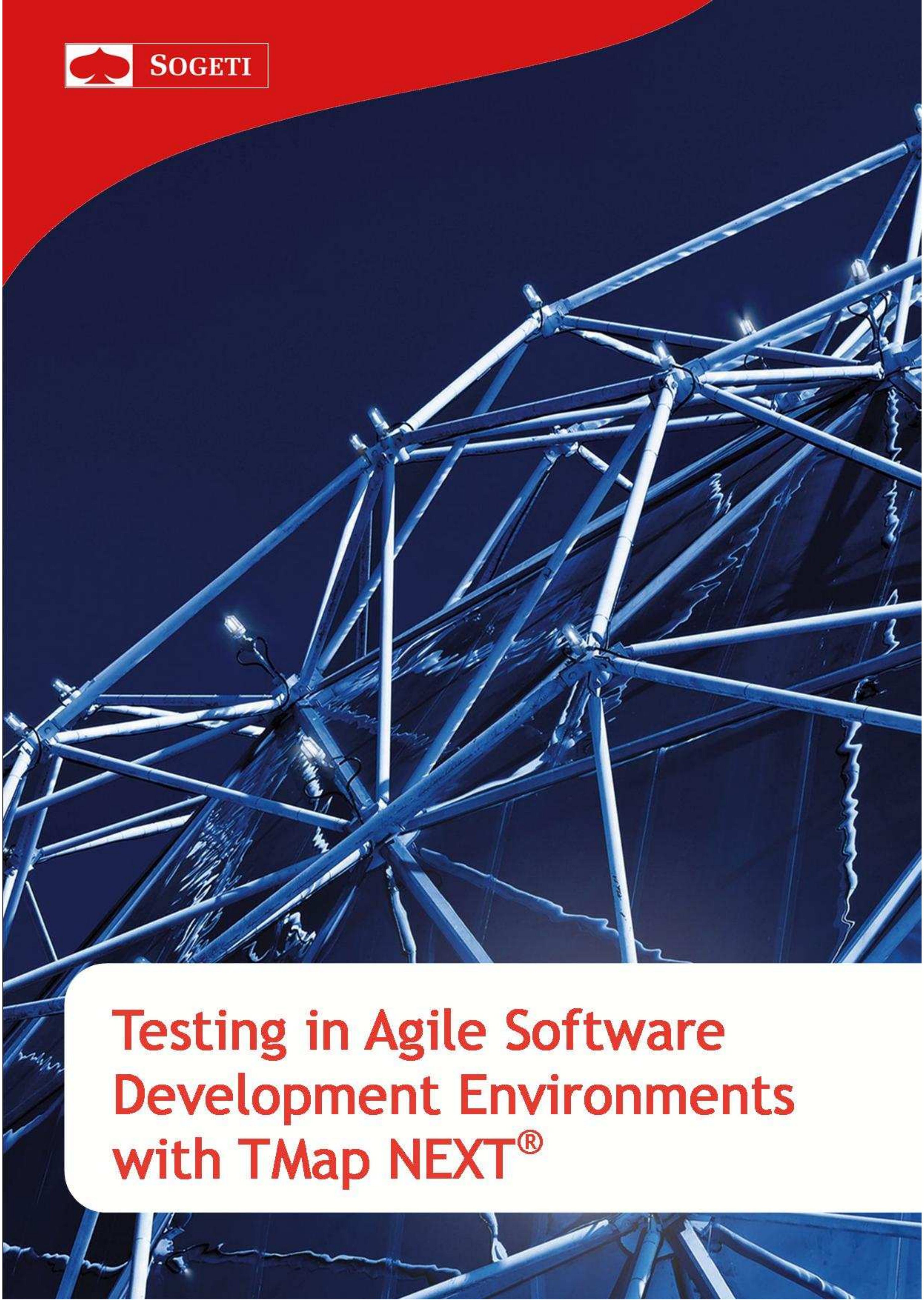
TESTING IN AGILE SOFTWARE DEVELOPMENT ENVIRONMENTS WITH TMAP NEXT®

authors: Leo van der Aalst en Cecile Davis
based on the original point of view paper

© 2011, Sogeti Nederland B.V., based in Vianen, the Netherlands.

This work (or any part thereof) may not be reproduced and/or published (for whatever purpose) in print, photocopy, microfilm, audio tape, electronically or in any other way whatsoever without prior written permission from Sogeti Nederland B.V. (Sogeti).

All rights reserved TMap®, TMap NEXT® and TPI NEXT® are registered trademarks of Sogeti.



Testing in Agile Software Development Environments with TMap NEXT®

TABLE OF CONTENTS

1 SUMMARY

Structured testing can be perfectly integrated in agile development. This *vision* paper is intended for everyone with an interest in Sogeti's vision on testing in relation to the agile manifesto[1] and, in general, the application of TMap NEXT® [2] in an agile software development environment (ASDE). This paper does not describe in detail how to use TMap NEXT® together with a specific agile method, like Scrum.

In agile processes a number of aspects pose a significant challenge to the traditional view of professional testing: lack of detailed requirements, reduced multipurpose process documents (test strategy, plans and cases etc.), always delivering working software, nightly integration and builds, user involvement, short time boxed iterations, potential technical requirements for testers, change of culture (self managed teams), distributed or off-shore teams and the fast pace of delivery.

Since testing is not only a vital element of agile projects but also an integral part of agile software development, Sogeti decided to state its vision on testing in an ASDE:

1. **Use the agile manifesto as a starting point**
   Our test vision is based on the four values of the agile manifesto together with the twelve principles. This means that each and every proposed test activity must be in line with the agile manifesto and its principles.
2. **Integrate the test activities in the ASDE**
   a. The test activities must be integrated in the development process itself, agile teams test continuously.
   b. All team members must be prepared to perform test activities. Although a professional tester should be part of the team, this does not mean that all test activities must be carried out by the tester.
   c. Testing should move the project forward.
   This means a shift from quality gatekeeper solely to collaboration with all team members to provide feedback on an ongoing basis about how well the emerging product is meeting the business needs.
   d. Test tools are increasingly important and should be used to support and improve the performance of agile teams.
   e. Test activities/acceptance criteria must be part of the definition of done.
3. **Find balance by making well-considered choices**
   The right balanced choices will always be context sensitive, which means that different factors like type of organization, type of project, business goals, available resources and available technology are taken into account.
4. **Reuse values of the traditional structured test approach of TMap NEXT®**
   The traditional test approach is still very valuable, but must be adapted to the agile way of working.

