International Journal of Control and Automation
Vol.9, No.2 (2016), pp.343-352
http://dx.doi.org/10.14257/ijca.2016.9.2.32

Design of Acceptance Test Process with the Application of Agile Development Methodology

Jung-Ah Shim[1], Hyun-Jung Kwon[2], Hyun-ju Jung[3] and Moon-Sung Hwang[4*]

[1,2,3,4*] *Department of IT Policy and Management, Soongsil University, Seoul 156-743, Republic of Korea*

[1] *rosapoodle@naver.com*, [2] *musehjkwon@gmail.com*,
[3] *hyunju104@gmail.com@gamil.com*, [4*] *ms.hwang@sabic-ip.com*

Abstract

An acceptance test refers to a test that confirms whether the specified requirements are met. Recent studies on acceptance test automation appear to focus only on easy and simple test automation, while the importance of living documentation for the general application life cycle, which is the ultimate goal, is overlooked. Compared to traditional development methodologies, it points to marked absence of systematic planning and prediction, overhead due to the application of new process and tools, and in turn, a decline in development productivity. This study designed an architecture that can be used repeatedly through the common application of the agile software. Concrete action plans for automating the acceptance test are presented.

Keywords*: Acceptance Test, Agile testing, Test Automation, FitNesse, QA, ATDD*

1. Introduction

Anyone who has a programming language and a development environment can develop software. However, development of the software is not the end; the operation of functions, stability, and scalability must be verified.

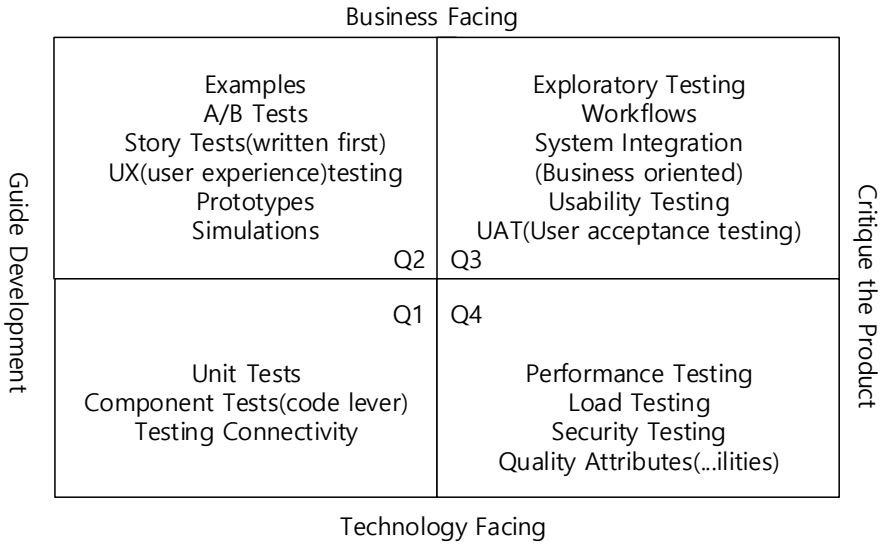

Figure 1. Agile Testing Quadrants

In particular, the agile development methodology which is under the spotlight recently emphasizes the importance of test in software development, which is referred to as test-driven development. Accordingly, the importance of test automation tools for automating

ISSN: 2005-4297 IJCA

the test process is also drawing attention. Test automation tools have a very broad range because software development tests are divided into unit test, integration test, system test, and acceptance test. Figure 1 [1] shows the test purposes of the agile methodology in four quadrants, and Q3 shows that when business-oriented test is performed to assess the product, customers or users perform tests in such a way that actual users are using the application. Acceptance test is one example. Unlike unit tests, acceptance test is targeted at verifying business functions rather than unit functions. As business functions are often implemented independently from different environments for different teams, tests on the interaction of these business functions are also included in acceptance test. For example, when business functions are implemented in different servers, acceptance test verifies whether or not these functions can properly interact with one another.

