Towards a Suite of Software Configuration Management Metrics

Lars Bendix, Lorenzo Borracci

Department of Computer Science, Lund Institute of Technology,
Box 118, S-210 00 Lund, Sweden
bendix@cs.lth.se, loreborra@gmail.com

Abstract. Software Configuration Management (SCM) is an important support activity in software development. However, its transparent nature as a service that makes life easier for others and as an insurance against disasters, often makes it difficult to justify investments in tools and processes that apparently do not have any direct return. We have made a first step towards establishing a model for showing the return on investment in SCM, making the costs and benefits explicit. In this paper, we also sketch how we plan to take the next important step and establish a set of metrics that can be used to manage and tune the SCM processes and tools.

1 Introduction

Development and maintenance of software require the use and interaction of many disciplines and groups of people. In the CMM model Software Configuration Management (SCM) is one of the key process areas that needs to be mastered to progress from the initial level of ad-hoc chaos to level 2 (Repeatable). However, unlike other key process areas, such as requirement engineering, SCM is not a discipline that ends up with a tangible product that can be weighed and judged for cost and quality. It is transparent (service and insurance) yet vital to help other disciplines carry out their work. Most of its benefits are "invisible" and spread over many groups (budgets), whereas the costs are mostly explicit and placed in the SCM group. So how do we justify initial or further investment in SCM?

It would be nice if there existed a simple model to calculate the Return On Investment (ROI) for SCM. Of previous academic work we are only aware of Larsen and Roald [LR98], that does not give a complete model, but measures some data pre and post implementation of an SCM tool and process. Many tool vendors have some promotional material, that use a simple ROI model [Merant01], but these models are not complete, nor are the values they claim credible. Such simple models may be adequate to justify the initial introduction of SCM, but they are not sufficiently complete to calculate the ROI of upgrading the SCM tool or introducing new SCM processes.

We want to do a more thorough analysis of the costs and benefits of investing in SCM to get a more complete picture. We are well aware that it is very difficult to get the full picture and that many costs and especially benefits are subjective and/or hard to quantify. This has to be dealt with in some way in such a model. Such a ROI model can have many uses and must be tailored for that. In our model, we aim both at companies that want to introduce SCM and at companies that want to upgrade their SCM tool and/or processes. Our motivation for doing this is the simplicity of handling only one model and the fact that it is indeed possible to use a complete model to establish the costs and benefits and calculate the ROI both for initial introduction of SCM and for later adding to a partial implementation of SCM.

However, the ROI model is not our ultimate goal. We find it more interesting to go one step further and provide help for the SCM manager to daily monitor, manage and tune their SCM tool and processes – and to be able to predict specific SCM costs and benefits for projects. Therefore we are looking for a set of SCM specific metrics. Traditionally metrics and SCM is thought of as data supplied by SCM to monitor other software development processes and not the SCM processes. In this work in progress, we will treat the ROI model as an intermediate result and use it as a stepping-stone towards uncovering a set of SCM specific metrics.

Establishing a model for calculating the ROI, as well as finding SCM specific metrics and benchmarks, is something that calls for both theoretical analysis and empirical validation. However, as it was also pointed out by the previous work of Larsen and Roald [LR98], this is something that requires a long period of time and this ongoing work is currently entering the phase of the empirical validation, so in this paper, we can present only the theoretical analysis.

In the following, we first present the ROI model for SCM and the analysis that led to it, then we outline how we intend to use the coming validation of this model to identify a set of SCM specific metrics and benchmarks, and finally we draw our conclusions.

2 Step One – A Return On Investment Model

To get the broadest possible coverage of potential costs and benefits, we wanted to analyse SCM from several different perspectives. The first – and most obvious – was to analyse costs and benefits of the four canonical activities of SCM, as laid out by standards like ISO 10007:2004 and ANSI/EIA-649: configuration identification, configuration control, configuration status accounting and configuration audit. To get a better idea of the extent of especially the benefits, we wanted to widen our analysis and focus not only on the coding phase of software development, but cover the entire life cycle from requirements analysis through to maintenance. Finally, we wanted to look at the people involved with SCM tasks to be sure to get a complete coverage of benefits and – in particular – costs.

In the following, we briefly describe the results of the analysis from each of the three perspectives. This is followed by the presentation of the model that results from these analyses.

2.1 Analysis by SCM activities

In general, SCM is looked at as a management discipline that can help in planning and running a project. As such, it provides a SCM plan that describes the processes that have to be followed and a tool – or set of tools – that to some degree can automate some of these processes. The general costs that we get are those associated with the tool(s) (licences, maintenance), the making and maintenance of SCM plans, and the training of people to understand and follow the established processes and tool(s). To know more about the benefits we treat each of the four activities in turn.