[1] http://agilemanifesto.org/

[2] Koomen, T., Aalst, L. van der, Broekman, B., Vroon, M. (2006), TMap NEXT®, for result-driven testing, 's-Hertogenbosch: Tutein Nolthenius Publishers, ISBN 90-72194-80-2

2 INTRODUCTION

This paper contains the following chapters:

Chapter 1	Summary.
Chapter 2	The target group and the scope of this paper are described. A short introduction to the testing challenges in an agile software development environment and the shift from traditional software development to incremental software development is described.
Chapter 3	In this chapter the high level vision on agile environments is given by showing the four values and the twelve principles of the agile manifesto.
Chapter 4	Since the agile manifesto doesn't say anything about testing, Sogeti gives its test vision in this chapter.
Chapter 5	For a common understanding, the common characteristics of the agile software development method in practice are described.
Chapter 6	In this chapter hints and tips are give on how to put Sogeti's vision on agile testing into practice using TMap NEXT® including the business driven test management approach.
Chapter 7	Glossary.

2.1 Target group and scope

This *vision* paper is intended for everyone with an interest in Sogeti's vision on testing in relation to the agile manifesto[3] and, in general, the application of TMap NEXT® [4] in an agile software development environment (ASDE). Especially for managers, who work in agile software development environments (ASDE) and want to know more about Sogeti's vision on testing in relation to the agile manifesto[5] and TMap NEXT® in general. This paper does not describe in detail how to use TMap NEXT® together with a specific agile method, like Scrum.

2.2 Agile software development environments pose challenges for testing

In agile processes a number of aspects pose a significant challenge to the traditional view on professional testing:

- Lack of detailed requirements.
- Reduced, multipurpose process documents (test strategy, plans and cases etc.).
- Frequently delivered working software.
- Intensive user involvement.
- Short time boxed iterations.
- Potential demands on the testers' technical skills.
- Change of culture (self managed teams).
- Dealing with distributed or off-shore teams.
- Fast pace of delivery.

[3] http://agilemanifesto.org/

[4] Koomen, T., Aalst, L. van der, Broekman, B., Vroon, M. (2006), TMap NEXT®, for result-driven testing, 's-Hertogenbosch: Tutein Nolthenius Publishers, ISBN 90-72194-80-2

[5] http://agilemanifesto.org/

How can 'complete' testing be carried out without detailed requirements in short iterations and with a scope that is uncertain? How much test documentation is 'sufficient'?

Is it possible to harness the apparent advantages of agile, with its emphasis on speed, customer responsiveness and flexible pragmatism, together with the structured discipline of testing, with its focus on defined plans, sign off levels, and sequential process steps?

With regard to the levels and phases of testing, what, in this context, is 'acceptance testing'? Does it mean attempting to substitute unit tests for acceptance tests? What about non-functional tests, the evaluation of quality characteristics such as performance, reliability, usability, scalability, etc.?

In this plethora of uncertainties, there is a further issue, that of the role of the professional tester. This is particularly important in the context of multi-site teams that are increasingly becoming the norm in large organizations. Do testers have the right skills and are they able to add value in this seemingly unstructured environment?

This vision document addresses the issues specifically related to testing in an ASDE.

2.3 Shift to an agile software development environment

In sequential software development lifecycles, the emphasis traditionally has been on defining, reviewing and subsequently validating the initial business requirements in order to produce a full set of high-quality requirements. Further levels of testing, such as system and user acceptance testing, are then planned to achieve coverage of these requirements and their associated risks.

This approach, combined with a full lifecycle testing strategy, utilizing effective document/code evaluation and other levels of development testing, such as unit and unit integration testing, can achieve very high levels of software quality.

However, in reality, projects invariably fail due to a lack of end-user involvement, poor requirements definition, unrealistic schedules, lack of change management, lack of proper testing and inflexible processes.

This traditional approach means that there often is a disconnection between users and testers. As a result, changes to requirements that often surface during the design or coding phase may not be communicated to the test team. This results either in false defects, or in a test strategy being incorrectly aligned with the real product risks.
To combat this, traditional projects exert a lot of effort in managing change because in long-term projects, change is inevitable.

The agile software development approach, based on iterative development in which requirements and solutions evolve in combination and change is embraced, would therefore appear to offer a potential solution to the problems in a traditional approach.

Incremental software development processes have been specifically developed to increase both speed and flexibility. The use of highly iterative, frequently repeated and incremental process steps and the focus on customer involvement and interaction theoretically supports early delivery of value to the customer.

Agile software development is not a method in itself. It is derived from a large number of iterative and incremental development methods. Methods such as:

- Rapid Application Development (RAD)
  - 80's, Barry Boehm, Scott Shultz and James Martin
- Scrum - The New New Product Development Game – Harvard Paper
  - 1986, Ikujiro Nonaka and Hirotaka Takeuchi
- Dynamic Systems Development Method (DSDM)
  - 1995, DSDM Consortium
- eXtreme Programming (XP)
  - 1996, Kent Beck, Ken Auer, Ward Cunningham, Martin Fowler and Ron Jeffries
- Feature Driven Development (FDD)
  - 1997, Jeff de Luca

In 2001 in Utah 17 representatives[6] of the above mentioned and other similar methods, met to discuss the need for lighter alternatives to the traditional heavyweight methodologies. In about three days they drafted the agile manifesto, a statement of the principles that underpin agile software development.

2.4 Reviewers

We would like to take this opportunity to thank all of the people who have contributed to the content of this paper: Ben Visser, Clemens Reijnen, Eddy Huisman, Fran O'Hara, Johan Vink, Julya van Berkel, Ken Brennock, Marc Valkier, Marco Jansen van Doorn, Rik Marselis, Rob Baarda and Robin Mackaij.

We have worked on this paper with great enthusiasm. We feel that implementing TMap NEXT® in an agile way will be a valuable addition in an ASDE.

Cecile Davis
Leo van der Aalst

[6] The 17 authors of the manifesto were: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland and Dave Thomas.

