Reaching Software Development Maturity with Continuous Delivery

Fredrik Stål
Viktor Ekholm
Faculty of Engineering, Lund University

June 11, 2013
Abstract

In the area of software configuration management a configuration manager works with almost every part in the software development process. The responsibility is to ensure efficiency at the highest possible level for a development team, which is simplified a great deal with a mature and well defined development process.

The purpose of this thesis was to analyze and locate key areas in a software development process for small teams and define methods of improvement that will serve as goals for to reach process maturity. The work is mostly based upon the concept known as continuous delivery, which aims at maintaining a releasable product after each software code integration. The project was carried out on a consulting firm called Softhouse that also has in-house development projects with mobile applications in teams ranging from one to seven developers.

The work started with interviewing the developers with the goal of finding the root-cause of the current process, as well as identifying both the possibilities of improvement and determine the requirements they had on their own work. To narrow the scope we focused on working together with a project that started and had its deadline during this thesis. That project was split into three smaller sub-projects, which gave us the chance to explore how more than one development process was normally carried out and how collaboration was done.

Together with the developers in one of the sub-projects we designed and implemented a new environment with the aim of solving both the perceived and real problems that were discovered during analysis. The purpose of the environment was to perform manual tasks by automation and introduce the starting points of continuous delivery. Measurements on the process was done before and after the improvement to find what benefits gained and any eventual costs. The results of the measurements were positive and although the change was a very small one the benefits was still detectable.

Based on our analysis and results from measurements we developed a maturity model. The primary objective of the model is define a set of goals to improve a software process. The model is split up into different levels of maturity and to reach a higher level the process first has to achieve the goals for the next maturity level. This maturity model is the result of our master’s thesis and is still in its experimental stages. It would require further testing and research to be fully developed.

Keywords

Software development maturity, software process improvement, software configuration management, continuous delivery, continuous integration, agile development
# Contents

Preface v  
Acknowledgments vii  

1 Introduction 1  
1.1 Problem Statement ................................................................. 1  
1.2 Purpose ..................................................................................... 2  
1.3 Contents .................................................................................... 2  

2 Our Context 3  
2.1 Project Types ............................................................................. 3  
2.2 Perceived Problems .................................................................... 3  

3 Software Development Maturity 5  
3.1 Software Development Maturity .................................................. 5  
3.2 Continuous Delivery ................................................................. 5  
3.2.1 Continuous Integration .......................................................... 5  
3.2.2 Continuous Delivery ............................................................. 5  
3.2.3 The Deployment Pipeline ....................................................... 5  
3.3 Patterns and Anti-Patterns .......................................................... 5  

4 Locating the Gremlin 7  
4.1 Define the Project Type ............................................................... 7  
4.2 Analyse the Current Process ....................................................... 7  
4.3 Discover the Real Problems ....................................................... 7  
4.4 Present the Requirements .......................................................... 7  

5 Designing the Solution 9  
5.1 Continuous Integration Server .................................................... 9  
5.2 Development Environments ...................................................... 9  
5.3 Branching Strategies ................................................................ 9  
5.4 Distributed Development .......................................................... 9  
5.5 The Value in Choosing .............................................................. 9  

6 Implementation and Results 11  
6.1 ASP.NET Website Project ......................................................... 11  
6.2 Android Project ...................................................................... 11  
6.3 iPhone Project ....................................................................... 11  
6.4 Results ...................................................................................... 11  

7 Evaluation 13  
7.1 Discussing the Results .............................................................. 13
7.2 Limitations .................................................. 13
7.3 Further Studies ............................................. 13

8 Conclusions ................................................. 15
  8.1 Results .................................................... 15
  8.2 Thoughts for the Future ................................. 16

Bibliography .................................................. 17

A Terminology ............................................... 19
This thesis is targeted for individuals with some expertise within the subject of software development, be it developers, project managers, configuration managers or undergraduate students in the field of computer science. We will attempt to keep our discussions at a level so that any person belonging to the aforementioned groups can make use of the information presented and discussed in this report.
Acknowledgments

We would like to thank our supervisor, Christian Pendleton, and our examiner, Lars Bendix, for their invaluable guidance and great engagement and interest throughout our work. We would also like to thank all employees of Softhouse in Malmö, the company where we carried out the work you are about to read, for their help and hospitality.
Chapter 1

Introduction

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 you 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 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 improve, since there is less chance of careless mistakes due to automation.

A fundamental building block in software process improvement is the practice of continuous delivery. Through this practice we expect less integration problems as software code is integrated in small portions several times per day. Through automated test executions upon integration, we receive iterative feedback which tells the status of our software.

1.1 Problem Statement

Softhouse 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 the in-house developers were in need of a common build server.

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. As there is no collective knowledge of processes, there is a lot of work made ad-hoc at project start. In a way this has a relation to open-source projects, were it is assumed that contributing developers will not contribute during the entire lifetime of the project.

Another problem is that no value is seen in taking the required time to mature and baseline processes. Work hours are considered to be billable and the customer is most certainly not interested in paying good money for advancing the processes of Softhouse after they already accepted them as developers. The interest for Softhouse in developing more efficient processes is to place more time and focus on developing the software and thus increase productivity.
1.2 Purpose

The purpose 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, Softhouse had multiple projects running simultaneously with different number of developers in each one, ranging from one to six. Our 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 beneficial areas during the course of the project. It was discovered that one of the smaller project 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 on Softhouse.

