Volume 5, Issue 10, October-2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Impact of Use Cases on Test Cases – Agile & Heavy Weight Methodologies

B. V. Ramana Murthy
Department of Computer Science & Engineering
Mahaveer Institute of Science & Technology
Hyderabad, Telangana State, India

Jogannagari Malla Reddy
Department of Computer Science & Engineering
Lingaya's University, Faridabad,
Haryana State, India

Abstract---- Software Engineering is our organized method that involves the phases like software development, operations, maintain and again, in this paper we attempt the significance of use case on Test Cases in the software Testing methodologies, use case are the starting point of analysis and are a popular way to express software requirements. Test Case acts as the starting point for the test execution, which has a set of test data. Here, we take we standing methodologies agile and weary weight / non agile to know the significance and impact of good / knowing use cases when we write test cases. An AHP technique has been used to know the relative preference between two alternate ranging from 0 to 9. This technique is uses between using agile methodology and non agile methodology, which computes the weightage of both the cases.

Keywords--- Agile, Non Agile, Use Cases, Test Case, Testing Methodologies, AHP

I. INTRODUCTION

Use cases are a popular way to express software requirements. They are popular because they are practical. A use case bridges the gap between user needs and system functionality by directly stating the user intention and system response for each step in a particular interaction.

Use cases are simple enough that almost anyone can read them. Even customers or users can read use cases without any special training. However, writing use cases takes some practice. It is not difficult to start writing use cases, but really mastering those takes training, practice, and insight.

One goal of writing use cases is to specify the system to be built, so that the resulting specification can be handed off to someone else for implementation. Another important goal is to allow potential customers to read the use cases and validate them. Long before you reach these goals, you will find that the process of writing use cases is itself a very useful way to clarify your own thinking about the system requirements by writing step-by-step descriptions, you force yourself to think through the details of the system's features.

A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

Building test cases is about doing your best to find the worst in a program in order to deliver a quality product to the customer. Without a well-designed and executed test plan, the results can be catastrophic and very costly. For example, because of an error in the software, the first flight of the Ariane 5 rocket resulted in its complete destruction and the loss of four satellites at a cost of more than $370 millions. In the U.S. alone, software bugs cost the economy $59 billion back in 2002. Clearly, software defects must be found and corrected before unleashing new programs to the customer.

What is a Test Case?

A test case has an input, an action and an expected result. In using test cases, the tester is trying to break the application. The whole point of using test cases is to find defects. Hopefully, serious defects that crash the system are found before your application is released to the customer. Test cases must exercise every feature of the application to prevent defects from being released. Each test case needs to contain a set of test steps of a feature or function. At the end of the test the expected results are compared to actual results to determine if the application is working as it should.

A test case can have information that includes the test case name, goal, environment, steps to take, input and expected results. A good test case consists of several ingredients: its chance of finding an error is high, it doesn't copy other test cases, it's most likely to find error or defects, it's not too simple or too complex and it is obvious to a tester when an application failure has occurred.

Test Cases and the Attitude of the Software Testers

There is a profound connection between creating and using test cases and the attitude of the tester. Test cases are about finding errors and breaking software. However, most people are more interested in building something than in breaking something. For that reason, we must be careful with how we develop and use test cases.

The goal of using test cases should be to point out the errors in a program. If this goal is set, then there is a higher probability of finding defects.

Techniques for Developing Test Cases

The software testing cycle varies between organizations. However, the testing cycle and the use of test cases usually progresses through the following stages :

1. Requirements analysis : During this stage, testers and developers work together to see how well it can be tested and within what parameters.
2. Test planning : The testing cycle requires a plan to organize testing efforts, including the creation of test cases.
3. Test development : Test cases are created to test the software.
4. Test execution : Testers run the software, including various test cases. Defects are reported.
5. Test reporting : Testers write their reports regarding the readiness of the software for release.
6. Test result analysis : Testers and customers determine which defects to fix and which can be delayed.
7. Retesting Defects : After a defect has been identified and corrected, test cases are run again by testers.
8. Regression testing : Test cases are run to determine if the software is still working after changes have been made to the code.
9. Closure : Documents, results, test plans, etc. are archived once the software passes all the tests

