system requirements and then code to make these tests pass. As tests like these grow into a suite covering the code, it works as a safety net for the developers; it can serve as an assurance that later changes in the code does not mess up what is already written. It can also serve as documentation of intended behaviour, in themselves describing the customer requirements. There are tools to help automatically run sets of tests while you code (or after). This can be done every time you have written new code, or every night if you wish to test the whole system. The focus has traditionally been on automating tests on what is called the ‘unit level’. This means testing the internal integrity of code units such as functions and classes. This will function as a safety harness for developers, as verification that changes or extensions to the system satisfy the unit requirements. All of this resides in the lower section of what is called the V-model, see Figure 1. Higher level, tests such as acceptance tests shown in the top right corner of the V, have usually been run manually. These tests enable the project members and customers to see if the system performs as expected, or even if the system described earlier really serves its purpose [1]. High-level tests performed by the customer or representative ensure that their current mental model of the real problem is compared to the actual system under development. A manual task such as acceptance testing is expensive and requires customer involvement. This kind of testing should verify the software, in making sure it adheres to the stated requirements. It should also measure the validity of the system, describing its actual use and how it helps the users perform their tasks.

In an ‘agile setting’, the customer is a part of the continuous development loop. Their requirements are hence refined, altered or confirmed each iteration. While testing is an integral part of any software development method, using an agile

Figure 1 - the V-model in software development

method means having to run the tests on a daily basis or even by the hour or minute. Testing is expensive and time consuming. Being able to automate this, partly or fully, thus reducing the need of a customer at hand at all times, could impact the design of software in a positive manner. One promising candidate for doing this has been developed over the last few years. One promise of Automated Acceptance Testing (AAT) is to make it easy for customers and developers to jointly express customer requirements directly as acceptance tests. These executable acceptance tests, in turn, can be used in test-driven development. This is called

---

1 …using tools such as jUnit for java and nUnit for .NET languages

2 Names on objects and complexity of the V-model vary greatly depending on who created the figures; this basic one is borrowed from Wikipedia in February 2009. A more complicated and exhaustive figure can be found at http://www.delta axiom.com/CJ256ED60045E95F/sysOakFil/plakat-konfigurationsstyring_T_amy$File/P-T-axiom-0707.pdf

3 The discussion if there’s any such notion as an ‘agile setting’ and what this would be is left out here
Acceptance-Test-Driven Development (ATDD). Two of the most used frameworks for such testing are Fit\(^4\) (Framework for Integrated Test) and Selenium. These tools are open-source, and in constant evolution. In Fit, business rules are expressed as simple table statements of input and expected output and can be executed through an underlying framework. The customer reads the tables and acknowledges that these are indeed the proper requirements; the developers start coding to make them run. If the developers are not using a test-first approach, the requirements can be described in some other way, and then later translated into acceptance tests as the need arises. Doing so, however, could in Lean terms be described as introducing waste. It is also a new translation of requirements, thereby introducing a possible source of misinterpretation.

While automated testing seems easier to use as tools for verification of stated requirements, I would like to look at how such tools and the process surrounding it affect the outcome. Does it mean developers produce the “right” system the first time?

There is a lot of activity in the developer community about automation of acceptance tests in an agile environment. Many gurus try to have a say in what this works for, and developers seem to discuss this lot in blogs, forums and agile community meetings. How does the introduction of automated acceptance testing impact the agile relations between customers and developers? Has it changed the way customers and developers communicate? There’s not much to be shown on research on this. I’d like to add to that.

II. RELEVANT PRIOR WORK

This work will focus on the use of acceptance testing in agile software development, especially on the use of tools for automated acceptance testing. This focus is chosen because acceptance tests are, or should be, strongly related to the customer requirements. Research published so far, as reviewed in a literature review presented at Agile 2008 [2], shows that there have not been a lot of studies on this topic. The literature review had a focus on the tool Fit, but we included search terms like ‘automated acceptance test’ and ‘executable acceptance test’ to cover also articles of interest beyond this tool. In the mentioned work, we identified 14 articles covering the field, of which only eight had some sense of empirical soundness. Inspired by constant comparison [3], we identified a set of topics that these articles covered. The most important findings follow.

Student experiments show that automated acceptance testing tools seem easy to learn and use [4], and good specifications can be produced [5]. Over half of students asked would rather use Fit than prose to describe system requirements; more than two thirds would consider it. They also find specifying Fit tests as easy as writing prose or unit tests [6]. These studies have been performed on students, and little is known about business cases. Melnik et al. studied one project where the customer successfully adopted and used Fit tests [7]. This customer, however, also had an information systems background. Studies do not show that it is difficult for non-technical persons to master the art of describing acceptance tests for automation, there’s just no proof for it being so.

