SCM: Status and Future Challenges

Reidar Conradi[1] and Bernhard Westfechtel[2]

[1] Norwegian University of Science and Technology (NTNU),
N-7034 Trondheim, Norway.
conradi@idi.ntnu.no
[2] Computer Science III, Aachen University of Technology,
Ahornstrasse 55, D-52074 Aachen, Germany.
bernhard@i3.informatik.rwth-aachen.de

This paper summarizes the state-of-the-art in software configuration management (SCM). Ten SCM themes are discussed and relevant research questions are proposed for each theme. The full version of this paper may be retrieved under http://www-i3.informatik.rwth-aachen.de/private/bernhard/westfechtel.html.

1. The version and product model: one or separate? These two models control the version space and product space, respectively. There are several research questions to clarify:

- *Q1.1: What product model to apply?* Most SCM tools are still based on files. It is highly desirable to support more sophisticated data models for representing typed objects, relationships, and attributes.
- *Q1.2: What version model to apply?* E.g. choose state-based versioning (variants and revisions) or change-based versioning (conditional compilation with more liberal delta combinations)? The two authors have proposed a unifying version model [CW97], capable of supporting both state- and change-based versioning – but limited experience exists.
- *Q1.3: Should the version model and product/object model be orthogonal to each other?* In many proposals, version model and product/object model are intermingled. Separating product/object model and version model appears attractive because the same version model can be combined with different product/object models. However, the implications on database design have still to be explored more thoroughly.

2. How to manage and evolve the meta-information, such as version rules? A software product may evolve into many revisions and variants, and many changes may be applied during its lifetime. Rule-based version construction supports the construction of consistent configurations from an intentional, high-level description. The result of version construction heavily depends on the quality of the version rules and the underlying deductive version engine. Not only does the product evolve; the version rules evolve likewise.

Research questions: Q2.1: To better understand the problem space (understanding the requirements, collecting empirical data on the structure of and evolution of version rules). Q2.2: To better understand the solution space (improving

the deductive capabilities of rule-based version engines and assisting the user in managing the version rules). Q2.3: Give effective user assistance in managing the combinatorial space of configurations and the interplay between product and version structure. Q2.4: In all this, collect empirical data on the structure and evolution of such rules, both for the version and the product space.

3. New media and delta techniques Traditionally, SCM has been applied to sequential texts only, i.e. "source programs" or other lifecycle documents. Efficient delta techniques exist for such texts, reducing the storage demand to 2-3% of the original one [HVT98]. Efficient diff/merge tools are also available. Later on, delta algorithms have been generalized to binary files.

Focusing on text and binary files is no longer sufficient. Multi-media data such as sound, pictures, and video also have to be stored efficiently. So far, this is addressed by compression techniques decoupled from versioning. In addition, we should also consider high-level deltas exploiting structural knowledge about software objects (e.g., deltas for HTML documents [WW98]).

Research questions: Q3.1: To develop new delta techniques for multi-media data. Q3.2: To explore the potential of more high-level structure-oriented deltas.

4. Workspace management and transaction control Central issues here are *high-level (intentional) configuration descriptions* that are expanded into *low-level (extensional) part-lists* to control check-out and check-in of workspaces, either shared ones for groups or private ones for individuals.

Recently some pragmatic tools have emerged for dynamic (re)configuration of files on distributed computers (e.g., file docking, incremental downloading of executables (web-applets), e-mail attachments). In all this, there is weak control in defining and maintaining local configurations.

Research questions: Q4.1: To develop flexible workspace architectures. Q4.2: To develop efficient and pragmatical configuration descriptions to control the above.

5. Distributed and cooperative work and relation to groupware SCM tools provide workspaces for organizing distributed and cooperative work. For example, ClearCase can support distributed workspaces with controlled check-out/in and mutual notifications. In addition, we may need temporary workspaces (bulletin boards) for short-term and dedicated communication and negotiation. Finally, SCM tools may provide cooperative transactions that allow for information interchange *before* check-in.

SCM tools tend to provide product-centered support for distributed and cooperative work. Conversely, groupware tools usually do not consider product structures, when organizing actors, groups, processes, and cooperation patterns. Thus, we have to investigate how to combine groupware and SCM support.

Research questions: Q5.1: To experiment with and assess the support from groupware tools on typical, cooperative SCM-scenarios. Q5.2: To configure and

evolve groupware support based on SCM-maintained information. Q5.3: To develop and validate flexible models for cooperating database transactions.