II. IMPACT OF USECASES ON TEST CASES

Usually step 1 is one which an Use Cases are developed by the Project Management Team, as discussed above Use Cases will give good insight to Test Cases. Use cases are defined as how an actor interacts with the system, where an actor is usually the user but could be another piece of software or a piece of hardware. Because the main part of a use case is a fairly concrete step-by-step description of that interaction — the actor does this, and then the system does that in response — use cases have been found good for aiding communication. Use cases can help a developer implement code that actually works the way a user expects, instead of the developer's guessing and ending up developing code that functions differently from the user's expectations or needs. Typically, use cases are considered a format for documenting requirements, which means we get the most value when the use cases are written early so they can be a basis for developing both the software and tests of the software. In many organizations, business analysts have the responsibility for defining requirements and thus would seem a simple answer to the question of who should design use cases. However, many other organizations don't have business analysts, and therefore use cases could be written by the project manager, a programmer, a user, or practically anyone for that matter.

Testers typically not use case authors**,** Testers might be unlikely use case authors at this point because they often do not become involved until later. That in itself is an issue because testers' effectiveness is diminished considerably when they are brought into a project too late, such as after the code has been written, which alas is very common. While indeed it is desirable for testers to become involved early, it is not the tester's job to define requirements. However, use cases are a helpful format for guiding testing. In fact, a valuable by-product of use cases is their natural linkage to test cases. At a minimum, a use case is tested by executing the use case. Consequently, if use cases have not already been written to document requirements, testers frequently will write use cases themselves to use as tests. Moreover, there's always the question of how accurate the tester's use cases are, since the fact the tester is writing them late in the process suggests there's inadequate information about the requirements upon which to base the use cases.

To be an effective basis for testing, the use case will describe what should be happening, not what is happening in the program. Moreover, use cases will describe not only the main path, which is how the use case is normally executed, but also alternative and exception paths.

Agile and Non Agile/Heavy Weight Methodologies(Water Fall , Spiral etc): Traditional Non Agile methods for developing software are rapidly declining in popularity as more recently developed Agile methodologies are increasingly adopted. But what's the difference between the two – and is Agile always better?

What is Non Agile ex: waterfall?-The waterfall model is one in which each phase of a product's life cycle takes place in sequence, so that progress flows steadily downwards through these phases like a waterfall.

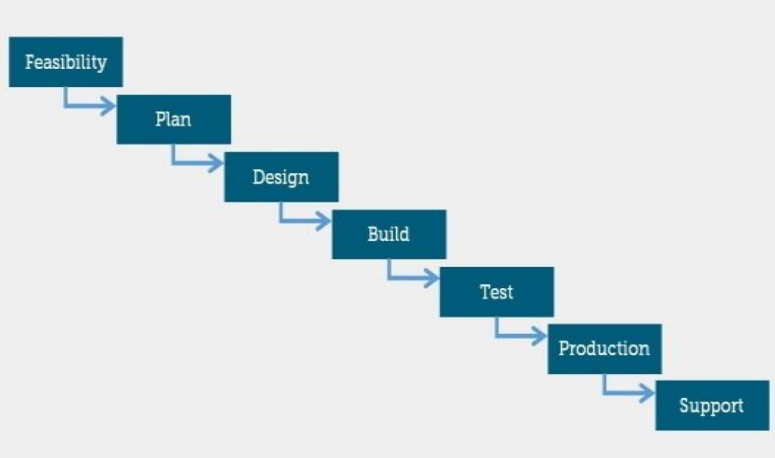


Fig. 1 : Waterfall model

Nobody invented the waterfall method. Rather it was inherited by enterprise software developers from other industries where, once a particular phase of production is complete, it was incredibly costly or impractical to go back and make changes. The waterfall was only codified when people subsequently realized that it wasn't the only way of doing things. In waterfall methodologies all the requirements gathering and design work is done before any coding takes place.
In general these methodologies have stages that deal with what you need to do before a project, during a start up phase, a planning phase, an execution phase and a closing phase. They also then have a series of processes for managing work packages, exceptions, reporting, risks and issues.

