Agile Testing: Best Practices to Ensure Successful Transformation to Agile Methodologies
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Why Agile Testing</td>
<td>4</td>
</tr>
<tr>
<td>Common Challenges Faced by CSPs During the Transition to Agile Testing</td>
<td>5</td>
</tr>
<tr>
<td>Agile Testing Methodology</td>
<td>6</td>
</tr>
<tr>
<td>Mind-Set</td>
<td>6</td>
</tr>
<tr>
<td>Skill-Set</td>
<td>7</td>
</tr>
<tr>
<td>Tool-Set</td>
<td>8</td>
</tr>
<tr>
<td>Management Support</td>
<td>9</td>
</tr>
<tr>
<td>Agile Testing Methodology Helps CSPs to Overcome Many Challenges</td>
<td>10</td>
</tr>
<tr>
<td>Conclusion</td>
<td>11</td>
</tr>
<tr>
<td>About the Author</td>
<td>11</td>
</tr>
</tbody>
</table>
Agile Testing

Abstract
Almost all the telecom service providers worldwide are facing tough times. Competition has increased and customers want more for less. Businesses are pressuring IT teams to change the way they develop and deliver software.
One solution that IT often tries, is to move from the Waterfall model to the Agile model. Unfortunately, few succeed in meeting the many challenges that they face. Many face resistance from independent software testing and quality assurance teams who believe that Agile development puts less emphasis on testing. They believe that it pressures testers to compromise on quality and release software to production even if it has only been partially tested, thus leading to poor quality which will have an adverse affect on the long term health of the organization and will nullify any short term gains that Agile brings.

This paper focuses on how Agile testing differs from the Waterfall model, and how you can successfully improve quality by moving to Agile testing. This paper will:
• Help to alleviate fears of many, including independent software test teams who believe that Agile compromises on quality.
• Help those who want to move towards Agile testing, but do not know how to start and what the benefits are.
• Those who tried to move towards Agile testing, but were not successful.

In brief, this paper will help CSPs ensure that their code is production-ready every 2 to 4 weeks, thereby greatly reducing the bottom-line, while at the same time, increasing the ability to respond to market needs faster than competition.

Introduction
In Agile, both testers and programmers are part of a single team. After each development cycle, tested software is released to internal/external customers without compromising on quality. The Agile development cycle is typically 2 to 4 weeks, and only the high priority features which can be developed in 2 to 4 weeks are included. Once these features are delivered, the features with the next highest priority are included in the next development cycle.

The Waterfall model has separate software development lifecycle phases, Requirements, Design, Coding and Testing. Software can not be released to the customer until the testing phase is complete. All the requirements are collected up front in as much detail as possible. This along with the other phases involved, increases the length of the development cycle to 4 to 8 months depending on project size, which is much longer than in the Agile model.

Agile helps shorten the development cycle from months to weeks, which helps businesses to improve their ROI and add the ability to cater to changing market needs. Agile has many benefits, but there are also many misconceptions.

The common misconception of the agile methodology is that it compromises on quality, as testers get less time for testing, they get influenced by developers, their voice/ concern are less heard, etc. Often this misconception is the result of incorrect implementation of Agile.
Another misconception is that Agile is just an iterative software development, which reduces the 4 to 8 months development cycle to 2 to 4 weeks and then squeezes all SDLC phases in that window. In doing so, this is simply an incorrect implementation of Agile, because testing is still done at a later stage similar to the waterfall model. This leaves less time or testing and bug fixes move to another iteration, where development fixes those problems in the next iteration. This leaves you with the same problems that you had in the Waterfall model because you cannot release the tested software to customer after each cycle. Reaching the stage where you can release code that is production-ready every 2 to 4 weeks is not easy and there are many challenges. This paper proposes an effective Agile testing methodology, which is designed to avoid common pitfalls and overcome the challenges.

Why Agile Testing

• **Faster time-to-market:** One of the biggest benefits of Agile testing is that it helps reduce the time to market relative to the Waterfall method. In a traditional waterfall model, once the development team completes coding, it takes about 3 to 6 months (depending on project size) to take this to production. In Agile both development and testing occur simultaneously. This allows you to eliminate the dedicated test cycle, thus reducing the cycle time by 25% or more.

