ISSN 2278-3091
Volume 9, No.1.3, 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse0891.32020.pdf
https://doi.org/10.30534/ijatcse/2020/0891.32020

The Evolution of Software Configuration Management

Syahrul Fahmy[1], Aziz Deraman[2], Jamaiah Yahaya[3], Akhyari Nasir[1], Nooraida Shamsudin[1]
[1]University CollegeTATI, Malaysia, fahmy@tatiuc.edu.my
[2]Universiti Malaysia Terengganu, Malaysia
[3]Universiti Kebangsaan Malaysia, Malaysia

ABSTRACT

Software Configuration Management (SCM) is a discipline in software engineering for managing changes to software products using standard processes and tools. This article presents the evolution of SCM since its inception, highlighting the components, application to other areas, change management and software quality. Research and development in SCM are highly motivated by the problems at hand in software development. SCM process and activities are sound, guided by international standards and industry best practice. Commercial and proprietary tools are aplenty, and the underlying techniques are no longer confined to SCM. SCM has been applied to other areas since the turn of the century and change management has become a tool-oriented process, rather than a management-oriented process. The role of human in SCM has yet to be studied extensively compared to other areas in software engineering. Software quality is associated with defects and quality factors are measured differently based on projects and metrics.

Key words: Software Configuration Management, Software Engineering, Software Testing, Software Quality.

1. INTRODUCTION

Software Configuration Management (SCM) can be loosely defined as "the ability of control and manage changes in a software project". It is part of a larger field of Configuration Management (CM), and primarily used to control the evolution of software systems [1]. Formal definitions from IEEE, the Software Engineering Institute and ISO are:

> "a supporting-software life cycle process that benefits project management, development and maintenance activities, quality assurance activities, as well as the customers and users of the end product" [2].
>
> "discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements" [3].
>
> "a management activity that applies technical and administrative direction over the life cycle of a product, its configuration items, and related product configuration information. It provides identification and traceability, the status of achievement, and access to accurate information in all phases of the life cycle" [4].

This paper presents the evolution of SCM, discussing its components, application to other areas, change management and software quality.

2. EVOLUTION

SCM can be traced back to the aerospace industry in the 1950s. Poorly documented engineering changes posed a problem to spacecraft production, and configuration management was applied to address it. When software managers faced similar problems of managing change, similar approach was adapted in the software development process [5].

2.1 1960s

Only a handful of SCM-related works were carried out and they were heavily funded by the government. Several SCM concepts was introduced including management concept for software systems [6]; concepts for documentation and procedures in SCM [7]; configuration as a control mechanism in software development [8]; and concepts for program specifications [9]. It is worth mentioning that knowledge regarding early SCM research and systems has disappeared as dedicated platform such as software engineering scientific conferences did not exist [5]. In addition, SCM was largely integrated in the operating systems and documentation describing early SCM systems is difficult to find.

2.2 1970s

The Software Crisis amplified software manager's problems of project cost and schedule overrun. Problems in software development were recognized and in 1973, software engineering was accepted as the solution to software manager's problem.

The discipline of SCM was presented by Bersoff in 1978, outlining its components and role in the management process of the system life-cycle [10] and SCM was formally discussed in the International Conference of Software Engineering in 1979 [11].

Majority of research focused on the concepts and tools for SCM. These include approach for adapting CM in the software development life-cycle [12]; the Source Code Control System (SCCS) to control changes to source code [13]; Software Upgrade for systems generation and CM [14]; and Make, a program for maintaining up-to-date versions of programs [15]. Variants of SCCS and Make are still in use today.

2.3 1980s

Software development started embarking on programming large and complex software systems (Programming-in-Large). Early SCM systems focused on file control, emphasizing on versioning, building and composition. Works in SCM concepts continued with CM concepts in software life-cycle [16]; configuration control in software life-cycle [17]; configuration control in software process [18]; and system design documentation for CM and software control [19].

Interest in SCM modelling started to grow for example model for version and configuration control [20]; model for change request [21]; model of a CM environment [22]; framework for an active SCM system [23]; and framework for integrating CM and process management [24].

Development of tools thrived during the 1980s with with RCS [25]; LIFESPAN [26]; Adele [27-28]; Source Control System [29]; Portable Configuration Management [30]; configuration management toolkit [31]; and Configuration Management Assistant [32].

2.4 1990s

As non-textual objects become common, new algorithms for storing and retrieving objects were needed. SCM started utilizing relational database and many of the SCM systems as we know it today came into the limelight.

Concepts of programming large software systems continued such as CM to manage large design projects [33]; procedures and tools for modifying large software systems [34]; and Programming-in-the-Large concepts including SCM [35].

General SCM concepts include user concepts of existing CM systems [36]; concepts of version control and CM in Ada [37]; effects of software development models on SCM [38]; and configuration approach to manage software development [39].

Conceptual SCM works has also attracted a lot of technical interest for example version control in hypertext systems [40]; distributed versioning on the Web [41]; lazy architecture for controlling change [42]; hierarchical and heuristics change detection [43-44]; event and lock mechanisms [45]; software merges using program slicing [46]; operation-based merging [47]; and integration algorithm [48].

Works in modelling too, has shifted to a more technical nature for example formal model for CM activities [49]; object-oriented semantic model of SCM [50]; process-oriented version and configuration control model [51]; model to support reuse [52]; model for configuration and version management [53]; model for managing changes to items of various types [54]; model for identifying and manipulating shared components in a software configuration [55]; framework for programming environments that handles versions and configurations [56]; and framework for process and version modelling [57].

Works in tools continued with HMS to support revision control and CM [58]; VMCM, a PCTE-based version and CM system [59]; change request management system [60]; Perforce SCM system [61]; automation of SCM in a project management system [62]; SRM to support software release management [63]; Distributed Version Control System to support software version control in distributed environment [64]; EPOS extensions for SCM [65]; APPL/A, to support change management [66]; structure-oriented merge tool for software documents [67]; and AVCS, an APL-oriented version control system [68].

2.5 2000s

Software development moved to a more distributed and heterogeneous environment where web services and Global Software Development transformed traditional software development landscape into a more decentralized platform.

Conceptual works continued focusing on technical aspects of SCM for example the use of aggregates as first-order entities to manage fine-grained artefacts [69]; scheme for revision identification of released components [70]; algorithm for detecting and visualizing structural changes [71]; application of aspect oriented programming to SCM to improve change control [72]; integration of collaboration into IDEs [73]; functional programming language for building software systems [74]; concern separation in the Stellation SCM system [75]; architecture evolution environment that integrates CM and architectural concepts [76]; impact of change to software product quality [77]; and change management process in the production system [78].

Works in modelling continued focusing on technical aspects of SCM for example semantic conflict detection for modelling language independent version control system [79]; model to support fine-grained version control [80]; formal merge

semantics [81]; language-specific operations for integration [82]; approaches for constructing version management tools [83]; technique for expressing fine-grained change [84]; mechanism for history-based change [85]; structure for uniform version management in component-based systems [86]; approach for conflict detection and resolution on models [87]; SCM approach for unified models [88]; component-based SCM model [89]; approach to modelling builds and class of build optimizations [90]; versioned hypermedia framework built on top of a SCM system [91]; return on investment model for in SCM [92]; and model and process for self-adaptation in change management [93].