2.1. Pros of the waterfall method

Potential issues that would have been found during development can be researched and bottomed out during the design phase. If appropriate meaning an alternate solution is selected before any code is written. The development process tends to be better documented since this methodology places greater emphasis on documentation like requirements and design docs. Many organizations find this reassuring.
Because the waterfall process is a linear one it is perhaps easier to understand, especially for non-developers or those new to software development. Often teams feel more comfortable with this approach.

2.2. Cons of the waterfall method

Often the people we're building software for (the client) don't know exactly what they need up front and don't know what's possible with the technology available. This way of working doesn't handle this well. Solution designers often aren't able to foresee problems that will arise out of the implementation of their designs. Changes to requirements (e.g. like those resulting from new technologies, changes in a market or changes to business goals) can't easily be incorporated with the waterfall method and there are often laborious change control procedures to go through when this happensThe process doesn't have its own momentum

2.3. What is agile?

An Agile software development methodology such as Scrum is one which eschews a linear, sequential approach in favour of an incremental, iterative one. Instead of extensive planning and design up front, Agile methodologies allow for changing requirements over time by using cross-functional teams – incorporating planners, designers, developers and testers – which work on successive iterations of the product over fixed time periods (time boxes). The work is organized in to a backlog that is prioritized in to exact priority order based on business (or user) value**.**
Empirical Analysis we find Impact and Significances of Test Cases and Use Cases on Agile & Non Agile Methodologies Now, Having come across Use Cases, Test Cases, Agile and Non Agile lot thinking has gone into it to know the weights which gives Impact and significances of use cases on test cases when we use this mythologies Agile OR Non Agile/Heavy Weight.
Finding weight ages of Test Cases (by using Use Cases) using Agile and Non Agile methodologies are carried out empirically by AHP technique. Questioner is framed to know the weights and on interaction with Project Manager the data has been evolved and results are interpreted.

III. PAIR WISE COMPARISION MATRIX USING ANALYTIC HIERARCHY PROCESS

The Pair wise Comparison Method was developed by Saaty (1980) in the context of the Analytic Hierarchy Process (AHP). This method involve pair wise comparisons to create a ratio matrix. It takes, as input, the pair wise comparisons and produces the relative weights as output
The method employs an Underlying scale with values ranging from 1 to 9 to rate the relative preferences fort criteria (see table).
Importance Definition

1 Equal importance
2 Equal to moderate importance
3 Moderate Importance
4 Moderate to strong importance
5 Strong importance
6 Strong to very strong importance
7 Very strong importance
8 Very to extremely strong importance
9 Extreme importance

Source: Saaty Scale for pair wise comparison
The following steps are involved in finding
weights using AHP Techniques

1 Create a ratio matrix- employs scale with values ranging from 1 to 9 to rate the relative preferences for two criteria
2. Sum the values in each column of the pair wise comparison matrix
3. Divide each element in the matrix by its column total (the resulting matrix is referred to as the normalized pair wise comparison matrix
4. Compute the average of the elements in each row of the normalized matrix, that is, divide the sum of normalized scores for each row .

5. These averages provide an estimate of the relative weights of the criteria being compared.
6. Impact and significances of use cases on test cases – Agile and Non Agile/Heavy Weight.

Case 1. Using Agile Methodology the Impact of Use Cases on Testing to get good test cases.
Step 1 create a ratio matrix: Two criteria's Use Cases & Test Cases, to find out relative preferences

Agile Methodology	**Use Case**	**Test Case**
Use Case	1	1/2
Test Case	3	1

Data Source: Project Manager
Step 2. Sum the values in each column of the pair wise comparison matrix

Agile Methodology	**Use Case**	**Test Case**
Use Case	1	½
Test Case	3	1
	4	1.5

Step 3:Divide each element in the matrix by its column total (the resulting matrix is referred to as the normalized pair wise comparison matrix

Agile Methodology	**Use Case**	**Test Case**
Use Case	0.25	0.33
Test Case	0.75	0.66

Step 4. Compute the average of the elements in each row of the normalized matrix, that is, divide the sum of normalized scores for each row .

Agile Methodology	**Use Case**	**Test Case**	**Weight**
Use Case	0.25	0.33	0.29
Test Case	0.75	0.66	0.71

Step 4 gives the weightages of UseCases based Test Cases using Agile methodology. It has been found that the Impact of UseCases on Test Cases when Agile methodology is used is 29% and without UseCases is 71%, it is an observation as the Agile in Incremental model the Impact of UseCases is less on writing TestCases, but it is significant percentage for writing Test Cases.

