INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 11 (6): 707-712, 2012
ISSN 1812-5638 / DOI: 10.3923/itj.2012.707.712
© 2012 Asian Network for Scientific Information

New Agile Testing Modes

[1]N. Ganesh and [2]S. Thangasamy
[1]Department of Computer Science and Engineering,
Anna University of Technology, Coimbatore, India
[2]Research and Development, Kumaraguru College of Technology, Coimbatore-641049, India

Abstract: Software is considered to have high quality, only if it has undergone a series of good test cases. Test-driven development is a software Development practice that leads to better quality and fewer defects in code. The purpose of study of this paper is to evaluate the difference between a traditional testing and agile testing, the testing techniques involved in agile projects, the different modes of testing, the do's and don'ts of testing and also about the set of activities an agile tester needs to keep in mind while working on agile projects. A study has been conducted in a software firm in Chennai, India. Two different testing modes have been proposed. Based on the testing modes that is projected in this article, the do's and don'ts of a software tester and their activities has been formulated. A simple yes/no type questionnaire is included in this article which is given to the testers before testing. Answering the simple yes/no questions makes a tester to be an efficient tester in his job. By following the do's and don'ts guidelines and the activities of a tester given in this article, it is found that the efficiency of an employee involved in agile testing has been improved considerably. It is concluded that testing plays a vital role for any project. Testing would be more efficient if a parallel testing could be done in par with the development team. Better quality could be achieved if the project team follows a continuous integration with a continuous delivery and a continuous feedback from the customer.

Key words: Test driven development, testing techniques, testing modes, test strategy, quality assurance, extreme programming

INTRODUCTION

Test-driven development (TDD) was one of the key practices that was followed in extreme programming (XP) (Beck, 2003) and was also used in the stand-alone process (Astels, 2003). The idea behind TDD is to write test cases before coding in small iterations. In agile projects, first the developer writes a test case to define the next functionality. Second, he writes the code, wherein the primary motive of the code written is to pass the test. Third, the code is re factored, if required (Reily and Gocher, 2009). These steps are iterated in short cycles through the whole development process. More the number of test case generated and the bugs rectified, better the product achieved where there is a dynamic change in the business environment, wherein the service oriented architecture is used by organizations for becoming more agile (Ihsan and Dogerlioglu, 2011).

Originally, TDD was introduced as a developmental practice and not as a testing method (Astels, 2003; Beck, 2003). The TDD yields better quality with fewer defects in code. Assessment of software testability can help in predicting the testing effort required for a given product (Singh and Saha, 2010). TDD is a simple practice, but developers sometimes do not apply all the required steps correctly. In agile software development, the TDD encourages communication between customers and developers and raises software quality thereby decreases bug density considerably. The global software development processes have reframed the traditional software approaches resulting in the changes on the preferences and the priorities between the developer and the client (Akbar *et al.*, 2011).

The purpose of the literature study was to evaluate the current available theories on TDD. The review process was designed using the well-known guidelines for systematic literature review (Kitchenham and Charters, 2007). TDD has been suggested to provide a variety of different benefits, such as better productivity (Erdogmus *et al.*, 2005; Janzen and Saiedian, 2006), better quality and high test coverage (Edwards, 2004; Nagappan *et al.*, 2008). Some studies also suggest that TDD may improve program design (Kaufmann and Janzen, 2003).

Always, small test suites that retain high fault detection are desirable as given by Roongruangsuwan

Corresponding Author: N. Ganesh, Department of Computer Science and Engineering, Anna University of Technology, Coimbatore, India

and Daengdej (2010). It has been proved from the available literature that software testing phase takes around 40-70% of the effort, time and cost (Kosindrdecha and Daengdej, 2010).

TDD applies at two levels in XP (Mugridge, 2003). The first level adapts the evolving design within an organizational context, with iterations and the planning game:

- The customer writes stories that will lead to the system development
- The developers estimate the cost for developing the user stories
- The estimates will help the customer to prioritize the stories that will fit into the next iteration
- Customer tests are developed for the stories that were chosen for iteration. They are used to determine whether a story is complete

As the system evolves, the value of different possible aspects of the system will become clearer as it begins to be fitted into its developmental culture.

