A Model for People-Centric Software Configuration Management

Syahrul Fahmy[1], Aziz Deraman[2], Jamaiah H. Yahaya[3], Amir Ngah[2] and Fouad Abdulameer Salman[2]
[1]*Faculty of Computer, Media and Technology Management, TATI University College, Kemaman, Terengganu, Malaysia.*
[2]*School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu, 21030 Kuala Nerus, Terengganu, Malaysia.*
[3]*Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia.*
fahmy@tatiuc.edu.my

***Abstract*—*Software Configuration Management* (SCM) is the adaptation of *Configuration Management* in software engineering to control changes to software products. Its implementation is guided by international standards and procedures, in addition to an array of supporting tools. However, the issues of project delays and products that do not fit its purpose still prevail in software development, questioning the practice of SCM by software organizations. Current research efforts in are mainly focused on technical issues, with little interest (if any) in the role of human in SCM implementation. This paper proposes an alternative view of SCM, which emphasizes the role of human in its implementation. The *People-Centric Software Configuration Management* (P-SCM) model comprises of four components namely *People, Process, Tools* and *Documentation*. It reveals the competency of the software project team, extensiveness of activities carried out by the organization, availability of supporting tools, and comprehensiveness of documentation. P-SCM supports software organizations in decision-making activities, provide insights to project discrepancies, identify best practices and pitfalls, support the identification of training needs and the selection of tools. Initial results reveal that P-SCM provides additional input to software organizations in project planning and outsourcing activities.**

***Index Terms*—People-Centric Approach; Software Configuration Management; Software Engineering; Software Quality.**

I. INTRODUCTION

Since the inception of Software Configuration Management (SCM) more than 50 years ago, research efforts in this field have been technically focused with the development of tools for addressing emergent issues in software development. Over the years, SCM has grown to be a mature field with solid processes and support tools.

As the 'cloud' re-defines software as 'services' and 'eco-systems', software development is no longer confined to geographical locations but scattered throughout the globe – Global Software Development. New approaches have emerged such as Agile and Lean that require specific human competency for successful adaptation and implementation.

However, the importance of human and competency has received little interest in SCM. SCM is viewed as a bureaucratic task and its implementation are too dependent on tools, missing out the big picture of change management. In addition, the issues of overdue projects, product revisions, and undelivered projects still prevail which questions the effectiveness of current SCM practices. As such, this paper proposes an alternative view of SCM implementation which is piloted by human.

This paper is organized as follows: Section II presents a brief overview of SCM while Section III presents a model for people-centric SCM entailing success factors and main components. Example of P-SCM implementation and comparison to traditional approach are presented in Sections IV and V respectively. Section VI presents the validation of the proposed model. Finally, conclusion and future work are presented in Section VII.

II. SOFTWARE CONFIGURATION MANAGEMENT

Two prevailing problems in software projects are falling behind schedule and software product that does not fit its purpose (lack of quality). These problems can be attributed to on-going changes made to software product, leading to more work than initially anticipated and the diminishing of quality as new changes are implemented. One approach for addressing these problems is through a systematic *Software Configuration Management* (SCM) practice.

SCM is the adaptation of *Configuration Management* in software engineering to control changes to software products. SCM implementation is guided by international standards including IEEE and ISO, sound procedure and process as stipulated by IEEE, SEI and ISO, and a vast collection of commercial support tools.

There are seven main standards related to SCM namely IEEE 828 [1], IEEE 15939 [2], ISO 10007 [3], ISO/IEC 12207 [4], ISO/IEC 15288 [5], ISO/IEC 15939 [6], and ISO/IEC/IEEE 24765 [7]. These standards can be classified into three: *General Standards* that provide a common vocabulary and measurement process for systems and software engineering (IEEE 15939, ISO/IEC 15939, ISO/IEC/IEEE 24765); *Life-Cycle Standards* that provide a common framework for life-cycle processes and activities (ISO/IEC 12207, ISO/IEC 15288); and *Configuration Management Standards* that describe configuration management processes within an organization (IEEE 828, ISO 10007).

Throughout the years, SCM has responded to issues at hand in software development through the development of tools for example Make in the late 1970s[8]; RCS in the mid-1980s [9]; CCM in the early 1990s [10]; Sun/Forte in the early 2000s [11]; and Gitless in 2013 [12]. SCM also has been applied to various areas in software engineering including *Software Product Lines* [13]; *Model-Driven Engineering* [14]; *Component-Based Systems* [15]; *Open-Source Systems* [16]; and the *cloud* [17].