3 THE AGILE MANIFESTO

Agile software development has many different 'flavours'. However, the foundation of any of these development methods known today can be found in the manifesto for agile software development. The agile manifesto consists of four values and twelve principles.

3.1 Four values

At http://agilemanifesto.org the manifesto is stated as:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions	*over*	*processes and tools*
Working software	*over*	*comprehensive documentation*
Customer collaboration	*over*	*contract negotiation*
Responding to change	*over*	*following a plan.*

While there is value in the items on the right, we value the items on the left more!"

Some ways wherein the first value expresses itself are the frequent face-to-face communication between individuals, the self-regulation of the multidisciplinary teams and the team's responsibility.

The second value is followed by working in short iterations with working deliverables at the end of each iteration, by prototyping and by efficiency in documentation.

The third value is supported by the involvement of customers in the team, frequent feedback and demonstrations.

The last value can only be true if all disciplines are involved early, and the process is implemented based on a strategy that considers changes being inevitable.

3.2 Twelve principles

The twelve principles behind the agile manifesto are:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity, the art of maximizing the amount of work not done, is essential.
11. The best architectures, requirements and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

4 HIGH LEVEL VISION ON TESTING IN AGILE ENVIRONMENTS

The manifesto together with the twelve principles that lie behind it, is the starting point for the way Sogeti looks at agile testing, for these cover the general ideas about agile methodologies.

Since testing is not only a vital element of agile projects but also an integral part of agile software development, Sogeti decided to state its vision on testing in ASDE:

1. Use the agile manifesto as a starting point.
   Our test vision is based on the four values of the agile manifesto together with the twelve principles. This means that each and every proposed test activity must be in line with the agile manifesto and its principles.
2. Integrate the test activities in the ASDE.
   a. The test activities must be integrated in the development process itself, agile teams test continuously.
   b. All team members must be prepared to perform test activities. Although a professional tester should be part of the team, this does not mean that all test activities must be carried out by the tester.
   c. Testing should move the project forward.
      This means a shift from quality gatekeeper solely to collaboration with all team members to provide feedback on an ongoing basis about how well the emerging product is meeting the business needs.
   d. Test tools are increasingly important and should be used to support and improve the performance of agile teams.
   e. Test activities/acceptance criteria must be part of the definition of done.
3. Find balance by making well-considered choices.
   The right balanced choices will always be context sensitive, which means that different factors like type of organization, type of project, business goals, available resources and available technology are taken into account.
4. Reuse values of the traditional structured test approach of TMap NEXT®.
   The traditional test approach is still very valuable, but must be adapted to the agile way of working.

It is important to realize that agile methodologies arose from software developer's side. Agile is not a prescriptive discipline, hence, specific processes like testing, configuration management, build and release processes are not discussed. However neutral the agile manifesto is towards the test discipline, the methods that embraced the agile philosophy never focused on how to integrate the testing discipline thoroughly in ASDEs, nor did they consider what the consequences for test professionals would be. Therefore many organisations struggle with the implementation of testing in an ASDE.

5 COMMON CHARACTERISTICS OF AGILE METHODS

Agile methods break the work that has to be done into small chunks of functionality that can be delivered in an iteration. Iterations (sprints) are short time frames ('time boxes') that typically last from one to four weeks. There is no long-term detailed planning at the beginning of the project, however, there is a global iteration planning. Detailed iteration planning is done at the beginning of the iteration concerned. Each iteration includes planning, requirements analysis, design, coding, unit testing, system testing and acceptance testing. At the end the process is evaluated and a working product is demonstrated to the stakeholders. Documentation is produced as required. An iteration may not add enough functionality to warrant a market release, but the goal is to have working software (with accepted defects) at the end of each iteration to show to the customer. Team & stakeholders discuss each open defect during the product demonstration.
The customer has the final verdict whether a defect is acceptable or not. Usually, multiple iterations are required to release a product.

Responsibility
Teams in agile projects are usually cross-functional and self-regulating and are as a team responsible for the results. Team members take responsibility for tasks that deliver the correct functionality an iteration requires. They decide as a team how to meet the iteration's requirements.

Efficient communication
Agile methods prefer co-located teams, where face-to-face communication, project notice boards and efficient forms of communication are preferred over large written documents. When a team works in different locations, they can use videoconferencing, voice, e-mail, etc. to maintain daily contact.

Customer representative
No matter what development disciplines are required, each agile team will contain a customer representative. This person is appointed by the stakeholders to act on their behalf and makes a personal commitment of being available for the team to answer questions and to give feedback. This person not only helps to explain requirements, especially when developing the acceptance test cases, they also have the power to prioritize requirements. At the end of each iteration, stakeholders and the customer representative review the progress and re-evaluate priorities with the purpose to optimize the return on investment and to ensure alignment with customer needs and company goals.

Daily communication
Most agile implementations use a daily face-to-face communication among team members. This specifically includes the customer representative and any interested stakeholders as observers. In a brief session, team members report to each other about what they did yesterday, what they intend to do today, and what their roadblocks are.

Definitions of terms used in this paper
There are many different views and definitions on agile software development terms. In this paper we use the terms as described below, but one should remember that some organisations use these terms differently:

- A project contains many iterations and a list of user stories (high level requirements).

User stories are very high level, many times just one line, but they can be gathered at a very early stage in the project and are usually emerging throughout the project.
- An iteration is a time boxed section of a project designed to deliver a subset of working user stories.
- A release is a point in the project where working software is delivered.

6 HOW TO PUT THE TEST VISION INTO PRACTICE

In this chapter Sogeti's vision on testing in ASDEs will be discussed in more detail.
In addition, some tips and hints will be given on how to put the test vision into practice.

The four, high level, test vision values are:
1. Use the agile manifesto as a starting point.
2. Integrate the test activities in the ASDE.
3. Find balance by making well-considered choices.
4. Reuse values of the traditional structured test approach of TMap NEXT®.

The above mentioned values are described in the following sections.

6.1 Use the agile manifesto as a starting point

When implementing an agile method, a lot of organizations pick one or two appealing values or principles from the agile manifesto and start to implement these favourite values. However, wise and sensible it may sometimes be to start with small steps, implementing only part of the values of the manifesto will not work in the end. It often happens that organizations stop implementing halfway or just do not spend as much attention to the other values as to the first. The manifesto is not a checklist you can pick from. The way these values are implemented is open, which is why there are different agile methods, but all values need to be implemented. And if an agile method is implemented partly, it will also have consequences on how to deal with test activities. Moving to agile is similar to any process change/improvement. That is, everybody needs to be clear on why the change, and the steps to be taken to change the process, is necessary.

