A Pattern Based Approach To Prioritize Test Cases For User Stories In An Agile Environment

Anita/Research Scholar*, Dr. Naresh Chauhan/Chairman and Head of Department.
YMCA University of Science & Technology, Faridabad, Haryana, India
anitaarora_20@rediffmail.com*, nareshchauhan19@gmail.com

ABSTRACT

An agile framework is designed for undertaking software projects at any level of development process. Methodologies that are used in agile are hybrid cum dynamic in nature. In every new iteration/sprint, team members use the lessons that are framed in the sprint review meeting. Sometimes, team members may reach to a better solution for the same problem for the upcoming sprint. This process is incremental in nature. Problem discovery and different solution proposing is reoccurring in every sprint life cycle. Specific patterns or templates are problem-solution pairs that provide tested solutions for commonly occurring problems in specific domain. The pattern concept is not the ultimate outcome but it can act as one of the aid in saving time for achieving good results. These patterns may be used to increase maintainability, comprehensibility, reusability and code quality. Effective reusability is the important factor in regression testing. In this paper, we have proposed an empirical approach using test pattern to prioritize test cases of a user story by considering the dependency factor into account.

Index Terms: Software testing, Agile Testing, Quality, Test Pattern, Test Template, Regression Testing, Test Case Prioritization.

I. INTRODUCTION

Emerging technologies and developments are supported and explored in software development domain by different software professionals so as to satisfy the dynamic or any requirements of customers. Temporal changes proposed by customers for her project may give birth to new technologies and developments. Also, in some cases market standards forcefully demands change in existing principles and policies. There may be two problems along with the introduced change so as to fulfill specific needs. First and foremost problem is acceptance among team members for adapting to new technology, new environment or new solution for existing problem. Secondly, risk involved while adapting to new change is cumbersome. Agile software development [1] is one of the emerging changes in software development [2] in the year 2001. In the initial phase, execution of agile faced many aforementioned challenges but newbie of agile tackled risks so well by planning and show outstanding results by using agile defined methodologies. Now, software professionals of upcoming market have realized the importance and need of this emerging area, that's why, transition is taking place from traditional [3] to agile processes in software industry so rapidly.

Also, a pattern or template is emerging trend for solution framing in less time. Execution of the pattern for so called discovered problem is now easy. Generally, patterns are developed for reoccurring problems for specific domain and out of all the solutions, best solution is considered as template for the future cases. One of the manifestos of agile reveals importance of customer. That's why, for saving time and cost of customer, agile may use patterns or templates for tested deliverable with good quality. In agile, milestones to be achieved by team members are user story. In one implementation, the technique includes identifying user stories for a desired application and identifying pattern types corresponding to...
the identified user stories. Selected pattern may be stored in the in-house databases and are executed so as to achieve customer satisfied results. Agile based organization that provides an integrated solution during or after the sprint may require significant software design [4] and testing patterns, and further to add on, these customized integrated solutions are based on the particular needs of various clients.

For a team to reach the threshold level which is client specific, the development team not only focus on their agile process, rather focus also shifted to available agile resources. This means that the client’s requirement must meet the market standards. This may be achieved if the code is adaptable in nature and tested during every sprint. Hence, the team must understand and learn approach for creating maintainable software. Patterns have been developed in different discipline of software engineering. Major objective of patterns is to produce high-quality solutions by considering expectations framed by client. Patterns aims towards reusability and support to simplify analysis and design processes by reusing existing solutions of existing problems. A comprehensive collection of patterns that are broadly used in software analysis and design is design pattern.

Researchers thought of this concept so as to solve problems like, how to use design in long terms, how to improve and transfer knowledge through design patterns [5], how to improve the log section for existing problem, how to overcome the communication problem, how to improve the common vocabulary of software development process to reduce the culture related barrier, etc. in an agile environment.

Research results predicts that software design pattern [6] have become very popular in object oriented paradigm [7], as it shows relationship and interaction between objects and classes of real world entities. A good quality design pattern mostly emphasizes problems of object oriented software analysis/design and present a solution which apparently improves its quality with respect to reusability, maintainability, comprehensibility and flexibility to changes. The flexibility in changes is the back bone of agile working. Thus, frequent changes execution in real world brings multiple test suits. Testing [8] those bulky suites is a biggest challenge as time is important factor during frequent delivery phase of user stories. During maintenance phase, backlog size also keeps on increasing, so, there is a need for managing all these test suites in an efficient manner.

