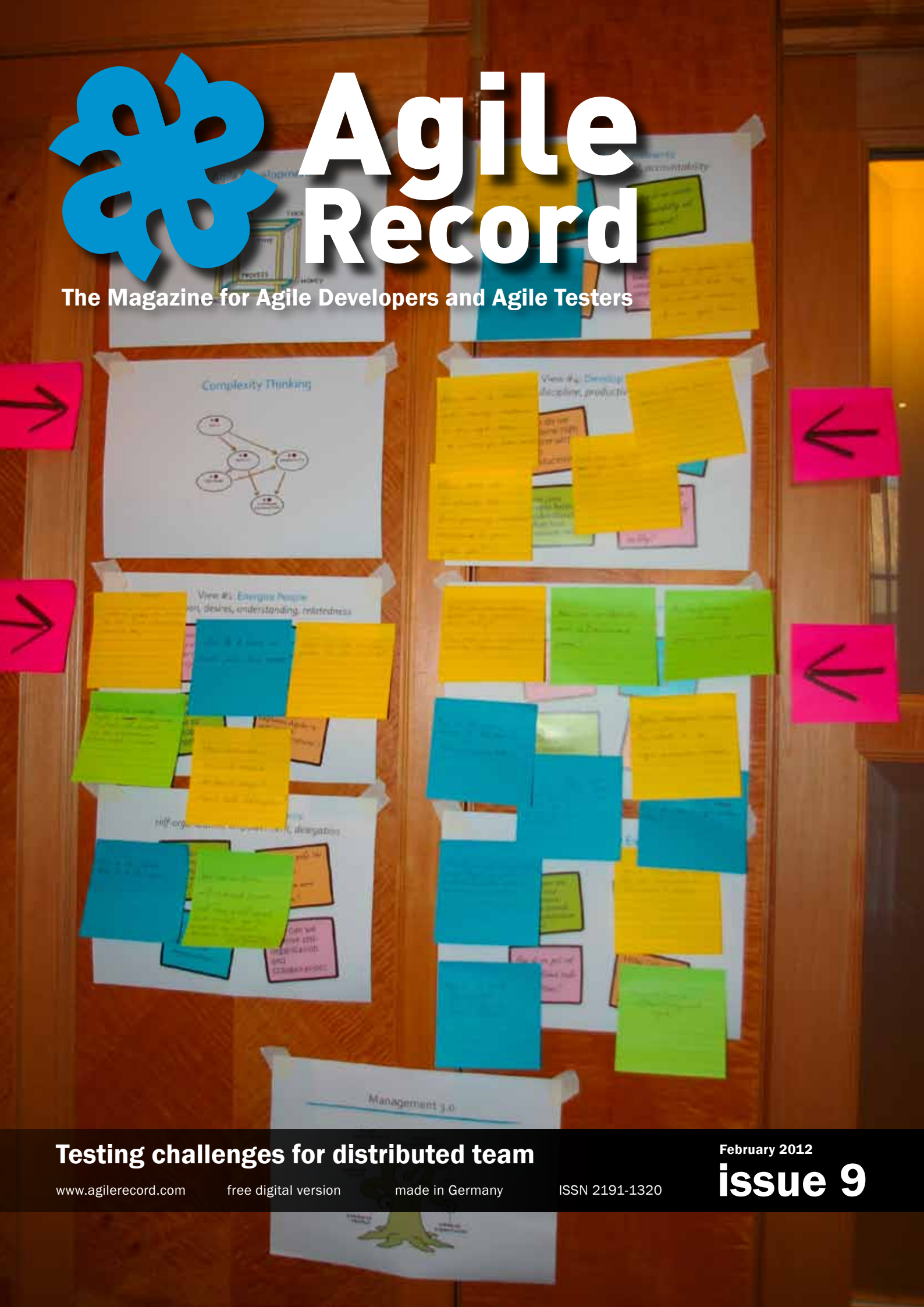