An acceptance test is the final test conducted at a point between the completion of development and the system release. The customer takes the lead in the test. The BC service includes alpha and beta services, while the SI test conducts tests led by the customer who orders SI. Unlike other tests at different levels, the purpose of an acceptance test is not finding defects but deciding whether to release the product. At the level of an acceptance test, complying with requirements and design specifications is certainly included in the scope, and legal matters and user experiences are verified, as well.

In other words, acceptance test-oriented development method is a representative agile development method and a communication method for producing software that correctly satisfies customer's requirements within a short development cycle by specifying and continuously using tests based on examples through collaboration.[2] In order to carry out this development method, the requirements must be automated as they are with no change, and repeated tests must be performed to ensure living documentation that always stays current. However, when we find examples related to such automation, they focus only on convenient and simple test automation and the importance of living documentation which is the ultimate goal is overlooked. Therefore, in this paper, acceptance test automation examples will be compared to find problems, and concrete and reusable specifications of conceptual agile development methodology and a basic architecture for tests will be presented.

2. Related Research

2.1. Classification of Test Automation Tools

Traditionally, acceptance test was performed directly by the QA (Quality Assurance) team, but the cost increased because too much manpower was inputted to the passive tests. To address this problem, the need for the introduction of test automation tools was raised. Test automation tools are classified by phase in Table 2 below. [3]

Table 1. Test Automation Tools are classified by Phase

SDLC	Automation Tools	Description
Design Phase	Specification-based test design tool	Test processes, data, drivers, etc. are generated from software specifications.
	Code-based test design tool	Test processes, data, drivers, etc. are generated from source codes.
	Test-based management tool	Supports planning, requirements, and bug tracking management.
Implementation	Static analysis tool	A tool for analyzing without running

Phase		programs. Analysis of complexity
	Dynamic analysis tool	System status is examined while the program is running
	Review and inspection tool	Guidelines and rules are inspected by analyzing source codes and design documents.
	Coverage measurement tool	Degree of completion of test case performance
	Performance and load simulation tool	Number of transactions per second is calculated after the occurrence of a system performance load.
Post-Implementation Unit Test Phase	Test performance tool	Performs unit and total system tests before acceptance test

The followings are the types and process of the acceptance test. [3]

Table 2. Test Type

Type	Main Activities	Verification Criteria
User Acceptance Testing	Confirms the appropriacy of using the system.	Usability and functionality
Operational Acceptance Testing	Inspects backup/restore, user management, maintenance, and security vulnerability.	Nonfunctional quality elements
Contract Acceptance Testing	Confirms if the terms to pass the acceptance of the contract are met.	Contract Terms
Regulation Acceptance Testing	Confirms if it is developed according to the regulation.	IT Compliance
Alpha Testing	Conducted by the potential customer in the developing organization.	User adaptability
Beta Testing	Conducted by the user and the potential customer on the site.	Environment adaptability

2.2. Comparison of Acceptance Test Automation Systems based on FitNesse

Fitnesse is a tool for enhancing collaboration in software development. Fitnesse enables customers, testers and programmers to learn what their software should do and to automatically compare that to what it actually does to. It compares customer's expectations to actual results [5].