Works in tools include TRICA, an integration tool coupled with SCM and issue tracking [94]; Palantir, a workspace awareness tool for SCM systems [95-96]; SCA, to detect semantic interference between parallel changes [97]; PARCS, to provide feedback to developers as to how a change affects the behaviour and performance of the overall application [98]; Odyssey-VCS, a version control system for fine-grained UML model elements [99]; Clever, a clone-aware SCM system [100]; Molhado, a hypertext versioning and SCM system [101-102]; Coven, an integrated programming environment and SCM system [103]; and MolhadoRef, a semantics-based SCM system [104].

2.6 2010s

SCM challenges in the 2010s include component-based development, dynamically bound and reconfigured systems, and web-based systems. Conceptual works in SCM include evolutionary software development and software change [105]; vision of software evolution based on a feature-oriented perspectives [106]; strategy that leverages common SCM patterns for software development [107]; management of software release [108]; obstacles to CM success in the aerospace and defence industries [109]; CM concepts and principles in distributed development teams [110]; CM and the process of maintaining system integrity [111]; integration decisions by release managers [112]; and hidden costs in merging changes [113].

Works in modelling continued on the technical path with a lightweight solution to version incompatibility [114]; controlled delegation and authorization scheme for version control systems [115]; change tracking algorithm for measuring changes to single lines of code [116]; real-time integration with automatic conflict detection [117]; automatic technique for constructing fine-grained version control system from an existing SCM repository [118]; operation-based conflict detection [119]; versioning support for mashup environments [120]; framework to detect changes between distinct versions of source code [121]; empirical study of build maintenance [122]; methodology to compute effort and evaluate the quality of merge algorithms [123]; framework to scope a change impact analysis technique [124]; approach to automatically identify minimal number of code modifications [125]; and approach to improve multi-stakeholder configuration process [126].

Works in tools include Pluto, a build system with incremental building [127]; Gitless, an open-source distributed version control system [128]; FSTMERGE for semi-structured merge operations [129]; EMFStore, an operation-based Version Control System for models [130]; Storyteller, a version control system to support software developers learning activities in collaborative development environment [131]; VERCAST, a version control system for managing application states [132]; Cored, a collaborative development environment [133]; NEEDFEED, a system that models code relevance to personalize a developer's change notification feed [134]; and CASI, a tool that informs the developers of changes that are taking place and the source code influenced by them [135].

Summarizing the evolution of SCM, research and development in this field are highly motivated by the problems at hand in software development. This is evident through R&D in programming large software systems in the 1980s; object-oriented systems in the 1990s; web services in the 2000s; and late binding systems in the 2010s (Table 1).

Table 1: Evolution of Software Configuration Management

	1950s	1960s	1970s	1980s	1990s	2000s	2010s
ISSUES	Documentation of engineering changes	Managing changes in software development	Software projects running over-time and over-budget	Large and complex software solutions	Object-oriented programming	Remote code management	Software as a Service
SOLUTION	*Configuration Management* theories and practices adopted into software development		Adaptation of SCM into software development process		Application of SCM process and tools into other areas and disciplines in software engineering		
R&D FOCUS	SCM concepts		SCM in software development life-cycle and SCM components modelling		SCM in dedicated development environments and granularity of artifacts		
STANDARDS			SCM acknowledged as a field in Software Engineering	828-1983 - IEEE Standard for Software Configuration Management Plans	828-1990 - IEEE Standard for Software Configuration Management Plans	828-2005 - IEEE Standard for Software Configuration Management Plans	828-2012 IEEE Standard for Configuration Management
					ISO 10007:1995 Quality Management - Guidelines for Configuration Management	ISO 10007:2003 Quality Management Systems - Guidelines for Configuration Management	ISO 10007:2017 Quality Management Systems - Guidelines for Configuration Management
TOOLS			First SCM tools	First integrated SCM systems	Workspace support in SCM systems	Process support in SCM systems	Distributed collaboration capability in SCM systems

3. COMPONENTS

There are three major aspects or components in SCM namely Process and Documentation, Tools, and People.

3.1 Process and Documentation

The concepts of SCM started as early as the 1960s with the ideas of configuration as a control mechanism in software development [8]; and concepts for program specifications [9]. The process underwent formalization throughout the 1970s with the recognition of software engineering as a new field in computing.

In 1983 IEEE published the first standard for SCM, the IEEE 828 Standard for Software Configuration Management Plans, which was revised in 1990, 1998, and 2005. The latest version was released in 2012 [136]. IEEE 828 establishes the minimum requirements for configuration management processes in systems and software engineering. ISO published a quality-related standard for SCM in 1995, the ISO 10007 Quality Management - Guidelines for Configuration Management, which was revised in 2003. The latest version was released in 2017 [137]. ISO 10007 provides guidance on the use of configuration management within the organization.

In addition to SCM-specific standards, there are also general standards related to SCM including IEEE 15939 [138]; ISO/IEC 15939 [139]; ISO/IEC/IEEE 24765 [140]; ISO/IEC 12207 [141]; and ISO/IEC 15288 [142].

Based on these standards, the generic process for SCM involves Management and Planning; Software Configuration Identification; Software Configuration Control; Software Configuration Status Accounting; Software Configuration Auditing; and Software Release Management and Delivery.

3.2 Tools

The corpus of research in SCM has been in the modelling and development of tools to address the issues at hand. This has translated into a plethora of tools and systems for SCM. As such, SCM implementation has been highly dependent on tools as reported by [143-146].

The first SCM tools emerged in the 1970s and targeted specific functionality for example SCSS [13] and Make [15]. First integrated SCM systems appeared in the 1980s, developed in-house and focussed mainly on file control. Examples include DSEE [147]; Adele [27]; and Aide-De-Camp [148]. Workspace and process supports were integrated into SCM systems throughout the 1990s and 2000s for example SourceSafe [149]; Sun/Forte [150]; Subversion [151]; CM/Synergy [152]; and ClearCase [153]. By 2010s, SCM systems have incorporated distributed collaboration capabilities for example Palantir [154]; Gitless [128]; and EMFStore [130].

To date, basic SCM tools are pervasive and the underlying techniques are no longer confined to SCM, but also in other areas including web protocol, services, and programming environments. Three types of tools that are common in SCM are versioning tools such as VERCAST [132], software build tools such as Pluto [127], and software release tools [155].

3.3 People

Works regarding people or human in SCM has focused on the process and outcome of SCM implementation for example:

- To analyse collaboration activities such as conflict history data [156]; revision history [157]; and the impact of a change made by one developer to other developers[124].
- To report the result of SCM tools and approaches in the classroom for example the use of version control system in student development projects [158]; implementation of a distributed revision control system as part of the undergraduate and graduate curriculums [159]; and the integration of configuration management into the IT curriculum [160].
- To identify the correlation between a commit's social characteristics and bugs [161]; the organization of bug reports into sets for effective management by developers [162]; and the relationship between developers' communication frequency and number of bugs [163] in debugging activities.

However, the responsibilities and activities that should be carried out by people are mentioned in SCM standards for example a generic skill set for SCM is outlined in the IEEE Software Engineering Competency Model [164].

4. APPLICATION TO OTHER AREAS IN SOFTWARE ENGINEERING

For the last 10 years, there have been much interest in the application of SCM to other areas in software engineering. This include Crosscutting Frameworks [165]; Scientific Workflow Management Systems [166]; embedded software systems [167]; and Product Line Engineering [168-169].

