AgileUAT: A Framework for User Acceptance Testing based on User Stories and Acceptance Criteria

Pallavi Pandit
Department of Information Technology
MITM
Indore

Swati Tahiliani
Department of Information Technology
MIST
Indore

ABSTRACT
User Acceptance Testing (UAT) has widespread implications in the software community. It involves not only the end-user, but the Quality Assurance (QA) team, developers, business analysts and top level management. UAT is conducted with the aim of developing confidence of the user in the software product. UAT is generally performed manually and not preferred to be automated. UAT frameworks exist for Agile methodologies such as Scrum. We propose a UAT process model which adapts the generic agile process model. Hence, it is able to encompass every agile methodology. AgileUAT, aims at generation of exhaustive acceptance test cases in natural language, based on acceptance criteria. It indicates whether the acceptance criteria is fulfilled or not, as a percentage value. The tool illustrates traceability among epics, user stories, acceptance criteria and acceptance test cases. We explore several different templates for user stories and acceptance criteria. In the future, we aim to provide a direct mapping between the acceptance criteria and acceptance test cases based on permutations and combinations using decision tables.

General Terms
Software Engineering -> Software Creation and Management -> Software Verification and Validation -> Process Validation -> Acceptance Testing

Keywords
Agile, UAT, user story, epic, acceptance criteria, traceability.

1. INTRODUCTION
Defects may occur at any stage of software development. If these defects are not fixed early, they become more and more expensive to fix. Testing helps us to measure the quality of the product in terms of defects found. Testing is conducted at many levels: Component Testing, Integration Testing, System Testing and Acceptance Testing[1].

Acceptance testing is when a user checks another's work for the purpose of accepting it. Acceptance Testing is establishing confidence in the user that the software product is fit for purpose. So, acceptance testing performs validation on the software product. Acceptance Testing is conducted by the user or customer, although it may involve other stakeholders.

Acceptance Tests can be classified as User Acceptance Tests (Internal Alpha Tests and External Beta Tests), Operational Acceptance Tests, Regulatory Acceptance Tests and Contract Acceptance Tests. The goal of user acceptance testing is reassurance. The motivations for UAT are presentation, demonstration, probing, usability and validation[2].

The V-Model illustrates the mapping between development phases and the corresponding testing phases. In this model, Requirements Gathering phase maps to Acceptance Testing Phase.

There are several approaches for User Acceptance Testing, viz., Requirements-based, Business Process based and Data driven[3]. We are following the Requirements-based process in which the user stories and acceptance criteria form the basis of the UAT process.

In User Acceptance Testing, manual testing is done by the user. Generally, UAT is not automated. Otherwise it would be considered as an automated test case for checking application functionality. However, if users are busy to test after every build or we have an understaffed testing team, we may consider automating certain tests[4].

We focus on UAT in Agile. Our approach consists of translating acceptance criteria into natural language tests for performing UAT. A case study exemplifies our work.

2. BACKGROUND STUDY
A user story is defined as "A user story is a tool in Agile development to capture a description of a software feature from an end-user perspective"[5]. The user stories have to address functional as well as non-functional characteristics. Every story includes acceptance criteria for these characteristics[6].

During each iteration, developers write code to implement the user stories, with the relevant quality characteristics, and this code is verified and validated via acceptance testing[6].

Acceptance criteria are said to be testable if they include functional behavior, quality characteristics, scenarios (use cases), business rules, external interfaces, constraints and data definitions[6].

There exists a traceability among epics and user stories, user stories and acceptance criteria, and acceptance criteria and acceptance test cases[7]. These traceability elements can be shown via a traceability tree or a traceability matrix or exported to an Excel sheet.

UAT best practices include focusing on requirements, designing systems for testability and consideration of usability testing[8]. Comprehensive UAT checklists [9][10] ensure that the process is carried out in the right manner. Guidelines for UAT are provided in [11].

3. RELATED WORK

3.1 Academia

[12] have proposed a model based technique for specifying user stories in the form of test models. These test models are enhanced with implementation details during sprint planning thus serving as a specification for developers. Testers further enhance them with test data and automatically generate test tables out of them using test generator. These tables can then be executed by Selenium.

3.2 Industry

[13] have classified the challenges for UAT in agile development model into four categories which include business challenges, people & process, governance and tools & automation. And to overcome these challenges they have proposed a UAT Centre of excellence (CoE) framework. The recommend unique UAT approach addresses all its challenges in an Agile development model, where the UAT team would work in tandem with the development and QA teams using CoE best practices to enhance test coverage and efficiency. eliminate many potential defects with early business validations and improves efficiencies through optimum automation of regression test beds.