Research efforts of more than half a century have not only provided SCM with a solid process; but also powerful tools to support its implementation. However, SCM has been viewed as a bureaucratic task, with overwhelming documentations [18-19]; implementation that are too dependent on supporting tools and missing out the big picture of change management [20]; which indirectly leads to SCM focusing on some software artefacts whilst ignoring others [21]. In addition, there seems to be little interest in the '*human factors'* in SCM, although is it recognized that SCM best practices are not observed by developers [22]. Human is a significant factor in SCM given available tools and standard procedures. Thus, this paper aims to highlight the role of human in SCM implementation.

III. People-Centric Software Configuration Management Model

This section presents the research activities leading to the development of the People-Centric Software Configuration Management (P-SCM) model. It started with the identification of critical factors, followed by the formulation of components and then the development of model.

A. Identification of Success Factors

Two surveys were carried out to identify SCM success factors and issues that inhibit its implementation in Malaysia. The first survey involved the administration of questionnaire to 3 types of respondents namely the Government, Higher Education Institutes (HEIs), and IT Companies [23-24]. A total of 19 responses were obtained from 3 government agencies, 5 HEIs and 11 IT companies. The second survey was a series of interview sessions with 12 key informants from 5 public HEIs [25].

B. Formulation of Components

Results of the surveys where analyzed and factors that support and/or inhibit SCM implementation are identified (Tables 1 and Table 2). In total, 4 major factors were considered crucial in SCM.

1) People

People are project team members who are directly involved in SCM implementation. People are expected to possess a certain level of competency to implement Process, operate Tools and generate Documentation. Competency is a blend of knowledge (familiarity and understanding of SCM, acquired through formal/ informal education/training in Computing); experience (mastery of SCM, gained through involvement in software projects/trainings); professionalism (code of conduct); training (development of skills and knowledge that relate to competencies in key areas of Computing); and SCM skills (the ability to perform assigned SCM task within a predetermined time/effort).

2) Process

Process refers to SCM policy and procedures that are practiced by software organizations. Standard SCM activities are: planning (coordination of activities throughout the project); identification of *Configuration Items* (CIs); control (change management process including the authority for reviewing and approving changes and the procedure for change request); accounting (recording/reporting of change implementation status); auditing (evaluation of CIs to ensure conformance to requirements and/ or a baseline conforms to its configuration information); delivery (building - combining the correct versions of CIs into an executable program and delivery - packaging and delivery of the software product to a customer or other entities).

Table 1
Issues in SCM Implementation

Issues	People	Process	Tools	Documentation
Ambiguous Requirements	✓			✓
Bureaucracy	✓	✓		
Change Request Procedure		✓	✓	✓
Competency of Staff	✓	✓	✓	✓
Conformance Issues	✓	✓		
Conformance to Directives	✓	✓		✓
Development Team Size	✓			
Infrastructure			✓	
Key Performance Indicator	✓			✓
Lack of Dedicated SCM Manager	✓			
Lack of Dedicated SCM Team	✓	✓		
Lack of Project Monitoring Tools		✓	✓	
Lack of Project Reporting Tools		✓	✓	
Lack of Understanding (SCM)	✓	✓	✓	✓
Maintenance	✓	✓	✓	✓
Management's Awareness	✓	✓	✓	✓
Management's Commitment	✓	✓	✓	✓
Number of New Applications		✓		
Poor Communication	✓	✓	✓	✓
Poor Documentation	✓		✓	✓
Responsibilities of SCM Management/ Implementation	✓	✓	✓	✓
Restructuring Exercise	✓	✓		
SCM Docs. not included in Delivery	✓	✓		✓
Software Quality Docs. not included in Delivery	✓	✓		✓
Software Quality Evaluation not Documented	✓	✓		✓
Suitability of Tools			✓	
Target Platform (maintenance)		✓	✓	
Target Platform (testing/ implementation)		✓	✓	✓
Task Assignments	✓	✓		
Technological Change		✓	✓	✓
Utilization of Tools	✓	✓	✓	✓
Vague Organization Direction				✓
Vendor Support (Tools)		✓	✓	
Vital Artifacts not Controlled	✓	✓		✓

Table 2
SCM Success Factors

Success Factors	People	Process	Tools	Documentation
Artifacts Versioning	✓	✓	✓	
Audit Results	✓	✓	✓	✓
Change Control Board	✓	✓		✓
Change Request Reports	✓	✓	✓	✓