Configuration identification. This activity deals with the recording and communication of information of identified configuration items and the baselining of these items. It establishes clear and effective naming conventions and how to hierarchically structure sets of configuration items. This creates a clear navigational structure for information retrieval and explicitly addresses traceability in different contexts.

Configuration control. Most important here are the change process and the Change Control Board (CCB) that defines how changes are handled. This ensures that the impact of changes is assessed and analysed, and that changes are planned and managed. This allows us to trace the status of changes as well as the entire product through its various baselines.

Configuration status accounting. This is a system of formatted reports created based on the data available in the SCM system. The associated cost is that of creating the original data, the benefits those of traceability and better information for management.

Configuration audit. Audits verify the functional characteristics and the form and fit of the product. The costs are those of carrying out the audits, the benefits are improved stability and quality of baselines/releases.

2.2 Analysis by the product's life cycle

Usually SCM is considered an activity that is aimed only at the coding phase of a software development project. However, all phases can and should apply the configuration management principles and methods to their work products. We will look at each of the life cycle phases in turn.

Requirements. By applying SCM to the requirements phase products we can get the same benefits (and costs) as for the Coding phase. Furthermore, we can obtain traceability between the work products of the different phases.

Design. This phase is equivalent to the Requirements phase.

Coding. The costs are getting tool(s), processes and training in place. Benefits come from being able to handle variations, reused code, work in parallel (and/or distributed), automated builds and facilitating the co-ordination and collaboration between programmers and teams. Furthermore, traceability, both between phases and between versions (or variants), is a benefit.

Testing. Testers benefit from the use of documented baselines to create the builds that are to be tested. This way is always clear exactly what is being tested.

Release. The use of documented baselines makes is possible to always recreate old releases. Furthermore, configuration audits ensures the quality of the release.

Maintenance. This phase probably has the most benefits from SCM. From the "information base" that SCM provides we obtain traceability and history that helps in tracking down and removing bugs. Benefits when adding new functionality are the same as for the coding phase.

2.3 Analysis by people involved

In the analysis so far, we have mostly looked at SCM from the company's point of view. In this section, we acknowledge that there are different roles that are related to using SCM and that these roles have different interests, as is also pointed out by Hass [Hass03].

Senior management. They share many interests and benefits with the company – better quality of the product, faster and more flexible response to customer requests and shorter time-to-market times.

Project management. Most benefits are related to the planning, scheduling and managing of activities. Status accounting gives improved insight into the project status and the use of a CCB ensures that all changes are controlled and impact analysed so they can finish in time and within budget.

Developer. As pointed out by Babich [Babich86], SCM is just as much about team collaboration and co-ordination as it is about management. In our case, we use the term developer in a very broad sense, as also people from other phases of the life cycle can use SCM for "developing" their products. The concepts of baseline and workspace allows for stability in the developer's daily work. The flexible merging of parallel work improves productivity, as people (or projects) do not have to wait for each other. The possibility to return to an older version gives courage and security. Traceability and the ability to highlight differences between two versions is a great help both during development and in particular during maintenance (debugging).

SCM manager. The fact that this role exists, indicate that there are also personnel costs from SCM. Someone has to take care of SCM plans and processes. Usually the SCM manager also takes part in the CCB meetings as secretary.

2.4 The resulting model

We are now ready to put all the pieces of costs and benefits together. However, before we can do that, we have to deal with a problem that we mentioned in the Introduction – the fact that that many costs and benefits are subjective or/and hard to quantify.

The aspect of subjectivity has to do with the fact that often we cannot exclude the specific context in which things are carried out. For instance, the degree of benefit we have from being able to handle variants depends on the degree to which we actually have to deal with variant products. Another factor that can cause problems is that it can be difficult or impossible to quantify a certain benefit or cost. For instance it would be difficult to quantify the actual advantage of getting to the market two months earlier than your competition – in some cases it would be a matter of survival of the company and as such the parameter should have an infinitely high value.