• **Increases revenue and market share:** We know that customers are looking for new offerings and that service providers who are able to bring better offerings to market early are able to gain much larger market share. We also know that early adopters are ready to pay a premium, which increases revenue and profitability. Agile testing helps the service providers bring new and better offerings to market before its competitors can.

• **Reduces cost:** Fixing a defect closer to its introduction is considerably less costly than fixing it at a much later time. In Agile, testing and programming happens almost in tandem, which considerably lowers the cost. Agile testing not only helps in early detection of defects, but it also helps prevent them. An Agile tester has specific skills and tools that not only detect defects early, but enables them to prevent the defects altogether.

• **Improves quality:** Agile testing place a very high emphasis on quality and automated testing. It brings many tools which help in automation and ensure that every piece of code gets tested. It helps in reduce production defects and improve quality considerably.

• **Happy customers and happy employees:** It is difficult for customers (end users) to put their requirements into a clearly written document before seeing the actual software. Agile testers work with customers (end users) to define the requirements. They also work simultaneously with programmers to clarify those requirements. This helps both the customer and employees, and reduces the requirement gaps that we often find in Waterfall projects.
Common Challenges Faced by CSPs During the Transition to Agile Testing

After seeing the benefits of the Agile testing model, many of the CSPs fail to release code that is production-ready every 2 to 4 weeks. This is because during the transition, they fail to deal successfully with the challenges related to the transition to Agile. Some of these challenges are listed below.

1. **Resistance to the transition:** The testing team often fears that it will lose the power to reject a product that it had before the transition to Agile testing. Under Agile, the entire team of testers and developers decides if and when to reject a product. Furthermore, the organizational structure, management hierarchy and a professional development path under Agile is less clear than in a traditional testing environment.

2. **Inability to achieve Agility:** Often the testing begins only after the code is written and the programmers do not collaborate with the testers during development. This leads to testing of old features, while the programmers are already coding new features before the discovery of bugs to the previous version. This leads to delays in the release of the old version to production until the bugs are fixed.

3. **Poor test design due to inadequate documentation:** The testing team continues to use old methods of test design that based on requirement documents/high level solution documents/design documents that lack information on what to test. This results in a poor test design that impacts on quality.

4. **Existing features get broken, when new features are added:** Many times features which worked previously get broken, which leads to concerns about poor quality. This occurs because in Agile features are added incrementally and the testing team needs to test new features and perform regression testing to make sure that the existing features still work. Testing team find it difficult to complete regression testing in a short cycle that also includes the testing of new features. As the regression test suite increases after each cycle, there is a tendency for the testing team to take short cuts that affect quality.

5. **Frequent requirement changes that impact on the quality of the testing:** In the Agile environment the requirements change frequently. If this is not properly communicated to testing team, defects will go undetected and get released to production.

6. **Performance issues identified much late:** The testing team focuses only on functional testing and ignores performance and load testing. This leads to the discovery of performance and load problems after the product is released to production.

7. **Managing test environments becomes difficult:** In Agile, testers receive and deploy new code almost daily. Management of the tests in different test environments becomes very problematic, which leads to testers and developers spending considerable time to manage the test environment instead of finding and fixing defects and issues.
8. Development team not able to collaborate with the testing team to improve quality: The development team finds it difficult to write automated unit testing, which is one of the major building blocks in Agile testing. The development team finds it difficult to use Test Driven Development (TDD), and instead of incorporating these practices, they question why testers are needed with TDD and automation of all unit tests. This attitude increases the fears of the testers that they will lose their jobs if the development team starts to use TDD and automates unit testing.

9. Managing communication becomes a challenge: In Agile, not everything is written or goes through controlled change process. Often, changes are discussed in daily status meetings and approved there only. If you are geographically separated and in a different time-zone, it becomes difficult to manage communication. This creates ambiguity, communication gaps that leads to complaints of chaos.

