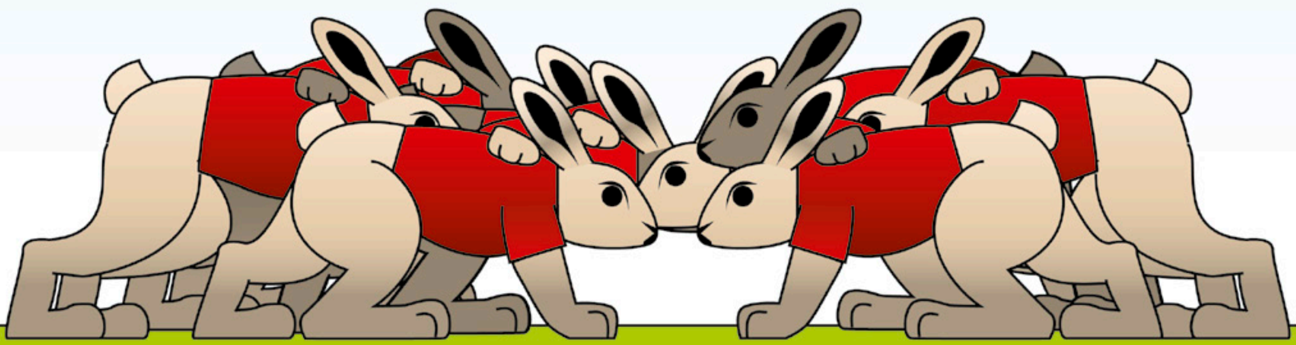

Delivering quality software with Agile

To deliver high-quality products to customers, Agile teams and product owners have to jointly ensure the quality of the requirements and manage them effectively. Agile quality practices have to be ingrained in the team's daily work. Ben Linders describes how it can be done.

Ben Linders

Quality software starts with quality requirements. When using Agile, this means that at the start of a sprint you need the stakeholders and the development team to agree on the products and the specific functionality that have to be delivered to your customers. It's important to have everyone involved committed to the same requirements: that ensures you're developing a product that your customers need and are willing to pay for.

You don't need to have buy-in on everything when you start a project. That's unfeasible, too expensive and too time-consuming. This is where the Agile method can be of great help. Agile teams use a product backlog to manage their requirements. Product owners prioritize the user stories based on the expected value they will deliver to customers.

To enable teams to start developing a product, make sure the stakeholders agree on the priorities. What needs to be delivered first, what do we need now, what should be in the upcoming iteration? That should provide sufficient commitment to warrant investing time and money, and for the team to start developing a first releasable product.

Stability

The manifesto for Agile software development prefers 'responding to change over following a plan'. But to develop software, you want to have stable requirements. The solution in Agile projects is to fix the requirements during an iteration but make them flexible over the lifetime of the project.

Agile teams treat requirements as stable during an iteration. When a user story is added to an iteration, the assumption is that there is a real need for software that fulfils the requirement. Time and money are invested at each iteration, so every decision to add a user story to a team's iteration backlog is actually a decision to invest.

The aim of requirements stability is not to prevent changes from happening. They will happen. Discouraging them or (even worse) ignoring them is no solution. What matters is that Agile teams are sufficiently able to deal with changes to the requirements and can maintain stability during product development, so they can deliver working software.

If there's a risk that a requirement underlying a user story might change on short notice, then it may be better to select another high-priority user story for the upcoming iteration. It's good practice to have user stories ready for the next two to three iterations, so that it's possible to switch user stories during the planning game. This also helps when a team is able to complete all the user stories before the end of an iteration; at that time they can discuss with the product owner which high-priority user story to add to the current iteration.

In their book Commitment, Olav Maassen and Chris Matts suggest that you never commit early unless you know why. I think this is a good approach to dealing with changing requirements. Commitment and stability support each other. By only committing to what needs to be done in the next iteration, you increase the stability of your requirements, which raises product quality.

Clarity

Another Agile quality practice is to have user stories of sufficient quality available at the start of a iteration – in other words, user stories that are 'ready'. Teams can use a Definition of Ready (DOR) to check the quality of the user stories. A DOR states the criteria that a user story should meet to be accepted into an iteration.

Similar to the Definition of Done (DOD), teams need to define their own DOR in discussion with their stakeholders, and adjust it when they see a need to do so. The DOR can include a checklist for user story quality. Many teams use Bill Wake's INVEST principle, which states that good user stories should be independent, negotiable, valuable, estimable, small and testable.

For a user story to be ready, there should be no unanswered questions. It should be clear for the development team what the product needs to do. Quality criteria such as throughput or capacity, reaction speed and system availability can be specified using acceptance criteria.

If it's impossible to remove all uncertainty about an important requirement, there are Agile practices you can use to clarify a requirement during an iteration. One solution is to use a spike story, a practice from Extreme Programming, to research a requirement or to investigate the feasibility of a technical solution. This can help you to drive out risk and uncertainty as early as possible.

Having several prioritized, sufficient-quality user stories ready at the start of an iteration helps to increase commitment from the stakeholders and the development team. It also lowers the chance that requirements will change during the iteration, which in turn raises product quality.

Whole team

Software reviews and inspections can also significantly improve product quality. These practices existed long before Agile was defined, but are still valuable for Agile teams. Coding guidelines can be used to spot potential code quality issues. Refactoring pieces of the software will help to improve code quality and reduce technical debt.

Team members can work in pairs to increase the quality of the software while they're writing it. They can take turns at the keyboard and switch roles to remain sharp and spot problems and opportunities to improve the code and reduce technical debt. Pair working also makes it possible for professionals to learn new skills or further develop existing ones. If you're very deep into something, chances are you'll start overlooking things. This is where pair working can help you; four eyes can see more than two.

When there's a major quality issue, swarming can be used to address it effectively and quickly. In this Agile practice, the whole team focuses on a single issue and together, the team members do whatever is required to solve it.

Multidisciplinary Agile teams should have all the skills and knowledge they need to deal with quality issues effectively. They have to know how to communicate and collaborate using the skills that individual team members possess to get the job done.

Ben Linders (info@benlinders.com) is an independent consultant in Agile, Lean, quality and continuous improvement, based in the Netherlands.

Edited by Nieke Roos

LDRA
Software Technology

Security for IoT Embedded Devices

Identify and eliminate security vulnerabilities

Verify security requirements

Ensure security requirement traceability

Visit ldra.com to start your free trial!

LOGIC TECHNOLOGY
HIGH TECH SOFTWARE CLUSTER

LDRA: Ensuring Software Security through Analysis and Test

www.logic.nl