

Maximizing the value of good testing practice in an Agile environment

Delivering on time, in scope, on budget and at the right level of quality



Agile is being adopted to speed up software development

In sequential software development lifecycles such as the V-model, the emphasis has traditionally been on defining, reviewing and subsequently validating the initial business requirements in order to produce a full set of high-quality requirements. Further higher levels of testing, such as User Acceptance and System 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 reviews and lower levels of developer testing, such as component and component integration testing, can achieve very high levels of software quality.

However, in reality, traditional software development approaches have a poor track record when it comes to delivering working software that meets business requirements, as well as on time and within budget. As industry analysts have discovered less than 50% of users are satisfied with quality and just over 33% are satisfied with the speed of software development.

Projects invariably fail because of lack of end-user involvement, poor requirements definition, unrealistic schedules, lack of change management, lack of testing, and inflexible processes.

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

The agile software development approach, based on iterative development in which requirements and solution evolve in combination, would therefore appear to offer a potential solution to these problems.

Incremental software development processes, such as XP and SCRUM, 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 poses a challenge for testing

However, while agile is indeed a new way of looking at a traditionally linear development process, it throws up its own distinct challenges: Should agile be adopted wholesale? Is it an appropriate approach to adopt for all situations? Or is a partial implementation or a mix of agile with traditional plan-driven approaches – in other words a hybrid solution - more suitable or indeed workable?

While the question of whether to adopt agile is largely context-specific and a feature of an organization's process culture, this paper address the issues specifically relating to testing in an agile environment.

In agile processes, the lack of detailed requirements and the fast pace of delivery pose a significant challenge to the traditional view of professional testing. As the Dilbert cartoon on agile programming ironically declares: "No more planning; No more documentation."¹ How can 'complete' testing be carried out without detailed requirements and plans? How much test documentation is 'sufficient'?

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 or vice versa! And how can the effectiveness of automated tests be measured? What about non-functional tests; the evaluation of quality characteristics such as performance, volume, reliability, usability scalability, memory usage, 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 multi-nationals. Indeed, do testers have the right skills or are they able to add value in this seemingly unstructured environment?

So, 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?

¹ Dilbert cartoon: Scott Adams Inc.

Maximize the value of good testing practice in Agile

At its heart, agile is all about quality and focusing on the customer, the very key characteristics that are the drivers for the traditional and professional tester. Capgemini Group therefore believes that testing has a significant role to play within an agile development environment and can actually strengthen the output of the process.

To maximize the value of the testing discipline, Capgemini suggests that:

- A Test-driven Development approach should be adopted as the most effective single practice to improve quality and eliminate duplication;
- Test strategy and planning should be carried out but they need to be flexible and adaptable to the new information produced at each iteration;
- Similarly the role of the tester has to adapt, but it becomes richer and more influential within the team process in which developers, users and testers each bring their particular expertise;
- Test automation, reducing the inefficient repetition of manual tasks, supports the fast and flexible agile principle.

This paper explores these assertions, based on our own practical experience of agile assignments.

Test-driven Development; putting tests before coding to ensure quality and productivity

The concept of Test-driven Development (TDD) is a software development technique that uses short development iterations, based on pre-written test cases, and integrates both testing and software design methodologies. In light of this integration, Capgemini believes it is probably the most effective single practice to improve quality and eliminate duplication.

TDD defines that one should: write the test; write the code and then refactor. Borrowing from early preventative testing principles, a key part of TDD, according to Kent Beck, involves: "Never write a single line of code unless you have a failing automated test".

The addition of 'automated' to well-known early testing practices means that developers are more likely to carry out unit/component testing because testing in this context involves writing code, which developers like and see as part of their job! Also these tests can be used for regression purposes, particularly as regression testing becomes more important as the iterations become more frequent.

This helps maintain stability particularly with high levels of change. The team will considerably benefit from automation when it comes to deploying software to 'live', as it not only reduces time, but instills confidence in the quality.