The UAT team gets involved early in SDLC and is engaged in the entire iterative process. The UAT team works with a story card acceptance criteria for each iteration. This helps

3.3 Tools for performing UAT

Several tools exist for performing UAT, which are outlined in [14]. Specifically, Cucumber, Jira, Fitnesse, Explorer, RSpec make use of acceptance criteria for designing acceptance test cases.

4. METHODOLOGY

We propose a logical framework for translating user stories and acceptance criteria into natural language user acceptance tests. We have studied the process model for agile and adapted it as per our requirements.

During **Pre-Iteration Planning,** first, we elicit epics/user stories in the form of a template: As a <user>, I want <feature> so that <benefit>. Afterwards, we input the acceptance criteria in one of the templates selected by the user. **During Iteration Planning**, we prioritize the user stories using the MoSCoW (Must, Should, Could and Would) acronym. **During Iteration Execution**, we extract the role (user) and feature of the epic/user story and the business value (benefit). We generate a test use case diagram with the user and functionality (feature). The test use case diagram acts like a basis to user acceptance testing (UAT). There is the same notation for both developers and end-users and testers. Hence, We stereotype use case diagram into test use case diagram. **During Iteration Wrap-Up**, the acceptance criteria are translated to acceptance tests (positive, negative and non-functional). **During Post-iteration consideration(Reports)**, we can view retrieve user stories can be done By user/ By role/ By date/ By theme/ By epic/By iteration/By priority. **During Post-iteration consideration(Traceability)**, we associate business requirements with user with feature(epics) with sub-feature(user stories) with acceptance criteria with user acceptance tests. Further, we generate an Excel sheet to show traceability.

During Post-iteration consideration(Defect Log), we will see how many acceptance tests have passed and whether the acceptance criteria is fulfilled or not.