There are also interest in the integration of SCM and other systems for example integration of version control, security analysis and patching support [170]; integration of architectural and configuration management system [171]; integration of branch and merge functionalities to LibreOffice [172]; modification of BitKeeper, a distributed version control system to handle product line requirements [173]; change-aware process environment for system and software engineering [174]; integration of programming language technology and version control [175]; and the integration of SCM techniques and reference modelling to manage model variants [176].

There are also interest in the area of Data Mining where SCM's repository data is used to predict pre-release defects [177]; guide programmers regarding change [178]; measure the extent of delay [179]; understanding problems in parallel development [180]; and to investigate collaboration efforts in open source projects [181].

The trend of applying SCM processes and tools to other areas and systems are expected to continue in the future especially in the fields of Big Data, Internet of Things and Crowdsourcing.

5. CHANGE MANAGEMENT

The initial concept of change management in SCM seems to have disintegrated throughout the years. The management of change is now focused at a finer level such as source codes [116]; Java entities [118]; models [130]; product variants [182]; and documents [172]. Change management has also been delegated to SCM systems and tools such as Eclipse [183] and Git [128]. These factors has made change management to be a tool-oriented process, rather than a management-oriented process.

6. SOFTWARE QUALITY

Interest in software quality for SCM started as early as the 1980s with studies of change control procedures in a Software Quality Control program [184]; a Software Quality Assurance program outlining the role of SCM in the implementation and maintenance phase [185]; and configuration management for attaining quality assurance [186].

To date, quality in SCM is mainly associated with source code defects and efforts are directed at reducing or eliminating them. Examples include real-time quality control through the analysis of code change [187]; consecutive changes and software defects [188]; tool for estimating defects and changes in software systems [189]; defect prediction with heterogeneous metric sets [190]; mechanism for presenting software defect metrics to aid analysis [191]; impact of product development strategy on defects [192]; and a decision support system to predict defects and enhance release management [177,193].

7. CONCLUSION

This paper has presented the evolution of Software Configuration Management, highlighting the components, application to other areas, change management, and software quality. In a nutshell, research and development in SCM are highly motivated by the problems at hand in software development. This is evident through efforts focusing on programming large software systems in the 1980s, object-oriented paradigm in the 1990s, web services in the 2000s, and late binding systems in the 2010s.

The process and activities in SCM is mature with the publications of international standards since the 1980s and revised periodically. Most of the research in SCM are technical in nature involving SCM concepts, models and tools.Commercial and proprietary tools are aplenty, and the underlying techniques are no longer confined to SCM, but in other areas as well such as web services.

SCM concepts and tools have been extensively applied to other areas since early 2000s and the trend is expected to continue. Change management has become a tool-oriented process, rather than a management-oriented process, drifting away from the initial purpose of SCM. The role of human in SCM has yet to be studied extensively compared to other areas in software engineering. Software quality in SCM is mainly associated with defects and quality factors are subjective and measured differently based on projects and metrics.

REFERENCES

1. Babich, W.A. (1986). Software Configuration Management, Coordination for Team Productivity. 1st ed. Boston: Addison-Wesley.
2. SWEBOK. (2014). Guide to the Software Engineering Body of Knowledge. IEEE Computer Society Press, Los Alamitos, CA, USA.
3. CMMI. (2010). CMMI for Development Version 1.3. Software Engineering Institute.
4. ISO 10007. (2017). Quality Management Systems - Guidelines for Configuration Management. International Organization for Standardization. (10 pages).
5. Estublier, J., Leblang, D., van der Hoek, A., Conradi, R., Clemm, G., Tichy, W., and Wiborg-Weber, D. (2005). Impact of Software Engineering Research on the Practice of Software Configuration Management. ACM Transactions on Software Engineering Methodology. 14(4): 383-430. https://doi.org/10.1145/1101815.1101817
6. Ratynski, M.V. (1967). The Air Force Computer Program Acquisition Concept. In Proceedings of the April 18-20, 1967, Spring Joint Computer Conference. NY, USA, 33-44. https://doi.org/10.1145/1465482.1465488
7. Searle, L.V., and Neil, G. (1967). Configuration Management of Computer Programs by the Air Force: Principles and Documentation. In Proceedings of the April 18-20, 1967, Spring Joint Computer Conf. NY, USA, 45-49. https://doi.org/10.1145/1465482.1465489
8. Oettinger, A.G. (1964). A Bull's Eye View of Management and Engineering Information Systems. In Proceedings of the 1964 19th ACM National Conference. NY, USA, 21.1-21.14.
9. Liebowitz, B.H. (1967). The Technical Specification: Key to Management Control of Computer Programming. In Proceedings of the April 18-20, 1967, Spring Joint Computer Conference, Atlantic City, USA, 51-59.