First of all, to solve the issue of a test process that is not based on the agile manifesto as a whole, it is necessary that all stakeholders involved are familiar with the manifesto and the agile method that will be implemented. This seems very simple: just give them a book to read. However, it is very important that people really understand the values and their consequences. The manifesto and its principles stand for a mindset; it is not a checklist.

Furthermore, it is most important that people have the same understanding.
If everybody has to learn the values on their own, for certain, different interpretations will arise. That will lead to miscommunication and misunderstanding. The best and most complete solution is therefore, to start by appointing an expert as coach, who can give training and support. When everybody is used to the new way of working, this coach will no longer be necessary.

6.2 Integrate the test activities in the agile software development environment

In this section five aspects with respect to the integration of testing into an ASDE are discussed.

6.2.1 Test continuously

In traditional development environments testing is often inserted at a later stage, although most test methods advise an early test involvement. In agile environments testing is, by nature, incorporated from the beginning and throughout the whole process. So, organizations need to think on how to deal with traditional test levels like system test, user acceptance test, end-to-end testing, and production acceptance test in an ASDE.

In this section two aspects with regard to continuous testing are discussed:
- Integrate testing with development activities.
- Apply test levels in an ASDE.

Integrate testing activities with development activities
Testing in agile projects cannot be seen as a separate activity but needs to be integrated in the entire process. Testing is not only a vital element of agile projects, it is an integral part of agile software development.

The underlying thought of the agile philosophy is to deliver business value as early as possible and in the smallest workable piece of functionality. To prove that the developed software works as desired *and* to prove business value, the user story and product need to be tested. To make agile work well, testing cannot be seen as a separate activity but needs to be integrated in the entire process.

Testers can add value in different areas by acting as a spider in the agile web. Besides being involved in testing, they will be involved in formulating the definition of done, in planning, in unit testing, in gathering all information necessary to form the test basis, in risk management, in retrospection, in test automation, etc. Therefore, the role of the tester is an important one. This early and high degree of involvement of testing in the entire project has a lot of benefits. For instance, including testing activities/acceptance criteria in the definition of done ensures that the software is considered ‘done’ only when the testing is fully ‘done’.

A high degree of involvement at an early stage is essential for testing in agile environments. All disciplines, testing included, must be involved as soon as possible in the process. This is an essential part of agility, required, for instance, by the value in the manifesto of *responding to change.* In agile environments, changing the requirements can be done more easily than in a non-agile environment. If the change is discussed and agreed upon within the team, which would also include the customer, then no one can really oppose to the change. Testing, especially, is of importance in an early stage of changing requirements, because of the value it can add to impact analysis and risk analysis. The sooner testing can respond to changes and take actions, the better.

Also, responding to change raises a need for test automation, efficient and effective documentation and more face-to-face communication. These differences require a different approach and different skills from testers (see also section 6.2.2).

Test levels in an ASDE
In traditional software development environments, test levels like system test, user acceptance test, end-to-end testing and production acceptance test are commonly used. However, in ASDEs, all team members have to work together to get the work done. In such an environment it is not relevant to talk about system test and acceptance test levels/teams with managers and budgets of their own.

If there is a need for some distinction, instead of test levels quality characteristics can be used per user story. This way, certain test activities can be grouped together. Quality characteristics can also be used to SMART-en up acceptance criteria in the definition of done.

And what about end-to-end testing? Is it possible to incorporate this test level in the agile process? This depends on the specific situation. Theoretically, end-to-end testing would benefit enormously from agile software development. However, it will usually be difficult to realize the end-to-end test within the agile process. Therefore, Sogeti usually advises to implement end-to-end testing as a separate phase after the agile project is completed.

6.2.2 The testers' role changes

Agile project teams have to work together closely and the team should contain the people required to make the project successful. In agile methods, involvement of all disciplines, including testing, is part of the philosophy. Therefore, working with multi-disciplinary teams is very important. So a tester must be part of the agile team.

Working in multi-disciplinary teams also means that all team members must be prepared to fill in other roles if necessary. For instance, a developer must be willing to fill in a testing role if the need arises. Especially, of course, when there is time pressure on the test activities. On the other hand, testers must be willing to assist other team members in their activities where they can according to the projects needs. This can demand more technical skills and knowledge from a tester.
Working together means two-way traffic.

The nature of the testers' role changes in iterative projects. Testers are no longer the high-profile victims, they are no longer the lonely advocates of quality. They are competent service providers, working in a team that wants to achieve high quality. The testers' role becomes richer and more influential. Agile methods require testers to be involved in the development project continuously and right from the start.

Testers need to have technical knowledge of the software they are testing and need to understand the impact on automation as well as the functional implications. Some iterations may be development heavy, some automation heavy, some test heavy; the agile tester needs to be adding value in all instances.

The personal characteristics that are especially important for a tester working in an agile team are:
- Communicative
- A retrospective attitude
- Flexible
- Pro-active
- Creative but practical
- Thinking in solutions
- Customer-oriented
- Open-minded
- A team player
- Interfering with everything like a spider in a web

Professional testers should adapt to fit this different role and so provide additional value by not only focusing on finding defects but also fulfilling a team role.

6.2.3 Testing moves the project forward

On traditional projects, testing is usually treated as a quality gate, and the test team often serves as the quality gatekeeper. The result of this approach exist of long, drawn out bug scrub meetings in which different parties argue about the priority of the bugs found in test and whether or not they are sufficiently important to delay a release.

In agile teams, the product is built well from the beginning, using testing to provide feedback on an ongoing basis about how well the emerging product is meeting the business needs.

This sounds like a small shift, but it has profound implications. The adversarial relationship that some organizations foster between testers and developers must be replaced with a spirit of collaboration. This is a completely different mindset.

Testers must be pro-active and work with the business stakeholders to understand their real needs and concerns. In traditional environments, this is usually called 'requirements elicitation'. In the context of agile development, the purpose of this discussion is not to gather a huge list of requirements but rather to understand what the business stakeholder needs from one particular user story.