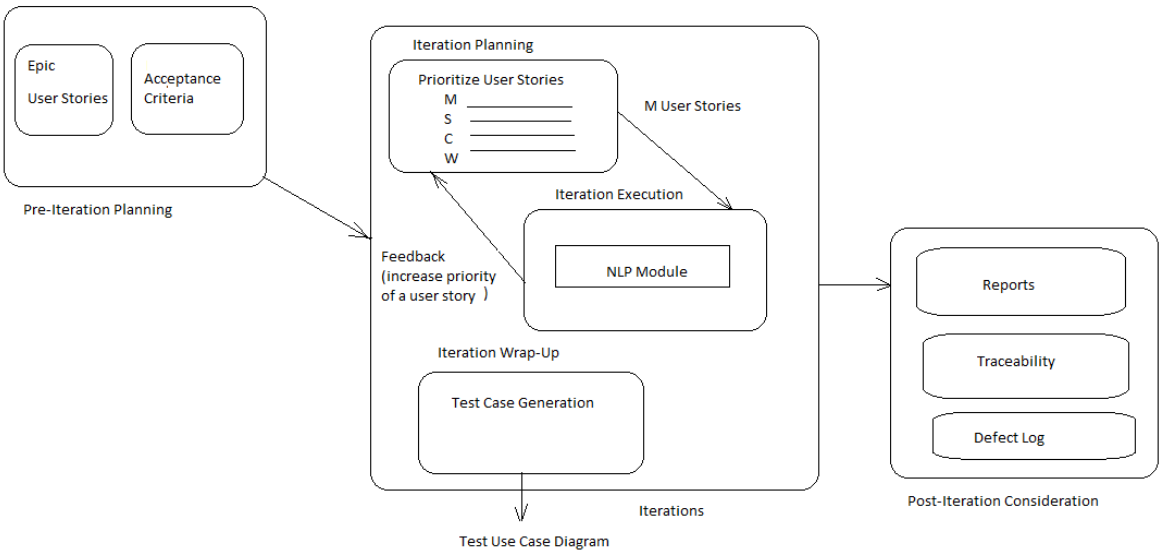

Fig 1: AgileUAT: our adaptation of the Generic process model of Agile

5. CASE STUDY

Element	**Template[15][16]**	**Example[7]**	**Extracted Information**
Epic 1	As a <role> I want <feature> so that <benefit>	As an internet banking customer, I want to avail the online banking facilities so I can work from home	Role1: internet banking customer Feature1: avail the online banking facilities Benefit1: I can work from home
User Story 1	As a <role> I want <sub-feature> so that <benefit>	As an internet banking customer I want to list my account balances so that I can understand my financial position.	Role1: internet banking customer Sub-Feature1: list my account balances Benefit2: I can understand my financial position
Acceptance Criteria 1	Given [inputs | preconditions] When [actions | triggers] Then [outputs | consequences]	Acceptance Criteria1 Given the customer has one credit account and one savings account When they have logged in successfully Then the two accounts will be listed in account number order (Account no, Name, Balance, Available Funds)	Inputs/Preconditions: The customer has one credit account and one savings account Actions/Triggers: When they have logged in successfully Outputs/Consequences: Then the two accounts will be listed in account number order (Account no, Name, Balance, Available Funds)
User Acceptance Test 1 (Positive)	Verify that preconditions occur, verify that action occurs, verify that outputs are generated	Steps: 1. Verify that customer has one credit account and one savings account 2. Verify that they have logged in successfully 3. Verify that the two accounts will be listed in account number order (Account no, Name, Balance, Available Funds)	10% acceptance criteria fulfilled
User Acceptance Test 2 (Negative)	Verify that invalid inputs occur, Verify that trigger does not occur, Verify that when outputs are not generated, an error message is displayed to the user	1. Verify that customer does not have one credit or one savings account 2. Verify that client cannot login successfully	20% acceptance criteria fulfilled

		3. Verify that message is displayed to the user	
User Acceptance Test 3 (Negative)	Verify that preconditions occur, actions or triggers do not occur, then error message is displayed to the user	1. Verify that customer has one credit account and one savings account 2. Verify that client could not login successfully 3. Display the error message to the user	30% acceptance criteria fulfilled
User Acceptance Test 4 (NFR)	Select attribute: performance	Verify that response time < 5 seconds	40% acceptance criteria fulfilled
User Acceptance Test 5 (NFR)	Select attribute: security	Verify that login is secure	50% acceptance criteria fulfilled
User Acceptance Test 6 (NFR)	Select attribute: availability	Verify that the service is available 24*7	60% acceptance criteria fulfilled
User Acceptance Test 7	Verify that service is working for the specified user(s)	For each user, run User Acceptance Test 1 through 6	100% acceptance criteria fulfilled

6. IMPLEMENTATION AND RESULTS

We have represented our solution structure as a DOM tree which can be written to XML format as shown below. We are associating the elements using a traceability links which will be shown as a traceability tree. The test cases are derived from the acceptance criteria and written to an Excel sheet.

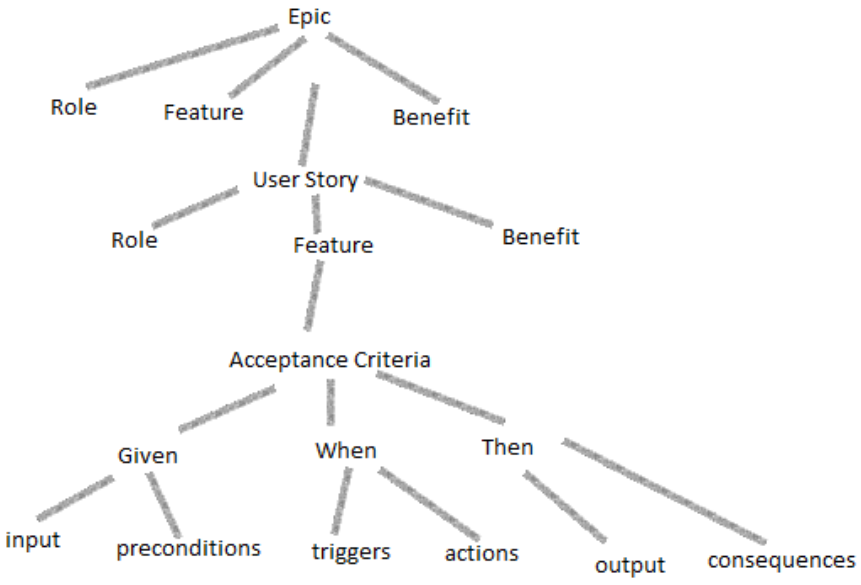

Fig 2: AgileUAT: DOM tree structure

6.1 Traceability Links

Here, we establish traceability links from user stories to test cases.

6.1.1 By Epic

Benefit 1->Role 1-> Epic 1->Feature 1-> Acceptance Criteria 1-> User Acceptance Test 1

6.1.1.1 By User Story

Benefit 1->Role 1-> Epic 1-> User Story 1-> Sub-Feature 1-> Acceptance Criteria 1-> User Acceptance Test 1

6.2 XML Representation

We are writing the generated test use case diagram in XML as per [17].