10. Bersoff, E.H., Henderson, V.D., and Siegel, S.G. (1978). Software Configuration Management. SIGSOFT Software Engineering Notes 3(5): 9-17. https://doi.org/10.1145/953579.811093
11. Sullivan, J.T., Correll, C.H., Pouzin, L., Lanzarone, G., Elliott, I.R., and Duby, J. (1979). Is Software Development Manageable? In Proceedings of the 4th International Conference on Software Engineering, Munich, Germany.
12. Caudill, R. (1977). Understanding the Developmental Life Cycle. In Proceedings of the June 13-16, 1977, National Computer Conference. New York, USA, 269-275. https://doi.org/10.1145/1499402.1499449
13. Rochkind, M.J. (1975). The Source Code Control System. IEEE Transaction on Software Engineering, 1(4): 364–370.
14. Pedersen, J.T., and Buckle, J.K. (1978). Kongsberg's Road to an Industrial Software Methodology. In Proceedings of the 3rd International Conference on Software Engineering, Atlanta, Georgia, USA, 85-93. https://doi.org/10.1109/TSE.1978.231510
15. Feldman, S.I. (1979). Make - A Program for Maintaining Computer Programs. Software: Practice and Experience, 9(3): 255–265. https://doi.org/10.1002/spe.4380090402
16. Bell, T.E. (1981). Structured Life-Cycle Assumptions. In Proceedings of the 1981 ACM Workshop/ Symposium on Measurement and Evaluation of Software Quality, Harold J. Highland (Ed.). ACM, New York, USA, 1-3.
17. Bryan, W., Siegel, S., and Whiteleather, G. (1981). An Approach to Software Configuration Control. In Proceedings of the 1981 ACM Workshop/ Symposium on Measurement and Evaluation of Software Quality, Harold J. Highland (Ed.). ACM, New York, USA, 33-47.
18. Berlack, H.R. (1981). Implementing Software Configuration Control in the Structured Programming Environment. In Proceedings of the 1981 ACM Workshop/Symposium on Measurement and Evaluation of Software Quality, Harold J. Highland (Ed.). ACM, New York, USA, 57-77. https://doi.org/10.1145/800003.807909
19. Burlakoff, M. (1984). An Approach to Software Design Documentation. In Proceedings of the 1984 Annual Conference of the ACM on the Fifth Generation Challenge, Richard L. Muller and James J. Pottmyer (Eds.). ACM, New York, USA, 116-120.
20. Walpole, J., Blair, G.S., Malik, J., and Nicol, J.R. (1988). A Unifying Model for Consistent Distributed SW Development Environments. SIGPLAN Notices, 24(2): 183-190.
21. Lacroix, M. and Lavency, P. (1989). The Change Request Process. The 2nd International Workshop on Software Configuration Management, Princeton, USA, 122-125. https://doi.org/10.1145/72910.73357
22. Miller, D.B., Stockton, R.G., and Krueger, C.W. (1989). An Inverted Approach to Configuration Management. In Proceedings of the 2nd International Workshop on Software Configuration Management, Princeton, USA, 1-4.
23. Sibley, E.H., Scallan, P.G., and Clemons, E.K. (1981). The Software Configuration Management Database. In Proceedings of the May 4-7, 1981, National Computer Conference, Chicago, USA, 249-255.
24. Bernard, Y. and Lavency, P. (1989) A Process-Oriented Approach to Configuration Management. In Proceedings of the 11th International Conference on Software Engineering, Pittsburgh, USA, 320-330. https://doi.org/10.1145/74587.74630
25. Tichy, W.F. (1982). Design, Implementation, and Evaluation of a Revision Control System. The 6th International Conference on Software Engineering, Tokyo, Japan, 58-67.
26. Pirie, I.W. (1986). The LIFESPAN System. SIGSOFT Software Engineering Notes 11(2): 27-28.
27. Estublier, J., Ghoul, S., and Krakowiak, S. (1984). Preliminary Experience with a Configuration Control System for Modular Programs. In Proceedings of the first ACM SIGSOFT/SIGPLAN SW Engineering Symposium on Practical Software Development Environments. NY, USA, 149-156.
28. Belkatir, N., and Estublier, J. (1987). Experience with a Data Base of Programs. In Proceedings of the Second ACM SIGSOFT/SIGPLAN SW Engineering Symposium on Practical Software Development Environments, Palo Alto, USA, 84-91. https://doi.org/10.1145/24208.24219
29. Murray, J. (1988). Source Control using VM/SP and CMS. SIGSOFT Software Engineering Notes, 13(2): 51-54.
30. Gordon, M. (1989). Combining Version Control with Automatic Program Building. SIGSOFT Software Engineering Notes, 14(6): 25-31.
31. Mahler, A., and Lampen, A. (1988). An Integrated Toolset for Engineering Software Configurations. In Proceedings of the Third ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, Boston, USA, 191-200. https://doi.org/10.1145/64135.64142
32. Ploedereder, E. and Fergany, A. (1989). The Data Model of the Configuration Management Assistant (CMA). In Proceedings of the 2nd International Workshop on Software Configuration Management, Princeton, USA, 5-14.
33. Banks, S., Bunting, C., Edwards, R., Fleming. L., and Hackett, P. (1991). A Configuration Management System in a Data Management Framework. The 28th ACM/IEEE Design Automation Conference, San Francisco, USA, 699-703.
34. Lockman, A., and Salasin, J. (1990). A Procedure and Tools for Transition Engineering. In Proceedings of the fourth ACM SIGSOFT Symposium on Software Development Environments, Irvine, USA, 157-172.
35. Tichy, W.F. (1992). Programming-in-the-Large: Past, Present, and Future. The 14th International Conference on Software Engineering, Melbourne, Australia, 362-367. https://doi.org/10.1145/143062.143153

36. Dart, S. (1991). Concepts in Configuration Management Systems. The 3rd International Workshop on Software Configuration Management, Trondheim, Norway, 1-18.
37. Micallef, J., Kaiser, G.E., and Perry, D.E. (1991). SETA1 Working Group on Ada Libraries, Configuration Management, and Version Control. In Proceedings of the First International Symposium on Environments and Tools for Ada, Redondo Beach, USA, 29-31.
38. Davis, A.M., and Bersoff, E.H. (1991). Impacts of Life Cycle Models on Software Configuration Management. Communications of the ACM, 34(8): 104-118.
39. Grinter, R.E. (1995). Using a Configuration Management Tool to Coordinate Software Development. In Proceedings of Conference on Organizational Computing Systems, Milpitas, USA, 168-177. https://doi.org/10.1145/224019.224036
40. Osterbye, K. (1993). Structural and Cognitive Problems in Providing Version Control for Hypertext. In Proceedings of the ACM Conference on Hypertext, Milan, Italy, 33-42.
41. Slein, J.A., Vitali, F., Whitehead, E.J.Jr., and Durand, D.G. (1997). Requirements for Distributed Authoring & Versioning on the World Wide Web. StandardView, 5(1): 17-24. https://doi.org/10.1145/253452.253474
42. Narayanaswamy, K., and Goldman, N. (1992). "Lazy" Consistency: A Basis for Cooperative Software Development. The 1992 ACM Conference on Computer-Supported Cooperative Work, Toronto, Canada, 257-264.
43. Chawathe, S.S., and Garcia-Molina, H. (1997). Meaningful Change Detection in Structured Data. In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, Tucson, USA, 26-37.
44. Chawathe, S.S., Rajaraman, A., Garcia-Molina, H., and Widom, J. (1996). Change Detection in Hierarchically Structured Information. SIGMOD Record, 25(2): 493-504.
45. Wiil, U.K. (1993). Experiences with HyperBase: A Hypertext Database Supporting Collaborative Work. SIGMOD Record, 22(4): 19-25. https://doi.org/10.1145/166635.166646
46. Gallagher, K. (1991). Conditions to Assure Semantically Consistent Software Merges in Linear Time. In Proceedings of the 3rd International Workshop on Software Configuration Management, Trondheim, Norway, 80-83.
47. Lippe, E., and van Oosterom, N. (1992). Operation-Based Merging. SIGSOFT Software Engineering Notes, 17(5): 78-87.
48. Yang, W., Horwitz, S., and Reps, T. (1992). A Program Integration Algorithm that Accommodates Semantics-Preserving Transformations. ACM Transactions on Software Engineering Methodology, 1(3): 310-354.
49. Heidenreich, G., Minas, M., and Kips, D. (1996). A New Approach to Consistency Control in Software Engineering. In Proceedings of the 18th International Conference on Software Engineering, Berlin, Germany, 289-297. https://doi.org/10.1109/ICSE.1996.493424
50. Render, H., and Campbell, R. (1991). An Object-Oriented Model of Software Configuration Management. In Proceedings of the 3rd International Workshop on Software Configuration Management, Trondheim, Norway, 127-139.
51. Ochuodho, S.J., and Brown, A.W. (1991). A Process-Oriented Version and Configuration Management Model for Communications Software. In Proceedings of the 3rd International Workshop on Software Configuration Management, Trondheim, Norway, 109-120.
52. Aquilino, D., Asirelli, P., Inverardi, P., and Malara, P. (1991). Supporting Reuse and Configuration: A Port Based SCM Model. The 3rd International Workshop on Software Configuration Management, Trondheim, Norway, 62-67. https://doi.org/10.1145/111062.111070
53. Wein, M., Cowan, W., and Gentleman, W.M. (1992). Visual Support for Version Management. The 1992 ACM/SIGAPP Symposium on Applied Computing: Technological Challenges of the 1990's, Kansas City, USA, 1217-1223.
54. Madhavji, N.H. (1991). The Prism Model of Changes. In Proceedings of the 13th International Conference on Software Engineering, Austin, USA, 166-177.
55. Nicklin, P.J. (1991). Managing Multi-Variant Software Configuration. In Proceedings of the 3rd International Workshop on Software Configuration Management, Trondheim, Norway, 53-57. https://doi.org/10.1145/111062.111068
56. Lin, Y-J., and Reiss, S.P. (1996). Configuration Management with Logical Structures. The 18th International Conference on Software Engineering, Berlin, Germany, 298-307.
57. Joeris, G. (1997). Change Management Needs Integrated Process and Configuration Management. SIGSOFT Software Engineering Notes, 22(6): 125-141.
58. Kramer, S.A. (1991). History Management System. In Proceedings of the 3rd International Workshop on Software Configuration Management, Trondheim, Norway, 140-143.
59. Berrada, K., Lopez, F., and Minot, R. (1991). VMCM, a PCTE Based Version and Configuration Management System. In Proceedings of the 3rd International Workshop on Software Configuration Management, Trondheim, Norway, 43-52. https://doi.org/10.1145/111062.111067
60. Okamura, K. (1993). Combining Local Negotiation and Global Planning in Cooperative Software Development Projects. In Proceedings of the Conference on Organizational Computing Systems, Milpitas, USA, 239-249.
61. Bjorkholm, T. (1997). Product Review: Perforce Software Configuration Management System. Linux J. 1997, 44es, Article 13 (December 1997).
62. Rosenblum, D.S., and Krishnamurthy, B. (1991). An Event-Based Model of Software Configuration

