Introducing SCM for a XP-team

Albin Rosberg
D06, Lund Institute of Technology
ie06ar3@student.lth.se
February 13, 2011

Abstract

In this paper a XP team will be examined from a SCM perspective; how they use different SCM practices, which problems occur and how they are handled by using SCM and how the given tools impact the overall SCM experience. A questionnaire regarding the previous SCM experience and knowledge of the team is conducted and branching patterns and dimensions are examined for the team and similar teams taking the same course. How much impact a coach has on the overall impression of SCM and how they chose to use (or not use) different SCM practices is examined. Simple encouragement in a few areas of SCM might lead to a team's deeper understanding of SCM and a higher acceptance of it's practice and implementation. Allowing the team to analyse problems and issues and their causes also aids the goal of a deeper understanding of SCM and tailored SCM systems.

Keywords: SCM implementation, Extreme Programming, Coaching

Index

1 Introduction

A course where the XP methodology is introduced and taught has been in the curriculum for students in year two on Computer Science at Lund Institute of Technology for several years now and provides the students with some experience in working as a part of a development team. There are 10 teams which consist of 8-12 students working as developers, 1-2 students coaching and a teacher acting as customer. The students are using Subversion and are mainly working with the Eclipse plug-in Subclipse.

1.1 Purpose

The purpose of this paper is to examine how Software Configuration Management can be introduced to a XP team, what impact the coach has and whether or not the SCM practices are adapted when introduced.

1.2 Concepts

SCM is defined as “the discipline of organising, controlling and managing the development and evolution of software systems” according to IEEE. SCM includes quite a few other terms as well, in this paper a few of them are mentioned. Dimensions of branching are descriptions of for what a branch is created (Appleton et al. 1998), which differs from branching patterns which in this paper describes when one should branch. In Change Management, the concept of Change Requests is used widely as what needs to be changed. After a change request is solved, there might be a Configuration Control Board (CCB) consisting of people from several parts of the company, including the Configuration Manager (CM), which reviews the change and either approves or disapproves it.

2 Background

The course (EDA260) is the first real encounter to SCM for a large amount of students. SCM is introduced briefly during the theoretical part of the course and the students get a chance to practice it during one laboratory session. During the laboratory session, the students are familiarised with basic CVS usage.

2.1 The course project

The course project is to develop a system for time taking of an enduro race. The team is given a number of stories each week, which are estimated by the team and prioritised by the customer. The team will use one work day (laboratory session) each week to attempt to implement the stories. This is repeated for 6 iterations during the course. The team is supposed to work with the XP practises defined by Beck (1999) and learn how to work as a developer in a small project.

2.1.1 SCM Tools
The teams are provided with a Subversion repository with accompanying trac and are working with the Eclipse plug-in Subclipse for most of their SCM needs.

Some teams also choose to automate releases by using Ant-scripts through Eclipse.

2.2 The team
The team in the interviews, Appendix A, consists of 8 students with a varied experience in version control and configuration management. Some have been using Subversion; others have no SCM experience outside the course. None of the students have any experience using the Subclipse plug-in.

2.2.1 The coach
The tasks of a coach are defined in Hedin et al. (2005) as running the planning meetings, coaching the team during the laboratory sessions, being responsible for “tracking the evolving architecture of the product and to make sure it is discussed within the team”. There is no real definition of the coach from a SCM point of view.

Each coach has a different focus study which is presented to the other coaches.

2.3 Software Configuration Management
Traditional CM consists of four operational aspects according to Dart (1991); Identification, Control, Status Accounting and Audit and Review. Asklund et al. (2004) speak of a different set of practices for a XP team: Configuration Identification, Configuration Control, Configuration Status Accounting, Configuration Audit, Version Control, Build Management, Workspace Management, Concurrency Control, Change Management and Release Management.

These aspects are the basis of configuration management and are, more or less, directly included in some of the XP practices (Asklund et al. 2004). Build Management, Change Management and Release Management are core practices for the course project and provide an interesting basis for reviewing.