There was no unified approach for the whole project, which had led to that developers on each part had implemented their project on an ad-hoc basis. Besides analysing the reasons and root-causes of the chosen methods we also wanted to implement an environment 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 make the company see the benefits of such process improvements from a practical perspective.

1.3 Contents

Below we describe the chapter outline and touch on what each chapter is intended to discuss, as a tool for you, the reader.

Chapter 3 will go more into detail and present a detailed in-depth study of this thesis. The chapter will describe what it means to have a mature development process and dig deeper into the practice of continuous delivery.

Chapter 4 handles the main problem analysis that were conducted during the course of the project and define the real problems in the studied process that are derived from the perceived problems described in chapter 1.

Chapter 5 takes the requirements that are defined in the previous chapter and present possibles solutions. It discusses the advantages and drawbacks of each presented solution and argues why we choose a certain solution.

Chapter 6 contains a description of how the selected solution was implemented into the software process and what measurements that were taken to arrive at results.

Chapter 7 evaluates and discusses the “validity” of the work. Any restrictions and limitations that are posed on the implementation are brought up and finally relations to other work in the same area and how to extend what was accomplished.

Chapter 8 is a summary of all the results of the thesis and potential thoughts for the future.
Chapter 2

Our Context

2.1 Project Types

As we have mentioned earlier on in this report we have targeted a project divided into a subset of three smaller projects for the purpose of analysis. These three projects are dependent on each other, or rather the two client applications are dependent on the server application. For the ease of the reader, we will define the types below before we delve into the detail of their characteristics.

**ASP.NET server application.** A website application written in Microsoft’s .NET framework, acting as the server-side application for the below client applications.

**Android client application.** Client for the Android mobile platform.

**iPhone client application.** Like the Android client, but a client for the iPhone/iOS mobile platform.

2.2 Perceived Problems
Chapter 3

Software Development Maturity

3.1 Software Development Maturity

3.2 Continuous Delivery

3.2.1 Continuous Integration

3.2.2 Continuous Delivery

3.2.3 The Deployment Pipeline

3.3 Patterns and Anti-Patterns
Chapter 4

Locating the Gremlin

4.1 Define the Project Type
4.2 Analyse the Current Process
4.3 Discover the Real Problems
4.4 Present the Requirements
Chapter 5

Designing the Solution

5.1 Continuous Integration Server
5.2 Development Environments
5.3 Branching Strategies
5.4 Distributed Development
5.5 The Value in Choosing
Chapter 6

Implementation and Results

6.1 ASP.NET Website Project

6.2 Android Project

6.3 iPhone Project

6.4 Results
Chapter 7

Evaluation

7.1 Discussing the Results

7.2 Limitations

7.3 Further Studies
Chapter 8

Conclusions

8.1 Results

The most promising result from this master’s thesis has been that even minor changes, which requires very little effort, have significant effects and benefits on a software process. Although, implementing a solution for one of the parts of the studied project proved only to be a challenge for reasons related to installations and first-time setup. Since neither of us have had much experience in this type of work, this resulted in unexpected problems.

We saw that the changes we introduced and have discussed will be very beneficial for a software company, especially in the long run. The results have also shown that a developer will increase his or hers interest in their own work, which is related to the fact that a process improvement is targeting just that. Given time a changed process will also become a natural way of working, which eventually leads to further improvement and higher levels of maturity.

There are still some issues that needs to be tackled linked to the view of concepts, such as continuous delivery and the tools developers can exploit to simplify activities. One of the most common issues we ran into during the project was a misguided view that tools are something of a wizard or a magic wand. The larger application, for example a continuous integration server application, is supposed to magically solve all your problems while you, the developer, only have to install the application and tell (configure) it what to do. The magic wand consists of tools like your version-control system that you somehow wave over your software code and every fault is corrected.

Another result of our thesis is that it has to, at least in some way, be defined who is responsible for maintaining the different environments. We have seen how the natural approach of everyone’s responsibility quickly leads to no one’s responsibility. In the same way, if too few employees are responsible then the question and the problem will still be present should they leave. This is even more true for this context, a consulting firm.

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 issues can not simply be abstracted into a general solution, but be solved differently from company to company. To reduce the friction in a software process improvement a maturity model, like the one we developed, will narrow the scope and help to create requirements to reach higher levels of maturity.

It proved very well to combine the principles of continuous delivery together with that of software development maturity. The end result became a model that is incremental by nature and more related to development activities and therefore more easily interpretable for developers. Although faster releases is a given benefit for continuous delivery this thesis has shown that the concept
is much more than that. The core feature, the build pipeline, has shown to be a well-designed
approach when implementing control and automation of your software.

8.2 Thoughts for the Future

As a final note it is worth taking the future into consideration. The solutions presented here in
this thesis might seem useless or no longer feasible in, for example, ten years. However we can,
with confidence, say that the problems related to software development that are the main focus for
a process improvement will still be and forever be present. The impact they have on development
might differ, but the root-causes will live on and need to be maintained.
Bibliography


Appendix A

Terminology