Management. In Proceedings of the 3rd International Workshop on Software Configuration Management, Trondheim, Norway, 94-97. https://doi.org/10.1145/111062.111074
63. van der Hoek, A., Hall, R.S., Heimbigner, D., and Wolf, A.L. (1997). Software Release Management. SIGSOFT Software Engineering Notes, 22(6): 159-175.
64. Korel, B., Wedde, H., Magaraj, S., Nawaz, K., and Dayana, V. (1991). Version Management in Distributed Network Environment. The 3rd International Workshop on Software Configuration Management, Trondheim, Norway, 161-166. https://doi.org/10.1145/111062.111083
65. Conradi, R., and Malm, C.C. (1991). Cooperating Transactions Against the EPOS Database. In Proceedings of the 3rd International Workshop on Software Configuration Management, Trondheim, Norway, 98-101.
66. Sutton, S.M.Jr., Heimbigner, D., and Osterweil, L.J. (1990). Language Constructs for Managing Change in Process-Centered Environments. In Proceedings of the fourth ACM SIGSOFT Symposium on Software Development Environments, Irvine, USA, 206-217. https://doi.org/10.1145/99277.99296
67. Westfechtel, B. (1991). Structure-Oriented Merging of Revisions of Software Documents. In Proceedings of the 3rd International Workshop on Software Configuration Management, Trondheim, Norway, 68-79.
68. Puntikov, N.I., Volodin, M.A., and Kolesnikov, A.A. (1995). AVCS: The APL Version Control System. In Proceedings of the International Conference on Applied Programming Languages, San Antonio, USA 154-161. https://doi.org/10.1145/206913.206995
69. Chu-Carroll, M.C., Wright, J., and Shields, D. (2002). Supporting Aggregation in Fine-Grained Software Configuration Management. SIGSOFT Software Engineering Notes 27(6): 99-108.
70. Brada, P. 2001. Component Revision Identification Based on IDL/ADL Component Specification. The 8th European Software Engineering Conference held jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Vienna, Austria, 297-298.
71. Ohst, D., Welle, M., and Kelter, U. (2003). Differences between Versions of UML Diagrams. The 9th European Software Engineering Conference held jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Helsinki, Finland, 227-236. https://doi.org/10.1145/940071.940102
72. Dolog, P., Vranic, V., and Bielikova, M. (2001). Representing Change by Aspect. SIGPLAN Not. 36(12): 77-83.
73. Cheng, L-T., de Souza, C.R.B., Hupfer, S., Patterson, J., and Ross, S. (2003). Building Collaboration into IDEs. Queue 1, 9 (December 2003), 40-50.
74. Heydon, A., Levin, R., and Yu, Y. (2000). Caching Function Calls Using Precise Dependencies. ACM SIGPLAN Notices, 35(5): 311-320.
75. Chu-Carroll, M.C., Wright, J., and Ying, A.T.T. (2003). Visual Separation of Concerns Through Multidimensional Program Storage. The 2nd International Conference on Aspect-Oriented Software Development, Boston, USA, 188-197. https://doi.org/10.1145/643603.643623
76. van der Hoek, A., Mikic-Rakic, M., Roshandel, R., and Medvidovic, N. (2001). Taming Architectural Evolution. The 8th European Software Engineering Conference held jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Vienna, Austria, 1-10.
77. Brudaru, I.I., and Zeller, A. (2008). What is the Long-Term Impact of Changes? In Proceedings of the 2008 International Workshop on Recommendation Systems for Software Engineering, Atlanta, USA, 30-32.
78. Dietel, K. (2004). Mastering IT Change Management Step Two: Moving From Ignorant Anarchy to Informed Anarchy. In Proceedings of the 32nd Annual ACM SIGUCCS Conference on User Services, Baltimore, USA, 188-190. https://doi.org/10.1145/1027802.1027846
79. Altmanninger, K., and Kotsis, G. (2009). Towards Accurate Conflict Detection in a VCS for Model Artifacts: A Comparison of Two Semantically Enhanced Approaches. The Sixth Asia-Pacific Conference on Conceptual Modelling, Volume 96, Darlinghurst, Australia, 139-146.
80. Junqueira, D.C., Bittar, T.J., and Fortes, R.P.M. (2008). A Fine-Grained and Flexible Version Control for Software Artifacts. The 26th Annual ACM International Conference on Design of Communication, Lisbon, Portugal, 185-192.
81. Bartelt, C. (2008). Consistence Preserving Model Merge in Collaborative Development Processes. In Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models, Leipzig, Germany, 13-18. https://doi.org/10.1145/1370152.1370157
82. Brosch, P., Langer, P., Seidl, M., and Wimmer, M. (2009). Towards End-User Adaptable Model Versioning: The By-Example Operation Recorder. In Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models. IEEE Computer Society, Washington, USA, 55-60.
83. Schmidt, M., and Gloetzner, T. (2008). Constructing Difference Tools for Models Using the SiDiff Framework. In Companion of the 30th International Conference on Software Engineering, Leipzig, Germany, 947-948.
84. Parnin, C., and Gorg, C. (2008). Improving Change Descriptions with Change Contexts. In Proceedings of the 2008 International Working Conference on Mining Software Repositories, Leipzig, Germany, 51-60. https://doi.org/10.1145/1370750.1370765
85. Omori, T., and Maruyama, K. (2008). A Change-Aware Development Environment by Recording Editing Operations of Source Code. In Proceedings of the 2008 International Working Conference on Mining Software Repositories, Leipzig, Germany, 31-34.