3 Method
The related articles will be used as background for comparison with the experience gained coaching the team. Information about the SCM background of team members will be gathered through a small questionnaire. The major part will be listening to any questions asked throughout the project, especially in cases where problems occur. Being perceptive; listening on conversations, continuously checking the repository and observing the team whilst programming provide vital progression information in how they use the tools provided and which features are used. Furthermore, looking at which difficulties they encounter and which of these difficulties are due to the tool (i.e. the SCM practice being poorly supported) and which difficulties are encountered due to lack of knowledge in SCM.

3.1 Course Start-up

During the start-up of the course, the following SCM ideas were presented to the team:

- Build Management
- Change Management
- Release Management

The team was asked to voice their general opinion on these SCM ideas and whether these would be of assistance during the course of the project.

3.1.1 Build Management

Past experience shows that building the project might not always be fast or easy. In order to make sure it is done with more ease, the system ought to be built frequently. Automating a build will manage the build process speed. A structure on *when* and *how* to build is needed in order to cope. Introducing the possibility of automated releases by using an Ant-script provided the team to make sure the builds are fast, thus not slowing down the implementation pace significantly. After each story was implemented the team was supposed to build the system, making sure it was working as intended. Using automated release, building frequently and using pair programming (Beck 1999) will eventually lead to the knowledge of how to build a system being spread throughout the team.

3.1.2 Change Management

There were several aspects of changes to be considered in a XP project; changes to stories and data structures, commits, architectural changes etc. No hard rules were set up, instead communication was highly encouraged and making sure the team was responsible for the code. Encouraging proper commit comments, continuous update of the team wiki and trac and forcing group discussion on major architectural and data structure changes was the overall plan laid out.

3.1.3 Release Management

The release management procedure was vaguely introduced at first, using Functional Branching (Appleton et al. 1998) in order to keep the repository clean. The branch would be releasable at all times and thus the any release would be smooth. A basic list of requirements was provided in cooperation with the team, setting up the requirements used for when to merge into the release branch. The streamed line pattern *Merge Early and Often* (Appleton et al. 1998) was chosen as benchmark in order to keep the release branch up to date at all times.

4 Results

Results from data mining and the questionnaire are represented to give basic data for the analysis and conclusions in this paper. The actual result data are represented in the appendices.

4.1 Sources of Error

Since the branching data was only gathered on two separate occasions and not continuously during the course, there are no guarantees that branching hasn't been used as suggested by the matrices. Continuous enquires have been made whether or not branches have been used, thought not noted down, and been consistent with the results.

4.2 Data mining other teams

By data mining the repositories of the different teams, the use (or lack of) of different dimensions of branching, as explained by Appleton et al. (1998), can be determined. The branches are used for either release purposes, experimental coding purposes or integrating purposes (i.e. not fully implemented code or refactoring). A matrix containing the results is shown in Appendix B. The results were gathered mostly by manually inspecting the repositories, or in some cases where the purpose was not clear; asking the teams. The data was gathered during two separate occasions with two weeks in between.

By the end of iteration 4, 50% of the teams were not using branching at all, 30% use it for experimental programming, 20% for release management and 40% for integrating purposes. Only one team, the team examined in this paper, use branching for all these three purposes.

By the end of iteration 6, one more team have used branching for the purposes of experimental coding and one team have branched for release.

4.3 Results from the questionnaire

Person A, B and C has had previous experience with SCM and recognise it as a core feature in a development team using the XP practises. The consensus of the team is that while SCM practice is a core feature, the Eclipse plug-in Subclipse has a very negative vibe and there is annoyance that it does not provide the features a Subversion tool should have. Person A stated that strict long transactions would be preferred for the project and Person B see the need of a more automated update-commit process. Person D, while having no previous experience with SCM, recognises long transactions as a vital feature due to human mistakes.

5 Analysis

Introducing SCM for a XP team of students has some major benefits compared to introducing it to companies. More often the developers working at companies have set habits of how things should work, which Rosberg et al. (2011) points out as a key factor to negative views regarding SCM. The second major factor which might make or break decisions regarding implementation of SCM in industrial life is money. Money is still a factor for a course, though it is easier to make a long term investment for educational purposes since there is no need for a monetary payback. The payback might instead revolve around efficiency, Pei et al. (2009) states that there is roughly 25% more lines of code produced by using some sort of CM in a project.