Agile Testing Methodology

All of the challenges faced by CSPs are due to the difference between Agile testing and the waterfall model. It not only requires changes to the way the testing team works, but also requires changes throughout the entire organization, from management to business people and programmers.

Based on the extensive experience that Amdocs has with Agile testing, it has come up with an Agile Testing Methodology. The Agile Testing Methodology has 3 pillars and one core foundation.

Mind-Set

Agile testing is not a mini-waterfall and it requires a change in the mind-set to understand its value and to reap its true benefits:

1. Everyone is responsible to ensure quality: In Agile, the entire team of programmers and testers is responsible to ensure quality. A tester brings the customer’s perspective and plays a very important role in the team. It is important that programmers write automated unit test cases, while the testers break the features down into test requirements that will help the developers to design the product accordingly. Both collaborate to make sure that no defects reported are from internal/external customers.

2. Adopt an Agile testing mindset: The role of the Agile tester differs from role in a waterfall model, where tester serves as a gatekeeper for quality and makes sure that that delivery is bug-free. In Agile, the entire team shares a passion for quality.
The Agile tester constantly improves skills, learns new things, and is passionate about helping the team to make the business succeed. The Agile tester always looks for new and effective ways to test. He focuses on team goals and asks for assistance when necessary. The tester focuses on delivering value and is flexible in responding to change. The Agile tester believes that prevention is better than cure, and always works with programmers to avoid defects.

3. **Collaborate with customers:** The Agile tester serves as a bridge between customer and developers. He works with the customer to understand and prioritize the requirements. The tester illustrates the requirements with examples and then turns those examples into executable tests. The tester encourages direct communication between programmers and customers. When requirements are misunderstood, customers, programmers and testers work together.

4. **Look at the big picture:** In Agile, teams work on small features, which can cause programmers to forget how each feature under development fits into the grand scheme of the product. It is the job of Agile testers to keep in this mind. The tester must always envision what impact a feature can have on other parts of the system and identify issues well in advance.

### Skill-Set

Agile is a new way of developing and testing software. It requires different skill sets for people working in Agile teams. It is important to recognize this and help team members to acquire new skills that will help them succeed.

1. **Automation:** An Agile tester should have the ability to automate repetitive tasks. He should always look for ways to automate, and where needed, get help from the developers. In Agile you develop small features iteratively and therefore must make sure that existing features don’t break when adding new ones. It will not be possible to keep pace with regression testing if it is performed manually. Therefore, testers need the skills to automate routine tasks and not be afraid to learn new scripts and tools.

2. **Writing test requirements that assist programmers:** In the Agile development cycle you do not have detailed requirements. Testers write the requirements in a way that helps developers to design and code the product in a way that prevents defects. Testers share these test requirements with customers so that gaps in the original requirements can be filled immediately. Agile values constant feedback and communication in order to avoid surprises later on.

3. **Skills to do different type of testing:** An Agile tester should have the skills and knowledge of different types of testing for functional, exploratory, end to end, usability, load, and performance. It may be required that for certain business requirements we require specific expertise, therefore the tester must be willing to learn new areas quickly and be able to apply them as when needed in the future.
4. Effective communicator and team player: An Agile tester must have excellent communication skills and possess the ability to collaborate as a team player. The ability to take initiate and a helpful attitude is a must.

Tool-Set

In addition to the Agile mind-set and skill-set, you will require different types of tool-sets to help you to successfully implement Agile testing. There is no one magical tool that will solve all your problems. Based on your needs, team experience and continuous feedback, you can decide which tools work best for you.

1. Development and build tools: Agile testing takes a holistic view of quality. Tools which help improve the quality of code from the start are very important because they help programmers handover the code to testers frequently and without much overhead. The tools below are recommended:

a. Source Control: This is also known by other names, such as version control. There are various tools, both open source (CVS) or Vendor tools such as ClearCase. Source control tools help identify which sections of code changed and should be covered during testing. Make sure to also control the versions of the test automation scripts.

b. IDE: An integrated development environment (IDE) tool, generally used along with source control tools. Examples are Eclipse, Netbeans, and IntelliJ IDEA. IDE highlights errors while writing code, and helps you write error-free code faster.

c. Build tools: There are various build tools available such as Ant and Maven. These tools not only manage the build, but also provide easy ways to report and document build results, easily integrate them with build automation and test tools.

d. Continuous Integration (CI): This is a core practice for Agile teams. A fully automated and reproducible build that runs many times a day is a key success factor for Agile teams. Common tools are CruiseControl and Hudson. These will reduce considerable amounts of time that is spent managing the environment.

e. Code coverage tools: Code coverage describes the degree to which the source code of a program has been tested. This will give an indication of whether we are doing enough testing. There are various tools available, for example Cobertura for Java.