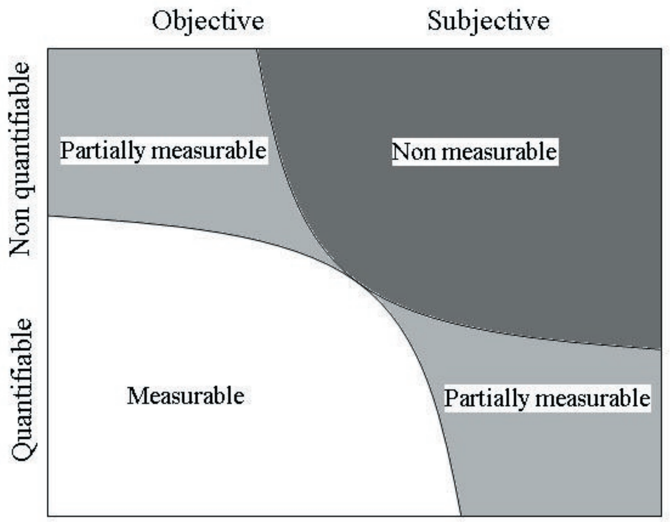

Figure 1. Characterising the measurability of parameters

To reflect these difficulties, we have divided up the parameters into three categories: measurable, partially measurable and not measurable, as shown in Figure 1. Our intention with this categorization is that the measurable parameters can be used in a general, basic ROI model, where the values of the parameters can be taken from common benchmarks and will vary very little from company to company. The partially measurable parameters will not apply to all companies and will have values that can vary wildly from company to company and as such has to be estimated by each company in question. Finally, the not measurable parameters will have to be considered an added bonus to the result from actually calculating a ROI – they do not apply to all companies and it does not make sense to try to estimate values for these parameters.

Using this categorization of costs and benefits, we obtain the two-by-three matrix of parameters shown below in Figure 2. We can see that for the costs, most are measurable, whereas for the benefits most are not measurable. This goes for the number of costs and benefits, not necessarily for their economical impact.

It is not a good property for a ROI model to have a high number of parameters that are not measurable. However, our model is still preliminary and we hope to be able to move parameters towards being measurable as we work with the empirical validation of the model. Furthermore, we can see that for the cost side most parameters are measurable whereas for the benefit side most parameters are not measurable. This means that if we use only measurable and partially measurable parameters to calculate the return on investment, we can be quite sure not to get unpleasant surprises from "hidden" costs and expect even more benefits in addition to the calculated ROI.

The model we present in figure 2 shows only an overview of the parameters. Obviously many details will be needed about the parameters to clarify their exact nature and definition. This can be done [Borracci05]; however, we leave it out here for reasons of space.

	Measurable	Partially measurable	Not measurable
Costs	• Tool costs • Licenses and maintenance costs • Training costs (cost of training the system administrator and the developers to the tool and the new SCM processes • Added work associated with new SCM tasks for the: o Config. Manager o Admin & o Developer technician	• Change process may be more complicated (loss of time and money waiting for authorization, average CCB-time) • Loss of time for doing the status accounting reports (depending on the tool and the level of automation)	• Fear of new procedures
Benefits	• Decrease of the externally reported defects (defect report arrival rate) • Less time per bugfix • Ability to trace the original product through its development • Save time with automated software builds • Manage versions, parallel work, automatic merges • Traceability implies less time for V&V and testing	• Decrease of number of staff changes / help to integrate new employees (less cost of training a new employee) • Allows to handle very complex activities (variation of a product) • Reusing existing code and reducing repetitive development efforts • Gain factor fixing bugs in different variants • Helps the maintenance • Changes are planned, their impact is assessed • Reducing the number of errors	• Employees are happier • Provides for communication and coordination in the group • Pleasure of working in stability with a baseline and an own workspace • Working from home and distributed development • Ability to bring out the product earlier • Decrease the time required to respond to user requests • Assures that the customer gets what he paid for • Audit at the end of each phase assures consistency of the work • Provides visibility of the project • Achieves a sense of organisation and control instead of chaos
	Quantitative <-> Qualitative		

Figure 2. The Return-On-Investment model

We have not stated an explicit formula to calculate the ROI in this paper, but have left it at the model showing the parameters. Such a formula can be made [Borracci05], but it becomes very complex and is probably of little use. What complicates matters the most is that some costs and benefits are one-time (like buying the tool) or once a year (like licenses), whereas others are daily (like the support for parallel work). Yet others are not linear (low benefit until you get to know the tool/process). Therefore it is difficult to make a precise calculation that takes into account all these factors. However, the previous study of Larsen and Roald [LR98] indicates that the ROI in SCM is high enough that such a precise calculation should not be necessary to justify the introduction of SCM tools and processes. When it comes to upgrading tools and/or processes, the number of relevant parameters will probably be low enough to allow a precise calculation. However, in many cases the most interesting will not be to calculate the profitability of SCM, but rather to estimate the SCM costs of a project as the tool(s) and processes are already given by the company standard.