Fit is intended to make the customer help write the tests. FitNesse\(^5\) is even more so, by allowing customers to access the test descriptions via a wiki server. Customers don’t always feel this case. In one project they were not interested in writing the specifications this way, and left this test coding to the quality analysts [8]. In hindsight, the authors suggest that the Fit documents should rather have been described as ‘specifications’. In the same manner, Melnik and Maurer describe a case where the customer brought, to each iteration meeting, previously made info-sheets describing diagrams, callouts and mock-ups [7]. Fit tests in another project were also accompanied by embedded commentaries and occasional diagrams [9]. Further, writing Fit tests seems to be less prioritized than other work, and one customer described how it wasn’t the difficulty of writing them that kept him from doing it, but the discipline [7]. Still, the same project members described how using Fit helped to discover inconsistencies and missing pieces. Having a set of Fit tests that cover the code and runs green (i.e. produces no errors) can lull developers into a “deceptive sense of security”, because no failing tests do not mean the needs of the customer is taken care of, or that the code is good [10].

Melnik et al. studied the quality of acceptance tests and resulting implementation. They found no evidence to support that good quality acceptance test specifications made by a customer team resulted in better quality implementation by a developer team [5]. They think this issue requires further study. Still, projects have been described as successful due to the use of Fit [8]. This was stated even if they claimed using Fit first hindered communication. The authors claim the project members let the tool drive the communication, making them focus too much on preparing ‘syntactically correct Fit documents’ instead of developing the most appropriate specification. Using Fit has been reported to help project members with conversation “‘...needed to have the features implemented in a way that is economically feasible, but also in a way that’s right.’” [9].

The review showed that only a few articles described automated acceptance testing. The topic was quite new, and conferences like Agile 2008 had several events on this. Further, there are a lot of blogs where developers describe their use of tools such as Fit. This indicates an interest within both the research and industry communities to look closer at AAT, and hopefully that a lot more has been written about this during the time since the review was written.

The literature summed up above has mainly had a focus on an agile development setting, and on automated acceptance

\(^4\) http://fit.c2.com

\(^5\) FitNesse (http://fitnesse.org) is an extension of the Fit framework, including a web server with a wiki where tests are created, maintained and can be executed
testing. This does not imply that there are no other important findings in literature concerning the use of automation of tests and acceptance testing outside the agile box. As a hint, the ACM Portal returns well above 13,000 hits on the search term ‘software test automation’ (the three words, not the three word string) A follow-up on this literature review will have to include also non-agile material.

III. RESEARCH OBJECTIVES AND QUESTIONS
Given the problem description the main goal of this PhD study will be to understand how communication between customers and developers are impacted by the development methods and the level of acceptance test automation:

- **What are the differences in use of acceptance tests for communicating customer requirements in various software development settings?**

I have intentionally decided to start out with a broad main goal, and rather focus on the agile setting in the research questions. One of the main differences between agile and ‘traditional’ software development is how developers treat the acceptance tests. As Scott Ambler puts it in Dr. Dobb’s; “[Agile software developers] recognize that acceptance tests are first-class requirements artifacts and not just test artifacts.”

The main goal will be studied by looking at some underlying factors:

**Research question i)** Why do developer organizations want to use automated acceptance testing?

Examples are:
- Increase collaboration between customers and developers
- Serve as continuous verification of the adherence to the stated customer requirements also at a high level
- Function as validation of the systems adherence to the customers real needs
- Increased security in code refactoring/development by catching errors early and making sure they don’t return
- Be more agile – quicker being able to see if the changes to the code are indeed supporting the customers needs or not
- A desire to try out new methods
- Other perceived benefits – for customers or developers

There are many reasons why organizations choose to use agile development strategies, and this also goes for automating acceptance tests. By discovering the intentions for using this kind of technique one can uncover the lacks of other techniques, thereby discovering the potential true benefits of AAT. To understand the use of automated acceptance tests one also has to peek at the manual acceptance testing.

**Research question ii)** How well do automated acceptance tests describe functional and non-functional requirements?

A system under development may consist of a set of requirements in various forms. It is interesting to see under which conditions AAT tools are, if at all, most valuable. Functional requirements are intuitively easy to express as tests; this is often not the case for non-functional requirements. The claimed abilities to describe non-functional requirements are not backed by empirical data. Being able to add empirical knowledge on these claims is needed. Requirements gathering and testing has traditionally been two separate activities, and quite often performed by different groups. Missing connections between these two processes can lead to tests that don’t test all requirements properly, or vice-versa; describing requirements that are difficult or impossible to test.

**Requirements question iii)** How do developers and customers cooperate on developing and maintaining customer requirements?