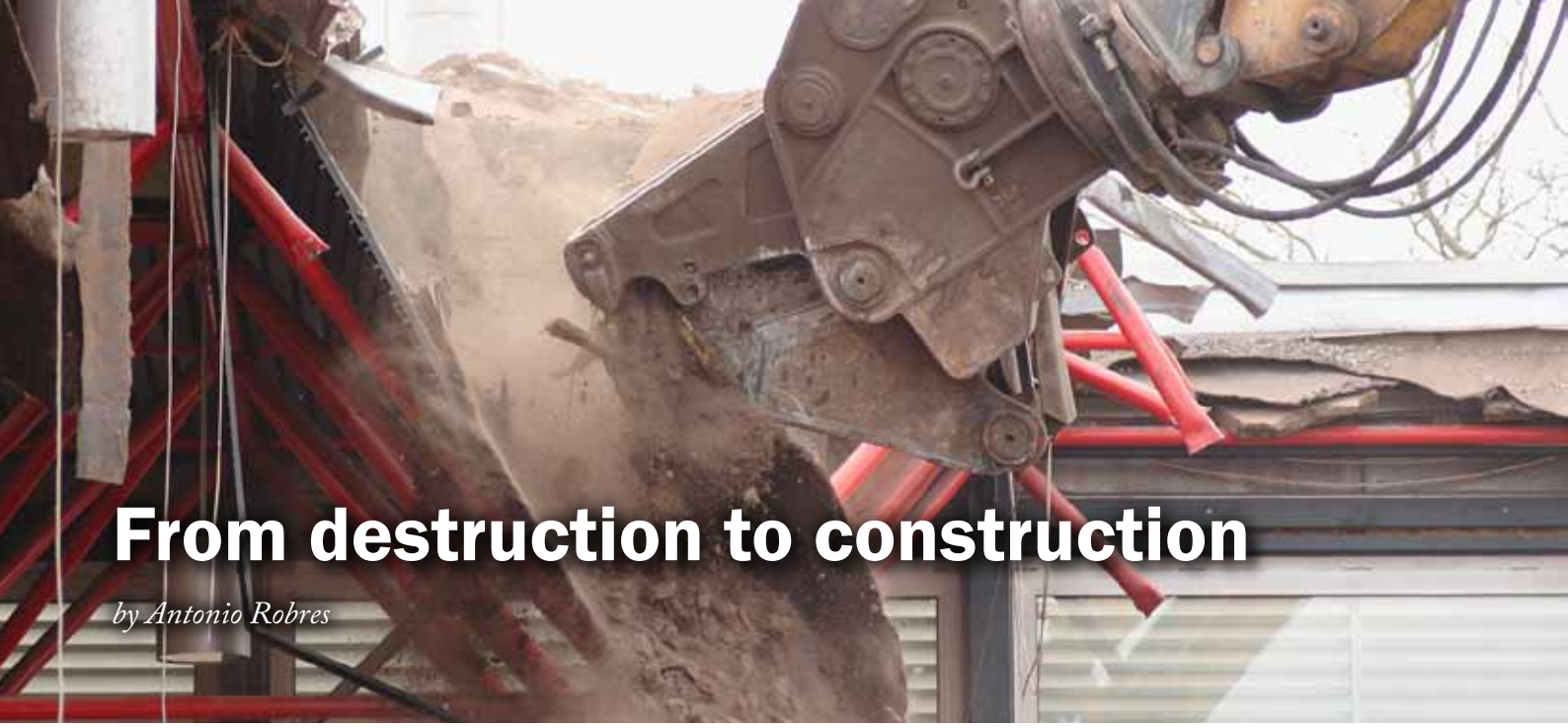