5.1 Tool impact analysis

From a SCM perspective, the project has so far not run as smoothly as possible. Using Subversion ought to give grand opportunities for branching without any loss of revision history. Subclipse has had a negative impact due to the seemingly incapability to merge branches successfully. Merging branches was tested thoroughly without success and due to the lack of success, the overall feeling towards using branches became more negative throughout the course of the project. Whilst branching patterns has been used extensively, without proper merging tools some of the benefits are nullified. Odd errors in synchronisation with the repository have also occurred at times, resulting in deleting the workspace and checking out the project once more. All issues with the tool leading to lost time, frustration and a negative view on the tool provided.

5.3 Overall branching analysis

The information provided in Appendix B show that a large amount of the teams do not use any branching patterns and apparently does not see the need for it. Several teams have had larger merge conflicts, resulting in branching for integrating purposes – keeping the repository clean and still not losing any work. Experimental branching, most often looking at how a server/client is set up, has been used on several teams. Branching provides an extra means of communication in the case of experimental coding, not only can someone be told which conclusions have been made but also how they came to the conclusions.

Release branching might be the most powerful tool in the course, yet only 30% of the teams had made branches for this purpose. Between the two data mining's only one additional team showed interest in the idea of having a branch for release after a discussion on the subject among the coaches. Branching patterns and dimensions of branching (Appleton et al. 1998) does not have the same natural feel to it as change management and build management has, it seems to be hard for the teams to see the real benefits from it and it does require the coach to actively promote it.

Two teams used branches for the projects to be reviewed, making sure the whole team had access to the information needed. Interestingly enough, one of these teams did not have a branch for their own release – only the release they were going to review.

5.2 SCM practise impact analysis

Among branching, update and commit; the students have found that both diff and log play an important part. Being able to see where and what went wrong by looking at the history and revision numbers to be able to understand present and future problems has proven important. With the difficulties of proper branch-merging in Subclipse, diff has proven invaluable to manually merge branches. The students were also able to diff when a previously working functionality ceased to work, often due to changes in logical tests. The diff command was not used as much as it could have been, by using it more actively the students would have saved valuable time searching for small logical errors. The difference between understanding the importance and using it properly was made present towards the end of the project with small errors taking a long time to solve. The team also discarded any ideas of branching, due to the inability to merge branches properly.

In an attempt to salvage any remains of the suggested release management, the streamed line pattern *Branch per Task* (Appleton et al. 1998) was suggested. Possibly cluttering the repository with branches being a risk, the advantage of having a working product at all times was alluring. The team being somewhat positive at first, disregarded the pattern as they encountered bugs and integration problems.

The team did however unwittingly change the change management draft by making a TODO-list available on the team wiki. The system worked as a basic change request (Dart 1991) and was used every time refactoring had to be done, or new tests had to be written. A team member recommended strict long transactions for the project, due to the heavy merge conflicts. It wouldn't be possible in a larger scale project due to the many commits, though in a small project it might be preferred. It would, however, not be needed if the commits were automated. A tool doing the update-run tests-update-commit sequence automatically was another suggestion from the interviews to help minimise potential damage caused by the programmers.

5.4 Coach as Configuration Manager

Whether or not a team will accept the SCM practices is not only due to the coach promoting it. The coach will have a big impact, though the tool might turn the team against it. The team will gain a lot from having a coach acting as the CM. A coach with a greater experience in the SCM tools will save some time, for instance in the case of repository not synchronising. If the coach keeps track on the configuration audit (Dart 1991) as well, the releases will be smoother. Making sure any plans set up are followed, preferably defining a full SCM plan, will in the end gain the whole team. Unfortunately as a single coach, time is not in surplus during the start-up phase of the project. A single coach will probably not have the time to act both as coach and CM. Ideally, the coach would also take the role as Configuration Control Board (CCB) in order to make sure all change requests are handled properly.