86. Kaur, P., and Singh, H. (2009). A Layered Structure for Uniform Version Management in Component Based Systems. SIGSOFT Softw. Eng. Notes 34(6): 1-7. https://doi.org/10.1145/1640162.1640167
87. Koegel, M., Helming, J., and Seyboth, S. (2009). Operation-Based Conflict Detection and Resolution. The 2009 ICSE Workshop on Comparison and Versioning of Software Models. IEEE Computer Society, Washington, USA, 43-48.
88. Kogel, M. (2008). Towards Software Configuration Management for Unified Models. In Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models, Leipzig, Germany, 9-24. https://doi.org/10.1145/1370152.1370158
89. Mei, H., Zhang, L., and Yang, F. (2001). A Software Configuration Management Model for Supporting Component-Based Software Development. SIGSOFT Softw. Eng. Notes 26(2): 53-58.
90. Gunter, C.A. (2000). Abstracting Dependencies between Software Configuration Items. ACM Transactions on Software Engineering and Methodology, 9(1): 94-131. https://doi.org/10.1145/332740.332743
91. Nguyen, T.N., Munson, E.V., and Boyland, J.T. (2003). Configuration Management in a Hypermedia-Based Software Development Environment. In Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia, Nottingham, UK, 194-195.
92. Bendix, L., and Borracci, L. (2005). Towards A Suite of Software Configuration Management Metrics. In Proceedings of the 12th International Workshop on Software Configuration Management. Lisbon, Portugal, 75-82.
93. Gacek, C., Giese, H., and Hadar, E. (2008). Friends or Foes?: A Conceptual Analysis of Self-Adaptation and IT Change Management. In Proceedings of the 2008 International Workshop on Software Engineering for Adaptive and Self-Managing Systems, Leipzig, Germany, 121-128.
94. Ki, Y., and Song, M. (2009). An Open Source-Based Approach to Software Development Infrastructures. In Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, Washington, USA, 525-529. https://doi.org/10.1109/ASE.2009.73
95. Sarma, A., Noroozi, Z., and van der Hoek, A. (2003). Palantír: Raising Awareness Among Configuration Management Workspaces. The 25th International Conference on Software Engineering, Portland, USA, 444-454.
96. Sarma, A., Redmiles, D., and van der Hoek, A. (2008). Empirical Evidence of the Benefits of Workspace Awareness in Software Configuration Management. In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Atlanta, USA, 113-123. https://doi.org/10.1145/1453101.1453118
97. Shao, D., Khurshid, S., and Perry, D.E. (2009). SCA: A Semantic Conflict Analyzer for Parallel Changes. In Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, Amsterdam, The Netherlands, 291-292.
98. Mostafa, N., and Krintz, C. (2009). Tracking Performance Across Software Revisions. In Proceedings of the 7th International Conference on Principles and Practice of Programming in Java, Calgary, Canada 162-171. https://doi.org/10.1145/1596655.1596682
99. Murta, L., Correa, C., Prudencio, J.G., and Werner, C. (2008). Towards Odyssey-VCS 2: Improvements over a UML-Based Version Control System. In Proceedings of the 2008 international workshop on Comparison and Versioning of Software Models, Leipzig, Germany, 25-30.
100. Nguyen, T.T., Nguyen, H.A., Pham, N.H., Al-Kofahi, J.M.,and Nguyen, T.N. (2009). Clone-Aware Configuration Management. The 2009 IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, Washington, USA, 123-134. https://doi.org/10.1109/ASE.2009.90
101. Nguyen, T.N., Munson, E.V., and Boyland, J.T. (2004a). Object-Oriented, Structural Software Configuration Management. The 19th annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications, Vancouver, Canada, 35-36.
102. Nguyen, T.N., Munson, E.V., and Boyland, J.T. (2004b). The Molhado Hypertext Versioning System. In Proceedings of the fifteenth ACM conference on Hypertext and hypermedia, Santa Cruz, USA, 185-194. https://doi.org/10.1145/1012807.1012859
103. Chu-Carroll, M.C. and Sprenkle, S. (2000). Coven: Brewing Better Collaboration Through Software Configuration Management. SIGSOFT Software Engineering Notes, 25(6): 88-97.
104. Dig, D., Nguyen, T.N., Manzoor, K., and Johnson, R. (2006). MolhadoRef: A Refactoring-Aware Software Configuration Management Tool. The 21st ACM SIGPLAN Symposium on Object-Oriented Programming Systems, Languages, and Applications, Portland, USA, 732-733. https://doi.org/10.1145/1176617.1176698
105. Rajlich, V. (2014). Software Evolution and Maintenance. In Proceedings of the on Future of Software Engineering, Hyderabad, India, 133-144.
106. Passos, L., Czarnecki, K., Apel, S., Wasowski, A., Kastner, C., and Guo, J. (2013). Feature-Oriented Software Evolution. In Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems, Pisa, Italy, Article 17, 8 pages.
107. Er, N.P., and Erbas, C. (2010). Aligning Software Configuration Management with Governance Structures. In Proceedings of the 2010 ICSE Workshop on Software Development Governance, Cape Town, South Africa, 1-8.
108. Marquardt, K. (2010). Patterns for Software Release Versioning. In Proceedings of the 15th European

Conference on Pattern Languages of Programs, Irsee, Germany, Article 17, 13 pages. https://doi.org/10.1145/2328909.2328931

109. Ali, U., and Kidd, C. (2013). Barriers to Effective Configuration Management Application in a Project Context: An Empirical Investigation. International Journal of Project Management, 32(3): 508-518.
110. Bendix, L., and Pendleton, C. (2013). The Role of Configuration Management in Outsourcing and Distributed Development. In Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia, Moscow, Russia, Article 8, 10 pages.
111. Lindkvist, C., Stasis, A., and Whyte, J. (2011). Configuration Management in Complex Engineering Projects. Procedia CIRP 11 (2013), 173 – 176.
112. Phillips, S., Ruhe, R., and Sillito, J. (2012). Information Needs for Integration Decisions in the Release Process of Large-Scale Parallel Development. In Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, Seattle, Washington, USA, 1371-1380. https://doi.org/10.1145/2145204.2145408
113. Premraj, R., Tang, A., Linssen, N., Geraats, H., and van Vliet, H. (2011). To Branch or Not to Branch?. In Proceedings of the 2011 International Conference on Software and Systems Process, Waikiki, USA, 81-90.
114. Almalki, J., and Shen, H. (2015). A Lightweight Solution to Version Incompatibility in Service-Oriented Revision Control Systems. The 24th Australasian Software Engineering Conference, Volume II, Adelaide, Australia, 59-63.
115. Chamarty, S., Patel, H.D., and Tripunitara, M.V. (2011). An Authorization Scheme for Version Control Systems. The 16th ACM Symposium on Access Control Models and Technologies, Shanghai, China, 123-132.
116. Fontana, F.A., and Zanoni, M. (2014). Tracking Line Changes in Source Code Repositories. In Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. New York, NY, USA, Article 68, 1 page.
117. Guimaraes, M.L., and Rito-Silva, A. (2010). Towards Real-Time Integration. In Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, Cape Town, South Africa, 56-63. https://doi.org/10.1145/1833310.1833320
118. Hata, H., Mizuno, O., and Kikuno, T. (2011). Historage: Fine-Grained Version Control System for Java. In Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th Annual ERCIM Workshop on Software Evolution, Szeged, Hungary, 96-100.
119. Koegel, M., Herrmannsdoerfer, M., von Wesendonk, O., and Helming, J. (2010). Operation-Based Conflict Detection. In Proceedings of the 1st International Workshop on Model Comparison in Practice, Malaga, Spain, 21-30.
120. Kuttal, S.K., Sarma, S., and Rothermel, G. (2014). On the Benefits of Providing Versioning Support for End Users: An Empirical Study. ACM Trans. Comput.-Hum. Interact. 21(2), Article 9, 43 pages. https://doi.org/10.1145/2560016
121. Li, Y., Wang, L., Li, X., and Cai, Y. (2012). Detecting Source Code Changes to Maintain the Consistence of Behavioral Model. The Fourth Asia-Pacific Symposium on Internetware. ACM, New York, NY, USA, Article 7, 6 pages.
122. McIntosh, S. (2011). Build System Maintenance. In Proceedings of the 33rd International Conference on Software Engineering Waikiki, USA, 1167-1169.
123. Mehdi, A-N., Urso, P., and Charoy, F. (2014). Evaluating Software Merge Quality. The 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14). ACM, New York, NY, USA, Article 9, 10 pages.
124. Sarma, S., Branchaud, J., Dwyer, M.B., Person, S., and Rungta, N. (2014). Development Context Driven Change Awareness and Analysis Framework. In Companion Proceedings of the 36th International Conference on Software Engineering, Hyderabad, India, 404-407.
125. Servant, F., and Jones, J.A. (2012). History Slicing: Assisting Code-Evolution Tasks. In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, Cary, USA, Article 43, 11 pages. https://doi.org/10.1145/2393596.2393646
126. Stein, J., Nunes, I., and Cirilo, E. (2014). Preference-Based Feature Model Configuration with Multiple Stakeholders. In Proceedings of the 18th International Software Product Line Conference, Florence, Italy, Volume 1, 132-141.
127. Erdweg, S., Lichter, M., and Weiel, M. (2015). A Sound and Optimal Incremental Build System with Dynamic Dependencies. SIGPLAN Not. 50(10): 89-106. https://doi.org/10.1145/2858965.2814316
128. de Rosso, S.P., and Jackson, D. (2013). What's Wrong with Git?: A Conceptual Design Analysis. In Proceedings of the 2013 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, Tucson, USA, 37-52.
129. Apel, S., Liebig, J., Brandl, B., Lengauer, C., and Kastner, C. (2011). Semistructured Merge: Rethinking Merge in Revision Control Systems. The 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, Szeged, Hungary, 190-200.
130. Koegel, M., and Helming, J. (2010). EMFStore: A Model Repository for EMF Models. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, Cape Town, South Africa, 307-308.
131. Mahoney, M. (2012). The Storyteller Version Control System: Tackling Version Control, Code Comments, and Team Learning. In Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity, Tucson, Arizona, USA, 17-18.
132. Lorenz, D.H., and Rosenan, B. (2014). Versionable, Branchable, and Mergeable Application State. In

Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, Portland, USA 29-42. https://doi.org/10.1145/2661136.2661151
133. Mikkonen T., and Nieminen, A. (2012). Elements for a Cloud-Based Development Environment: Online Collaboration, Revision Control, and Continuous Integration. In Proceedings of the WICSA/ECSA 2012 Companion Volume, Helsinki, Finland, 14-20.
134. Padhye, R., Mani, S., and Sinha, V.S. (2014). NeedFeed: Taming Change Notifications by Modeling Code Relevance. In Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, Vasteras, Sweden, 665-676.
135. Servant, F., Jones, J.A., and van der Hoek, A. (2010). CASI: Preventing Indirect Conflicts Through A Live Visualization. In Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, Cape Town, South Africa, 39-46. https://doi.org/10.1145/1833310.1833317
136. IEEE 828. (2012). IEEE Standard for Configuration Management in Systems and Software Engineering. The Institute of Electrical and Electronics Engineers. (71 pages).
137. ISO 10007. (2017). Quality Management Systems - Guidelines for Configuration Management. International Organization for Standardization. (10 pages).
138. IEEE 15939. (2008). IEEE Standard Adoption of ISO/IEC 15939:2007 - Systems and Software Engineering - Measurement Process. The Institute of Electrical and Electronics Engineers. (40 pages).
139. ISO/IEC 15939. (2007). Systems and Software Engineering - Measurement Process. International Organization for Standardization. (38 pages).
140. ISO/IEC/IEEE 24765. (2010). Systems and Software Engineering - Vocabulary. International Organization for Standardization. (410 pages).
141. ISO/IEC 12207. (2008). Standard for Systems and Software Engineering - Software Life Cycle Processes. International Organization for Standardization. (123 pages).
142. ISO/IEC 15288. (2008). Systems and Software Engineering - System Life Cycle Processes. International Organization for Standardization. (70 pages).
143. Young, J.C. (1988). SofTool Users Group. SIGSOFT Software Engineering Notes 13(1): 68-70.
144. Martinis, J. (1990). Softool Change/Configuration Management. SIGSOFT SW Engineering Notes, 15(3): 51-.
145. Sheedy, C. (1991). Sorceress: A Database Approach to Software Configuration Management. In Proceedings of the 3rd International Workshop on Software Configuration Management (SCM '91), Peter H. Feiler (Ed.). ACM, New York, NY, USA, 121-126.
146. Titze, F. (2000). Improvement of a Configuration Management System. The 22nd International Conference on Software Engineering. ACM, New York, NY, USA, 618-625.
147. Leblang, D.B. and Chase, R.P.Jr. (1984). Computer-Aided Software Engineering in a Distributed Workstation Environment. The first ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments. New York, USA, 104-112. https://doi.org/10.1145/800020.808255
148. Aide-De-Camp. (1989). Aide-De-Camp Software Management System: Product Overview. Software Maintenance and Development Systems, Inc.
149. Microsoft. (2000). Sourcesafe Product Documentation, Microsoft, Inc., Seattle, USA.
150. Sun/Forte. (2000). Teamware Product Documentation. Sun MicroSystems Inc, Mountain View, USA.
151. Pilato, M. (2004), Version Control with Subversion. O'Reilly & Associates, Inc., Sebastopol, CA, USA.
152. Wright, A. (1990). Requirements for a Modern CM System. CaseWare, Inc.
153. Flemming, T., Christensen, Abbott, J., &Pflaum, G. (2003). Rational ClearCase UCM Migration: A Case Study. Rational Report.
154. Sarma, A., Redmiles, D., and van der Hoek, A. (2012). Palantir: Early detection of Development Conflicts Arising from Parallel Code Changes. IEEE Transactions on Software Engineering, 38(4).
155. Klepper, S., Krusche, S., and Bruegge, B. (2016). Semi-Automatic Generation of Audience-Specific Release Notes. In Proceedings of the International Workshop on Continuous SW Evolution and Delivery, Austin, USA, 19-22.
156. North, K.J., Bolan, S., Sarma, A., and Cohen, M.B. (2015). GitSonifier: Using Sound to Portray Developer Conflict History. In Proceedings of the 2015 10th Joint Meeting on Foundations of SW Engineering, Bergamo, Italy, 886-889. https://doi.org/10.1145/2786805.2803199
157. Huang, S-K., and Liu, K-M. (2005). Mining Version Histories to Verify the Learning Process of Legitimate Peripheral Participants. THE 2005 International Workshop on Mining Software Repositories, Saint Louis, USA, 1-5.
158. Makiaho, P., Poranen, T., and Seppi, A. (2014). Version Control Usage in Students' Software Development Projects. The 15th International Conference on Computer Systems and Technologies, Ruse, Bulgaria, 452-459.
159. Gowtham, S. (2014). Revision Control System (RCS) in Computational Sciences and Engineering Curriculum. In Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment, New York, USA, Article 76, 3 pages.
160. Jiang, K., and Kamali, R. (2008). Integration of Configuration Management into the IT Curriculum. In Proceedings of the 9th ACM SIGITE conference on IT Education, Cincinnati, USA,183-186.
161. Eyolfson, J., Tan, L., and Lam. P. (2011). Do Time of Day and Developer Experience Affect Commit Bugginess?. In Proceedings of the 8th Working Conference on Mining Software Repositories, Waikiki, USA, 153-162.
162. Bortis, G., and van der Hoek, A. (2013). PorchLight: A Tag-Based Approach to Bug Triaging. In Proceedings of

the 2013 International Conference on Software Engineering. IEEE Press, Piscataway, USA, 342-351. https://doi.org/10.1109/ICSE.2013.6606580

