Change Sets Revisited Configuration Management of Complex Documents

MacKay, Stephen

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

Change Sets Revisited and Configuration Management of Complex Documents

(Position Paper)

Stephen A. MacKay

Software Engineering

March 1996

Canada

NRC No. 40194

This paper also appears in the *Proceedings of the 6th International Workshop on Software Configuration Management (SCM-6)*, Berlin, Mar. 25-26, 1996. Published as *Lecture Notes in Computer Science (LNCS) 1167, Software Configuration Management*. I. Sommerville (Editor). Springer-Verlag, Berlin. 1996. pp 277-281.

ISSN 0302-9743

ISBN 3-540-61964-X

Additional copies are available free of charge from:

Communications Office
Institute for Information Technology
National Research Council of Canada
Ottawa, Ontario, Canada
K1A 0R6

Des exemplaires supplémentaires peuvent être obtenus gratuitement à l’addresse suivante:

Bureau des communications
Institut de technologie de l’information
Conseil national de recherches du Canada
Ottawa (Ontario) Canada
K1A 0R6

Change Sets Revisited and Configuration Management of Complex Documents

Stephen A. MacKay

Institute for Information Technology—Software Engineering
National Research Council of Canada
Ottawa, Ontario, Canada K1A 0R6
MacKay@iit.nrc.ca http://wwwsel.iit.nrc.ca/

1 Introduction

The SCM-5 workshop in Seattle provided a forum for software configuration management (SCM) researchers, tool developers and users to come together and discuss relevant problems in the field. The workshop concluded with a number of unsolved problem areas. This document summarizes those areas and—drawing from our research experiences—discusses a few of the challenges in greater detail.

2 What are the Problems?

2.1 SCM models

Feiler's models of configuration management (check-out/check-in, composition, long transaction and change set) [Feil91] no longer adequately represent the current generation of commercial configuration management tools or the emerging tools and research systems. Are there better or expanded models to represent workspace concepts? Do we need new models for concurrent development in a widely distributed environment, or can we adapt the existing ones? Are there graphic representations and visualization methods better than the overworked version graphs? Is modelling just an irrelevant academic exercise?

2.2 SCM architecture

Software projects involving multiple companies benefit from common configuration management tools. However, each group is reluctant to change its own culture. Similarly, customers obtaining updates and patches from development tool vendors are not likely to use the same CM tools as the vendors. There is a clear need to separate the architecture of CM tools from the implementations. Can we define a common architecture for commercial CM tools that would allow software teams to interact even if they are using CM tools from different vendors? Are there existing relevant standards?

2.3 SCM and process

The popularity of ISO 9000 and CMM certification has made companies more aware of software development processes. What is the relationship between SCM and the overall software process? Is CM merely one part of the software process or is CM itself the process? How do the various standards on software processes view CM?

2.4 Distributed, concurrent development

Commercial CM tool vendors are beginning to provide support for widely distributed, concurrent software development, but there is little agreement on the mechanisms. Are there models or appropriate graphic representations of distributed, concurrent development that would aid in user understanding and acceptance of this powerful paradigm? What are the significant problem areas (scale, merging, group dynamics, etc.)?

2.5 CM of complex documents

Most current CM systems store non-textual configuration items in the repository as frozen "binary" entities (possibly compressed). This is unsatisfactory in 1996. How can we do proper CM of word processor produced documentation; multimedia documents; databases; project files for advanced graphical user interface generators; and "source code" for non-textual languages? How can we determine what has changed in a non-textual configuration item? Can we represent or determine the differences between products composed of more complex components?

The remainder of this document begins by looking at CM models for widely distributed, concurrent, software development projects. It then continues the discussion of change sets begun in Seattle. It concludes with a brief discussion of some of our preliminary thoughts on configuration management of non-textual components.