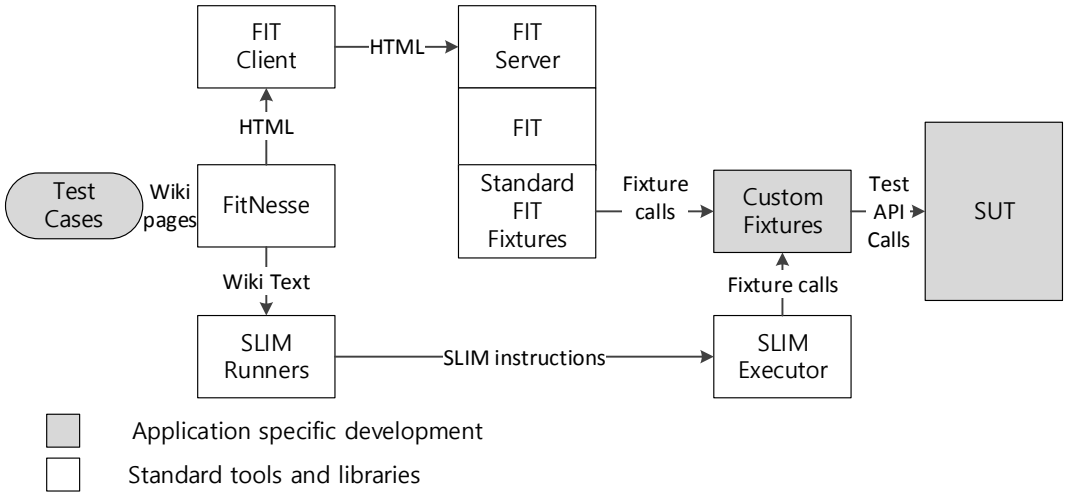

Figure 2. FitNesse Architecture

To apply test automation, a proven test methodology and an automation framework can be used. However, using a framework requires us to learn the techniques of the test tool and much time and effort is required for customizing. The wiki-based FitNesse [6][7] compared here is based on the concept of Fit (Framework for Integrated Test) which allows us to perform testes in a Web browser by configuring test tables on the Wiki page. Therefore, an automation framework that combines STAF with FitNesse and the combination of FiNesse [5] and Selenium are compared before an acceptance test automation architecture is presented.

Table 3. Comparison of Conventional Automation Systems Based on Fitnesse

Comparison	Combination with Selenium [8]	Combination with NTAF [5]
Common points	. Easy to manage because it is based on FitNesse . Short-term tools are used . Focused on accommodating various test environments after development	
Advantages	.Fast test preparation is possible through recording method	.Flow control for wiki test case tables is possible. .Various keywords are provided
Disadvantages	.Tests are easily broken even by slight UI changes.	.When complexity increases, it is difficult to achieve the standard quality of specifications which is the goal of development.

3. Designing Acceptance Test Automation Architecture

3.1. Concluding Automation Architecture Driver

Non-functional requirements are reviewed through the main process in the concept of "specifics using examples," which is the ideal model of acceptance test-driven development

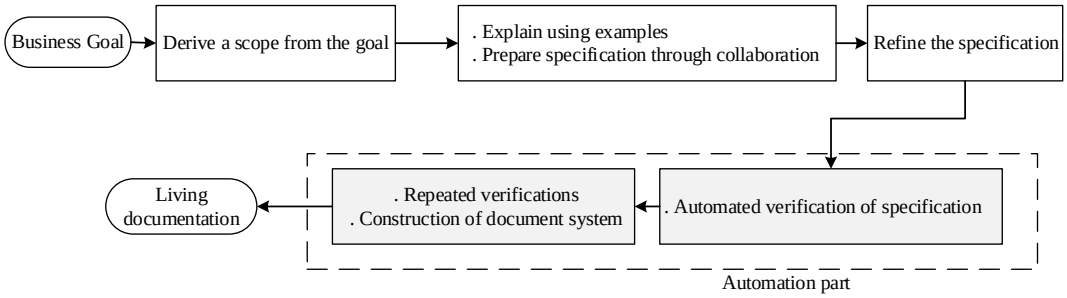

Figure 3. Main Process of Specifications using Example

In Figure 1, the processes related to automation are "Automated verification of specifications" and "Repeated verifications", which lead to "Construction of document system". These are core tasks in order to implement the value of specification using examples beyond simple automation for simple repetition. Applying the conventional test automation often generates omissions and disagreements of information in the technical automation code reconstructed based on business specifications. However, specifications using example must be automated while maintaining the requirements specification including examples that can be used by all team members As quantitative evaluation indicators are required to evaluate objective changes, main quality properties of the agile test automation related to the architecture drive and the mutual relationship were studied. Twelve quality properties — mentioned in "The Test Automation Manifesto" [6] were used to check the quality properties of general agile test automation.

