



testing solutions group

# TSG Quick Reference Guide to Agile Development & Testing



Testing Solutions Group Ltd  
5th Floor, 117-119 Houndsditch  
London EC3A 7BT

Tel: +44 (0)20 7469 1500

© Testing Solutions Group, 2008/9

[www.testing-solutions.com](http://www.testing-solutions.com)  
[www.tsg.learntesting.com](http://www.tsg.learntesting.com)

## What is Agile Development?

**There are various opinions on what defines ‘agile development’, but most would agree that the following describe a set of basic principles.**

- Incremental development - delivering a number of (useful, usable) deliveries;
- Iterative development - allowing requirements to evolve (change and be added to) as the project develops;
- People-oriented development - relying on the quality of developers and testers rather than well-defined processes; allowing the agile team to manage themselves; expecting daily interaction between developers and the business;
- Technical and engineering excellence – achieved through disciplined approaches to the development, integration and test of products.

Naturally, agile projects in the real world do not all necessarily conform to the ‘ideal’, but most would incorporate the above principles.



## Manifesto for Agile Software Development

The manifesto below describes the non-profit Agile Alliance’s view of agile as an alternative to traditional, process-led, highly-documented methodologies ([www.agilemanifesto.org](http://www.agilemanifesto.org)).

*“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*

*That is, while there is value in the items on the right, we value the items on the left more.”*

As it says, and is seen on agile projects, the items on the right are still valuable. For instance, we still need documentation, but not to excess and we generally find that agile projects are highly-dependent on a fair degree of automation.

## Scrum Intro

Scrum is by far the most popular approach to managing agile projects; it was initially used in 1993, and the first book on it was published in 2001. Scrum is not an anagram – it comes from sport (either rugby or American football) – and refers to the idea of people putting their heads together to decide on their next action (teams working with Scrum have a short daily scrum meeting to communicate progress).

Scrum is not officially a development methodology (i.e. it does not tell you what method to use for describing requirements, or how to write code) and is better described as a project management framework within which an agile methodology is applied by the developers (XP is the favourite although many organizations create their own hybrid methodology).



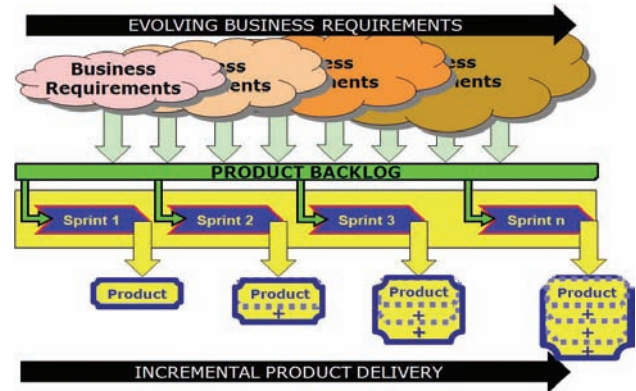
## Scrum Sprints

A Scrum project comprises a number of sprints, with each sprint resulting in new functionality that can be delivered to the customer.

New functionality may build on previously-delivered functionality or introduce new features.

Each sprint will typically last between one and four weeks and the number of sprints needed to complete the project will often be unknown at the start. This is because on typical agile projects the requirements are not fully understood (let alone specified) at the beginning of the project – they are allowed to evolve – like in real life.

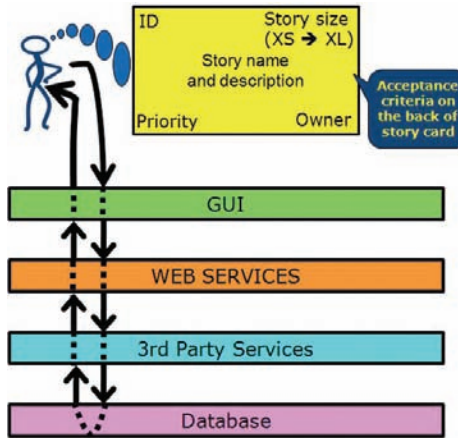
Customer requirements are collected in the 'Product Backlog'; at a high level these may be specified as use cases. The high level requirements (in whatever form) are broken down into 'stories'.



## Story Cards

Stories are often written on A5 cards, which can then be moved across wall charts to show progress with the story; their small size forces concise stories consisting (at a minimum) of:

- the business requirement;
- how it may be tested;
- an estimate of its size; and
- its priority.