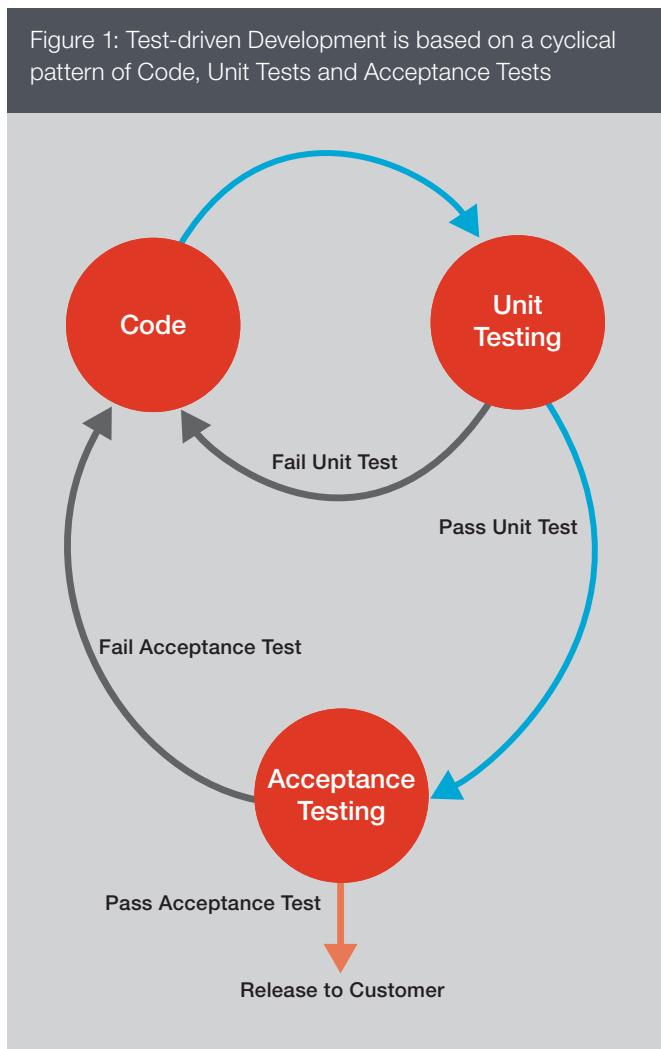
Test-driven Development also means that testing takes on a 'specification' role. In the absence of detailed requirements, test cases define the required behaviour of the system. In other words, a feature is not specified, until its acceptance test has been written, and a feature is not done until all of the acceptance tests have passed. In addition, reviewing the test cases directly with the business is vital in order to avoid possible rejections of the system when the business does get involved.

In this way, unit and acceptance tests become key requirements/features and design artifacts as is illustrated in Figure 1:

- Unit: Drives design-executable design specifications; does the code do what the developer intended?
- Acceptance: Defines completion-executable requirements; does the system do what the customer requires?



Figure 1: Test-driven Development is based on a cyclical pattern of Code, Unit Tests and Acceptance Tests



Test cases by their very nature are specific, can add detail and reduce the ambiguity of requirements. In a recent assignment, the requirements of a client were kept vague for two key reasons. First, not all of the requirements could be fleshed out initially. Second, the client did not want to waste time on requirements that they felt would be wrong no matter how long they spend working on them up-front. The development of the test cases, involving the users, allowed the requirements to be incrementally clarified. The test cases provided very clear scenarios, including input data and expected results, which helped the users to understand the requirements.

Generally, Capgemini found from practical experience that using TDD has in many cases been very positive for the production of high quality software in highly iterative and incremental environments. This is particularly the case when combined with:

- Continuous build and integration, where automated unit/ component tests are run daily on integrated builds; and
- Coverage measurement – as one of the ‘done’ criteria - to achieve high degrees of structural coverage of code.

TDD is not without its own issues, relating to educating developers as to what constitutes good unit tests. We have come across many examples of automated unit tests that simply check for the existence of code rather than testing what the code does.

Similarly, some developers write unit tests that just focus on checking the main (simple) positive path – known as ‘happy path’ testing – and so results in incomplete testing. But on balance, TDD is undoubtedly a powerful practice that ensures good testing practice is used.

Developing a flexible test strategy remains important

While TDD is of undoubted benefit in agile, it is not a replacement for thinking through an appropriate testing strategy. We have seen that, as practical experience in implementing agile matures, there is an evolving trend in the use of hybrid approaches that combine both elements of agile with more traditional development methodologies.

In development, this can include the use of traditional plan-driven approaches for longer term estimation and planning, which are particularly useful in fixed-price projects. These can then be combined with agile’s highly iterative and incremental approach.

The fact that a significant number of organizations implement elements of agile and combine this a la carte selection with components of traditional development means that a test strategy still needs to be considered and defined.

The learning point is that defining an adaptive risk-driven testing strategy to ensure efficient and effective testing is just as important in agile as it is in traditional sequential lifecycles, but must be adapted to this more flexible and fluid environment.