2. Tools for requirements and examples: Including requirements in a detailed MS Word document does not ensure that you will not have any requirement gaps. Agile addresses requirements in a much better way. Customers, programmers and testers use various tools to minimize this gap. The tools below can be used to help.

a. Wiki: In Agile, use a Wiki instead of long documents. The Wiki is a space on the web where you can share work and ideas, pictures and links, videos, etc. This facilitates communication across different teams in different time zones and locations.
b. **Checklist:** This has many uses. One way you can use this is to create a template for requirements to make sure that all aspects of a requirement dealt with. You can create Wiki templates for detailed descriptions, for design, etc.

c. **Mind map:** Mind Maps are diagram created to represent concepts, words, or ideas linked to central key concept. You can simply use the whiteboard or software. It is very powerful tool for summarizing key concepts all at once.

d. **Mock up:** This tool can be used to show a customer, how the UI will look before programmers start to write code. This provides early feedback from the customer. You can use a paper prototype or screen shots from an existing application. It does not need to be fancy or pretty, or to take a lot of time to create. It should be understandable to both the customer and the developer teams.

e. **Spread sheet and Flow diagram**

3. **Tools for test automation:** For Agile testing to be successful, it is very important that you automate repetitive tasks. There are various tools available which help with automation.

   a. **Unit level tools:** These tools are used by development to automate unit testing. Unit test tools are specific to language in which you do coding. For example, Junit for Java and Nunit for .Net. You can use TDD to further improve automation.

   b. **Behavior Driven Development (BDD):** This is an excellent tool for writing test specifications which can be read by customers and can be used by developers to automate testing. Some of the BDD tools are easyb and JBehave for the Java, and NBehave and NSpec for .Net.

   c. **API layer functional test tools:** Fit and FitNesse tools can be used for API layer functional test tools. These tools promote collaboration between different team members to come with the correct tests.

   d. **GUI automation tool:** QTP is one of the popular tool used for automated testing the GUI.

4. **Tools for improving communication between distributed teams:** In Agile, we work in a fast paced manner and lot of communication occurs between customer, programmers and testers to make sure that everyone is aware of progress and is well-informed. There is a variety of tools available to help improve communication, such as Screen sharing tools, Video Calls, Wiki and travel. You might already be using these tools or be aware of them. I highlighted them because even when we are familiar with them, we do not always use them to solve day-to-day problems.

**Management Support**

Any change management requires support from management. This holds true for Agile testing as well. It is the core foundation of successful Agile testing. Below is a list of ways that management can help adoption of Agile succeed.

1. **Start with a small, clear and compelling business need:** Start with a small project that has a clear and compelling business need in order to receive active involvement from the business people.

2. **Select the right people:** Put the right people on the Agile team as Agile testers. Select people who are proactive, eager to learn new things, have scripting knowledge, are willing to collaborate with programmers and customers, and passionate about improving quality as a team.
3. **Accept Failures**: Journey to Agile will not be smooth. There may be initial failures, but it is important that the team recognize this and make sure not to repeat the same mistakes twice. Management should make sure to build a culture where people feel comfortable to share the failures and learn from them.

4. **Provide budget and time**: Agile adoption will require training to enable teams to learn new skills, and may require the purchase of new tools. It is better to start with an Agile coach who can help the team during the starting phase and reduce risk.

**Agile Testing Methodology Helps CSPs to Overcome Many Challenges:**

(Amdocs) Agile testing methodology is a proven framework which will help CSPs to successfully deal with the challenges faced by them and ease their journey towards Agile Testing.