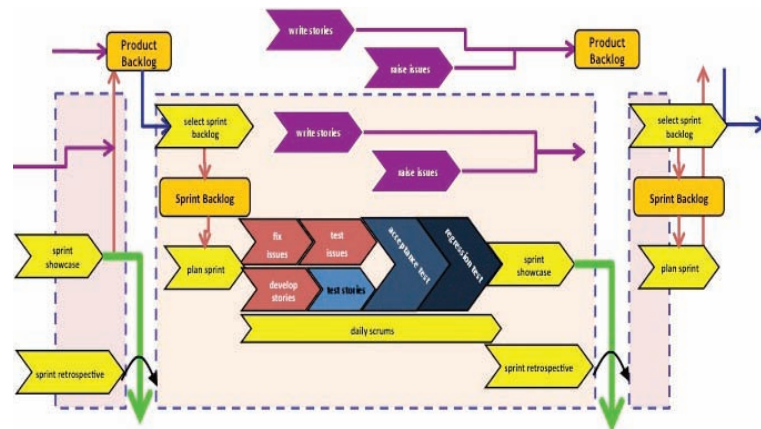
As a story describes business requirements from the user's perspective it is typically an end-to-end scenario, which can require implementation across a number of system layers in a single sprint (see above).

## Inside the Scrum Sprint

At the start of the sprint the Scrum 'Players' (see later) agree which stories from the 'Product Backlog' should be implemented in this sprint. The selected stories make up the 'Sprint Backlog'.

Next, planning for the sprint is carried out (yes, you plan in agile!), to decide both the scheduling of development and test activities (e.g. which stories will be implemented first) and team members' roles and responsibilities.

Agile follows the same generic process used across all software development and testing as can be seen in the representation of a Scrum sprint below:



The stories are coded and tested (ideally using test-driven development), and immediately integrated into a new system build, whereupon automated regression testing ensures the new code does not adversely affect any of the rest of the system. New functionality is next tested against the story's original acceptance criteria, before, where appropriate, acceptance testing by the users is performed. Towards the end of the sprint, before delivery to the user, a final regression test is carried out to ensure all the newly implemented stories work together and with the rest of the system.

The deliverable from the sprint is demonstrated to the end users at the 'sprint showcase' meeting, where the team have the opportunity to show the users (and management) what they have created.

The final activity carried out by the sprint team is the 'sprint retrospective', in which the team review how well they performed in this sprint and identify where improvements will be made in future sprints – thus process improvement is built-in to the Scrum framework.

Throughout the sprint, stand-up scrum meetings are held at the start of each day to ensure everyone on the team knows what everyone else is doing, and for the Scrum Master to determine what 'impediments' need to be removed to allow the



team to progress most efficiently. Impediments may be distractions arising from previous projects, the lack of suitable tool support, etc. These meetings are carefully managed and kept short – no longer than 15 minutes.

The stories are often created within the sprint by a team member in the role of a business analyst - the completed stories are deposited in the product backlog to be considered for implementation in future sprints.

Issues with the system are raised both by the sprint team and by external stakeholders, such as customers and users and may include bug fixes and enhancements. These can be treated similarly to stories and are entered into the product backlog so that they can be selected (or not) for inclusion in the next sprint. When an issue is selected for handling in a sprint then it appears in the sprint backlog and the issue is fixed and tested in much the same way as a story.

## The Scrum 'Players'

- Sprint Team – performs the sprint, ideally cross-functional (everyone can do everything), but in reality it's more likely to be a mix of developers, testers and, sometimes business analysts.
- Scrum Master - who protects his team from interference by management and others.
- Product Owner – the representative of the business.

## Scrum Deliverables

The deliverable from each sprint:

- is agreed between the customer and the sprint team at the start of that sprint;
- generally comprises the highest priority stories currently in the 'Product Backlog', e.g. the functionality that the customer wants as soon as possible.
- must be an achievable set of work that is good enough to give to the users and that the team believe they can implement in the sprint.

If there are unfinished stories at the end of a sprint they are put back into the product backlog. Normally these are picked up and completed in the next sprint, although if project priorities have changed during the course of the sprint or there is a technology-based reason for not doing them they may be left in the product backlog.

## Scrum & Testing

Typical practices used by the sprint team in the area of testing are:

- Test-driven development (TDD), where the code is written to pass tests that were written in advance.
  - the tests are based on the stories;
  - generally uses automated unit test tools;
  - TDD is sometimes seen as a form of programming.

- Automated builds and continuous integration.
  - the system is continuously updated and regression-tested as code is booked-in;
  - ensures speedy identification and correction of integration and regression issues.
- System testing (both functional and non-functional) against both the user stories and high level requirements.
- Acceptance testing (which may include organizing the end users).
- Regression testing, to ensure any last minute changes have no adverse side-effects.

Within a truly cross-functional sprint team all testing tasks could be performed by any team member, but in most projects each task is performed by a particular role (e.g. TDD by developer, final regression testing by tester, story testing by BA).