In the present model, we have not considered such parameters as compliance requirements and support of development methods. We believe that these are political issues that will not be influenced by SCM economics nor by the ability or not of SCM to support them – we may, though, be wrong about this.

3 Step Two – Looking for Metrics

Now that we have a model for the costs and benefits of SCM, we can use that model and its parameters to arrive at what we are really looking for – a set of metrics that are targeted specifically at measuring the SCM processes.

The project information base, that the SCM repository constitutes, is an obvious source of data for metrics. In fact the Configuration Status Accounting activity is mostly concerned with putting together data that can be used to monitor and manage projects and processes. However, these metrics are mostly targeted at general software engineering processes and very rarely are the SCM processes ever considered. We know of only the two cases of Farah [Farah04] and Jönsson [Jönsson04] and feel that the field needs more work to advance it to a more mature and complete state.

Just like other processes, SCM processes need to be kept an eye on and to be improved. This cannot be done if we do not have data from a set of SCM specific metrics that can be used to measure the performance of the SCM processes. We need information about the current state to make data-driven decisions about changes. And we need to track our progress to be able to assess the impact of SCM process changes.

In a project course at our department [HBM05], teams have to do four releases during the course of six XP-iterations (each iteration being 14 hours of work). We keep an explicit SCM metric for the time it takes to produce a release (extract code from the repository, compile it and carry out unit and acceptance tests, and put together system, source code, manual and documentation in one package ready to ship to the customer). The first release is done manually in 2-4 hours, teams improve for each release and most teams have an automated forth release – with a record to beat of 38 seconds. The customers guarantee that the teams do not trade quality for speed.

A set of SCM metrics and associated benchmarks can also be used to predict SCM costs for new projects drawing on data from old and current projects. But we need to find and define a set of SCM metrics and to collect data. The SCM metrics will tell us when our SCM processes are working as expected – and more importantly, the anomalies will give us early warning about SCM processes with potential problems.

In parallel with the project where we validate the ROI model, we also intend to establish a tentative set of SCM metrics. During a longer time-span we want to experiment with those metrics to see what stories they can tell about the company's SCM processes and a possible change of tool. We do not expect all the parameters from our ROI model to become useful SCM metrics – and we expect more to pop up.

Already now we have some ideas for what metrics could be used for improving the SCM processes. The time to produce a release, as mentioned above, is just one. Others could be: number of merge conflicts, time to do a configuration audit, accuracy of impact analysis – and many more. However, we still need to do a lot of work here and would like to discuss our preliminary findings and our ideas for the continued work in a forum of experts.

4 Concluding Remarks

We have screened our proposed ROI model with the local branch of a global company. They have found it useful and want to adopt it, not for calculating the ROI of introducing SCM as they already has that in place, but to evaluate the profitability of proposed changes to their SCM tool and/or processes.

During the course of the model's validation, we want to look for a set of SCM specific metrics to help manage and tune SCM processes. The empirical validation of the ROI model will surely show that some parameters have only marginal effect and are best left out in order to reduce the model's complexity. We also expect new parameters to emerge that we have overlooked. And finally, it is our hope that with practical experience, we can move some of the parameters towards being more measurable.

The ROI model that we have presented is just to be considered an intermediate step; our ultimate goal is to uncover a set of "pure" SCM specific metrics. This is work in progress that we want to evaluate and discuss now that we are in the transition from phase one (the ROI model) to phase two (the SCM metrics).

References

[Babich86]: Wayne A. Babich: *Software Configuration Management – Coordination for Team Productivity*, Addison-Wesley Publishing Company, 1986

[Borracci05]: Lorenzo Borracci: *A Return on Investment Model for Software Configuration Management*, Masters Dissertation, Lund Institute of Technology, May 2005.

[Farah04]: Joe Farah: *Metrics and Process Maturity*, The Configuration Management Journal, December 2004.

[Hass03]: Anne Mette Jonassen Hass: *Configuration Management Principles and Practice*, Addison-Wesley Publishing Company, 2003.

[HBM05]: Görel Hedin, Lars Bendix, Boris Magnusson: *Teaching eXtreme Programming to Large Groups of Students*, Journal of Systems and Software, January 2005.

[Jönsson04]: Henrik Jönsson: *Graphs for Change Requests*, The Configuration Management Journal, December 2004.

[LR98]: Jens-Otto Larsen, Helge M. Roald: *Introducing ClearCase as a Process Improvement Experiment*, in proceedings of the SCM-8 Symposium, Brussels, Belgium, 1998.

[Merant01]: *Assessing Return on Investment for Enterprise Change Management Systems*, Merant White Paper, 2001.