It is just not sufficient to say that unit/component and acceptance testing should be carried out, without considering if this is an appropriate strategy to address the risks identified. But an over-reliance on detailed plans and heavy-handed change control will not work either.

System risk is a function of requirements, complexity and technology, amongst other factors, and so the appropriate test strategy has not only to adapt to this, but also needs to be cognizant of the particular methods and processes used to develop the system and how effective they are.

Reducing a test strategy to the use of two predefined levels of testing is often an over-simplification. Experience shows that as a result, risks are frequently insufficiently identified or addressed. For example, agile teams rarely articulate non-functional quality criteria such as reliability, usability, performance, scalability and memory usage in their user stories or test cases.

Consequently, teams seldom have tests designed that provide sufficient information about these attributes. Partly because agile is a developer-centric method, and professional testers have been left a little behind in the early waves of agile, many organizations have therefore failed to implement appropriate testing strategies shaped by risk management. In our opinion, this should be addressed.

A testing strategy is an essential element of smooth performance and project management; agile projects are not immune to these – in fact agile relies heavily on a disciplined (project management) process. A testing strategy, for example based on TMap®, provides the business rationale and context and ensures most importantly that the development is aligned with the business needs and wider enterprise goals.

Role of the tester has increasing influence

Initially when agile was focused on unit and acceptance testing, it seemed that the system tester did not have a role, and that testing could be carried out as part of other positions or functions. This was exacerbated by the traditionally independent perspective of system testing and the significant cultural change that it was thought would need to be made by professional testers in order to become fully integrated agile team members.

But as the adoption of agile matures, so too is the role of the tester now receiving increasing attention, and a growing recognition of the value of the tester's contribution to the overall process.



Cem Kaner² indeed points to a more positive scenario: "The nature of the tester's role changes in iterative projects. We are no longer the high-profile victims, we are no longer the lonely advocates of quality, we are merely [sic] competent service providers, collaborating with a group that wants to achieve high quality."

So for testers this is good news as their role potentially becomes richer and more influential, in a team process in which developers, users and testers each bring their particular expertise whether it is technology or domain knowledge.

But some testers will have to adapt to this new context. Traditional 'independent' functional testers are at a disadvantage as they may not be able to add as much value in an agile environment as they would in waterfall or V Model development. Agile, in the true sense of the word, means fast-paced and rapidly changing and this means that the roles and responsibilities of the team are also likely to flex and change quickly.

Testers need to be technically aware of what they are testing and understand the impact on automation as well as the functional implications. Some iterations may be development heavy, some automation heavy, some test heavy, and the agile tester needs to be adding value in all three instances.

Our experience is that an effective agile tester must understand how to write and interpret code, how to write automated test scripts and analyze their results, as well as understand how to functionally test a component to an acceptable level of coverage.

Testing in this context then is a much more value-adding role, albeit a different one, and one that requires a tester to be involved in the development project continually, and right from the start. In other words, the test team should be integrated into a composite group that includes both developers and business analysts, sharing responsibilities.

In this position, their role becomes more comprehensive as they are in a position to influence more widely and at an earlier stage than in a traditional software development lifecycle. This can only be of benefit to the project, and the ability to meet the business goals.

The typical responsibilities of a tester in this more influential agile role include:

- Facilitating communication between technical and business stakeholders, providing continuous feedback progress reports and decision support, for example in defining acceptance criteria;
- Supporting early validation of requirements;
- Creating automated acceptance tests and expanding their scope;
- Performing manual/exploratory tests on early-stage code;
- Writing and interpreting code to help fix defects etc;
- Estimating, planning and advising the team of overall risks and trends;
- Ensuring that best testing practice – for example test design techniques and testing coverage metrics – are used.



² Cem Kaner is Professor of Software Engineering, Department of Computer Sciences, Florida Institute of Technology, USA.

Our experience shows that the perfect agile tester is someone who has a software development background, but has transitioned into a testing role and has built up a wealth of experience - or a traditional tester with strong development skills. Experience and breadth of skills are both essential.

Technical know-how is not enough. But with these skills of dedicated testers come real benefits, as Bret Pettichord³ outlines: a focus on customer usage over technical implementation and a focus on uncovering flaws over confirming completeness.

Professional testers therefore can adapt to fit this enlarged role and so provide additional value by not only focusing on finding defects but also playing a team and liaison role with the key stakeholders throughout each iteration.

Increased focus on automation tooling

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

If agile is about turning the “knobs to 10”⁴, then the role of automation in agile is to remove the mechanical, routine tasks. Due to the 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.

