Reaching Software Development Maturity with Continuous Delivery

Viktor Ekholm
Fredrik Stål

Faculty of Engineering, Lund University

Having a mature software development process signifies that it has undergone certain improvements to maximize productivity while the users of the process find it simple to work with.

In this master's thesis we have studied an approach to increase the level of software development maturity in a company by analyzing key areas in software projects on the company.

We were able to find that the most important first step in a process improvement is to develop a template for a common process.

As fine spirits mature and develop a more complex and interesting taste over the years, the same can, almost, be said about software development. Having a mature software development process may not produce any extraordinary taste sensations, but it does give a comfortable and effective work process. The prerequisites are to put in effort to actually mature the software process.

In short, improving your software process aims to rid you of any manual, repetitive processes and possible downtimes due to events you have to wait for to complete. Unnecessary rework is considered an anti-pattern and thus a very bad practice. Automating this hard labor intends to give developers a break from tedious, manual tasks, but in theory quality and consistency would also improve. This is because we have a smaller risk of random mistakes due to automation.

This master's thesis have studied a practice, where we aim to always have a releasable product, as a means for software process improvement. Through this practice we expect less integration problems as software code is integrated in small increments several times per day. Through automated test executions upon integration, we receive iterative feedback which gives the status of our software.

The company is first and foremost a consulting firm, but also conduct some in-house development. These development projects became our focus and it was initially perceived that one of our premises was that the in-house developers did not have a common development process.

Through interviews with developers and stakeholders, our take on the in-house environment was that the nature of the company required developers to be given assignments on other companies between projects. Their knowledge of processes is then temporarily lost for the in-house development when they leave. As there is no collective knowledge of processes, there is a lot of work only suitable for the moment at project start. This is usually called *ad hoc*. In a way the context has a relation to open-source projects, where we have to expect that developers come and go and thus require a process that would make it simple to start contributing.

It was perceived that no value is seen in taking the required time to mature and baseline processes. Work hours are considered as billable and the customer is most certainly not interested in paying good money for advancing the processes at the company after they already have accepted them as developers. The interest for the company in developing more efficient processes is to place more time and focus on developing the software and thus increase productivity.

One of the purposes of this thesis was to, through interviews and analysis, locate key areas in the development process that could benefit the most from improvement. Although being a smaller company, they had multiple projects running simultaneously with different number of developers in each one, ranging from one to six. The first target was only the projects that developed applications for the Android platform and try to extend the process by adding additional features and tools. The intention was to improve, not only to increase efficiency of the development, but also the communication with customers.

After a certain amount of weeks we changed our target to a newly started project that had three smaller parts. These parts consisted of an ASP.NET website, an Android and an iPhone application, respectively. The motivation of this change of target was to have the opportunity of accessing and monitoring a live project taking shape in real time, and not just basing our work on theoretical or old projects. Also, the diversity aspect of this three-parted project interested us.

The primary goal then became to analyze the process used for each part and together with the developers work with improving key areas during the course of the project. It was discovered that one of the smaller projects contained a deployment procedure that was performed several times a day to a testing environment, which we saw as a perfect opportunity to study how processes are carried out at the company.

There was no unified or general approach for projects, which had led to that developers on each part of the studied project had performed implementations on an ad hoc basis. Besides analyzing the reasons and root causes of the chosen methods we also had a goal to develop a common process with the purpose of introducing automated procedures for as many of the manual activities as possible. This would further benefit our own work, but our initial hypothesis was that it would also help to increase knowledge of well-defined practices and assist in increasing the level of software development maturity on the company, which we had as a third goal.

Based on our analysis and studies we were able to determine the most simple solution for the problems we had found. The key areas of the development process turned out to be development activities such as building, testing and deploying software. By ensuring that these activities could be automated in an abstract process applicable to every project, we had developed a common process with the aims of fulfilling our goals.

In the prototype solution, development activities are separated into sequential and automated stages. Each stage has the capability of reporting its results as feedback to developers and once a stage (i.e. an activity) passes, the next stage will be triggered. For every new iteration, any additional changes to the software are collected to ensure that the latest version of the application is built and tested.

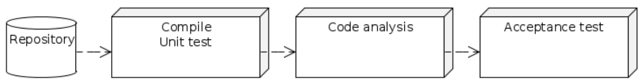

Fig. 1. An example of the prototype solution.

After we had made a prototype we held a demonstration for the developers in the project, in which we showed them how to set up the solution from scratch. Our setup handled builds, unit tests and deployments of the application automatically once a new version of the software was created. This demo with the developers generated a lot of valuable discussion and it was decided that the developers would try our implemented solution during the remaining couple of months work on the project.

The work that has been conducted during this thesis gave us a lot of ideas, some of them which we were able to fulfill with implementations. No one can deny that reaching high levels of software development maturity requires effort from the majority of the organization, but there is no particular reason as to why this should not be one of the most prioritized goals. A strong argument is that our implemented solution was able to increase efficiency by almost 10%.

We saw that the changes we introduced and have discussed will be very beneficial for a software company and the benefits are discovered almost immediately after implementing solutions. The results have also shown that the commitment of developers will increase because they no longer are forced to endure tedious, manual tasks followed by a new version of the software. In the long run, an improved process will also become a natural way of working, which eventually leads to further improvement and higher levels of maturity.

The above mentioned issues are what we found in our context, taking another context into consideration might have a lot more issues, or perhaps fewer. Our work showed that a general solution can only be abstracted for a specific domain, which is different from company to company. By first defining what goals our improvement aims to fulfill we can reduce the friction in a software process improvement, remove ad hoc procedures and be one step closer to ultimately reach higher levels of software development maturity.