Management of test cases is one of the major issues in regression testing. Critical faults may be identified through various regression testing approaches such as test case prioritization [9], regression test case selection [10], test suite reduction or retest all. In this paper, we have proposed a technique to prioritize test cases using test pattern which is based on good design of the original user story.

The rest of the paper is organized as follows: section II reviews the related work. In section III, the proposed test pattern is provided by considering the coupling and cohesion principle. Hypothesis studies and results are discussed in section IV. The final section gives conclusion of the proposed work and what inventive measures should be taken to making this pattern a utility model.

II. RELATED WORK

In [11], a comparative analysis was performed among different patterns to find out which design pattern is more error prone. For this purpose, researchers had taken five open source systems and among those, java files, which contain bugs, were observed for pattern occurrence. Four design patterns were considered for evaluation; Singleton, Factory, Composite and Adapter. Different hypothesis were established and statistical tests were performed. It had been evaluated that Adapter pattern is more error prone as compared to other patterns. One other reason of being more error prone is that it is excessively used in all the projects especially Hibernate Project. So, when a pattern will be excessively used then its tendency of producing errors would be more for sure. On the basis of Gamma [12] [13], adapter constitutes structural design pattern defects [14] [15] [16] which in this case is missing design
pattern. As missing design pattern is the most common among four types of defects. So, there is a need of improvement in converting poor design into good design.

Architecture design and detailed design are important for getting the outline of the project having different user stories. Using coupling, it is possible to identify relationship among different user stories. Sometimes, design phase of software development life cycle is ignored while working with test cases. Test case prioritization methods including statement level coverage, branch level coverage and function level coverage, consider individual module and accordingly for that particular module test cases are prioritized. Our proposed approach is based on design for doing prioritization of test cases.

The advantage of design patterns is that it can speed up the development process by providing tested patterns. The study revealed that the information extracted through design patterns is very important during the maintenance of software applications. Furthermore, the reverse engineering of patterns from existing legacy applications and their reusability for developing new applications enable software developers to leverage best practices encapsulated as design templates.

III. PROPOSED WORK

A design and testing based test pattern is proposed in this paper which is for prioritization of test cases in an agile environment during regression testing. A regression testing is that phase of software testing in which one or more of user stories are added or removed in the existing active system. Due to this change, effects may be numerous in nature which may go unnoticed and may have substantial change in the outcome of the system. So, there is a need to fix this problem. This effect may spread in multiple directions with multiple intensities. In this paper, we have focused on a more severe module after introducing minute change in the original user story. The identification for the most severe module having strongest intensity is done using first order dependence matrix concept that was introduced by Myers. Before elaborating more about identification of severe user story, pattern for prioritization of test cases has been introduced.

The test pattern for the common problem of frequent change in an agile context is the reason for its existence. As customer or one of his representatives is always at the development site for supplying instant feedback, so, by this specific activity his contribution is more demanding for considering him in many other related tasks that are significant for user story enrichment.

Customer collaboration with teams is one of the manifesto in agile during user story development stage. Customer may or may not be technical sound but his instant feedback is useful for detecting and removing defects at an early stages (at the time of unit testing, integration testing, incremental testing or regression testing during sprint) of the sprint. Customer cannot frame good test cases for defect detection but skeleton of user story is specified by him/her. The test pattern is a kind of pattern which is simple and effective to use for customer satisfaction. Summary for the pattern is explained below:

- **Name:** Test pattern.
- **Motivation:** To improve design, save time and to have good quality.
- **Consequences:** less defects for user story in small iteration/sprint.
- **Related pattern:** pattern design for software testing using finite automata machine [17] [18] [19] [20].
- **Implementation:** Microsoft excel is the good application to manage and order different activities related to this pattern.

