The Role of Testing in Agile and Conventional Methodologies

Agustin Yagüe and Juan Garbajosa

Universidad Politecnica de Madrid (UPM), System and Software Technology Group (SYST)
E.U. Informatica. Cra. Valencia Km. 7, Madrid 28031, Spain

Abstract. Testing in software engineering is a well established practice. Though the scope of testing may differ depending on the community, e.g. for some communities is a process in itself while for some other communities is a an activity or a task under verification and validation, many fundamental issues around testing are shared by all the communities. However, agile methodologies are emerging in the software engineering landscape and are changing the picture. For instance, in agile methodologies it may happen that code is written precisely to pass a test. Moreover, tests may replace the requirement specification. Therefore the concepts underlying test practice are different in conventional and agile approaches. This paper analyses these two different perspectives for testing, the conventional and the agile, and discusses some of the implications that these two different approaches may have in software engineering.

1 Introduction

According to SWEBOK [1] testing is an activity performed for evaluating product quality, and for improving it, by identifying defects and problems. Classical methodologies considered testing as an activity which started only after the coding phase was complete, with the limited purpose of detecting failures. This understanding has been changing with time. Nowadays testing is seen as an activity which should be part of the whole process. But as we shall describe below very different perceptions coexist under the term testing. The term *testing* has been used for years to refer different concepts such as: testing techniques —like black-box and white-box—, testing strategies —unit, integration, acceptance or system testing—, testing practices —TDD (Test Driven Development), ATDD (Acceptance Test Driven Development), STDD (Story Test Driven Development)— or testing methodologies —TMAP [2]—.

International standardization organizations have documented testing practices from different points of view in multiples standards, some of which are [3–7]. Testing is being used by all software and system communities. They share some basic techniques and approaches but very often these approaches are applied in different phases of the development process, with a different scope, and by different actors of the process.

The relevance of the concept of testing is increasing in Conventional and in the so called Agile methodologies. Agile methodologies emerged as a reaction to some challenges that the software industry was facing such as unavoidable market changes and

Yagüe A. and Garbajosa J. (2009).
The Role of Testing in Agile and Conventional Methodologies.
In *Proceedings of the 7th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*, pages 93-100
DOI: 10.5220/0002205100930100

a progressively shorter time to market [8]. Agile methodologies intend to increase the product quality and to reduce the cost caused by changes in requirements by simplifying the requirements management and documentation tasks. To achieve this Agile methodologies are characterized by a number of values, principles and practices, promote a fast and continuous communication between customers and the development team and promotes testing as a leitmotiv [9, 10]. Agile in general, and one of the schools in particular, Extreme programming (XP) [11], are in some way responsible for the increasing popularity of testing. They write user needs, sometimes as ”user stories”, with associated acceptance criteria as a way to represent system requirements. Therefore acceptance tests are written to validate user needs, at an early stage, and to drive the implementation; even these tests may play a role of specifications. Code is implemented to success the acceptance tests. However in conventional methodologies acceptance tests are considered during the last stage of the development. Hence, testing may have radically different goals, such as requirement validation in conventional approaches or even, replacing requirement specification in some agile methodologies. It can be asserted that testing is evolving to achieve new roles in software/system development. This evolution of testing is associated with new tools supporting some of the topics mentioned above.

An issue is that some concern is being raised with the respect to skills and background of testing practitioners. Sometimes testing tools and practices are being used with different objectives with that were defined . As consequence project resources, in terms of time and money, are wasted. This paper analyzes testing both from the Conventional and Agile perspectives. It shows how similar techniques can be used with a very different purpose. Commonalities and differences are presented and discussed.

The remainder of the paper is structured as follows: section 2 presents the use of testing techniques in Conventional and Agile methodologies. Section 3 shows the different scope of testing activities in both methodologies. Later, section 4 introduces testing from a research perspective. Finally, section 5 presents a number of conclusions.