Success Factors	People	Process	Tools	Documentation
Clear Organization Direction				✓
Clear Requirements		✓		✓
Comprehensive Documentation	✓			✓
Conformance Documentation	✓			✓
Conformance to Directives	✓	✓		
Controlled Artifacts Documentation	✓	✓	✓	✓
Dedicated Development Team	✓	✓		
Dedicated Maintenance Team	✓	✓		
Dedicated SCM Manager	✓	✓		
Easy to Use Tools	✓		✓	
Efficient Communication Channel	✓	✓	✓	
External Consultant for SCM Implementation	✓	✓		
High Morale of Staff	✓			
Implementation of Audits	✓	✓	✓	✓
Implementation of Key SCM Proc.	✓	✓	✓	✓
Infrastructure Support		✓	✓	
Management's Commitment	✓	✓		
Policy for Change Control	✓	✓		
Reports Generation		✓	✓	✓
SCM Awareness	✓	✓		
Software Development Reports		✓	✓	✓
Software Quality Documentation		✓	✓	✓
Tools to Support Implementation		✓	✓	
Training	✓	✓	✓	
Use of Project Management Tools	✓	✓	✓	
Use of SCM Tools	✓	✓	✓	
Use of Software Libraries	✓	✓	✓	
Use of Standard Change Req. Forms	✓	✓	✓	✓
Utilization of Support Tools	✓	✓	✓	

The execution of Process will result in the SCM Plan (SCMP), a living document that is used throughout the project. SCMP provides a systematic view of the current configuration, supports decision-making activities in processing change request, provide information on the project/ software product status, and facilitate future enhancements through detailed documentation. The SCMP takes shape during Planning where the Contractual, Organizational, Project and Software Quality Requirements are identified (Figure 1).

During the Identification process, contractual and organizational requirements dictate the type of artefacts (CIs) that are going to be controlled. List of CIs and baselines are appended to the Project Documentation. In Control, contractual and organizational requirements characterize the change control authority and change request procedure. These information are then appended to Project Documentation.

In Accounting, change requests and CIs approval are processed. Change requests documentations, approved CIs and baselines are appended to Project Documentation. In Audit, the contractual, organizational, project and software quality requirements are taken into consideration and referred to in determining the approach for evaluation. Audit results are appended to Project and Software Quality Documentations.

Software product and other relevant documentation are added to the SCMP as appendices. In delivery, the Project and Software Quality Documentations are included in the software package to verify conformance.

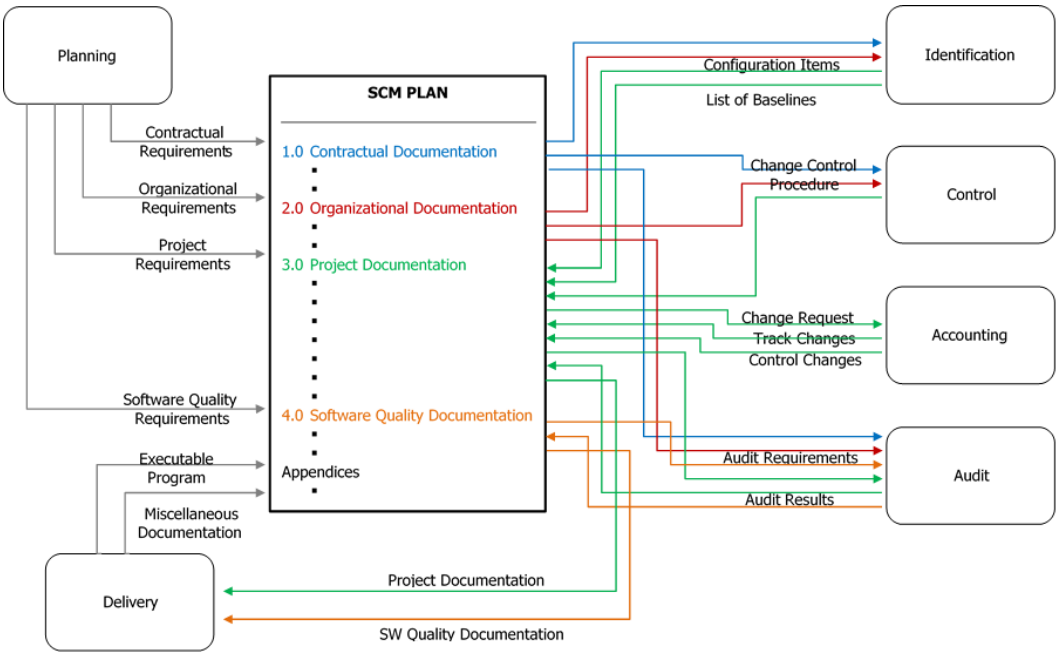

Figure 1: The SCM Pelan

3) Tools

Tools are means to support implementation. Proprietary and commercial SCM tools include versioning tools (tools for managing versions of a CI or a group of CIs, also referred to as revision control or source control tools); software build tools (tools for creating executable application from program source code); and software release tools (tools for distributing the final version of a software product). Additional tools include planning tools (management of planning activities); change request tools (management of change requests); reporting tools (project reporting); and audit management tool (management of project and software audits).