XP is well suited when the scope and value of a computer system are not well understood, where the design of the system has to evolve along with the requirements. As a concrete system evolves, the customer determines the value and makes his decision. The source code based testing techniques can be used to identify the test data, test sequence and the dependencies available for each test case as given by Kosindrdecha and Daengdej (2010).

The second level of TDD is as described by Beck (2003), which is at the level of micro-iterations in completing a task:

- An area of design or task requirement has to be chosen to test drive the development
- Design a concrete test that is as simple as possible while driving the development as required and check that the test fails
- Alter the system to satisfy the test and all other tests

Principles of test driven development:

- Test as early as possible: The potential impact of a defect rises exponentially over time and in such a case of this sort a test first approach like TDD is highly recommended
- Test as often as possible: The chance of impinging on a defect is higher when the frequency of testing is increased
- Test just as much is needed for the situation: Intensity of testing should be based on the business criticality of the software being developed
- Pair testing: Pair testing involves testing the software as a duo

The objective of testing is to understand the risk of putting software in to production as given by Mustafa *et al.* (2007). By and large, testing is done in both the models, but the methodology adopted and the naming convention differs. Some of the traditional model includes the waterfall model, spiral model, the rapid application development model, the prototyping model and so on (Zhang *et al.*, 2010). Some of the agile models are XP, scrum, feature driven development, crystal and so on. Identifying the bugs and generating the test cases is one of the most time consuming and costly step in software testing phase (Keyvanpour, 2011).

Problems: The problems here in this study illustrate the generic common queries that the agile team has when it is working on a project.

They are listed as follows:

- How agile testing is to be conducted?
- Should the Quality Assurance (QA) team be part of the development team?
- Can the development team and the testing team be fit in the same sprint?
- Who does the QA work?
- What techniques should the team employ?
- Should the development team and the test teams work closely?

The answers for these queries are explained later in our discussion section, by using a testing model. Table 1 shows Difference between Traditional testing and agile testing.

Table 1: Variance in traditional and agile testing

	Traditional testing	Agile testing
Change	Should separately be managed and controlled	Accept when it comes
Planning	Detailed upfront test design has to be given	Can be planned when the team moves to the next functionality
Documentation	Good documentation has to be maintained	Only as much as needed for the work
Handoffs	Should have a formal entrance and exit criteria with client sign-offs whenever needed	Can be collaborated with the existing development team
Automation	It is done after the full code is been created	All levels, built by anyone from the team, form an integral part of the project

RESEARCH APPROACH

The research is based on an interpretive case study conducted in select Indian organizations engaged in Information Technology (IT) outsourcing to global clients. The case study research strategy is useful for investigating phenomena that are under-researched, complex or difficult to extract from their underlying contexts. We have adopted an interpretive approach since it is through multiple, inter-subjective views of actors working within the IT cultural enclave environments where concept, theories and rich insights about the phenomena can emerge. This context-dependent knowledge can prove useful in gaining expertise of understanding a practical Indian setting, an outcome relevant to the research objectives. There are three types of studies using the case study method, namely; (1) Intrinsic case study: Researcher wants a better understanding of the particular case, (2) Instrumental case study: A particular instance is examined to provide insight into an issue or refinement of theory and (3) Collective case study: Researchers may jointly study a number of case studies in order to inquire into the phenomenon, population, or general condition. Since, it is a phenomenon of the usage of the combination of methodologies such as XP and the scrum, the agile testing plays a vital role in such software developmental projects. Hence, we have adopted the instrumental case study approach to provide better insights into the issue.

DISCUSSION

Case description-the background: The answers for the problems listed earlier are explained with the help of a case study that is conducted in an Indian software company.

The case study is framed based on the author's involvement in an agile software project that has been developed in a software company located in Chennai, India. The software company which the author is referring to is one of the pioneers in handling projects on image and video optimization. As an extension of this work, the software team has developed logic and is developing the beta version of the product to compress the image and retrieve it back without any loss on the original image. They are also leaders in creating solutions for correcting the online competitive examinations. The company has its offices in various locations across India and has its global operations in countries like USA, Japan and Philippines. As per the policies of the organization, the company name and the project names are to be kept anonymous.

