(Agile) development needs (agile) testing and one-team vision

10 tips for implementing agile development with quality

Along the (not so long) story of software engineering, we have experienced a continuous transformation for improvement, in terms of more industrialized, faster and higher quality development. We keep on continuously improving the way we develop software, as an intrinsic part of business and society, pushed by a changing context in the way we consume and access software. “Change is the only constant in life”. We need to respond to two main aims: Firstly, we need to deliver software faster. Secondly, we need to improve quality in an changing, challenging and more complex world, because software defects and low quality directly impact the business (reputation, business processes behavior and associated revenue, user experience, social consequences…).

In this context, agile is a development approach that inspires a transformation wave in almost all organizations, in conjunction with the DevOps vision. However, agile does not mean (only) going faster by doing (more or less) the same. We change because we want to work better. And if this is the aim, no agile implementation can be really agile unless (1) agile testing and continuous quality assurance are considered and applied as key activities of software development, and (2) teams become really one-team mind-shaped, with diverse profiles (developers, business, architecture experts, UX, testers,…) aimed at working together. And this usually requires cultural changes, and a plan to re-skill technical capabilities and cross-wise abilities.

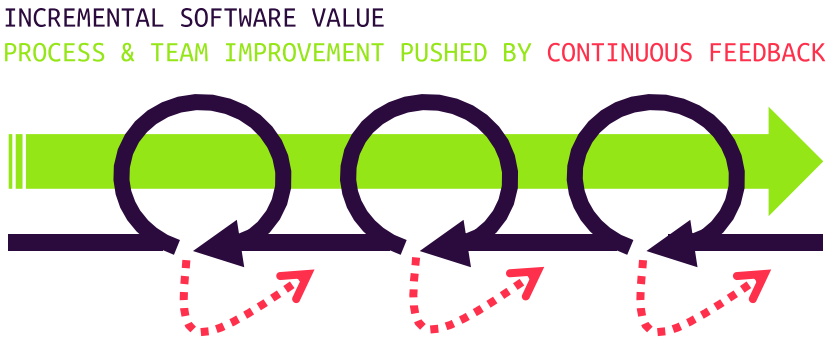
In this post, let me explain you my top 10 tips for implementing agile development with focus on delivering software continuously with quality:

1. Understand and spread agile mindset

Agile development, in a nutshell, means continuous incremental development in iterations with continuous feedback for improving next iterations, based on a one-team vision aimed at working on a same objective: continuously delivering software with enough speed and quality/value. Two common characteristics of working teams aimed at implementing agile approaches are:

- **Incremental & iterative:** Structure development objectives in short cycles that finish with working software and feedback.

- **Cooperative & adaptative:** Emphasize collaboration and communication and respond quickly and as early as possible to everyday changes.



“Without enough continuous quality feedback, no agile development iterations exist”.

2. Setting up agile teams with one-team vision

Continuous approvals between departments as a fragmented responsibility chain based on an organization with professionals belonging to field departments is not agile. The product is the center in agile, and the agile team is responsible for incrementally improving the product at any time with quality, by following a one-team vision. For a rich one-team vision, diverse people need to collaborate from the beginning to the end, applying the most of pairing tasks and cocreation. Because the mix of different points of view and expertise working together is the essence of agile.

When creating an agile team, we need to take into account:

- **Set up a diverse team.** Typically (taking SCRUM as a popular example) a Product Owner for leading the product, and a SCRUM master for leading the process and the team improvement are designated. Then, mix valuable people: business, developers, architect, UX expert, and for sure, tester and quality assurance roles. Each member is part of a valuable chain, even if they are not 100% assigned to an agile team.

- **State, at least, five common objectives/responsibilities:** (1) Incrementally improve the product, (2) deliver valuable software which means software with quality, (3) incrementally improve how we work, (4) individually and commonly be up-to-date and trained regarding modern technologies, and (5) pair tasks and point of views.

3. Push for continuous quality as a shared team objective

Continuous quality assurance is the insurance but also the promoter for achieving valuable software. It is about testing in different levels, but also about disambiguate and make requirements testable, about code analysis, about quality measurement, about expectations management, about user experience… The value of the delivered software can rapidly decrease and impact business if quality is not a stablished shared objective of agile teams. This motivates sometimes the necessity of a "Quality Lead", for promoting quality and technical excellence, for measuring quality status and for monitoring how quality evolves, as critical feedback iteration by iteration.

A typical question is who tests in agile, and where to test. Since there are different sources of potential defects, we need to test at different levels. Defects may be caused when programmers write code contributions in isolation, when this code contributions are merged with others, when user stories are not clearly understood, when integrating different products that are part of the same end-to-end business processes, when putting new versions into production environments, etc. Defects may also be of different types: functional, performance, security, usability... Therefore, testing activities need to be done by different professionals in agile. For example, developers should write their own unit test cases and analyze its code as shift-left quality assurance techniques and some integration test cases, while other integration and acceptance functional tests should be usually performed by testing roles, which need to be part of the user stories writing from the beginning. Some types of testing (performance, security, mobile, etc…) may require, depending on the project, specialized testing professionals to be part of different agile teams. Furthermore, we have testing aimed at validating new functionalities, and testing aimed at validating that new changes do not break previously validated functions (regression testing). And finally, test cases may be executed manually or even automatically. So… someone is still going to think that testing is not a critical activity in agile, that requires a strategy within agile projects?