6. Process support issues Most SCM processes are well established and repetitive, thus ripe for computerized support. Indeed, the more advanced SCM tools offer facilities for process support, e.g. Adele, Continuus and ClearCase.

At least five process areas must be supported: (1) change control and status reporting, which is already supported in many SCM tools; (2) management (project planning, cost estimation, etc.), which is covered by project management tools; (3) QA and auditing, which is supported by many methods and tools with weak coupling to SCM; (4) regeneration, which is well supported by Make and related tools; (5) cooperation/negotiation support, which is provided by groupware tools and some advanced SCM tools. All in all: some of the total process support is supported by the SCM tools, some by a spectrum of other tools.

Research questions: Q6.1: Can required process support for SCM be made independent of basic product/versioning management? Q6.2: What are the mutual dependencies and interfaces of SCM and process tools?

7. SCM tool architecture Software architectures have attracted much attention recently. Since current SCM tools tend to grow larger and larger, developers of such tools would clearly benefit from a well-defined architecture. However, virtually no such proposal exists, although we have outlined a layered architecture with coarse-grained components, such as: a versioned database (the "facts"), a rule base, a version engine for defining and evaluating rules, a workspace manager, a transaction manager, etc. in an earlier paper [CW97]. The layering is still subject to debate — as well the scope of SCM (e.g., to what extent is process support included?).

Research questions: Q7.1: What are kernel SCM functionalities? Q7.2: How should the layering and interfaces be?

8. Industrial Experiences SCM is an established and recognized area of software engineering, with a spectrum of methods, techniques and tools available. We should then expect that there is a huge body of empirical data to demonstrate its effectiveness – but not so.

Much data has been collected to assert the effectiveness of delta storage on the low end. At the high end, it is much more difficult to obtain measures of productivity improvement resulting from the introduction of SCM tools. Some work has been done e.g. in the European ESSI program, where several process improvement experiments were concerned with the introduction of SCM tools. One of these reported a 36 % reduction in external reports of major errors after ClearCase had been installed, although no strict causality can be assumed.

Research questions: Q8.1: Design a common and moderate metrics to assess the impact of SCM tools, both on the product and process side. Q8.2: Perform empirical studies from industry, using such metrics.

9. SCM Maturity Models Both ISO-9001 and CMM (level 2) require product management, i.e. SCM. However, there are many aspects to consider when introducing a SCM tool into an organization. Thus, it makes sense to distinguish between levels of SCM maturity, e.g.:

1. No SCM procedures – chaos.
2. Get the overall process in shape. Organize old/new versions in different catalogs.
3. Introduce simple SCM tools, such as RCS / SCCS and Make. Improve the process.
4. Upgrade to a medium-level SCM tool, like CVS, with explicit product definitions. Consolidate the process.
5. Finally introduce a more complete SCM tool, like ClearCase. Also allow a distributed process.
6. Total SCM, with complete support processes across projects and products.

Research questions: Q9.1: How to design such an maturity scale? Q9.2: How to use and validate such an maturity scale?

10. SCM technologies also for CAD/CAM, VLSI and office automation Managing consistent configurations of versioned documents is a problem that occurs not only in software engineering, but also in electrical, mechanical or chemical engineering, in office automation, etc. Although many similarities do exist, disciplines such as SCM and EDM/PDM (engineering/product data management) and corresponding tools have evolved fairly independently.

Research questions: Q10.1: To investigate whether common version- and configuration management are applicable on hybrid hw/sw products. This may assume a common base model, and how to separate domain-independent from domain-specific aspects.

Acknowledgements Thanks go to our nearest colleagues and to many discussions in the SCM workshop series.

References

[CW97] Reidar Conradi and Bernhard Westfechtel. Towards a Uniform Version Model for Software Configuration Management. In *Reidar Conradi (Ed.): "Proc. Sixth International Workshop on Software Configuration Management (SCM'7)"*, pages 1–17, Boston, USA, 18–19 May 1997. Springer Verlag LNCS 1235.

[HVT98] James Hunt, Kiem-Phong Vo, and Walter Tichy. Delta algorithms: An empirical evaluation. *ACM Transactions on Software Engineering and Methodology*, 7(2):192–214, April 1998.

[WW98] E. James Whitehead and Meredith Wiggins. WEBDAV: IETF Standard for Collaborative Authoring on the Web. *IEEE Internet Computing*, pages 34–40, Sep/Oct 1998.