3 Models for Concurrent, Distributed Development

Today's new culture of software development relies on teams of developers equipped with desktop workstations or personal computers. The teams are frequently distributed worldwide and may not be reliably networked. This environment brings special problems, particularly in areas such as: distribution across time zones; access to the repository by intermittently connected developers; and sharing the repository across company boundaries.

3.1 Version-oriented CM

Version-oriented configuration management focuses on defining and managing product versions through the handling of revisions and variants at the individual component level. The component and product versions are the first-class entities, managed by the developer. One of the common features among version-oriented models is the use of the directed acyclic version graph. Each node in the graph represents a version of the component or product and each edge between nodes represents the transition between versions (an *is-version-of* relationship or the "delta" between the versions) [vand95].

At the component level, version graphs quickly prove inadequate. While they can easily represent the migration path for a short time, they do not scale for longer projects nor do they handle components that are undergoing significant concurrent modification [MacK95]. Trying to study relationships between components using their individual or combined version graphs is difficult. The pictures provide a clear history of each individual component, but are of little help in determining which version of one is related to which version of the other.

Product-level version graphs usually result in a simpler picture, but they do not convey enough information. For example, when analyzing product migration, it is difficult to determine what constitutes a change between versions or how two arbitrary versions are related because the deltas (edges) are not first-class entities. For highly portable products with many active versions, it is difficult to express the application of a single change to a variety of versions.

For concurrent development, version graphs introduce artificial branches that have little to do with the structure of the product, making the model more difficult to understand and maintain. Commercial CM systems implementing version-oriented models, usually discourage branching for full concurrency, even though it is the only mechanism they provide for development to proceed simultaneously on a single configuration item [MacK95].

3.2 Change-oriented CM

Change-oriented configuration management focuses on managing logical changes to a baselined product. Here, the description of the change—known as a change set—is a first-class entity, managed by the developer. The versions are derived by applying relevant change sets to the baseline. Developers therefore work with product-level deltas, collecting all those individual components that are relevant to the particular change, excluding other groupings that made sense in other situations (like initial product design). This structure reduces considerably the difficulty of managing the revision and release process [Wein95].

Feiler notes that concurrency control is outside the change set model, but he goes on to state:

> Change sets can also be used to support distributed concurrent change without centralized coordination. Each site generates change sets independently. Once the changes sets are exchanged between sites, each site can, at its leisure, combine change sets. The result is that the system evolves at both sites. If assignment of changes to sites is planned carefully, conflicts in change sets can be kept to a minimum. [Feil91, pg. 43]

Managed carefully and supported with appropriate CM tools, change sets provide exactly the concurrency management required in the widely distributed development environment. Importantly, the mechanism scales down to smaller teams as well.

The *workspace* mechanism [Dart90, Dart92]—where developers can get and modify components from the repository independently of other developers—is a natural way of implementing change sets. Augmented with

Dart's *transparent view* and *transaction* mechanisms, the change set model becomes a powerful and complete method for describing configuration management in widely distributed environments.

Change sets have often been viewed unfavourably, characterized as a *Chinese menu* approach in which individual revisions are tracked and then collected into logical groups to define a version. Often a check-out/check-in methodology is used to manage the revisions. This approach represents a limited view of change sets, trying to superimpose a version graph on the change set model. The research community needs to find representations and visualizations that free us from version graphs.

Two visualization techniques, described at SCM-5, provide a starting point. The *Database and Selectors Cel (DaSC)* approach, developed in our laboratories at the National Research Council of Canada, characterizes change sets as groups of layers stacked on top of a known baseline [Gent89, MacK95]. Tandem Computers' *Fully Populated Paths* mechanism uses Railroad Diagrams (resembling DaSC laid on its side) to show the relationships among change sets [Schw95]. Railroad diagrams look familiar to people comfortable with version graphs, but they convey significantly more information. One of the useful outputs from SCM-6 would be progress towards a uniform graphical notation for change-oriented configuration management.