Agile
Record
The Magazine for Agile Developers and Agile Testers
Complexity Thinking
Management 3.0
Testing challenges for distributed team
www.agilerecord.com
free digital version
made in Germany
ISSN 2191-1320
February 2012
issue 9



From destruction to construction

by Antonio Robres

Destructing the software

Since the beginning of the software testing discipline, the main objective has been to find defects in the software implemented before clients find them. This objective tries to save maintenance costs and to prevent that those errors impact the final customers.

With this philosophy, the waterfall model was created. In this model, the testing process is performed only at the end of the development process, and the only aim was finding defects in the software developed, e.g., by destroying the software implemented using techniques and heuristics.

Some years later, the V and W models attempted a new approach with testing departments trying to involve the testers in several stages of the development. In this models the testers participate in all stages of software development and are responsible for the validation and verification of software requirements, software design and implementation. However, the role of testers using these methodologies continues to be a destructive one because the only aim of testers is to detect defects in all the development stages.

With the above models, the testing team does not add quality to the product. It only determines whether the product has enough quality to pass to the next stage of development or not.

This destructive role of the testing department is the reason why other departments (programming and business teams) see it as an enemy, because the only purpose of testers seems to be criticizing the product. This confrontation may influence the relationship between the testers and programmers, and it may impact on the quality of the product.

Agile methodologies: New tester role?

With the new agile methodologies, the testers' role in software development changes radically. The testers or testing departments are not isolated from the development team. The tester is a member of the development team and takes part in all the product development.

The key point for this integration with the team is participating in all team activities, along with developers and stakeholders. In this set-up, testers are involved in the following activities:

Sprint Planning

Insprint planning, the tester has several important tasks for the proper planning and prioritization of user stories.

The first is the possibility of planning the different test activities with the whole team including the programmers, business analysts and stakeholders. The participation in the planning of the tests activities helps programmers and stakeholders see the different activities performed by testers and the effort involved in every activity.

Another task in sprint planning is to give the whole team another point of view about the different user stories. For example, it is possible that a user story for which the software implementation is very simple can require considerable effort from the testing team because the functionality has a big impact on the product.

During sprint planning, the Product Owner explains the different user stories and their acceptance criteria. It is very important for the tester to understand all the functionality explained by the Product Owner. At this point, it is also essential to try to make a first version of the acceptance tests to set accurate acceptance criteria with the whole team. These tests can be useful for the development team for understanding the stakeholder requirements and for implementing the unit tests and software development.

Finally, the participation of the tester in the "definition of done", i.e., setting the requirements to close a user story, is very important. Depending on the complexity and the risks of the user story, the Product Owner and the tester can require documentation,

unit testing, performance testing, etc. The tester should review all the tasks included in the "definition of done" before closing the user story.

Retrospective

Another important activity included in the agile methodologies is the retrospective. This activity is used by the team for a continuous improvement in the whole team (not only in the source code).

In the retrospective, the tester has the opportunity to expose, if needed, the handicaps or deficiencies detected in the testing process during the sprint development. It is also important to try to give feedback to all the team about the quality of the product and the development process in order to increase the product quality.

The testers in the team can also ask for and get feedback from the rest of the team, including the programmers and stakeholders, to increase the efficiency of the testing process within the sprints.

Demos

At the end of every sprint, the team performs a demonstration to the stakeholders to show all the user stories and functionalities developed and implemented by the team.

The testing presence is critical because the tester can obtain feedback from the stakeholders and customers, and information about their interests. This information can be used later to improve the acceptance tests and to apply to the test designs in the next developments.

Finally, it is highly desirable to show the tester activities of the sprint during the demonstrations to explain all the tasks related to quality assurance and testing. It is also essential to show all the tests types performed by the team during the sprint (unit testing, functional testing, acceptance testing, integration testing, performance testing...).

Tester activities using agile methodologies

To become a constructive member instead of a destructive one, some activities in the sprint developments need to be performed at several levels.

ATDD (Acceptance Test Driven Development)

One of the most important contributions in each of the sprints are the ATDD practices.

Using ATDD, the testers can transform user stories or customer requirements into executable tests to verify and evaluate the software implementation of the programming team.

The tests designed using ATDD become executable specifications and can be used to support the programmers and analysts to develop the functionality required by the stakeholders. The tests should be designed before development begins.

The tests designed using ATDD must be exhaustive, describing the whole functionality from the user story, and they must be very simple (KISS concept). These characteristics are vital to avoid ambiguity, errors or misunderstandings between programmers and customers.

Using this practice, the testers provide a point of view which is aligned between the customers and programmers and which focuses on avoiding problems rather than detecting them. Testers add quality to the product and are valuable team members.