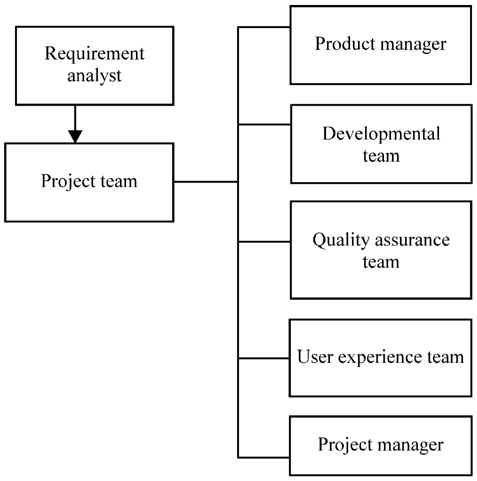

Fig. 1: The project team of the firm where the study has been made

Table 2: Type of testing and execution status of the concerned

Type of testing	Responsible	When to execute
Unit testing	Developer	Coding
Functional testing	Tester	Throughout the sprint
System testing	Tester	Throughout the sprint
Integration testing	Developer/Tester	Throughout the sprint
Regression testing	Tester	Throughout the sprint
Performance testing	Tester	Stabilization sprints/last sprint in the release cycle

The project team: The below illustration shows the agile cross-functional development team which includes a product manager, developmental team, QA team, user experience team and a project manager. The project team of the firm where the study has been made is presented in Fig. 1.

The project manager supervises the release schedule and helps resolve logistical issues, but otherwise does not participate in the development process. The developmental team, consisting of three to eight developers, implements and delivers to QA a feature or set of features per sprint based on a prioritized backlog determined by product management.

Test strategy: During the test strategy design, decision on the scope of testing such as unit, integration and system testing and types of testing such as performance, load, regression, etc is taken. The acceptance criteria should generally define upfront for each story and should be available in the beginning of each sprint. Based on the skill set available and time, decision on level of automation for the project is also decided. The decision of which testing is done in the sprint and which one is done outside the sprint is also taken. Table 2 shows the types of testing and the responsibilities of the agile team in executing the tests.

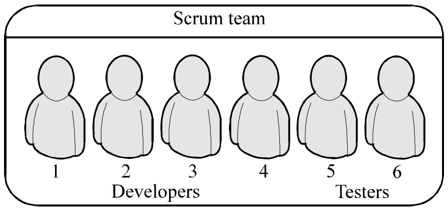

Fig. 2: Scrum team in the linear embedded testing model

Testing techniques: The acceptance testing, unit testing and the TDD are classified under the category of confirmatory testing, which means these are the mandatory testing that has to be performed under a single sprint. The exploratory-based testing, session based testing and the scenario based testing is classified under the broad category of investigative testing, which means certain mining aspects have to be performed while testing. It also denotes testing out of the box.

Apart from testing, the project may be highly successful only if it makes continuous integration with continuous delivery and also gets continuous feedback from the customer end.

Testing modes: The testing modes are broadly classified in to two different groups. They are the linear embedded testing model and the other is the rotational embedded testing model.

Linear embedded testing model: This model is a common model that is been used in many of the software projects, wherein the developer will be involved in coding and the testers will be embedded along with the development team and they work on generating several test cases and executing them. But in this scenario, people that are the developer whoever is involved in coding will always work as the coder till the end of the project and the persons whoever is involved as a tester will act as a tester till the project gets completed. Figure 2 illustrates the scenario.

In such a scenario, there are several chances for both the developers and testers to get fed up with their roles of coding and testing, respectively.

Rotational embedded testing model: In agile projects that deploys a combination of both XP and scrum methodology, the developers whoever is involved in coding can be converted to play the testing role after one particular sprint is completed. Normally, a sprint lasts long for a period between 1 week to 15 days. In this case, both the tester and the developer will be happy in playing multiple roles. As well, the developer as he would have played the role of a tester in the same project, he would be doubly careful in coding the task that is assigned to him. Figure 3 illustrate the scenario.

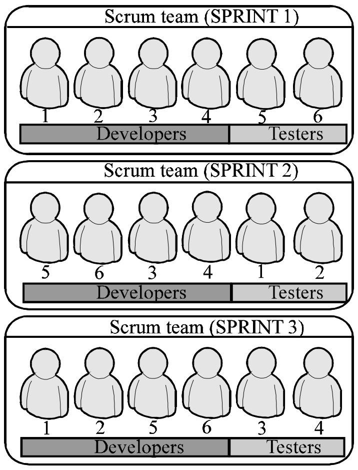

Fig. 3: Role played by scrum team members using rotational embedded testing model

The do's and don'ts in agile testing: The below guidelines will be highly helpful for an agile developer cum tester in a real-time agile developmental project.