Another typical question: how different is testing when working in agile? Testing is not even more a late phase in the development. Therefore, testing is a continuous activity during the agile development, at different levels, of different types and performed by different roles. It is also important to note that intelligence in testing is even more important in agile. As there are continuous changes in the software in each iteration and release, not all types of testing and at all levels can be executed. This situation requires: (1) more prioritization of test cases, (2) a strategy of testing balanced with the predicted risk which needs to decide which testing is applied at iteration, release and organization levels. If you still don't know about CognitiveQA, this is an essential solution to support intelligent testing in agile. Bots with reasoning and predictive models in the background may be robotic members of agile teams that assist in the collaboration and cocreation that characterizes agile.

4. Avoid creating unfeasible expectations

Agile is also about transparency. And transparency needs to be supported by artifacts, ceremonies, planning and measurement (see next tips). Going ahead without feasible expectations should be forbidden in agile. Because all supposed agility rapidly falls down without aligned expectations. This is why we advocate for concepts like *"iteration 0"* for planning without delivering valuable software, but creating valuable organization for the team, or ceremonies that need to be computed as part of the development time because of their value. And this is why we work in iterations in which we plan what we can do and what it cannot be done in an iteration. Agile is about reducing risk in each iteration by setting up feasible ambition in an incremental process of delivery.

5. Co-create artifacts for driving the agile process

Communication needs to find a balance between face-to-face conversation and explicit artifacts that drive and support the process, while they are aimed at reducing knowledge debt. In this context, a specification strategy needs to be defined. Artifacts like user stories are recognized as agile artifacts, but again, they need to be conceived as co-created artifacts. Every team member has something to say regarding a user story: business people, for sure. But also testers (who provide a different way of thinking about corner cases, positive and negative results, specific scenarios, etc.), UX professionals, developers (who need to clearly understand, and not assume, what do be implemented), architects (who can put his view on the complexity of implementation), etc.

Artifacts like user stories are not bureaucracy. If conceived like this, they can be removed. Instead, use them as co-creation artifacts aimed at pushing for early and continuous communication. Things will go better if we work around explicit user stories that, while being created and refined, provide alignment, clarification and work improvement for different team members.

6. Celebrate ceremonies

Agile ceremonies are the institutionalization of valuable meetings for improving the product, the process and the team. Sprint reviews in SCRUM, for example, are aimed at following-up the product, while daily meetings are very short and are aimed at emerging blocking problems as soon as possible and reviewing day-to-day plans. But we also have, for example, retrospectives aimed at improving the software development process and the team. The list of items to be considered and followed-up in the agile process is typically defined in the *definition-of-done (DoD).* The DoD needs to incorporate quality assurance & testing activities in order to ensure that such activities are planned, performed and checked.

7. Plan & measure

The pursue for quality in agile software development requires planning when applying "industrialized" or large-scaled agile. For example, no sprints prioritization is possible without a high-level release planning that acts as a roadmap for the product. Or we cannot apply agile testing without a release and organization strategy (see tip 3). Or another example: we cannot check if the quality is being decreased or increased across iterations if we don't have a minimum set of Quality KPIs (Key Performance Indicators) to objectivize the quality evolution and use it as feedback to refine the quality assurance strategy.

Agile gives us the opportunity to adapt the quality strategy iteration-by-iteration. This means planning and taking decisions, which need to be informed and supported by measured quality evolution.

8. Go for technical excellence

Agile also pushes for technical excellence, which is also the base for improving quality. Being updated about modern architectures, new tools, new best practices, etc. is essential. This is why specialized and open-minded people is required for being part of agile teams. Moreover, such people need to be professionals that participate in communities, further than being closed to the boundaries of an agile team.

When talking about Sogeti, for example, as a company that provides development and quality services, we talk about an example of community, with professionals of development, testing, UX, architecture, business… that share best practices and technical expertise. Such Sogetians are professionals able to participate in agile projects, with the value-added of being part of a community of knowledge exchange and expertise evolution with focus on specialization.

9. Manage "agile of agiles"

In large organizations, there is more than one agile team. It means that more than "product owners", sometimes we have "subproduct owners" that belong to the same organization. Users use software as a whole, regardless internal products organization, since business processes usually cross the boundaries of one single product. This creates the concept of "agile of agiles" (in SCRUM, "Scrum of Scrums"). It means that a coordination between agile teams need to exist, stablished as an upper organization layer, in which, typically, at least all product owners need to collaborate. Moreover, agile teams (squads) may organize their members across so called *chapters*, in order to exchange expertise and knowledge regarding a common discipline.

When managing quality, some activities also need to be planned and performed at "agile of agiles" level. Regression test automation, for example, should address the most critical business processes of a company, and therefore it is typically managed at "agile of agiles" level, together with a master test & quality strategy, software products ambition or cross-wise architectural blueprints.

This is what we have as a great value in Sogeti: *chapters* of specialized people in development and quality assurance & testing in order to be part of agile teams, and to be part of "agile of agiles" specialized groups.

10. Improve

Agile is not about fundamentalism, it is about continuous improvement in the way we develop software with quality. Do not forget to be questioning the way we work continuously, and how we can refine it to achieve better results. Learning by doing and improving from success and failures are a must in agile (like in our life)!

In conclusion

The aim of agile development approaches is twofold: faster continuous development and better quality. Continuous development activities require continuous feedback, mainly provided by ceremonies and continuous testing & quality assurance activities. QA activities are an indissoluble part of agile development, as they are a continuous insurance and leader of quality, and a main source of continuous feedback. Then, if you want to be more agile, let's define, organize and apply an (intelligent and agile) testing and quality strategy. Otherwise, you will apparently run faster (in the beginning) ... but probably you won't be as agile as expected.

And what about moving towards DevOps? Hope that you read my next post… ;)