Table 4. Check the Quality Properties [6]

Test Automation Manifesto (Specification Layer)	Description	Architecture Driver
Concise	Test should be as simple as possible and no simpler.	Readability
Self Checking	Test should report its results such that no human interpretation is necessary.	
Repeatable	Test can be run repeatedly without human intervention.	
Robust	Test produces same result now and forever. Tests are not affected by changes in the external environment.	Maintainability
Sufficient	Tests verify all the requirements of the software being tested.	
Necessary	Everything in each test contributes to the specification of desired behavior.	
Clear	Every statement is easy to understand.	Readability
Efficient	Tests run in a reasonable amount of time.	Maintainability
specific	Each test failure points to a specific piece of broken functionality	
Independent	Each test can be run by itself or in a suite with an arbitrary set of other tests in any order.	Maintainability

Maintainable	Tests should be easy to modify and extend.	Maintainability
Traceable	Tests should be traceable to the requirements; requirements should be traceable to the tests.	Traceability

Thus, key quality attributes that need to be specially emphasized in acceptance test automation along with the principles of the agile test automation are identified in Table 3.

Table 5. Key Quality Attributes of Acceptance Test Automation

Functions	Targets
Readability	Automation specification for business staff and developers
Maintainability	Automation specification according to the changes of requirements and SUT
Traceability	Requirements and tests
Accessibility	Automation specification of stakeholders for close collaboration

3.2. Automated Design Process

Two outputs of test automation tools for acceptance test-oriented development are specifications that can be read by people and automation code in a programming language.

Even in the traditional test method, the legibility of test case gives additional advantages to the test purpose and the maintenance. In case of a development led by an acceptance test such as "Specifications Using the Exercises," it is an essential component as a foundation of the communication for basic collaboration and mutual understanding. Apply the SoC (Separation of Concerns)[9], one of the programming principles, to the automation layers to separate them into two clearly separated layers based on the viewpoints of each stakeholder. Then, reduce basic complexity of each part to primarily secure the legibility and maintainability. This is the most basic property to be read as specifications, while each separate layer describes each concern only. This is to enhance the cohesiveness of each document, and then secure fundamental legibility and help to understand the document. Especially, it separates technical contents including motions and realizations of each stage from practicable specifications to remove the complexity of the contents. Practical verification of such principles includes applying the SRP (Single Responsibility Principle) of the object-oriented design (SOLID). In order to check this, conduct an SRP test on specifications and automation codes, verify the suitability of the responsibilities of specifications and automation code layers, and conduct secondary layer segregation. The specification layer is a part where requirements are clearly stated through the acceptance criteria. The specifications using the exercises describe testable requirement, mainly the exercises in the table format. This part can be configured, being divided into two parts: a part to describe brief requirements in the user story style and a part to describe according exercises. When describing the specifications, the exercise part in the table format is somewhat related to the actual test operation sequence, while direct input value and the resulting value are included. To improve the legibility and understanding of the test operation sequence or the test step definition, following the traditional S-S-V-T structure enables more clear description.