163. Abreu, R., and Premraj, R. (2009). How Developer Communication Frequency Relates to Bug Introducing Changes. In Proceedings of the Joint ERCIM Workshop on Software Evolution and International Workshop on Principles of Software Evolution, Szeged, Hungary, 153-157.
164. IEEE Software Engineering Competency Model (SWECOM). IEEE Computer Society Press, https://www.computer.org/web/peb/swecom-download, retrieved Sept 2018.
165. Arimoto, M.M., Cagnin, M.I., and de Camargo, V.V. (2008). Version Control in Crosscutting Framework-Based Development. In Proceedings of the 2008 ACM Symposium on Applied Computing, Fortaleza, Brazil, 753-758.
166. Ogasawara, E., Rangel, P., Murta, L., Werner, C., and Mattoso, M. (2009). Comparison and Versioning of Scientific Workflows. In Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models. IEEE Computer Society, Washington, USA, 25-30.
167. Mohan, K., Xu, P., Cao, L., and Ramesh, B. (2008). Improving Change Management in Software Development: Integrating Traceability and Software Configuration Management. Decision Support Syst. 45(4): 922-936. https://doi.org/10.1016/j.dss.2008.03.003
168. Anastasopoulos, M. (2009). Increasing Efficiency and Effectiveness of Software Product Line Evolution: An Infrastructure on top of Configuration Management. In Proceedings of the Joint ERCIM Workshop on Software Evolution and International Workshop on Principles of Software Evolution, Szeged, Hungary, 47-56.
169. Buchmann, T., Dotor, A., and Westfechtel, B. (2013). MOD2-SCM: A Model-Driven Product Line for Software Configuration Management Systems. Information and Software Technology 55(3): 630-650.
170. Braun, B. (2008). SAVE: Static Analysis on Versioning Entities. In Proceedings of the fourth international workshop on Software Engineering for Secure Systems, Leipzig, Germany, 25-32.
171. Roshandel, R., van Der Hoek, A., Mikic-Rakic, M., and Medvidovic, N. (2004). Mae - A System Model and Environment for Managing Architectural Evolution. ACM Trans. Softw. Eng. Methodol. 13(2): 240-276. https://doi.org/10.1145/1018210.1018213
172. Pandey, M., and Munson, E.V. (2013). Version Aware LibreOffice Documents. The 2013 ACM Symposium on Document Engineering, Florence, Italy, 57-60.
173. McVoy, L. (2015). Preliminary Product Line Support in BitKeeper. In Proceedings of the 19th International Conference on Software Product Line, Nashville, USA, 245-252.
174. Hajmoosaei, M., Tran, H-N., Percebois, C., Front, A., and Roncancio, C. (2015). Towards A Change-Aware Process Environment for System and Software Process. In Proceedings of the 2015 International Conference on Software and System Process, Tallinn, Estonia, 32-41.
175. Swierstra, W., and Loh, A. (2014). The Semantics of Version Control. In Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, Portland, USA, 43-54.
176. Pichler, C. (2010). A Framework for Handling Variants of Software Models. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, Cape Town, South Africa, 345-346. https://doi.org/10.1145/1810295.1810385
177. Tosun, A., Turhan, B., and Bener, A. (2009). Practical Considerations in Deploying AI for Defect Prediction: A Case Study within the Turkish Telecommunication Industry. The 5th International Conference on Predictor Models in SW Engineering, Vancouver, Canada, Article 11, 9 pages.
178. Zimmermann, T., Weisgerber, P., Diehl, S., and Zeller, A. (2004). Mining Version Histories to Guide Software Changes. In Proceedings of the 26th International Conference on Software Engineering, Edinburgh, United Kingdom, 563-572.
179. Herbsleb, J.D., Mockus, A., Finholt, T.A., and Grinter, R.E. (2000). Distance, Dependencies, and Delay in a Global Collaboration. In Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, Philadelphia, USA, 319-328.
180. Perry, D.E., Siy, H.P., and Votta, L.G. (2001). Parallel Changes in Large-Scale Software Development: An Observational Case Study. ACM Trans. SW. Eng. Methodol. 10(3): 308-337. https://doi.org/10.1145/383876.383878
181. Yamauchi, Y., Yokozawa, M., Shinohara, T., and Ishida, T. (2000). Collaboration with Lean Media: How Open-Source Software Succeeds. In Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, Philadelphia, USA, 329-338.
182. Schwagerl, F., and Westfechtel, B. (2017). Perspectives on Combining Model-Driven Engineering, Software Product Line Engineering, and Version Control. The Eleventh International Workshop on Variability Modelling of Software-intensive Systems, Maurice H. terBeek, Norbert Siegmund, and Ina Schaefer (Eds.). New York, USA, 76-83.
183. Matsuda, J., Hayashi, S., and Saeki, M. (2015). Hierarchical Categorization of Edit Operations for Separately Committing Large Refactoring Results. In Proceedings of the 14th International Workshop on Principles of Software Evolution (IWPSE 2015). ACM, New York, NY, USA, 19-27.
184. Cox, P.R. (1982). Elements of a Software Quality Control Program. The ACM '82 Conference. New York, USA, 2-4.
185. Gustafson, G.G. and Kerr, R.J. (1982). Some Practical Experience with a Software Quality Assurance Program. Communications of the ACM 25(1): 4-12.
186. Zychlinski, B.Z., and Palomar, M.A. (1984). A Software Quality Assurance Program Through Reusable Code. In

Proceedings of the 3rd Annual International Conference on Systems Documentation, Mexico City, Mexico, 107-113.

187. Heinemann, L., Hummel, B., and Steidl, D. (2014). Teamscale: Software Quality Control in Real-Time. In Proceedings of the 36th International Conference on Software Engineering, Hyderabad, India, 592-595.
188. Dai, M., Shen, B., Zhang, T., and Zhao, M. (2014). Impact of Consecutive Changes on Later File Versions. In Proceedings of the 3rd International Workshop on Evidential Assessment of Software Technologies, Nanjing, China, 17-24. https://doi.org/10.1145/2627508.2627512
189. Malhotra, R., and Agrawal, A. (2014). CMS Tool: Calculating Defect and Change Data from Software Project Repositories. SIGSOFT Softw. Eng. Notes 39(1): 1-5.
190. Nam, J., and Kim, S. (2015). Heterogeneous Defect Prediction. In Proceedings of the 2015 10th Joint Meeting on Foundations of SW Engineering, Bergamo, Italy, 508-519.
191. Henderson, C. (2008). Managing Software Defects: Defect Analysis and Traceability. SIGSOFT Softw. Eng. Notes 33, 4, Article 2, 3 pages.
192. Ramler, R. (2008). The Impact of Product Development on the Lifecycle of Defects. In Proceedings of the 2008 Workshop on Defects in Large Software Systems, Seattle, USA, 21-25. https://doi.org/10.1145/1390817.1390823
193. Rohini B. Jadhav, Shashank D. Joshi , Umesh G. Thorat, Aditi S. Joshi. A Software Defect Learning and Analysis Utilizing Regression Method for Quality Software Development. International Journal of Advanced Trends in Computer Science and Engineering. Volume 8, No.4, July – August 2019 https://doi.org/10.30534/ijatcse/2019/38842019