The tests designed for ATDD should be expressed using business language in order to validate them with the Product Owner or with customers. Later, these tests can be implemented using a technical language aligned with the programmers' to support and to help in the implementation of the functionality. Using this model, the team can achieve full alignment between business facing and technical facing aspects.

Finally, it is also vital to automate the acceptance tests. This automation should, where possible, be at the behavioral level of the applications instead of via graphical user interface. Using this approach, the team can obtain tests without GUI dependency that are easier to maintain. This also has the advantage of being able integrate the acceptance tests into the continuous integration platform to execute the tests continuously.

Integration Testing and TDD

After the acceptance test implementation, tests must be designed and developed for the lower levels (unit testing and integration testing). These tests can be designed and implemented by developers using TDD (Test Drive Development) practices.

Usually, unit tests are performed by the programmers using xUnit frameworks; this depends on the programming language used for the application.

At the unit test level, the number of tasks for the testers in the team may be less, but they exist.The testers can support the programmers' activities by helping them in the unit testing implementation and providing continuous feedback to improve the coverage of unit tests and make them efficient.

The correct implementation of the unit tests depends also on the acceptance test designs, because they are a good input for the programmers. It is very helpful to request feedback from developers during the implementation of unit test to improve the previously designed acceptance tests.

Maintaining good collaboration between testers and developers is essential during the acceptance and unit test design, and during the implementation. As a result, the quality of the tests will improve and trust between the teams will be increase from the points of view of both parties.

One practice recommended to improve the skills of programmers and developers is performing pair programming sessions that include the unit testing implementation. This practice allows all team members to learn testing and coding skills to improve the quality of the product and to maintain a good relationship between departments.

Continuous integration

Another significant tester activity during product development is the implementation of a continuous integration platform.

Using the continuous integration platform, the team can automatically execute source code compilation, the acceptance tests, the unit tests and integration tests. It is also useful to obtain quality metrics such as coverage, or static analysis to evaluate the quality of the source code.

With the metrics obtained and by using the continuous integration platform, the QA and testing team can analyze the product development and the quality in every component of the source code. This analysis can be useful to find the parts of software most critical in order to increase the effort to improve the quality of the respective code.

If all the development team works within the platform, which enables the team to detect when an error has been introduced into the source code and fix it faster, thereby increasing the team's velocity. It is vital to analyze the impact of the new code introduced into the product and to detect the complexity of the different component of the source code.

Exploratory testing

It is commonly believed that exploratory testing has no place in agile methodologies because all the tests are performed in the acceptance tests and unit tests (except performance or security tests). This is absolutely false, because exploratory testing can bring much value in the sprint development by performing exploratory testing of the functionality and user stories already implemented.

What advantages are obtained by using exploratory testing?

- The tester can interact with the software using other techniques or models, which are not part of acceptance tests and unit tests.
- The tester has the freedom to test without any script or pattern. This characteristic allows testers to be more creative than when using other test techniques.
- The testers can learn about the software during test design and test execution. The knowledge gained will be useful later to perform better designs and acceptance tests in the next sprints.
- Exploratory testing enables continuous improvement in the application and the business facing aspects.

A significant aspect during sprint planning is to schedule exploratory testing tasks in order to obtain another point of view of the product. This testing can be performed not only by testers, but also by the programmers. A good practice can be to plan some exploratory testing sessions where a tester and a programmer pair up to improve the testing skills for both.

Other testing activities

Other activities in the sprint can be the design and implementation of other testing types, such as performance testing, security testing or usability testing.

It is essential that these tests are planned and prioritized in the sprint planning as a user story or as a task with established acceptance criteria. They must be aligned with the programmers and stakeholders. Using this approach, it is possible to design and implement activities as a user story or a requirement.

For these activities, it will be mandatory to use specialized tools and to agree about the tests to be performed with the development team. This will enable support from all the component parts of the team to be obtained regarding the implementation.