5.4.1 SCM plan

Establishing a proper SCM plan, making sure the teams agrees with it and understands it would help ease the workload on both team and coach. The SCM plan could be generic at first, with basic information on when to commit, what should be in commit comments, when to build and the like. As the project progresses, more detailed information would be provided, i.e. which branching patterns should be used. The SCM plan would need to be revised continuously with regards which features are supported by the tools provided. It’s important that the SCM plan is not too heavy at first, when introducing new practices to a team baby steps are recommended (Rosberg et al. 2011) – the plan has to grow naturally.

6 Conclusions

Software configuration management is a very powerful tool when used right and with the features needed provided, though introducing it to a team has difficulties. The overall response from students encountered with the request to use SCM practices seems to be positive, though when the tools are not working as expected – they are rather quick to discard the ideas. In order to properly introduce SCM for a XP team, more knowledge of the actual version control tool is preferred. A basic SCM plan ought to be provided and the team should extend the plan during the course of the project. Time is a major factor for a coach and doing the job of coach, CM and CCB is rather impossible. In a pair coach relationship, the two coaches could more easily divide the different jobs and thus perhaps be more successful. The coach has to be prepared with backup plans, different branching patterns perhaps, in case the SCM tools do not provide the necessary features.

7 References

[1] Pei, S., Chen, D., (2009), *The Implementing of Software Configuration Management Based on CMM*, 2009 5th International Conference on Wireless Communications, Networking and Mobile Computing

[2] Dart, S., (1991), *Concepts in Configuration Management Systems*, Proceedings of the 3rd international workshop on Software configuration management (Proceeding SCM '91)

[3] Sillito, J. Murphy, G.C., De Volder, K., (2006) *Questions Programmers Ask During Software Evolution Tasks*, Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering (SIGSOFT '06/FSE-14)

[4] Masticola, S.P., (2007), *Lightweight Risk Mitigation for Software Development Projects Using Repository Mining*, Fourth International Workshop on Mining Software Repositories (MSR'07:ICSE Workshops 2007)

[5] Hiltunen, M.A., (1998), *Configuration Management for Highly-Customizable Software*, Proceedings. Fourth International Conference on Configurable Distributed Systems (Cat. No.98EX159)

[6] Rosberg, A., Lindahl, A., Rudner, M., (2011), *How do you introduce SCM to a company that does not see the need?*, unpublished

[7] Asklund, U., Bendix, L. (2004), Ekman, T., *Software Configuration Management Practices for eXtreme Programming Teams*, 11th Nordic Workshop on Programming and Software Development Tools and Techniques (NWPER'2004)

[8] Beck, K. (1999), *Embracing Change with Extreme Programming*, Computer vol. 32 no. 10, IEEE Computer Society

[9] Hedin, G., Bendix, L., Magnusson, B., (2005), *Teaching extreme programming to large groups of students*, Journal of Systems and Software vol. 74 no. 2

[10] Appleton, B., Berczuk, S. P., Cabrera, R., Orenstein, R., (1998), Streamed Lines*: Branching Patterns for Parallel Software Development*, Proceedings for PLoP '98 conference (PLoP '98)

[11] Moreira, M.E., (1999), *The 3 Software Configuration Management Implementation Levels*, J. Estublier (Ed.): SCM-9, LNCS 1675, pp. 244-254, 1999, Springer-Verlag Berlin Heidelberg 1999

Appendix A: Interviews

Person A

Interviewer: Albin Rosberg
Language: Swedish

- Kände du till konfigurationshantering innan du påbörjade kursen?

Kännt till länge. Läst på Wikipedia samt checkat ut mjukvara att kompilera. Använde först på riktigt i DWWW 2009 och sedan i projektet i ingproc våren 2010.

- Vilka verktyg kände du till och har arbetat med innan? (t.ex. Git, ClearCase, Subversion, CVS)

SVN i terminalen och med Subversive i Eclipse. Git har jag lekt kort med på egen hand. Bazaar har jag checkat ut någon mjukvara någon gång vill jag minnas.