## Testing within Sprints

In an 'ideal' sprint the deliverable is ready for use by the users, which means all the aforementioned testing is performed within the sprint (as shown previously). At the end of a sprint, when a story is "done" that means it is built, tested and acceptable for use in production.

Many projects find this difficult, which leads to two alternatives; testing being performed as a parallel, but offset activity, or testing being handled in an occasional special testing sprint.

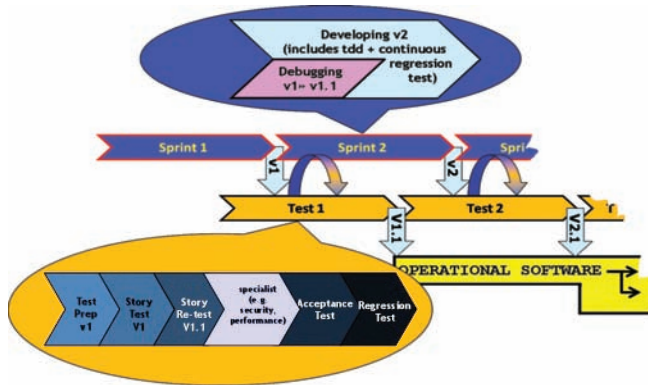


## Parallel Test 'Sprints'

In the parallel test 'sprint' the testers first prepare and then test the output from the last development sprint before sending fault reports back to the development team, who are now working on their next sprint.

Thus, the developers have two distinct tasks to perform in their sprint – they will develop their next deliverable, but they will also have to debug the faults found by the testers in their previous deliverable. They end up having to work on two versions at the same time – so configuration control and managing their development/test environments will be more difficult.

Debugged code is returned from the development team, re-tested and finally the system is acceptance and regression tested before shipping to the users. With this parallel and offset 'test sprint' you



can see that the time between the business user requesting the implementation of a story and it being delivered to them is extended to considerably longer than the simple sprint length. "Done" at the end of the development sprint no longer means "done, tested and acceptable for production".

## Occasional Test Sprints

An alternative to running parallel test sprints is to decide that after a number of 'normal' development sprints the sprint team will perform a 'test sprint' on the deliverables produced in the preceding development sprints. In a cross-functional team, the whole team may be involved both in the development and in the test sprints. The most obvious disadvantage of this approach is that the response time between accepting a user story for development and delivery back to the user can now be quite long. "Done" at the end of the development sprint no longer means "done, tested and acceptable for





## Agile – Key Points

### Key People:

- customer (product owner)
- scrum master
- team – cross-functional – developer, analyst and test skills

### Key Processes:

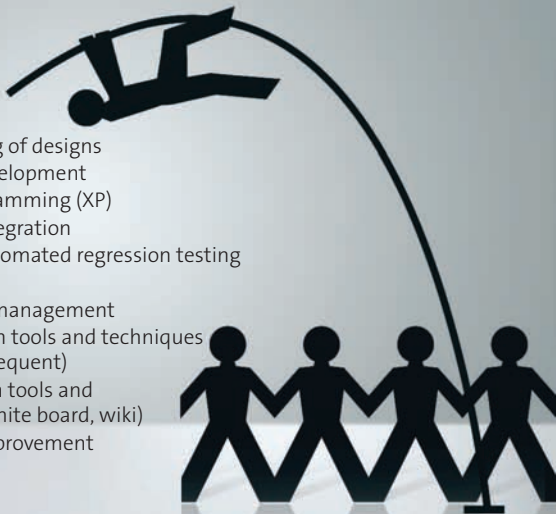
- sprint planning meeting with sprint backlog prioritization
- daily scrum meeting with daily priorities and problem solving
- sprint retrospective with continuous improvement ideas

### Key Techniques:

- self-managing teams
- story cards
- white-boarding of designs
- test-driven development
- eXtreme programming (XP)
- continuous integration
- continuous automated regression testing
- story testing
- configuration management
- communication tools and techniques (face to face, frequent)
- documentation tools and techniques (white board, wiki)
- continuous improvement

### Key Words:

- trust, honesty, feedback, customer focus, embracing change.



For details on how TSG can add value to your organisation, contact Bernard Melson on [bmelson@testing-solutions.com](mailto:bmelson@testing-solutions.com) or call +44 (0)20 7469 1500.

Testing Solutions Group Ltd  
5th Floor, 117-119 Houndsditch  
London EC3A 7BT.  
Tel: +44 (0)20 7469 1500.  
[www.testing-solutions.com](http://www.testing-solutions.com)  
[www.tsg.learntesting.com](http://www.tsg.learntesting.com)