Prioritization of test cases is possible when test case generation has already taken place for the specified user story. For building test scripts, Selenium IDE (Integrated Development Environment), a prototyping
A tool can be used. This tool is based upon record and play functionality. After generating test cases, the next step is to analyse the test cases and call graph creation using first order dependence matrix (introduced by Myers) for the user story. This call graph is applicable for one user story at a time and consists of linkages of ovals for each task of the user story. Different tasks (Class) of the story are connected by coupling mechanism and one class is tightly connected with its data and methods by cohesion principle. Three algorithms are used here. First and foremost algorithm is A1 that includes four steps.

A1:

1. Create call graph for the given user story
2. Identify coupling and cohesion among the tasks of the call graph
3. Find the first order dependence matrix.
4. Referring test cases using Selenium IDE (this is original test suite before testing the user story after a specified change).

Consider for example, a user story is, “getting the patent to be granted by government of India to have hold on all the rights in the specified territory by any patentee” (S). In this user story, primary two parties are patent office and applicant. Secondary party is public and it is not always active as public role arises when public has some opposing point in the invention to be granted. Steps for the user story S in brief are:

- Filing the application with complete specification in the patent office by applicant for his invention. (c1)
- Invention is published online in the patent office’s official journal (weekly issue) after 18 months. (c2)
- Request for examination for the said invention within 48 months. (c3)
- First examination report by patent office to applicant. (c4)
- Office Action/Office Action Reporting is generated by patent office/applicant until satisfaction level is reached for putting an application in order for grant within 12 months. (c5)
- Pre grant opposition done by opponents/public to patent office. (c6)
- Response sent by applicant to patent office. (c7)
- Grant of patent for 20 years by patent office. (c8)

So, three classes namely patent office, patentee and public are interconnected with each other with respect to the single user story S with one or another way. At first sight, they seem to form a sequential pattern but actually not (Refer Figure 1). Thus, it is clear that classes are dependent on each other especially primary one. Secondary, one is active only in case of pre grant opposition after publication. Suppose there is a change (c9) in the user story which says that opposition from public is allowed after grant of patent also or after c8. Now, situation is little bit complex as public class is becoming active. If it would have been an undirected graph then situation would have been more tedious. But in this case, directed call graph, can easily give dependency classes.

Applying the algorithm A1, most intensive class as per step 3 of the algorithm, here are, applicant and patent office both, patent office directly and patentee indirectly by using first order dependent matrix values. In step 5, test cases are referred for the specified (most severe) classes.

Adding this new step (c9), will introduce some ambiguous linking between primary class methods. Here, test pattern is proposed for such kind of scenarios, where direct as well as indirect class dependency exist and need is to prioritize the test cases so as to gain quality.
After applying algorithm A1, next step is to find the similarity with the existing patterns available in the design pattern history. Analysis says that aforementioned scenario matches with observer pattern which is a part of behavioural pattern. Specifically, in this case, objects of classes are not directly dependent on each other rather one is directly and other is indirectly dependent. Steps for algorithm 2 (A2) are given below:

A2:

1. Find the directly dependent class and indirectly dependent class from the call graph.
2. Identify the objects of the changing task's with respect to the dependent classes (directly and indirectly).
3. Find the relationship of objects of all the related classes in specific methods.

After applying algorithm A2, relationship of classes for the specified user story is shown in Figure 2. In this figure, dotted lines represent indirect relationship and solid lines represent direct relationship between classes for the user story S. Also, for every class in the relationship diagram, object list and method list is there. Complexity of the change is dependent on number of classes present in the user story. But in agile culture, less is more concept is used, so, small user story would have small number of classes in one sprint and relationship would be strong.

Time of delivery of user story to customer is very frequent in nature. So, there is a need to find defects at early stage of development. Next step is to use the already generated test suite (TS) of S. If original generated TS contain 500 test cases (tc1-tc500) then after this little change in maintenance phase time to execution would be very less, so, it is not possible to execute complete TS. So, finally last but not the least a prioritization algorithm A3 is given below for the sake of saving time.

![Figure 1. Patent Grant Procedure](image-url)
A3:

1. Create the dependence diagram; it is a smart way of representation for classes linkages in a user story, (refer Table 1) and is to be displayed along with the story board (refer Table 2).
2. When a significant change is encountered in the user story, observe the dependence diagram and relationship diagram as discussed earlier to identify the intensive classes and objects (with relevant data members and methods) of the user story. Also, consider test suite for this user story with suitable data entries.
3. Sort the records of the test suite on the basis of dependent and indirectly-dependent objects by considering relationship diagram and dependence diagram (Checking non null entries in the test suite table).
4. Apply second sort now on the basis of maximum number of non-null entries for existing objects in the classes for one particular user story.
5. Frame new test suite table after doing prioritization.