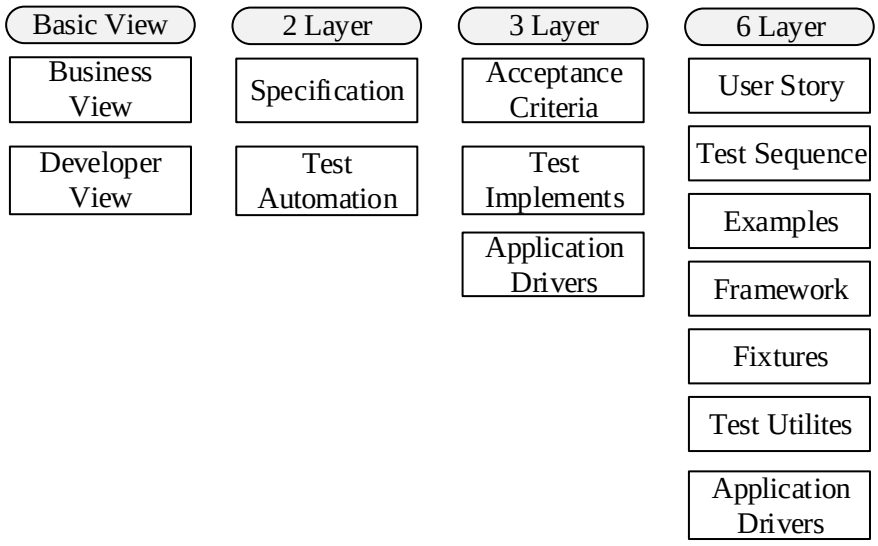

Figure 4. Automated Design Process

3.3. Classification of Test Automation Tools

In order to systematically carry out an actual application project, a process for acceptance test automation was designed and applied. It was carried out in four steps: Establishment of strategies − Derivation of requirements − Preparation of requirements specification and test case − Construction of test automation. In step 2, ‘ Derivation of requirements’ and step 3, ‘ Preparation of requirements specification and test case’ to verify the requirements, the specification was improved through mutual verification by applying repetitive iteration.

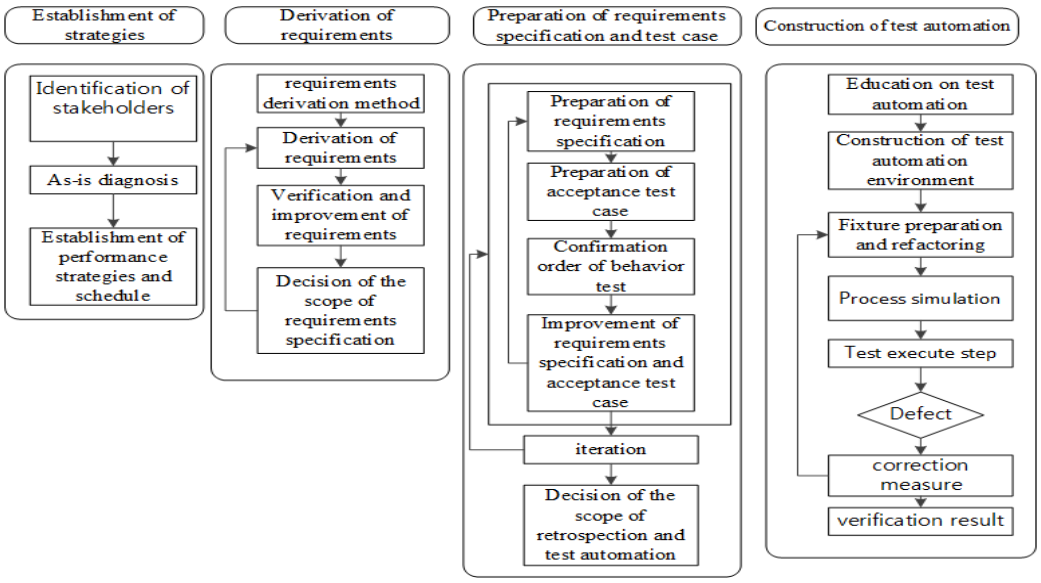

Figure 5. Acceptance Test Automation Process for Rebuilding

This repetitive improvement was performed until the specification part is clearly composed of the three phases of user story – step define – example as proposed in this paper. Furthermore, in step 4, ‘Construction of test automation’, repetitive iteration was applied to the process of ‘fixture preparation – fixture refactoring – test suites composition – test performance.’ Through this fixture refactoring process, method/class

extract were performed repeatedly until the form of custom fixture – test utilities – application driver was established according to the architecture proposed in this paper. The proposed acceptance test automation architecture will provide proven guidelines that can be used repeatedly to achieve the basic quality when applied to various acceptance automations in the long term. By providing an automation architecture, such concretized acceptance test can be used in combination of various specification-based test frameworks such as FitNesse, thereby improving quality.