- Lite kort: vilka fördelar ser du med att använda någon form av konfigurationshantering i XP-kursen / i ett XP-team?

Otänkbart att inte använda. Man behöver arbeta med samma kodbas men utan att vara samma "fysiska" filer. Tryggt att man jan reverta i fall att något går ner. Decentraliserat system hade kanske varit roligare? Nu måste vi lita på att cs underhåller sin server och tar backups.

- Vilka problem har du stött på?

En jättestor mergekonflikt, vad trodde du? Jobbigt att brancha i svn. Build in mergern känns otroligt dum ibland. Strict long transaction hade nog varit bra om det fanns inbyggt.

Person B

Interviewer: Albin Rosberg
Language: Swedish

- Kände du till konfigurationshantering innan du påbörjade kursen?

Japp.

- Vilka verktyg kände du till och har arbetat med innan? (t.ex. Git, ClearCase, Subversion, CVS)

Subversion, CVS och Git.

- Lite kort: vilka fördelar ser du med att använda någon form av konfigurationshantering i XP-kursen / i ett XP-team?

Gemensam kod, det gör att kodintegrationen utförs kontinuerligt. Detta blir viktigare allt eftersom teamets storlek ökar.

Vidare finns alla versioner tillgängliga i repositoriet så om något skulle gå fel finns det möjlighet att gå tillbaks till en tidigare version.

- Vilka problem har du stött på?

Jag finner inga direkta nackdelar, utan ett konfigurationshanteringsverktyg så är man förlamad. Det bör poängteras att det hade varit trevligt om verktyget skulle kunna utöva update-update-commit rutinen automatiskt (med relevanta tester innan commit) i bakgrunden hela tiden, det är lätt att glömma bort.

Person C

Interviewer: Albin Rosberg
Language: Swedish

- Kände du till konfigurationshantering innan du påbörjade kursen?

Ja det gjorde jag.

- Vilka verktyg kände du till och har arbetat med innan? (t.ex. Git, ClearCase, Subversion, CVS)

Har använt svn tortoise (trac).

- Lite kort: vilka fördelar ser du med att använda någon form av konfigurationshantering i XP-kursen / i ett XP-team?

Xp kursen hade gått helt fantastiskt dåligt utan versionshantering och dessutom väldigt ineffektiv.

- Vilka problem har du stött på?

Det är dumt att Subclipse fungerar så dåligt med inställningar och tag/branch-ning.

Person D

Interviewer: Albin Rosberg
Language: Swedish

- Kände du till konfigurationshantering innan du påbörjade kursen?

Hade hört om det, men aldrig använt mig av det själv innan.

- Vilka verktyg kände du till och har arbetat med innan? (t.ex. Git, ClearCase, Subversion, CVS)

Kände inte till några speciella verktyg utan bara att det fanns och vad den användes för.

- Lite kort: vilka fördelar ser du med att använda någon form av konfigurationshantering i XP-kursen / i ett XP-team?

Tror det hade varit svårt att göra ett sånt här projekt utan ett sådant verktyg.

- … och vilka problem.

Kan inte komma på något på rak arm utan det är ofta mänskliga misstag som commit innan update etc sådant hade hänt även om man inte använt sig av konfigurationshantering.

Person E

Interviewer: Albin Rosberg
Language: English

The interviewed person did not respond to the questionnaire and instead simply answered that the person knew nothing of SCM and had never worked with it before.

Appendix B: Data mining

Date: 2011-02-14

	No branching	Experimental	Release	Integrating
Team 1	x			
Team 2	x			
Team 3	x			
Team 4			x	x
Team 5	x			
Team 6		x		x
Team 7	x			
Team 8		x	x	x
Team 9		x		
Team 10				x
	50%	30%	20%	40%

Date: 2011-02-28

	No branching	Experimental	Release	Integrating
Team 1		x		
Team 2	x			
Team 3	x			
Team 4			x	x
Team 5	x			
Team 6		x	x	x
Team 7	x			
Team 8		x		x
Team 9		x		
Team 10				x
	40%	40%	30%	40%