4 CM of Complex Documents

The future of software development will not remain focused on managing changes to files containing only ASCII text. Already developers—even in traditional environments—are faced with revisions of: documentation produced by word processors or page layout programs; test case data stored in databases; soft-copies of design drawings; and binary resource descriptions. We are now beginning to add to the mix: multimedia and hypertext documentation (e.g., HTML, HyperCard, etc., with embedded sound and video); data maintained in personal or shared productivity tools (e.g., Lotus Notes); project files for advanced graphical user interface generators (e.g., XVT) and compilation environments; and even full visual programming languages (e.g., Prograph CPX). Full configuration management of these components is difficult, so little commercial CM tool support is available. Most tools only permit storage of a complete, compressed copy of the component in the repository. A few, like Voodoo, store a compact delta of the binary files.

We believe change-oriented methodologies, particularly DaSC, will support a number of these new application areas. We have recently begun exploring some of them, but it is too early to publish results. Two clear issues have emerged. Managing revisions while editing documents stored in proprietary formats is extremely difficult. The vendors of the tools that create these documents must provide: a powerful document editor with appropriate calls to manage a change set methodology; a document editor with sufficient hooks to allow the addition of extra functionality; or enough information about the document formats to allow companion tools to be written. If they fail to meet this challenge, customers will migrate to competing vendors. There is a great challenge for the CM research community to investigate ways to bring the variety of documents under common configuration management.

Another challenge is in representing the differences between products composed of more complex components. Whether we are looking for tools to automatically generate differences between two known versions, or for representations of the differences that the software developer can visualize and manipulate, the problem is equally challenging. There are many opportunities for discussion and further research on this topic alone.

5 Acknowledgments

I would like to thank my colleagues, past and present, in the Software Engineering Group for their many contributions to our DaSC project and for reviewing this position statement. I would like to thank especially Charles Gauthier, Morven Gentleman, Anatol Kark, Darlene Stewart and Marceli Wein for their efforts and support.

6 References

[Dart90] Susan Dart. Spectrum of functionality in configuration management systems. Carnegie Mellon University, Software Engineering Institute *Technical Report: CMU/SEI-90-TR-11*, Dec. 1990. 38 pages.

[Dart92] Susan Dart. The past, present, and future of configuration management. Carnegie Mellon University, Software Engineering Institute *Technical Report: CMU/SEI-92-TR-8*, Jul. 1992. 28 pages.

[Feil91] Peter Feiler. Configuration management models in commercial environments. Carnegie Mellon University, Software Engineering Institute *Technical Report: CMU/SEI-91-TR-7*, Mar. 1991. 54 pages.

[Gent89] W.M. Gentleman, S.A. MacKay, D.A. Stewart, and M. Wein. Commercial realtime software needs different configuration management. *Proceedings of 2nd International Workshop on Software*

Configuration Management (SCM), Princeton, NJ. Oct. 24–27, 1989. Published as *Software Eng. Notes*, 17(7): 152–161; 1989. NRC 30695.

[MacK95] Stephen A. MacKay. The State of the Art in Concurrent, Distributed Configuration Management. *Proceedings of 5th International Workshop on Software Configuration Management (SCM-5)*, Seattle, WA. Apr. 24–25, 1995.

[Schw95] Bill Schweitzer. Fully Populated Paths: A Conservative, Simple Model for Parallel Development. *Proceedings of 5th International Workshop on Software Configuration Management (SCM-5)*, Seattle, WA. Apr. 24–25, 1995.

[vand95] André van der Hoek, Dennis Heimbigner, and Alexander Wolf. Does Configuration Management Research Have a Future? *Proceedings of 5th International Workshop on Software Configuration Management (SCM-5)*, Seattle, WA. Apr. 24–25, 1995.

[Wein95] M. Wein, S. A. MacKay, D. A. Stewart, C.-A. Gauthier and W. M. Gentleman. Evolution Is Essential for Software Tool Development. *Proceedings of the 1995 International Workshop on Computer-Aided Software Engineering (CASE-95)*, Toronto, Ontario, Jul. 9–14, 1995.