Tasks that can normally be automated within agile teams include:

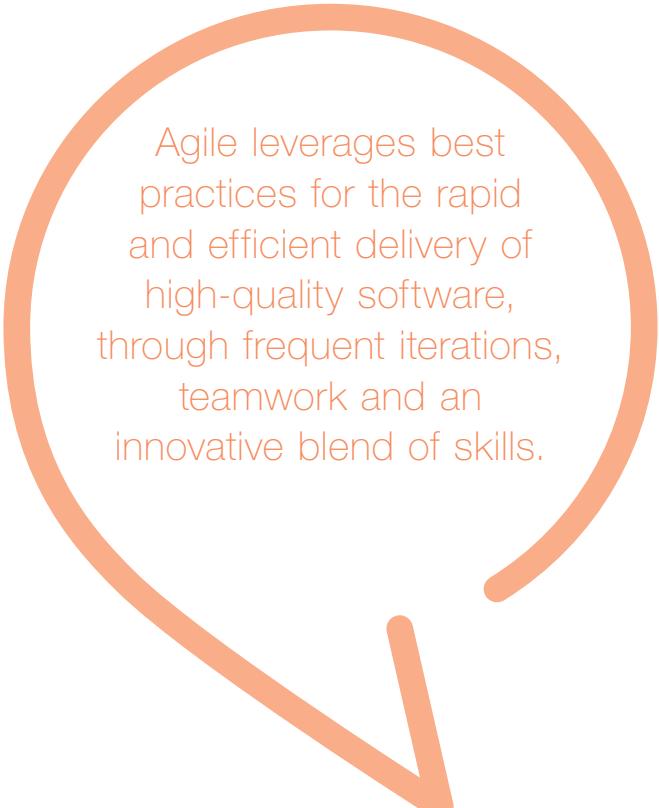
- Build and integration process: Usually in agile teams, this process happens on a very regular basis (every night), resulting in a new build every day, with almost zero manual effort being put toward this task. This requires good configuration management and build tools;
- Unit Test: These are part 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 use of static analysis tools review the code against coding standards and 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;

- Functional Testing: Previously, 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; execution should be automated.

Conclusion

Agile leverages best practices for the rapid and efficient delivery of high-quality software, through frequent iterations, teamwork and an innovative blend of skills. It therefore requires a particular mindset, one based on adaptability and flexibility, in comparison to traditional plan-driven approaches.

Testing is therefore adapting the traditional principles of a hierarchical test strategy and applying them with rigor and discipline, but without the heavy hand of documentation. And professional testers are rising to this challenge of more closely collaborating with developers, analysts and other business stakeholders, operating as fully-integrated team members. In the search for faster software development, this can only be of benefit to organizations that need to keep one step ahead.



Agile leverages best practices for the rapid and efficient delivery of high-quality software, through frequent iterations, teamwork and an innovative blend of skills.

3 Bret Pettichord is a software testing expert and an influential author and speaker. www.bret.pettichord.com

4 Kent Beck, the creator of Extreme Programming and one of the 17 original signatories of the Agile Manifesto in 2001

About Capgemini and Sogeti

With almost 140,000 people in over 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2013 global revenues of EUR 10.1 billion. Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Sogeti is a leading provider of technology and software testing, specializing in Application, Infrastructure and Engineering Services. Sogeti offers cutting-edge solutions around Testing, Business Intelligence & Analytics, Mobile, Cloud and Cyber Security. Sogeti brings together more than 20,000 professionals in 15 countries and has a strong local presence in over 100 locations in Europe, USA and India. Sogeti is a wholly-owned subsidiary of Cap Gemini S.A., listed on the Paris Stock Exchange.

Together Capgemini and Sogeti have developed innovative, business-driven quality assurance (QA) and Testing services, combining best-in-class testing methodologies (TMap® and TPI®) to help organizations achieve their testing and QA goals. The Capgemini Group has created one of the largest dedicated testing practices in the world, with over 12,300 test professionals and a further 14,500 application specialists with Testing experience, notably through a common centers of excellence with testing specialists developed in India and elsewhere.

For more information, please visit:

www.capgemini.com/testing
www.sogeti.com/testing

Contact

For more information about how Capgemini and Sogeti's Testing Services can help organizations achieve their testing and QA goals, please contact your local Capgemini or Sogeti account manager or our Global Testing Services Team:

Mark Buenen
Global Service Line Testing
Vice President
mark.buenen@sogeti.com