Case 2. Using Non Agile Methodology the Impact of Use Cases on Testing to get good test cases.
Step 1create a ratio matrix: Two criteria's Use Cases & Test Cases, to find out relative preferences

Non Agile Methodology	**Use Case**	**Test Case**
Use Case	1	7
Test Case	2	1

Data Source: Project Manager
Step2:. Sum the values in each column of the pair wise comparison matrix

Non Agile Methodology	**Use Case**	**Test Case**
Use Case	1	7
Test Case	2	1
	3	8

Step 3:Divide each element in the matrix by its column total (the resulting matrix is referred to as the normalized pair wise comparison matrix

Non Agile Methodology	**Use Case**	**Test Case**
Use Case	0.33	0.875
Test Case	0.66	0.125

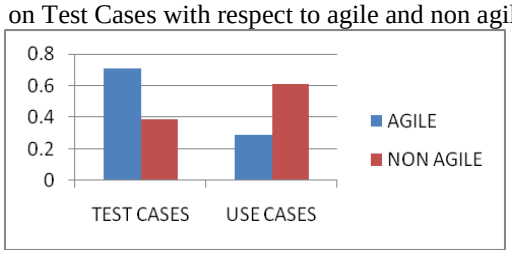
Step 4.Compute the average of the elements in each row of the normalized matrix, that is, divide the sum of normalized scores for each row .

Non Agile Methodology	**Use Case**	**Test Case**	**Weight**

Use Case	0.33	0.875	0.61
Test Case	0.66	0.125	0.39

Step 4 gives the weight ages of Use Cases based Test Cases using Non Agile methodology. It has been found that the Impact of UseCases on Test Cases when Non Agile methodology is used is 61% and without Use Cases is 39%, it is an observation as the NonAgile has Impact of Use Cases on writing Test Cases and it significant percentage for writing Test Cases

Results Interpretation : Based on the weightages of case 1 and case 2 , the following table gives the clear information about the significances of Use Cases on Test Cases with respect to agile and non agile methodologies



IV. CONCLUSION

The weightage of Use Cases on Test Cases for Agile is 71% whereas for Non-Agile is 39% this shows the significance of Use Cases on Test Cases is important but the weightage of significance depends on the type of methodology we use, here weightage is in terms of generating good test cases. Empirical analysis of significance shows that the use cases plays dominant role in writing good test cases, percentage of weightage variations has been found from the result generated when we apply the concept of Agile Vs Non-Agile methodologies, Non-Agile taken more when compared to agile, the results shows in equivalence with Software Engineering Principals as Non-Agile is not incremental model and changes after testing will cost more, good test cases are to be incorporated to get good test cases the percentages of use cases weightage must be more. The impact of good use cases produces good test cases upon which better testing environment can be performed.

REFERENCES

[1] A. Bahram, Object oriented systems development : using the unified modeling language, Mc-Graw Hill, Singapore, 1999

[2] C. Nebut, F. Fleurey and Y.L. Traon, Automatic Test Generation: A Use Case Driven Approach, IEEE Transaction on Software Engineering, Vol.32, No.3, 2003

[3] D. Wood and J. Reis, Use Case Derived Test Cases, Software Quality Engineering for Software Testing Analysis and Review Online. http://www.stickyminds.com/

[4] I. Jacobson, G. Booch, J. Rumbaugh. The Unified Software Development, England,1992.

[5] J. Gutierez, Escalona M.J. and Torres M.M, An Approach to Generate Test Cases from Use Cases, Proceedings of the 6th International Conference on Web Engineering. pp. 113-114,2006.

[6] J. Heumann, Generating Test Cases from Use Cases, Rational Software, IBM,2001

[7] J. Jansen, Using an Intelligent Agent To Enhance Search Engine Performance,1996, http://www.firstmonday.org

[8] R.V. Binder, Testing Object-Oriented System. Addison-Wesley. USA,2000.

[9] Rational, Mastering RequirementsManagement with use,2003

[10] T. Stanley, Intelligent Searching Agent on Web,http://ariadne.ac.uk/issue7/search-engine