2 Basic Testing Techniques

In process models, Testing is used as a term with two different meanings: the practice of defining and executing test, that is a task of the development process, and at the same time, as a synonym of the verification and validation process. This is more evident if we consider different domains such as telecom, space, or automotive. A common aspect in Conventional and Agile methodologies is that the software must be tested using some techniques. This section will describe the role of testing techniques considering two main categories: white-box and black-box testing techniques.

White-box testing (basis path, control flow, data flow or Branch Testing among others) is code-centric and it uses an internal perspective of software based on the source code. A key issue concerning this technique is that, as it is based on the actual implementation of the software, if some changes are done in the implementation, in the most of the cases, tests will need to be changed. That is the reason why it requires programming skills to identify the test cases. Even though white-box testing could be applied to different testing strategies (unit, integration and system testing), it is typically applied to unit testing.

The actors that use white-box testing are different in Conventional and Agile methodologies. In Agile methodologies developers are in charge of executing white-box test using unit testing as a part of the validation process. As developers are working with the source code, they know: the software structure to define the appropriate test cases and, when a test case fail which lines of code are are involved in the revealed a problem. In Conventional methodologies, test engineers define test cases and testers are the responsible to execute this type of tests.

As oppose to white-box, black-box testing (equivalence partitioning, boundary value analysis or cross-functional testing, among others) uses an external perspective of the software without to be focused on source code. Test cases are focused on the different inputs of the software and their corresponding output values. Therefore, the main goal is to test software/system requirements through their functionality. This technique could be applied at different levels of testing – unit and acceptance testing – with different levels of abstraction and complexity for test cases. When black-box is applied in Conventional methodologies is done by software testers, but when it is applied in Agile methodologies, two main perspectives are considered: developers, who know the right type of each parameter, running black-box tests as unit tests; and testers, that were not involved in the software development, running acceptance tests.

3 Testing Activities

This section will describe the role of testing as a process in the scope of a global development process. It will analyze the relevance of testing in Agile and Conventional methodologies to study how different roles are played. Four different testing activities are identified: unit testing, integration testing, acceptance testing and system testing [1]. The way of applying these testing activities is radically different in Conventional and Agile methodologies.

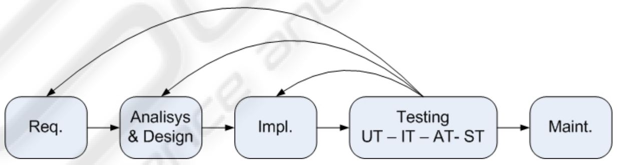

Fig. 1. Conventional testing approach.

Conventional methodologies pay special attention to the elaboration of software requirement specification. The refinement of a requirements document is done in several iterations within the requirement elicitation process. This approach addresses the development process from requirements to code and from code to tests. Although testing is important in Conventional methodologies, it is directly dependent of the rest of activities of the development process. The validation process is understood as a complement to the development process. Figure 1 depicts an abstraction of this model. In this case, all of the activities are applied as a sequence, what means that integration testing could not be done before to do unit testing, what it is extensible to the rest of the activities. Therefore, source code is the one that determines the testing process. An example of this can be found in [12]: the testing process could not start before the previous process, implementation, is finished. Meanwhile, in the V-model [13], and also in [7], the definition of test cases is integrated from the first stages of the process model: acceptance and system test are derived from the requirement specification document, integration tests are derived from design phase and unit test cases are derived from the module implementation. In this approach, test cases are defined by the testing team that is not involved in the implementation.

In Agile approaches, testing is the core of the methodology. Tests drive the development process. To compare with Conventional approaches, two strategies are considered: Test Driven Development (TDD)[14, 15] and Acceptance Test Driven Development(ATDD)[16]. Both strategies share a common point of view of the development process but with different starting points.