During these discussions, the tester must ask questions designed to uncover assumptions, understand expectations around non-functional needs such as performance, reliability, security, etc., and explore the results the business stakeholder is requesting. This will improve the quality of the software product.

One, very effective, way of involving testers early is to introduce test driven development. This way, testers, programmers and business representatives can all learn from each other, while interpreting the requirements together.

6.2.4 Test tools are necessary

Over time, test tools have become increasingly important to the performance of agile teams. This is not just because teams sometimes have to be technically oriented, but because the right tools can help a team to become more efficient.

If agile is about speed, efficiency and flexibility, then the role of automation in agile is to support this and remove mechanical, routinely and time consuming tasks. Due to the required speed of agile, management of test data and environments needs to be very efficient and effective with little if any room for unnecessary manual effort. Although this seems logical, it is still very hard for people working in an ASDE to decide upon the tools to be used and how these tools should be implemented and used. Tasks that can normally be automated within agile teams include:

- Build and integration process.
  Usually in agile teams, this process happens on a very regular basis (every night), resulting in a new build every day, with almost zero manual effort being put into this task. This requires good configuration management and build tools.
- Unit (integration) Test.
  These are part of the nightly build and integration process, allowing the development team to get instant feedback on the quality of their code. The execution of these unit tests requires no manual intervention.

- Static Analysis Tools.
  Instead of doing manual code reviews, the analysis tools review the code against coding standards to uncover defects. The manual reviews can be kept for particular types of defects or more complex code.
- Test data and environment management.
  Available tools can generate data to manage the test environment.
- Regression Testing.
  To be able to follow the quick pace of agile software development, agile teams cannot do without automated regression testing.
- Functional Testing.
  Until now, functional automation testing has focused on regression testing. However agile teams are pushing for functional testing to be automated earlier and earlier in the development lifecycle, so that it is the design of test cases rather than the execution that is important. The use of Model Based Testing (MBT) tools in combination with the automation of test case execution is a (almost) perfect solution for functional testing in agile environments. Test execution should be automated as much as possible.

6.2.5 Testing is part of done

In traditional software development environments with strict boundaries between development and test, it often happens that user stories are declared 'done' before it is tested properly.

Of course, agile teams only count something as 'done,' when it has been implemented, tested and bugs have been fixed.

Incorporating testing in the definition of done gives more control on iteration planning, on testing risks, more early involvement for testers and therefore more grip on the test process.

6.3 Find balance by making well-considered choices

The most important consideration when using an agile software development method, is finding the right balance in the choices that have to be made. When you look at the four values of the agile manifesto, you could think about a balanced choice for each value between the left and right item.

In practice, however, we see agile project teams struggle mostly with finding balance between:
- Working software and comprehensive documentation.
- Covering the risk and the available time and money.

Therefore these two balances will be discussed in the following sections.

6.3.1 Find the balance between working software and comprehensive documentation

Does agile mean the end of all documentation? To those looking at adopting agile practices, especially if they come from more traditional project management backgrounds, agile can seem quite loose, especially in the area of documentation.

The agile manifesto values working software over comprehensive documentation. And in agile projects working software is perhaps the ultimate quantification of your projects status. This may take some time to get used to.

Agile methods are all in favor of documenting only what is necessary. They simply value working software over comprehensive documentation. However, what level of documentation is necessary may vary per project. Besides that, just deciding on what to document and what not, is not enough. Whenever decisions are made on, for instance, the extent of documenting requirements, it must be thought over how the information about these requirements that is not in the documentation is communicated between the different "requirements stakeholders". There should be a balance between documentation and communication so that important information does not get lost.

In addition, there is a number of principles behind the manifesto that elaborate on this value:

- Working software is the primary measure of progress.
- The teams' highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Sometimes end-user documentation is used as one of the criteria for 'done'. It can also be used to document decisions for team continuity. And documentation may even be needed for regulatory compliance.

With regard to the quantity of documentation to be produced, the team could be asked to answer two simple questions:

- Does the documentation add value?
- Is the team better off writing it?

6.3.2 Find the balance between risks versus time and money

An agile project, like any other project, never has unlimited time and money for testing purposes. Such constraints in terms of time and money represent constraints on the test result to be achieved and therefore reduced coverage of the product risks. As such it is important to make well-considered choices in relation to the optimum division of the available time and money across the user stories that require testing. In fact we need to determine product risks and test strategy in an *agile business driven test management way*.

Based on the insight resulting from the product risk analysis, high risk user stories can be tested more thoroughly than those representing a lower risk. A product risk is the chance that the product (user story) will fail in relation to the expected damage if it does.

Product risk = Chance of failure * Damage

When performing the product risk analysis, all stakeholders should be member of the agile team and participate in the analysis.

In an agile environment it is difficult to perform a product risk analysis for the release as a whole. There is just not enough detail available of individual user stories. So, in practice a product risk analysis can be performed at the start of each iteration.

Since an iteration will deliver a few user stories only, it does not take much time to perform the analysis (it could be done on a whiteboard). During an iteration, some product risks can change from high to low or vice versa and new product risks can emerge. The product risk analysis evolves with the product.

The next step is to determine the testing strategy. In this activity, the outcome of the product risk analysis is used to determine which combinations of quality characteristic and user story are to be tested with what test intensity.

All user story and quality characteristic combinations are carried out by the agile project team members. And, as with the product risk analysis, the test strategy will be evaluated regularly and will evolve with the product.

The test strategy is the foundation for every test action, activity, process or project. It holds the justification of what will be tested and how this will be done. Proper risk analysis drives this justification: "No Risk, No Test". The test strategy describes the test goals, how they will be approached and how (product) risks are covered. Risks influence priorities in agile methodologies. Areas that carry the greatest risks need to be prioritized highly. Furthermore, risks will trigger the noted flexibility of agile environments in deciding what to do and what not to do in terms of time, costs and quality. The test strategy thus facilitates the whole agile team.

6.4 Reuse and adapt the values of TMap NEXT®

Many people think that agile projects are chaotic, unorganized and uncontrolled. On the contrary. Agile projects will not be successful when they lack discipline or structure. Because of this prejudice, however, structure is often thrown out as is the baby with the bathwater.