In a traditional ‘waterfall process’ system development process, customer (and thus system) requirements are generally described in the beginning of the project. They are then used by developers (including testers) to create a system according to these specifications. In an agile setting, customers are involved in a development method with very short iterations between deliveries. This makes the customer able to see the (partial) results, inspiring changes and new requirements. There are few empirical studies on how these short iterations affect cooperation in various settings, for instance when using a test-first development method or not. Do the customers and developers sit down together and produce the actual tests, or do they cooperate/agree on a description that ‘someone’ (most likely a developer/tester) later translates into tests? My impression from interviews and literature review described in [2] is that customers rarely sit down and help to describe these tests. That does not necessarily mean doing so would be a waste of time. I would like to know if this lack takes place because it isn’t fruitful, or because the possible benefits are not known or spread. The customer role in an iterative setting is quite different from the waterfall setting, and having to sit down and write tests is perhaps even more difficult to find time for.

I will not look into automation of acceptance tests being used instead of manual acceptance tests. I have not yet been able to find any evidence for this being possible, nor am I sure it would be a positive thing. The important question is, I think, how these automated tests are used both between and within iterations to force the developer to focus on customer requirements as opposed to code syntax and unit test definitions.

**A. Narrowing the scope**

Testing is in itself a very extensive area, with much published literature. In order to narrow down the scope I will focus on the customer aspects. This does not mean that developer aspects will be left out entirely. It rather means that the study...
will look at how developers use customer requirements to develop the acceptance tests (or how this is done in cooperation), and how these in turn are run in order to ensure adherence to the customer requirements. Topics such as code quality will not be looked at, nor will I try and find out if developers should use both automated acceptance tests and unit tests at the same time. To what extent AATs can serve as an extra internal safety net is also beyond the scope of this thesis. Due to time constraints and wish to focus, the candidate has decided to not perform an experiment or create a tool as part of this work. While it could prove beneficial to perform such a study, doing both action research/case studies and experiments would be too much work, and focus would be lost. The candidate finds it more important to work closely with the companies in one way; this does not mean that other researchers should feel discouraged from doing so.

B. Hypotheses
I have not yet gathered enough data to be able to define a set of hypotheses for the study. My initial feeling is that that using automated acceptance testing tools such as Fit/FitNesse is of most value when you do so in a test-driven fashion. This is when results from every meeting with the customer are fed directly into the testing tool. The tests in turn are supposed to fail; it’s the customer requirements that changed. This means the developers are constantly working to comply to the most up-to-date requirements. In the case study I have looked at so far, they did not use Fit for communication with the customer. Neither did they take on a test-first approach. This means that changing requirements led to rewriting the business code first. When in a hurry, they skipped updating the tests. If you don’t do test-first, broken tests means more boring work. This, I believe, in turn resulted in up in a messy situation no one wanted to take care of. Hopefully my next interviews with these groups can shed some light on this.

IV. Empirical study design
This PhD will be based on combining literature review with case studies:

Study A) Literature study
- **Goal:** To describe the current empirical knowledge on using automated acceptance testing in software development as explained in literature
- **Result:** One literature review has already been published [2]. This study could be classified as an integrative literature review [11]. We identified relevant publications by searching ACM Digital library, IEEE Explore and Web of Science for a set of strings we considered to ensure the relevant hits. The resulting set of articles was filtered based on title and source to weed out non-relevant material. The next review step was to evaluate each of the fifty remaining papers’ abstracts according to topic and empirical value. After looking at the remaining 19 articles in more detail, 14 were considered to be of value to the review. The next step involved was reading these articles and, inspired by constant comparison [3], identifying central concepts and topics through repeated analysis of the literature. We chose to include a set of important but non-empirical articles in the review, as this is a new field and we felt it vital not to let potentially valuable findings out. As a result of this, the study can not be considered a full systematic review [12]. This work has been used in the section covering relevant prior work. I am considering expanding this article into a journal article. This work will, if so, include updating the review (searches were made in 2007), setting automated acceptance testing into a context of other testing practices (test automation has been an issue for decades). It will also be decided if a full systematic review is of value in this area. During our initial review we found that the topic has not produced many articles with empirically valuable knowledge, and I so far believe that performing a full systematic review according to Kitchenham would lead to a lot of work for potentially very little results.

- **Status:** Work on how to expand the literature in the best way has started. Further, I would like to look more closely at literature describing theories of customer and developer communication.

Study B) Case studies / Action Research study
- **Goal:** Collect data from industrial settings in order to analyse and understand the use of automated acceptance testing
- **Result:** One to two conference articles per case study. Merging smaller case studies into one journal article is a probable way forward, depending on findings.