Skills required in agile testing

The adaptation to a new role of the tester is not easy, and it may be necessary to have several skills to perform the associated tasks and activities described:

- **Testing knowledge and testing concepts:** This is needed as a basis for testing concepts to perform the testing activities. The tester needs to have knowledge about testing techniques, heuristics, oracles or tools to perform all the activities in agile testing.
- **Coding skills:** If testers are to be valuable for the development team, they should speak the same language. Coding skills are important to help the programming department in the unit testing tasks or to evaluate the software designs. This skill can be useful to build trust between testers and programmers.
- **Testing tools:** These skills are important to enable the testing activities to be carried out efficiently. It is also essential to use the tools to automate the tests and to facilitate the tasks in the sprint.
- **Business skills:** All the tests performed during the sprint should be aligned with the business and the product. For this reason, knowledge of the business domain is very important to create acceptance tests that are useful for the product. This knowledge is also useful to establish a good relationship between Product Owner and customers.
- **Communication:** The testers in agile methodologies mediate between the business layer and technical layer, so it is essential to have some communication skills. It is vital to be an active listener in order to obtain all possible information from both parties and act accordingly.The communication between team members should always be constructive to avoid problems.

- **Team player:** As mentioned at the beginning of the article, one of the major changes in the tester activities is the relationship with the team. In agile methodologies, the tester is more involved in the team and participates in all the activities. One of the key points of the tester in the agile methodologies is the opportunity to be an active member of the team; somebody who adds value to the product. An additional task in the team is that of a "preacher" of quality who convinces the team that quality is the responsibility of the whole team, not only of the tester.
- **Continuous learning:** One desirable skill for the tester is curiosity and being prepared for continuous learning to offer a continuous improvement to the team. Improving knowledge and skills means improving activities performed during development. As a result, efficiency will be higher and the quality of the product is increased. Testers must be able to request and accept the feedback obtained by the rest of the team and apply it to perform continuous improvement – personally and professionally.

Conclusions

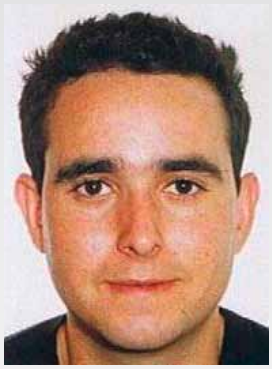
The tester role has changed in the agile methodologies. In the past, the waterfall or V model saw the role of tester as destructive; somebody trying to find as many defects as possible in the product at several levels to save money and decrease any adverse impact on the customers.

With the agile methodologies, the role of the tester becomes more constructive and adds value to both the team and the product. To increase this value, the tester needs to help align stakeholders and programmers. They should be involved at all the stages of the product development.

To be a valuable team member and to add value to the product, it is important to follow several recommendations:

- Be an active participant in all team activities during the sprint, such as sprint planning, retrospectives and final demos.
- Use ATDD to create executable specifications and evaluate them with the stakeholders and with programmers.
- Give support and help the programming team with the unit and integration testing. Try to perform pair programming and pair testing sessions.
- Use a continuous integration platform to compile and deploy all the code, and to execute all the acceptance and unit tests.
- Obtain the metrics from the continuous integration platform to analyze product quality.
- Give and receive continuous feedback using a constructive communication with all the team and to practice continuous improvement.
- Apply clear communication with the team members to create trust.
- Be a quality preacher in the team. Promote and encourage the development team to use good development practices and help them increase product quality. Remind the team that quality is the responsibility of the whole team.
- Enjoy! Make the rest of the team have fun with their work!

> About the author



Antonio Robres
is a QA engineer at Telefonica I+D in Barcelona, Spain. He studied Telecommunications Science at the Polytechnical University of Catalonia in Spain and has a Masters Degree in telecommunication administration. Also, he is an ISTQB© Certified Tester at Foundation Level. He has been working for 6 years in the field of software testing and quality engineering for different companies such as Telefonica, Gas Natural or Grifols. His work focuses on the design and execution of several testing projects, mainly in the fields of mobile devices and web applications. He is also involved in the design and development of test automation projects with open-source tools. He was a speaker at the QA&TEST 2010 and at VLCTesting 2011. He is a writer in the testing blog www.softqatest.com. He also is a member of the TestQA association and a member of the SSTQB©.