A structured testing approach offers the following advantages:

- It delivers insight into, and advice on, any risks in respect of the quality of the tested system.
- It finds defects at an early stage.
- It prevents defects.
- The testing is on the critical path of the total development as briefly as possible, so that the total lead time of the development is shortened.
- The test products (e.g. test cases) are reusable.
- The test process is comprehensible and manageable.

When implementing an agile method, organizations often find it difficult to decide on what practices to keep and what practices to throw out. Usually, organizations find it easier to start afresh without much consideration for the consequences of throwing out old structures, and very often the baby is thrown out with the bathwater.
When organizations refrain from this decision process, they might save time at the beginning, but in the end the result can be loss of insight into the quality of the tested system, lacking reusability, lacking maintainability, more defects and delays in testing due to an unmanageable test process and most of all loss of all the advantages of an efficient running agile project.

To be able to decide upon which test structures and activities to keep and which to discard, it is necessary to have a good knowledge about the agile method to be implemented. An expert as coach, who can give training and support and who is aware of all the issues coming up during a transition from one method to another is almost indispensable. Usually, many practices and procedures, whether or not adjusted, can be reused.

In the following sections some examples will be given of how to adapt the four essentials of TMap NEXT® to an ASDE.

6.4.1 TMap NEXT's adaptability

In agile development reacting to change, learning and improving the way of working within the team are fundamental elements. TMap NEXT® supports this, since one of the TMap NEXT® essentials is "adaptability" with almost the same elements (in TMap NEXT® called "properties") as mentioned above.

TMap® is an approach that can be applied in all test situations and in combination with any system development method, explicitly including agile development methods.
It offers the tester a range of elements for his test, such as test design techniques, test infrastructure, test strategy, phasing, test organisation, test tools, etc. Depending on the situation, the tester selects the TMap® elements that are appropriate. Sometimes, only a limited number of elements need to be used; at other times, a wide range of elements will be appropriate. This makes TMap® an adaptive method. But remember, where adaptation is required, balanced choices must be made too.

The adaptability of TMap® is not focused on a specific aspect of the method, but is embedded throughout the method. Adaptability is more than just being able to respond to the changing environment. It is also being able to leverage the change to the benefit of testing. This means that TMap® can be used in every situation and that TMap® can be used in a changing situation. During the execution of projects, changes occur that have an impact on earlier agreements. TMap® offers the elements to deal with such changes.

TMap's adaptability can be summarised in four adaptability properties:
- Respond to changes.
- Reuse products and processes.
- Learn from experiences.
- Try before use.

6.4.2 TMap NEXT's approach towards techniques and tools

TMap NEXT® supports effective and efficient testing and test automation, with a complete toolbox. With this tool box, the tester possesses a great number of options to meet the test challenges in an ASDE successfully.

Many techniques can be used in an ASDE. Some techniques may have to be adjusted slightly to the ASDE. Useful techniques are: test estimation, defect management, product risk analysis, test design and product evaluation.

To execute tests efficiently, test tools are necessary (see also section 6.2.4).
In TMap NEXT® the test tools are classified in four groups:
- Tools for planning and managing tests.
- Tools for designing tests.
- Tools for executing tests.
- Tools for debugging and analyzing code.

Using tools can have the following advantages:
- Increased productivity.
- Higher testing quality.
- Increased work enjoyment.
- Extension of test options.

6.4.3 TMap NEXT's business driven testing

Agile development is about doing the right things for the right reasons, all aimed at delivering value to the end-user with the right quality for the right price. The TMap NEXT® business driven test management approach also aims to do this.

In general, it can be said that the business driven test management approach (BDTM) aims to achieve a balance between the investment in money and time on the one hand, and the result to be achieved and the risks covered on the other.

It is important to give serious thought to the test strategy, also in agile environments. Choices must be made on what to test and how intensively to test it. Such choices depend on the risks that the team (including the client!) believes it will incur, how much time and money is available, and the result the organization wishes to achieve. The fact that these choices are based on risk, result, time and cost is called 'business driven'.

Thanks to the BDTM approach, TMap® pays explicit attention to communication by speaking the 'language' of the client. BDTM aims to set up and manage a test process in close collaboration with the client, seeking a balance between the aspects of result, risk, time and cost.

Result
Where in more traditional environments usually the scope of the project is fixed and time and quality may vary, in the agile approach time and quality are fixed and scope can vary. During the project, the client is frequently involved in making choices relating to the required result. The advantage is that the test process continues to match the wishes and requirements (= test goals), and therefore the expectations, of the organization as closely as possible.

Risk
The test effort is related to the risks for the client of the system to be tested. As a result, the deployment of manpower, resources and budget focuses on those parts of the system that are most important to the client. TMap NEXT® uses the test strategy to distribute the test effort over the system parts, thus allowing the client to gain insight into the extent to which risks are covered or not. In agile projects, the team will be leading in deciding on risks and priorities, since the team has the best insight in the risks and priorities and how they change during the development process (see also section 6.3.2).

Time
The schedule of the test process is related to the possible requirements relating to the end date of the test project and the formulated test goals. In agile, iterations are time boxed, so time is fixed within iterations. However, the number of iterations in a project is not always set on beforehand. The test goals to be achieved can change during the project (flexible scope). In the mean time, the client has an adequate picture of the lead time and the relationship with the test goals.

Cost

The budget for the test process is related to an imposed budget, if any, and the formulated test goals. If changes are made that have an impact on the required testing intensity of the user stories, this is translated immediately to new or adjusted test goals and an amended budget. As a result, the client has at all times an adequate picture of the required budget and the relationship with the test goals.

6.4.4 TMap NEXT's life cycle model

To have controlled, repeatable testing, a process has to be in place to facilitate this. The TMap NEXT® life cycle model offers this and can be adapted to fit agile software development.

Agile test processes are much more diffuse and test activities are *performed in parallel* rather than sequential. The activities from the TMap NEXT® test lifecycle can be found in agile too. A clear picture of the TMap NEXT® agile test lifecycle is depicted below.