Many of the challenges which CSPs face requires work on more than two areas in addition to management support. For example, resistance in transition to Agile can be eliminated by Management support and a change in the mind-set. Management can show the growth path to testers, and create a collaborative culture between programmers and testing team. Educating testers about why this change is required from an organizational standpoint.

In the same way, if CSPs are not able to achieve agility, then testers may require new skill sets and specific tool sets. This will require management support for budgets for training and the purchase of new tools. In some situations, it may require all three areas (mind-set, skill-set and tool-set) to be used in parallel.

According to the Amdocs Agile methodology, the following steps should be followed to achieve true agility.

1. **First identify the challenges which prevent a transition to Agile testing.** Talk with team and the other stakeholders to prepare the list of challenges.

2. **Prioritize these challenges.** Which are the most important for the team and the other stakeholders. Select the top three challenges from the list. Never take more than three at a time.

3. **Use the Agile testing methodology to identify why you have these challenges.** Always involve the team in the identification process. For example, you might need a specific skill on how to automate regression test cases and a new tool for CI (Continuous Integration).
4. Once you identify where the gap exists, take an implementable action. Create a plan and start tracking it. Use the PDCA cycle (Plan, Do, Check, Act).

5. After you deal with the top three challenges, find the next three and repeat steps 1 to 4. Repeat this process until you eliminate all your challenges. The key to success is not to repeat the same mistake twice and to actively involve team members in the process.

Conclusion
In order to be successful, Agile testing requires a change in the mind-set, and a different skill-set and tool-set. It also requires management support, without which Agile testing will not succeed. Agile testing will reduce the total development cost, reduce risk and will help communication service providers to launch products and services much earlier to market.

About the Author
For further queries on this white paper, contact Avinash Garg, Project Manager, Amdocs Consulting Division, Amdocs. Avinash has over 16 years of experience in handling various telecom development projects and has helped organizations to become Agile and reap true benefits of Agile.
About Amdocs

Amdocs is the market leader in customer experience systems innovation. The company combines business and operational support systems, service delivery platforms, proven services, and deep industry expertise to enable service providers and their customers to do more in the connected world. Amdocs’ offerings help service providers explore new business models, differentiate through personalized customer experiences, and streamline operations. A global company with revenue of $3 billion in fiscal 2010, Amdocs has approximately 19,000 employees and serves customers in more than 60 countries worldwide. For more information, visit Amdocs at www.amdocs.com.

Amdocs has offices, development and support centers worldwide, including sites in:

**THE AMERICAS:**
- BRAZIL
- CANADA
- COSTA RICA
- MEXICO
- UNITED STATES

**ASIA PACIFIC:**
- AUSTRALIA
- CHINA
- INDIA
- JAPAN
- SINGAPORE
- THAILAND
- VIETNAM

**EUROPE, MIDDLE EAST & AFRICA:**
- CYPRUS
- CZECH REPUBLIC
- FRANCE
- GERMANY
- HUNGARY
- IRELAND
- ISRAEL
- ITALY
- THE NETHERLANDS
- POLAND
- RUSSIA
- SOUTH AFRICA
- SPAIN
- SWEDEN
- TURKEY
- UNITED KINGDOM

For the most up-to-date contact information for all Amdocs offices worldwide, please visit our website at www.amdocs.com/corporate.asp

---

Copyright © 2013 Amdocs. All Rights Reserved. Reproduction or distribution other than for intended purposes is prohibited, without the prior written consent of Amdocs. The trademarks and service marks of Amdocs, including the Amdocs mark and logo, Intentional Customer Experience, CES, Clarify, Ensemble, Enabler, Return on Relationship, Intelecable, Collabrent, IACCT, DST Innovos, Silbo Graphic Software, Ipsos, Chimes, Sightline, JacobsRimell, ChangingWorlds, jNetX, OpenMarket Inc., MX Telecom Inc., MX Telecom Ltd, Streamrcco, and Bridgeweaver Systems are the exclusive property of Amdocs, and may not be used without permission. All other marks are the property of their respective owners.