Do's in agile testing:

- Encourage test automation
- Domain knowledge is crucial for the QA members
- Certain degree of technical competency essential for the QA members
- Testing as early as practical and continuously
- Develop acceptance test cases before the software is developed
- Developers have the responsibility to create automated unit tests
- Make testing and feedback a continuous activity in a sprint rather than a phase towards the end.
- Testers must be in continuous conversation
- Continuous build and integration practices will aid in testing

Don'ts in agile testing:

- Expect a signed off/base lined requirements
- Create an elaborate test plan
- Development team and the QA team have separate daily stand ups and separate planning and estimation process
- Follow manual testing entirely

Table 3: Self evaluation questions

Q. No.	Answer each question as Yes/No	Y/N
1	Are the testers considered as part of the sprint team?	
2	Is the tester satisfied in writing well-documented incident reports?	
3	Can the output of each sprint immediately usable by the users?	
4	Are you highly productive when working alone?	
5	Do you strongly believe that its unfair to question the decisions made by business analysts	
6	Do you think regression testing is a waste of time?	
7	Do you find inconvenient and uncomfortable in short deadlines	
8	Are the developers willing to use test driven development	
9	Is continuous integration and automated regression testing implemented?	
10	Will you consider test automation to be your own problem?	
11	Can you work on code in par with a developer?	
12	Will the lack of detailed specifications make you uneasy?	
13	Do you stick with your plans on any situation?	
14	Will you take responsibility with rest of the sprint team for the deliverables?	
15	Are you happy by following the procedures?	

- Disconnect from the customers and development team
- Participate only towards end of sprint
- Ignore system testing since unit tests exist

Typical activities of an agile tester:

- Testers should raise queries on the testing needs for the requirements and should aptly identify the missing door
- Understand the acceptance criteria of a story and perform testing based on the acceptance criteria
- Try some negative testing within the limit of the story
- In case of failing scenarios, raise a defect on a card
- Notify the status and defects that have been identified so far during DS
- In case of passing scenarios automate the story covering all the acceptance criteria
- Automate whatever defect is fixed if it is not covered in any story
- Perform exploratory testing to look into some missing functionality that has not been covered in the story (may need interaction with business analyst)

Tester's agile adoption questionnaire: The answers to the below mentioned sample questionnaire make the traditional tester to get converted to an agile tester. The more the number of Yes he/she gives in answering, the more they can adapt themselves to agile testing. Self evaluation Questions are presented in Table 3.

Debugging: The below flowchart in Fig. 4 explains how to debug a test case and effectively eradicate the bug from a session.

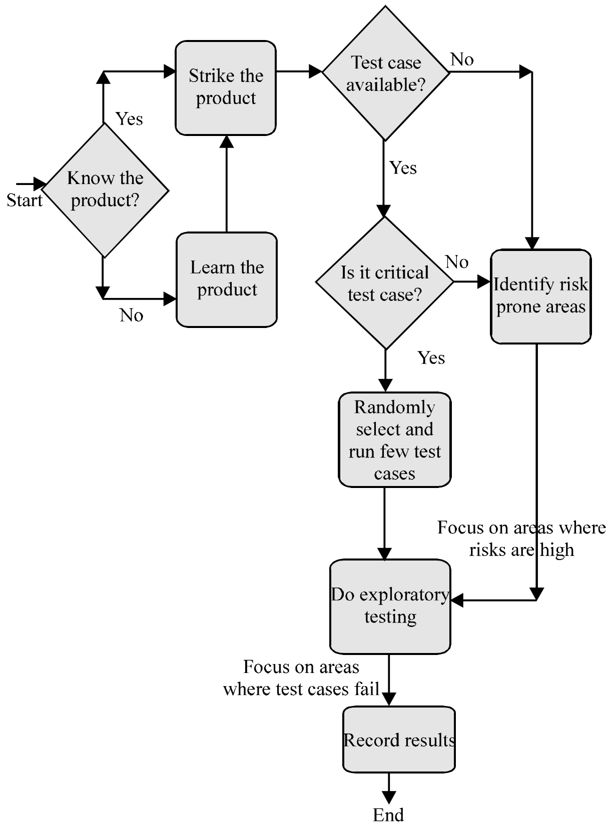
Start
Know the product?
Yes
No
Strike the product
Learn the product
Test case available?
No
Yes
Is it critical test case?
No
Yes
Identify risk prone areas
Randomly select and run few test cases
Focus on areas where risks are high
Do exploratory testing
Focus on areas where test cases fail
Record results
End