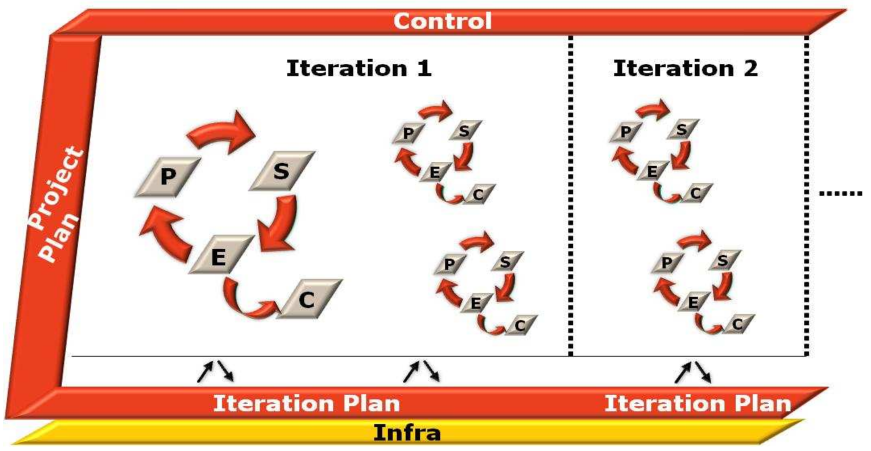


Figure 1: TMap NEXT® agile test lifecycle.

Explanation to figure 1:

A project contains one or more iterations. Per project a compact project test plan is made. An iteration contains one or more user stories. Per iteration a compact iteration test plan is made. The preparation (P), specification (S), execution (E) and completion (C) phases apply to each individual user story. The control and infrastructure phases are continuous phases covering the whole project.

When using TMap NEXT® in an ASDE, keep the following in mind: be adaptive, use TMap NEXT® in an agile way.

Per TMap NEXT® phase some tips and hints are given:

Planning

- A compact high level long term plan (project test plan) covering a number of iterations is typically produced to ensure the test strategy, plan and organisation spanning multiple iterations is defined and agreed upon. This may be produced in 'iteration 0'. The use of a project plan can help teams that are used to a more sequential form of development methodology move to an agile methodology. The expectation should be that this plan will change through the project.

- At the start of each iteration a compact plan is also produced to detail the test approach for the iteration in question (iteration test plan). This plan may also need to change based on new information coming to light during the iteration. In principle, however, these iteration test plans are more or less fixed, in order to protect the team from too many disturbances.

- All the above plans are 'lighter' than those used with sequential development. Documentation does not necessarily mean a pile of paper. Agile practices strive for efficient and effective documentation. If that means test plans can be found in pictures of a whiteboard, that is fine.

- The project test plans are continuously adapted as appropriate during iterations reflecting the iterative nature of agile software development.

- The iterations are typically fixed in duration (time boxed) so that the iteration plan reflects the testing of the highest priority scope (in terms of 'user stories') (See also section 6.3.2).

Control

- Agile culture involves team empowerment, hence the 'manage' element of control is typically more a facilitation activity. In agile environments the team is responsible for the work that has to be done and for solving the problems that arise. The members of this team have to trust each other and in turn they have to be trusted by their management.

- Daily stand-ups are a typical mechanism for agile 'control' which also provide visibility and supports reporting on the testing activities. At the daily meetings the team will discuss the schedule, blocking issues and corrective actions. The testers will also ask if other team members need anything from them.

- Overall control is present through the BDTM aspects and could be made visible by showing the status per BDTM aspect, result, risk, time and cost.

Preparation, Specification and Execution

These activities are performed, normally in parallel to each other and per user story within each iteration.

Preparation

- Evaluation of requirements and user stories (the assessment of the test basis), is normally an informal, ongoing interaction with the product owner, customer (for business insight) and developer (for technical insight). This means that if the tester gets sufficient answers to questions, the assessment of the test basis is done and no testability review report is needed.
- Assignment of techniques will usually include experience based techniques such as for instance, exploratory testing.
- (Re-)evaluation of product risks.

Specification

- Automated test scripts are typically produced within agile teams (preferably at the unit test level, but also at other levels as part of a test driven development approach).
- The level of documentation of manual test specification is typically light (procedural information would not be documented unless absolutely necessary).

A decision needs to be made at the organisation level with respect to what level of documentation is required. For example, having well documented test cases means that more junior testers can execute the test cases, however this takes more time during the initial documentation of the test cases and later at the maintenance stages. Also, important information that is not documented still has to be communicated somehow.

- In parallel to the evaluation of requirements and user stories the tester will create the test specifications. In addition, the answers given during the evaluation by the product owner, customer and developer could become a part of the test specifications. This could very well mean that in the end test specifications contain more (functional, technical, etc.) information of the product then written down in requirements and user stories.

Execution

- Typically with the high level of automation, there is constant regression testing performed at the unit level and at the higher levels as part of a continuous build and integration strategy.
- Pre or smoke tests are normally not required as there is continuous automated regression testing and there are no handovers to independent test teams. However, pre or smoke tests can be used to check if a test is ready to run in a regression test.
- 'Check and assess' is typically relative to the teams definition of working software being 'done' and potentially shippable.
- As the focus is on working software, defects are often repaired almost as soon as they are found. However, not all found defects are fixed immediately. In addition to this, in agile software development projects sometimes already solved defects seem to re-appear in a following iteration. Therefore registration (tracing and tracking) of defects is important. Also, a possible need for reliable metrics must be considered here too.

Completion

- At the end of each iteration the process is evaluated as part of a 'retrospective'. The lessons learned should be incorporated as much as possible in the next iteration test plan.

- Testware preservation is decided upon at the end of each iteration. However, it sometimes can be mainly performed as part of the final iteration, but dependencies between iterations must be taken into account here.

- Testers as well as developers should be using the configuration management process to manage and control test ware.

- In agile environments, responding to change is considered to be more valuable than following a plan. However, responding to change means more than responding to changing requirements. What if team members or roles change? In such situations, transferability of, for instance, test scripts become an issue. Responding to changes can be quite hard if little or no attention has been given to maintainability, transferability and reusability. This is where testware management comes in. Testware management improves maintainability, transferability and reusability and therefore it is an important facilitating area for an agile team.

Infrastructure

- The infrastructure is typically set up and maintained by the team members themselves. So the team must have members with sufficient technical knowledge.

- The tester will also set up or arrange to have the test environment and data to be set up. Ideally this should be something testers can do themselves in an early stage.

