Quality == Speed

I like the sound of this brisk statement. In my recent experience and discussions with fellow Agile practitioners, I am more convinced than ever that this statement is true. It doesn't matter what subject or profession, any neglect on quality will always come back to haunt you and your project. I will discuss a few recognizable examples we see every day, each sprint and all year round.

Aiming for quality or aiming on costs // quality in engineering skills
First, is it really necessary to mention that in order to create great software you need great - at least good - programmers and excellent tools to support them? Yes and no perhaps. Why is it that in our field of expertise we always have to defend our teams for the costs (wages) they carry? A greater and ability experience should be worth something. Research has shown good programmers can be more efficient by a factor 10 of than average programmers [2]. This is nothing new. However, it does seem paradoxical that even experienced managers are addicted to use “cheap labor” to staff new teams. Why? If you can afford a team of above average engineers, the speed of coding is most likely going to be well above average. These guys will have a profound understanding of quality code [3] and nothing will be holding back the team from becoming very, very efficient. This is why I am convinced that any investments made in enough above average programmers to set up a team, for arguments sake say 15 Euro more per hour than average pay, will be worth your money.

People over process, but still process // quality in communication processes
The following seems to happen quite often. At the start of an important project the project leader organizes a team kick off; somehow he fails to explain and discuss the communication lines you have with your client. This can be risky. Team members might be losing precious time in unstructured discussions about what is required. They contact the wrong people, or the right people with the wrong questions, or the right people and questions but at the wrong time. Delays and unclear requirements will pile up. In no time any team will feel the urge to improve on these processes and will hopefully do so. Yet, at a time where projects are getting smaller and customer expectations only bigger, the need for clear communication from the start is paramount. Yes even in Agile & Scrum you need to have this well organized. Quality in communication processes is often an underestimated factor in software engineering. Potentially it is one of the biggest drains and wastes of energy and time [4]. So take a little time to establish clear ground rules for interacting with your Product Owner and client.

Pipelines and usage // quality in deployments pipelines
In the not so distant past, 10 years ago, it was state of the art to have an automated build in place. By having these in place, your code would be compiled and unit tested every night. In the last few years the continuous-x-movement has provided us with numerous tools and platforms in which almost every step of the software process can be automated. There is no argument left to *Not* implement the smallest feedback cycle possible for programmers as described in chapter 5 of the CD Bible. But, better yet also for Product Owners and clients. Programmers can get their direct feedback on their code check in, based on coding guidelines, pre written acceptance tests. Enabled by the continuous deployment [5] on specific environments, Product Owners and

clients can actually experience how new code – the newest version of the product - is working, and how the UX design looks and feels. If these benefits are so huge, why don't all projects use them already? Well, setting up a build pipeline takes time and requires knowledge of infrastructure and a lot of tools. Large companies have dedicated teams to set up and maintain these pipelines, so development teams can focus on actual software. Smaller companies may well be helped by CD consultants helping them, and in doing so increasing the quality of their delivery process. But even for smaller projects it is still well advised to do so. Not convinced yet? Check out Jez Humble's post and come back if you are still having second thoughts.

Conclusion

As you might have noticed, I am convinced about the enormous role a quality-in-everything approach can have in software projects. Quality on all levels but also the right level of quality for a specific project. The examples mentioned above will obviously not hold for some demo projects in which a only new design needs to be demonstrated. However, the moment you are starting a new project or team (or redefining existing ones), that is the time to think and design your strategy based on a good quality approach and implement as good as possible right from the start. Take a look at some of the points below and ask yourself if your project is sufficiently equipped:

• Team mission is clear and shared
• Enough talented engineers
• You (as manager), the team and your customer agree on a realistic planning
• Communication process is clear
• Team & customer are aligned on strategy
• Continuous delivery pipeline set up
• Releasing goes without stress
• Code quality is measured and good
• Delivery keeps a steady pace
• When finishing a project no skeletons are found

This list can go on and on of course. Remember nothing is set in stone, but a good start can provide a necessary jumpstart for a successful project.

Contributions & References

[1] Leading image: 3d-systems-project-ara-high-speed
[2] Origin of 10X: 10x Software Development - Origins of 10X
[3] Clean code: http://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship
[4] Communication in software projects: White paper Effective project communication:
[5] Differences Continuous X:
Continuous-Delivery-vs-Continuous-Deployment-vs-Continuous-Integration