4. Conclusion

Modern software testing methods are designed to define features and functions that business experts desire, growing out of past automation tests to verify development product and find faults at the end of the development cycle. However, such core value is conceptual and lacks in practical application and instances. With many supporting automation tools, this pattern was flooded with application methods in many different dimensions. It was even altered as a simple UI-based acceptance test method. This study proposed an acceptance test automation method for "specifics using example" based on establishing architecture. The results can serve as a practical basis for typical and repetitive execution of the conceptual agile acceptance test-driven development methodology to secure fundamental quality. However, it is difficult to measure performance and effects with the proposed acceptance test automation architecture alone. Therefore, further research is needed for the confirmation procedure about its positive effects on the development organization and process as well as on actual customer satisfaction.

References

[1] L. Crispin, "Agile Test Planning with the Agile Testing Quadrants", ADP Testing Workshop, **(2009)**.
[2] "Agile Testing and Quality Strategies: Discipline Over Rhetoric", http://www.ambysoft.com/essays/agileTesting.html#AgileRequirementsStrategies
[3] ISTQB Test Fundamental Syllabus -http://www.istqb.org/downloads/finish/16/15.html
[4] B.W. Boehm, J.R. Brown and M. Lipow, "Quantitative evaluation of software quality", *In:* Proceedings, pp. 592-605.
[5] J.C. Na, Y. Oh and S. Ryoo, "Implementing an Automated Testing Framework through the Integration of FitNesse and STAF", Journal of KIISE : Computing Practices and Letters, vol. 16, no. 5, **(2010)**, pp. 581-585.
[6] G. Meszaros and S.M. Smith, "The Test Automation Manifesto. Extreme Programming and Agile Methods", XP/Agile Universe 2003, vol. 2753, **(2003)**, pp. 73-81**.**
[7] E.H. Kim, J.C. Na and S.M. Ryoo, "Implementing an Effective Test Automation Framework", 2009 33rd Annual IEEE International Computer Software and Applications Conference, compsac, vol. 2, **(2009)**, pp. 534-538.
[8] Selenium Documentation Team, "Selenium Documentation", SeleniumHQ, Sp. JetBrains and Atlassian, Available : http://seleniumhq.org/docs, **(2012)**.
[9] Separation of Concerns (SoC), http://en.wikipedia.org/wiki/Separation_of_concerns.
[10] R. Mugridge and W. Cummingham, "Fit for Developing Software", Prentice Hall, **(2005)**.
[11] Y.S. Hwang, S.M. Jung and C.D. Hwa, "A Keyword-based UI Test Framework for Web Services", Journal of KIISE: Software and Applications, vol. 38, no. 12, **(2011).**

Authors

Jung-Ah Shim, She received her MBA in Culture and Art Management from Hongik University in Korea(2013). She has worked as an expert in Communicology and Social Network Service Public-Relations field for 8 years. Now she is an administrator in the Ministry of the Interior.

Hyun-Jung Kwon, She received her bachelor's degree in Computer Science from Seoul Women's University in Korea, (1999). She worked in the IT field and Cyber Security Center. Her research interests are in areas of Cyber Security, Personal Information Protection Policy and Technology.

Hyun-Ju Jung, She received her Master degree in Journalism and Broadcasting from DongGuk University in Korea, (2011). He worked in the National assembly as a public relations specialist over 11 years. His research interests are in areas IT sector policies

Moon-Sung Hwang, He received his bachelor's degree of chemical engineering in Korea University, Seoul (1985) and Master's degree of business administration (MBA) in Korea University, Seoul (2008). He worked in multinational company over 20years. His research interests are in the areas of business process.