On the one side, TDD is based on unit testing. The first step is the definition of a requirements repository, usually called *product backlog*. After that, developers write unit tests to satisfy the software requirements and then they start to implement the source code that is needed to pass the defined unit tests. In the case of TDD, each requirement is represented with an artifact, sometimes called *User Story*, with their own unit tests. Therefore, the complete list of user stories and their corresponding unit tests have the same role of the software requirement specification (SRS) in conventional methodologies.

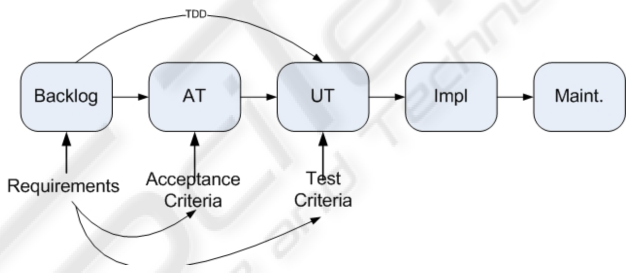

Fig. 2. Agile testing approach.

On the other side, ATDD is based on acceptance tests. The process is quite similar to TDD but the process is boosted by acceptance tests. In this case, each user story has associated a set of acceptance criteria. Then, for each acceptance criteria, a set of acceptance tests to be successful are defined. Once acceptance tests are clear, developers start to implement the source code to pass each acceptance test. As in TDD happens, the complete list of user stories (product backlog) and their corresponding acceptance tests have a similar role of SRS in conventional methodologies.

Figure 2 shows the main testing strategy in Agile methodologies. Agile methodologies use unit and acceptance testing as the main testing activities. Integration and system testing are implicitly applied in the development process, that is each new piece of code that is written is integrated with complete system, therefore, it is not necessary the definition of specific integration test cases. The language used to write the test cases depends on the implementation language and the tools that are available to execute the test cases, e.g. Java and jUnit. In the case of acceptance tests, specific languages and tools to describe test cases are needed. FIT or Easyaccept[17, 18]are tool samples.

4 A Discussion on Some Open Testing Topics

SWEBOK [1] provides a broad discussion on testing, not including Agile approaches as such. There is an existing breach between Agile and Conventional methodologies, as reported in [19] it is mainly focused in the ability of Agile to manage large projects, among other reasons. A key aspect of *Agile* is face to communication, this issue in relation with requirements has been studied in [20]. Assessments of agile in relation with other process models can be found in literature[21–23].

In Agile approaches the definition of tasks related to requirements is very often kept informal but testing tasks are very detailed. Nevertheless there are not studies that compare the results of an agile and conventional processes referred to the same product. It is clear that it would be expensive to have two teams developing the same product, as well as some doubts on the value of the conclusions. In [24, 25] the differences between requirements specification in conventional and Agile approaches are analyzed. Conventional methodologies are focused on the *anticipation abilities* and can be termed as *plan based* [26, 27] because these process models are defined in such a way that the later an error is discovered, the more expensive will be to correct it. With the objective of reducing the high impact of errors in the cost of a project, Agile methodologies propose to increase the relevance of testing. In this sense, Agile methodologies perceive each change like a chance to improve the system and increase the customer satisfaction. Agile teams do not try to avoid changes but try to understand what is behind them, to deliver the highest value to the customer.

Some studies show that product development in Agile environments is very different to that in conventional environments[28–30]. There are several experience reports, such as [31–33], that describe success stories of using Agile approaches. However, they do not usually provide enough details about the experiment or do not give quantitative results. Others studies give recommendations and general rules for the use of Agile methodologies [34, 35]. Nevertheless from these studies it is possible to understand that testing is a fundamental practice in agile software development and XP took it to the extreme by iteratively developing test in tandem with writing code.

Some studies relate requirements and testing. Ricca [36] shows the results of a series of experiments about using *acceptance test* as a support for clarifying requirements and conclude that this technique lead to a significantly better understanding of requirements. Other works in this line are [37–39].