Fig. 4: Representation of a test case debugger

CONCLUSION

Testing plays a vital role for any project. Testing would be more efficient if a parallel testing could be done in par with the development team. Better quality could be achieved if the project team follows a continuous integration with a continuous delivery and a continuous feedback from the customer.

As a future direction, when the bug has arrived, before directly implementing the test cases, the criticality of the bug has to be inspected at first and then adaptation has to be implemented based on the criticality and priority of the bug. It is the QA team's responsibility to work in par with the development team and bring out a better qualitative product.

REFERENCES

Akbar, R., M.F. Hassan, A. Abdullah, S. Safdar and M.A. Qureshi, 2011. Directions and advancements in global software development: a summarized review of GSD and agile methods. Res. J. Inform. Technol., 3: 69-80.

Astels, D., 2003. Test Driven Development: A Practical Guide. 2nd Edn., Prentice Hall, Upper Saddle River, New Jersey, ISBN-13: 978-0131016491, Pages: 592.

Beck, K., 2003. Test-Driven Development: By Example. Addison-Wesley, USA.

Edwards, S.H., 2004. Using software testing to move students from trial-anderror to reflection-in-action. Proceedings of the 35th SIGCSE technical symposium on Computer Science Education, (SIGCSE'04), ACM, New York, USA., pp: 26-30.

Erdogmus, H., M. Morisio and M. Torchiano, 2005. On the effectiveness of the test-first approach to programming. IEEE Trans. Software Eng., 31: 226-237.

Ihsan, B. and O. Dogerlioglu, 2011. Impacts of service-oriented architecture transformation on organizational structures. J. Applied Sci., 15: 2791-2799.

Janzen, D.S. and H. Saiedian, 2006. On the influence of test-driven development on software design. Proceedings of the 19th Conference on Software Engineering Education and Training, (CSEET'06), IEEE Computer Society, Washington, DC., USA., pp: 141-148.

Kaufmann, R. and D. Janzen, 2003. Implications of test-driven development: A pilot study. Proceedings of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages And Applications, (OOPSLA'03), ACM, New York, USA., pp: 298-299.

Keyvanpour, M.R., H. Homayouni and H. Shirazee, 2011. Automatic software test case generation. J. Software Eng., 5: 91-101.

Kitchenham, B. and S. Charters, 2007. Guidelines for performing systematic literature reviews in software engineering. Keele University and Durham University Joint Report, Technical Report EBSE 2007-001, 2007. http://www.dur.ac.uk/ebse/bibref.php?id=51

Kosindrdecha, N. and J. Daengdej, 2010. A test case generation process and technique. J. Software Eng., 4: 265-287.

Mugridge, R., 2003. Test driven development and the scientific method. Proceedings of the Agile Development Conference, June 28, 2003, Salt Lake City, UT, USA., pp: 47-52.

Mustafa, G., A.A. Shah, K.H. Asif and A. Ali, 2007. A strategy for testing of web based software. Inform. Technol. J., 6: 74-81.

Nagappan, N., E.M. Maximilien, T. Bhat and L. Williams, 2008. Realizing quality improvement through test driven development: results and experiences of four industrial teams. Empirical Software Eng., 13: 289-302.

Reily, T. and A. Gocher, 2009. Beautiful Testing: Leading Professionals Reveal How They Improve Software. 1st Edn., O'Reilly Media Inc., USA., ISBN-13: 978-0596159818, Pages: 352.

Roongruangsuwan, S. and J. Daengdej, 2010. A test case prioritization method with practical weight factors. J. Software Eng., 4: 193-214.

Singh, Y. and A. Saha, 2010. Predicting testability of eclipse: A case study. J. Software Eng., 4: 122-136.

Zhang, X., T. Hu, H. Dai and X. Li, 2010. Software development methodologies, trends and implications: A testing centric view. Inform. Technol. J., 9: 1747-1753.