- In order to deliver working software, a robust and reliable test environment is essential. For agile environments a stable test environment is crucial because iterations or increments must prove their value in a short period of time. Problems that arise in the environment may block the progress of all other activities and the impact on the iteration can be huge. The test environment also plays a major role in integration tests. The greatest part of integration testing can be part of an iteration deliverable. It can be incorporated into the delivery of a "potentially shippable product", e.g. in a simulated production environment. A stable test environment is a very important facilitating area for an agile team, enabling it to deliver reliable working software at the end of each iteration.

Which tips and hints apply to your situation can and will differ from situation to situation. So this means that the given hints and tips have to be adjusted to your own situation.

6.5 Concluding

Testing in agile software development environments ensues a different way of working. Moreover, a different mind-set is necessary. It will not be sufficient to follow a list of rules, applicable in ASDEs. A professional agile tester will need a thorough understanding of the agile approach along with a readiness to step out of the test compartment. Doing whatever is necessary in an effective and efficient way, in order to be a valuable agile team member, must be the goal of an agile tester. TMap NEXT® can support the agile tester to achieve this.

7 GLOSSARY

BDTM	Business driven test management is aimed at enabling the client to manage the test process on rational and economic grounds. Important BDTM aspects are: result, risk, time and cost.
Chance of failure	The chance of failure is the chance that a product (component) will fail during operational use because it contains a fault. The chance that the product will fail increases with the frequency of its use.
Damage	Damage relates to the negative impact resulting from the failure of the product. Product failure may result in damage for multiple stakeholders.
Definition of Done (DoD)	The definition of done is a checklist of valuable activities required to produce software. The DoD is a simple list of activities (writing code, coding comments, unit testing, system testing, acceptance testing, release notes, design documents, etc.) that add verifiable/demonstrable value to that product. The Definition of done not only describes activities but also the criteria that must be met before a task is completed ("done"). Focusing on value-added steps allows the team to focus on what must be completed in order to build software while eliminating wasteful activities that only complicate software development efforts. Definition of done not only describes activities or tasks but also the requirements that must be met before a task is completed ("done").
Exploratory testing	Is the simultaneous learning, designing and executing of tests, in other words every form of testing in which the tester designs his tests during test execution and the information obtained is reused to design new and improved test cases.
Extreme programming (XP)	Extreme programming is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements.
Iteration	An iteration is a time boxed section of a project designed to deliver a subset of working user stories.
Product risk	The chance that the product fails in relation to the expected damage if this occurs: Product risk = Chance of failure * Damage where Chance of failure = Chance of defects * Frequency of use.
Product risk analysis	Analysing the product to be tested with the aim of achieving a joint view, of the more or less risky characteristics and parts of the product to be tested so that the thoroughness of testing can be related to this view.
Project	A project contains 1 or more iterations and a list of user stories. At project level a set of user stories (requirements) is delivered.
Rapid application development (RAD)	Rapid application development refers to a type of software development methodology that uses minimal planning in favour of rapid prototyping. The 'planning' of software developed using RAD is interleaved with writing the software itself.
Release	A release is a point in the project where working software is delivered to a team. During an iteration testers can expect multiple releases from development.
Requirement	A requirement is a singular documented need of what a particular product or service should be or perform. It is most commonly used in a formal sense in systems engineering or software engineering.
Retrospection	The evaluation of the process at the end of an iteration.
Rational unified process (RUP)	The rational unified process is an iterative software development process framework created by the rational software corporation, a division of IBM.
Scrum	Scrum is an iterative, incremental framework for project management and agile software development based on lean.
Stand-up meeting	A stand-up meeting (or simply stand-up) is a daily team meeting held to provide a status update to the team members. The 'semi-real-time' status allows participants to know about potential challenges as well as coordinate efforts to resolve difficult and/or time-consuming issues.

glossary

Test driven development (TDD)	Test-driven development is a software development technique that relies on the repetition of a very short development cycle: first a failing automated test case that defines a desired improvement or new function is written, then code to pass that test is written.
Test level	A test level is a group of test activities that are managed and executed collectively.
Test strategy	The distribution of the test effort and coverage over the parts to be tested or aspects of the test object aimed at finding the most important defects as early and cheaply as possible.
Testing	A process that provides insight into, and advice on, quality and the related risks.
User story	A user story describes desired functionality from the customer (user) perspective and is not technical. A good user story describes the desired functionality, who wants it, and how and why the functionality will be used. The basic components of a user story are sometimes dubbed as the three C's: Card - the written description of the story, serves as an identification, reminder, and also helps in planning. Conversation - this is the meat of the story; the dialogue that is carried out with the users; recorded notes; mock-ups; documents exchanged. Confirmation - the acceptance test criteria that the user will utilize to confirm that the story is completed. A well-written user story follows the INVEST model: Independent, Negotiable, Valuable, Estimable, Small, and Testable. Though this definition of a user story can be preferred in order for the team to get a good idea of what a customer needs, basic user stories usually comply to the following line: As a <Role> I want to <Functionality> so I can <Goal>.

8 ABOUT THE AUTHORS

Cecile Davis

Cecile Davis is a test consultant, focussing on improvement models like TPI NEXT® and People CMM®. Her specialism lies in the subject of agile testing.

She started her IT career in 1998, programming C at a software company and progressing into test engineering soon afterwards. She has been involved in several agile test projects. She is RUP-certified, co-writer of TPI NEXT® and founder of the SIG on agile testing within Sogeti. She is involved in several (national) SIG's related to this subject.

Besides this, Cecile likes to develop training material and teach. She is becoming a regular speaker at national and international conferences.

Leo van der Aalst

Leo van der Aalst has almost 25 years of testing experience and developed amongst others services for the implementation of test organisations, test outsourcing, test-governance and software testing as a service (STaaS).

He is co-author of the TMap NEXT® and TMap NEXT® BDTM books. Leo designed the EXIN tasks for TMap NEXT® certification, is lector Software Quality & Testing at Fontys University of Applied Sciences, member of the standardization committee 381007 "Software and Systems Engineering" and member of the Innovation Board Testing.

Besides all this, Leo is a much sought-after teacher of test training, a regular speaker at national and international conferences, and he is the author of several articles.

Want to know more?
For more information about how Sogeti's Testing Solutions can help organizations achieve their testing and QA goals, please contact Sogeti at tmap@sogeti.nl
or visit: www.sogeti.nl or www.tmap.net