Another key issue is test automation, while Conventional approaches pay attention to generation of test inputs from different sources, guided by source code coverage criteria rather than requirements coverage; Agile approaches are more interested in tools to automate the testing process considering that if the testing environment is as much automated as possible, more and more often, test cases could be run and more automated tools to support testing could be integrated. Continuous integration tools and framework such as Hudson, Cruise and Cruise Control or Apache Continuum represent a proof of it [40].

Finally, specific testing methodologies such us TMAP [2] describe testing as a independent project, that should be done in parallel with the development project. TMAP it is not aligned with another methodologies. They stress the importance of testing management.

5 Conclusions

Testing plays an important role in different process models, ranging from Conventional to Agile. Although testing techniques (white-box and black-box) are applied in both, actors are rather different and the role of testing is different, and while practices may look similar, i.e. how to produce a unit test, may be common to conventional and agile, they are not at a semantical level. While in conventional methodologies a test is performed to check that code is according to a specification, in agile the test is part of this specification, or the specification itself. Test design activities are some how different in Conventional and Agile. Last trends are giving more relevance to unit and acceptance testing. Another fact is that testing is being done as soon as possible in the development process what it is aligned with the reduction of the cost impact of errors detected in the testing phases. The changing role of testing has also strong implications on the overall process definition that standards will need to adequately consider.

Acknowledgements

Thi work has been partially sponsored by the Spanish MEC project OVAL/PM and MICYT within the FLEXI FIT-340005-2007-37 (ITEA2 6022) project.

References

1. Abran, A., Moore, J.W., Bourque, P., Dupuis, R., Tripp, L.L.: Guide to the Software Engineering Body of Knowledge (SWEBOK). IEEE (2004)
2. Koomen, T., van der Aalst, L., Broekman, B., Vroon, M.: TMap Next, for result-driven testing. UTN Publishers (2006)
3. ISO: ISO/IEC 12207. Software Engineering - Life Cycle Processes. ISO (2008)
4. ISO: Systems Engineering - System Life Cycle Processes. ISO (2008)
5. IEEE: IEEE Std 1008-. Standard for Software Unit Testing. IEEE, pub-IEEE-STD:adr (2002)
6. BSI: Software testing. Software component testing. BSI (1998)
7. European Consortium for Space Standardisation, E.: Software - part 1: Principles and requirements. E-40 (2003)
8. Boehm, B.: A view of 20th and 21st century software engineering. In: ICSE '06: Proceedings of the 28th international conference on Software engineering, New York, NY, USA, ACM (2006) 12–29

9. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: Manifesto for agile software development (2001)
10. Shore, J., Warden, S.: The Art of Agile Development. O'Reilly Media, Inc. (2007)
11. Beck, K.: Extreme Programming Explained: Embrace Change, Second Edition. Addison Wesley Professional (2004)
12. Royce, W.W.: Managing the development of large software systems: concepts and techniques. In: ICSE '87: Proceedings of the 9th international conference on Software Engineering, Los Alamitos, CA, USA, IEEE Computer Society Press (1987) 328–338
13. the: The V-Model. http://www.v-modell-xt.de (2005)
14. Janzen, D., Saiedian, H.: Test-driven development: Concepts, taxonomy, and future direction. Computer, 38 (2005) 43–50
15. Fraser, S., Astels, D., Beck, K., Boehm, B., McGregor, J., Newkirk, J., Poole, C.: Discipline and practices of tdd: (test driven development). In: OOPSLA '03: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, New York, NY, USA, ACM Press (2003) 268–270
16. Koskela, L.: Test Driven: TDD and Acceptance TDD for Java Developers. Manning Publications (2007)
17. Sauve;, J.P., Neto, O.L.A., Cirne, W.: Easyaccept: a tool to easily create, run and drive development with automated acceptance tests. In: AST '06: Proceedings of the 2006 international workshop on Automation of software test, New York, NY, USA, ACM Press (2006) 111–117
18. Cunningham & Cunningham, I.: Fit: Framework for integrated test. (web)
19. Vilki, K.: Juggling with the paradoxes of agile transformation. Flexi Newsletter, 2 (2008) 3–5
20. Cao, L., Ramesh, B.: Agile requirements engineering practices: An empirical study. Software, IEEE, 25 (2008) 60–67
21. Boehm, B.W., Turner, R.: Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods. In: ICSE, IEEE Computer Society (2004)
22. Boehm, B.W., Turner, R.: Management challenges to implementing agile processes in traditional development organizations. IEEE Software, 22 (2005) 30–39
23. Larman, C., Basili, V.R.: Iterative and incremental development: A brief history. Computer, 36 (2003) 47–56
24. Paetsch, F., Eberlein, A., Maurer, F.: Requirements engineering and agile software development. In: WETICE '03: Proceedings of the Twelfth International Workshop on Enabling Technologies, Washington, DC, USA, IEEE Computer Society (2003)
25. Sillitti, A., Ceschi, M., Russo, B., Succi, G.: Managing uncertainty in requirements: A survey in documentation-driven and agile companies. In: METRICS '05: Proceedings of the 11th IEEE International Software Metrics Symposium, Washington, DC, USA, IEEE Computer Society (2005) 17
26. Miler, R.: Managing Software or Growth without fear, control, and the manufacturing mindset. Addison-Wesley Professional (2003)
27. Boehm, B.W.: Software Engineering Economics (Prentice-Hall Advances in Computing Science & Technology Series). Prentice Hall PTR (1981)
28. Dyba, T., Dingsoyr, T.: Empirical studies of agile software development: A systematic review. Information and Software Technology, 50 (2008) 833–859
29. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In: ICSE '00: Proceedings of the Conference on The Future of Software Engineering, New York, NY, USA, ACM Press (2000) 35–46
30. Ceschi, M., Sillitti, A., Succi, G., De Panfilis, S.: Project management in plan-based and agile companies. Software, IEEE, 22 (2005) 21–27

31. Schwaber, K.: Agile Project Management With Scrum. Microsoft Press, Redmond, WA, USA (2004)
32. Mann, C., Maurer, F.: A case study on the impact of scrum on overtime and customer satisfaction. In: ADC '05: Proceedings of the Agile Development Conference, Washington, DC, USA, IEEE Computer Society (2005) 70–79
33. Capiluppi, A., Fernandez-Ramil, J., Higman, J., Sharp, H.C., Smith, N.: An empirical study of the evolution of an agile-developed software system. In: ICSE '07: Proceedings of the 29th international conference on Software Engineering, Washington, DC, USA, IEEE Computer Society (2007) 511–518
34. Nerur, S., Mahapatra, R., Mangalaraj, G.: Challenges of migrating to agile methodologies. Commun. ACM, 48 (2005) 72–78
35. Baker, S.W., Thomas, J.C.: Agile principles as a leadership value system: How agile memes survive and thrive in a corporate it culture. In: AGILE '07: Proceedings of the AGILE 2007, Washington, DC, USA, IEEE Computer Society (2007) 415–420
36. Ricca, F., Torchiano, M., Penta, M.D., Ceccato, M., Tonella, P.: Using acceptance tests as a support for clarifying requirements: A series of experiments. Information and Software Technology, 51 (2009) 270 – 283
37. Park, S.S., Maurer, F.: The benefits and challenges of executable acceptance testing. In: APOS '08: Proceedings of the 2008 international workshop on Scrutinizing agile practices or shoot-out at the agile corral, New York, NY, USA, ACM (2008)
38. Feather, M.S., Smith, B.: Automatic generation of test oracles from pilot studies to application. Automated Software Engg., 8 (2001) 31–61
39. Ho, C.W., Johnson, M., Williams, L., Maximilien, E.: On agile performance requirements specification and testing. Agile Conference, 2006 (2006) 6 pp.–52
40. Kim, S., Park, S., Yun, J., Lee, Y.: Automated continuous integration of component-based software: An industrial experience. (2008) 423–426