IV. TEST PATTERN

This pattern is based upon the observer pattern under behavioral category. But, in comparison to previous one, this pattern does not intimate its dependents rather test cases are prioritized for the purpose of saving time in an agile context by considering dependence diagram of the story board which is updated from time to time. Main components of this pattern (shown in Figure 3) using A1, A2 and A3 are:

(1) Call graph and test case generator-Generator
(2) Dependent and indirectly dependent class identification-Finder
(3) Prioritization of test cases using dependence diagram-Prioritizer

<table>
<thead>
<tr>
<th>Table 1. Dependence Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEPENDENCE DIAGRAM FOR $1</strong></td>
</tr>
<tr>
<td><strong>CLASS #</strong></td>
</tr>
<tr>
<td><strong>DIRECTLY DEPENDENT</strong></td>
</tr>
<tr>
<td><strong>INDIRECTLY DEPENDENT</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2. Story Board</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>US #</strong></td>
</tr>
<tr>
<td>S1</td>
</tr>
<tr>
<td>Task2</td>
</tr>
</tbody>
</table>
V. RESULTS & ANALYSIS

The purpose of this analysis is to prove that quality component is optimal one as compared with non-prioritization scheme of regression testing. Generated test suite (TS) is given in Table 3 and after considering dependence and relationship diagram, new test suite is shown in Table 4 in which sorting is applied.

Suppose O1-O4 are objects of public class, Op1-Op3 are objects of patent office and Opp1-Opp2 are objects of applicant class from Figure 2. After applying A1-A3 net results for sorted test cases is in table 4. Non zero entries in Table 1 for test cases 1,2,3,4,5 are 7,6,5,5,4. Out of all, find the nonzero entries for direct class objects. Directly dependent class is patent office and indirectly dependent class is applicant.
Objects of directly dependent class are Op1-Op3. After doing sorting on this basis, series of test cases would be by excluding 5 number test case as all the entries are null for specified objects (Op1-Op3). The rest of four test cases are needed to be sorted.

Ascending order series for four test cases for non-zero entries is 1,2,2,3 for test cases number 4,1,2,3. Out of these results, maximum number is 3. So, priority would be high for 3 number test case. Out of tc 1, 2 which one will take priority depends upon non zero entries for indirectly dependent classes. Test case 2 has higher priority as compared with test case 1 as Op1 and Op2 has potential to affect Opp1 and Opp2. So final results is in Table 4. If there is no such difference then either of two test cases may be considered.

<table>
<thead>
<tr>
<th>TC #</th>
<th>O1</th>
<th>O2</th>
<th>O3</th>
<th>O4</th>
<th>Op1</th>
<th>Op2</th>
<th>Op3</th>
<th>Opp1</th>
<th>Opp2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>-</td>
<td>3</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>4</td>
<td>-</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

VI. CONCLUSION & FUTURE WORK

In this paper, a test pattern has been proposed which is mainly useful for saving time during regression testing phase in an agile environment. Also, relationship diagram and dependence diagram are discussed for the purpose of finding linkages between various objects of various classes. Main components of the pattern are generator, finder and prioritizer. The pattern, which has been proposed here, is only applicable for small stories having small number of classes. A lot of scope is there to implement this pattern in other kind of applications like web application, embedded etc..., in which there can be n number of classes. In addition, pattern can be implemented for the purpose of similar kinds of problems. This solution is not only the solution but also other solution pair can also exist for similar kind of problem.

Analysis for this test case prioritization method can be done with the help of velocity metric, which is used in agile development. To calculate velocity agile team’s product owner, can add up the estimates of the features, user stories, requirements or backlog items. Further, new metric can be created which can be useful for supporting above work. Also, generation of test cases may be done by using different open source agile tools.
VII. REFERENCES


[13] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Book titled Design Patterns: Elements of Reusable Object-Oriented Software, 1994