The project funding my PhD includes a set of SME’s that are involved in acceptance testing in various ways. While many of them have had a lot of focus on testing before, using automated acceptance testing has only recently become an issue for most. This means I, as a researcher, can perform case studies in companies that adopt (or tries to adopt) this development method.

Case 1: The software division in a company, with about 50 employees that builds receiving stations for data from meteorological and Earth observation satellites. Traditionally following a plan-based development approach, they a couple of years ago started using Scrum in some projects. In the spring of 2009 they started their first project using automated acceptance-test-driven development. It used FitNesse, and while this is a small project (two developers, and 3 months development time) it will have given them some impressions of the tools and method. They also plan to use the same development method in follow-up projects to the current one (of course provided their evaluation is positive)
- Somewhat experienced in agile methods
- Beginning to use automated acceptance-test-driven methodology
- First reactions seem positive
- The group started using tools that I brought up, perhaps this case is the most promising candidate for action research
- Being able to follow them in later projects will of course increase the knowledge from this case study

**Case 2:** This consultancy company offers services in the area of object oriented systems development, and has been using a variety of agile methods for quite some time. They have already been studied using semi-structured interviews, leading to articles presented at Agile 2008 [2] and HICSS 2009 [13]. We interviewed four project members each from two teams. The teams were at the time using Selenium and Fit. Neither of the projects used a test-driven approach. The studies showed that while they had some problems, they in general felt that using such tools gave advantages. Recent talks with the company that they have stopped using these tools. I will follow up on this, and hopefully be able to gain insights into why this is the case.

- Experienced in agile methods
- Used an automated acceptance-test framework for some time, albeit not with a test-driven approach
- Interesting contrast case as they found using such tools not worth the hassle – trying to uncover why this was so might lead to insights
- At first they thought to be worth-while as internal documentation and as a high-level safety-net, but for some reason this was not enough for them. It would be interesting to find out why

**Case 3:** Company 3 delivers services in the field of electronic document access and geographical IT. They have been implementing agile methods (Scrum) for a few years, but are now looking at using an acceptance-test driven method.

- Somewhat experienced in agile methods
- Beginning to use acceptance-test-driven development (ATDD) methodology

In addition to the companies taking part in the project funding this thesis work, contact has been established with yet another company.

**Case 4:** A software consulting company with approximately 200 employees, supporting large corporations and the public sector agencies. They operate mainly with advisory services, technology services and application management. According to my contact they use test-driven development methods in approximately fifty percent of their projects, and a small part of these use behaviour-driven design⁷. This is in other words a company with long traditions using agile methods.

- Experienced in agile methods
- Uses a test-driven approach in 50\% of their projects, some of these with behaviour-driven design (BDD)
- Being the mature case, this company will hopefully be interested in letting me study them deeply

Initially the cases will be studied using semi-structured interviews, hopefully eventually also with customers. As expected with scientific methods, there are critiques to this approach (see [14] for a summary of twelve methodological problems). I will in my work try to answer this critique for all cases. One important factor is triangulation. I will try to join project group meetings, attend retrospectives, and possibly study data on test usage to study data from different angles. The companies 1, 2 and 3 have been part of research projects for quite some years. Depending on the results from early studies, it might be possible to turn the case studies into action research studies. This will be done in accordance to the five principles of canonical action research [15]. The study will follow a cyclical process model proposed by Susman and Evered [16], which contains discussion with the companies, planning action, taking action, evaluating the outcome of the action, and finally specifying the learning others can have from this.

**Studies contributing to answering research topics**

It is too early to clearly define which of these possible case studies that will contribute to answer each research question. The study consists of both mature and immature teams; they differ in terms of using agile methods, using test-driven design as well as using automated acceptance tests. They are also consultants, in-house and mass-market³ developers. This should hopefully be a set of companies that will enable a fruitful study. When it comes to establishing valuable knowledge from a small set of case studies I lean on Flyvbjerg, stating that also case studies can contribute to scientific development [17]. Flyvbjerg does not, on the other hand, claim that this is easy.

**Status for case studies**

Case 1 has finished their first project using FitNesse. Meetings to debrief on their use, and also to gain insights on their general approach on aligning requirements and tests are scheduled.

Case 2 was interviewed once already, two years ago. In talks with scrum master he described how they have stopped using the tools. More information on this is needed, and interviews are scheduled.

Case 3 has used FitNesse in one small project. The first action in this company is performing interviews looking at aligning requirements and tests.

Case 4 has been initially positive to cooperation, but getting from there to specifying the actual dates for interviews has proven difficult. Work is still ongoing...

V. **Literature**


---

⁷ http://dannorth.net/introducing-bdd. A design method driven by business values not unlike AAT-driven design such as with Fit

³ Albeit for a specialized and small mass-market