<Epic>

<Role>internet banking customer</Role>

<Feature>avail the online banking facilities</Role>

<Benefit>I can work from home</Benefit>

<User Story>

<Role>internet banking customer</Role>

<Feature>list my account balances</Feature>

<Benefit>I can understand my financial position</Benefit>

<Acceptance Criteria>

<Given>

<input></input>

<precondition>The customer has one credit account and one savings account</precondition>

</Given>

<When>

<action>When they have logged in successfully</action>

<trigger></trigger>

</When

Table 1: Acceptance Test Cases from Acceptance Criteria1

<Then>

<output>Then the two accounts will be listed in account number order (Account no, Name, Balance, Available Funds)</output>

<consequence></consequence>

</Then>

</Acceptance Criteria>

<Benefit></Benefit>

</User Story>

</Epic>

6.3 Acceptance Test Cases for Acceptance Criteria 1

Test case generation for acceptance criteria is exemplified in [18]

Suggested Test Cases based on the case study are generated below. More test cases can be generated by the end-user or QA team.

Given the customer has one credit account and one savings account Step1

When they have logged in successfully Step2

Then the two accounts will be listed in account number order (Account no, Name, Balance, Available Funds) Step3

S. No.	AC Step	Prerequisites	Positive Scenario/Negative Scenario/NFR Scenario	Acceptance Test Scenario description	Acceptance Test Scenario Steps	Input Values	Expected Result(s)
AT S1	Step 1	None	Positive	Verify that the customer has one credit account and one savings account			
AT S2	Step 2	Step1	Positive	Verify that the customer has logged in successfully			
AT S3	Step 3	Step2	Positive	Verify that the two accounts will be listed in account number order(Account no, Name, Balance, Available Funds)			

7. CONCLUSION AND FUTURE WORK

We have used the simplest acceptance criteria template(Given, When, Then). This template can be extended to use multiple ANDs for different conditions. Our work can be extended to incorporate these multiple ANDs and generate test cases accordingly. One of these examples is shown below[7]:

Given the customer has twenty five accounts step1
And they have logged in successfully step2
And they are on the first page of the list step3
When they activate the Next Page button step4
Then the list will be cleared step5
And the list will be populated with the last five accounts step6
And the Previous Page button will be enabled step7
And the Next Page button will be enabled step8

Our work can be extended to group related stories into themes using Natural Language Processing.

We will generate a test use case diagram from epics/user stories. This diagram is an extension of the use case diagram for testing purpose. It provides same notation which can be used by both developers and testers. It is used for visualizing roles, functionality listed in a user story.

8. REFERENCES

[1] Graham D., Veenendaal E., Evans I., Black R. Foundations of Software Testing. 2008 Cengage Learning EMEA

[2] Michael Bolton. DevelopSense. User Acceptance Testing – A Context-Driven Perspective.

[3] USER ACCEPTANCE TESTING (UAT) PROCESS.Version 1.0.March3, 2008. British Columbia. Information and Technology Management Branch.IM/IT Standards and Guidelines.

[4] http://www.searchsoftwarequality.techtarget.com/answer/Automating-user-acceptance-test-cases

[5] http://www.searchsoftwarequality.techtarget.com/definition/user-story

[6] ISTQB Agile Tester Syllabus

[7] http://www.batimes.com/articles/user-stories-and-use-cases-dont-use-both.html

[8] https://www.develop.com/useracceptancetests

[9] www.testingpro.net/2013/07/user-acceptance-testing-uat-checklist.html

[10] www.iste.uni-stuttgart.de/fileadmin/user_upload/iste/se/links/links-se/checklists/download/Accpetance.html

[11] docs.oracle.com/cd/E14004_01/books/DevDep/Testing_Guidelines3.html

[12] Löffler R., Güldali B., Geisen S. Towards Model-based Acceptance Testing for Scrum.

[13] http://searchsoftwarequality.techtarget.com/tip/Streamlining-user-acceptance-testing-UAT-with-Agile

[14] www.testdriven.com/tag/acceptance_testing_tools_post_tag/

[15] https://www.mountaingoatsoftware.com/agile/user-stories

[16] http://guide.agilealliance.org/guide/gwt.html

[17] Dranidis D., Tigka K. Writing Use Cases in XML

[18] http://testerstories.com/2011/08/be-acceptable-write-tests-from-stories/