4) Documentation

Documentation refers to reports and records generated throughout the software project for ensuring conformance to requirements; as basis for management decision-making activities; and as project documentation. Documentation is dependent on the SCM process undertaken and embodied in the SCMP. Types of documentation include contractual (external stakeholders requirements); organizational (internal stakeholders requirements); software quality (product quality requirements based on the ISO 25010 standard); and project (management reports i.e. development plan, CIs, change control, project status, audit, builds and releases). Complete SCM documentation is listed in Table 3.

Table 3
Documentation in P-SCM

Type	SCM Process	Example
Contractual	Planning	• Vendor Control
Organizational	Planning	• Organization Policy • SCM Organization Chart • Schedule
Software Quality	Planning	• Quality characteristics • Evaluation strategy
	Audit	• Functional Conf. Audit • In-Process Audit
Project	Planning	• Life Cycle Process • Tool Selection • Branching/ Merging Strategies
	Identification	• Requirements/ Specifications • CIs Identification Scheme • Versioning Techniques
	Control	• *Software Change Request* Procedure • Configuration Control Board
	Accounting	• Approved CIs • Change status • List of CIs & baselines
	Audit	• Functional Conf. Audit • Physical Conf. Audit • In-Process Audit
	Delivery	• Executable Program • Project Documentation • Audit Results

C. Development of Model

P-SCM outlines SCM implementation based on the roles played by People, Process, Tools and Documentation (Figure 2). People implements Process, operates Tools, and generates Documentation. Tools are used to implement specific Process and generate Documentation. Process generates new and updates existing Documentation. Documentation guides People in implementing Process and decision-making activities.

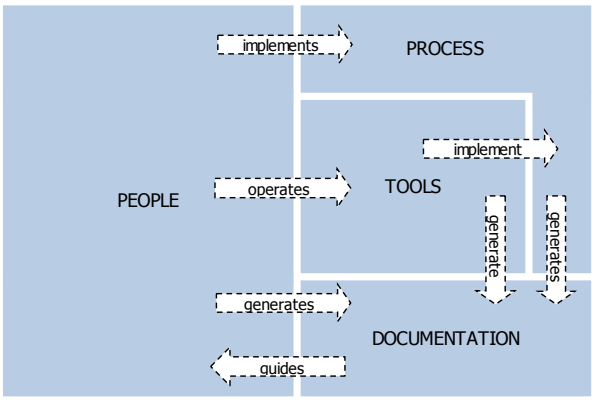

Figure 2: People-Centric Software Configuration Management Model

P-SCM does not replace traditional approach for SCM, but complements it through an alternative view of its implementation which is guided by human (People). P-SCM is scalable and can be tailored to suit the needs of specific software organizations. It can also be used in other areas of software engineering such as software quality.

IV. Example of P-SCM Implementation

P-SCM assesses the competency of the software project team (People), extensiveness of SCM activities (Process), availability of support tools (Tools), and comprehensiveness of the SCMP (Documentation) (Figure 3).

The criteria for assessing People are knowledge, experience, professionalism, training and SCM skills. Results reveal the competency of the project team based on the *Software Engineering Competency Model*. The criteria for assessing Process are the execution of standard SCM activities based on international standards including IEEE 828 and ISO 10007; the criteria for assessing Tools are the availability and suitability of tools for implementing SCM; and the criteria for assessing documentation are the comprehensiveness of the SCM Plan as outlined by international standards.

Results of P-SCM implementation would support software organizations in managing software projects. P-SCM implementation for active projects would support decision making activities and provide insights to project discrepancies (if any). Implementation for past projects would support identification of best practices and project pitfalls, enabling better planning for future projects. Implementation for future projects or at a start of a new project would support project planning and identification of training needs and tools procurements. P-SCM could also be administered to outsourced projects, supporting organizations in selecting suitable software vendor(s).

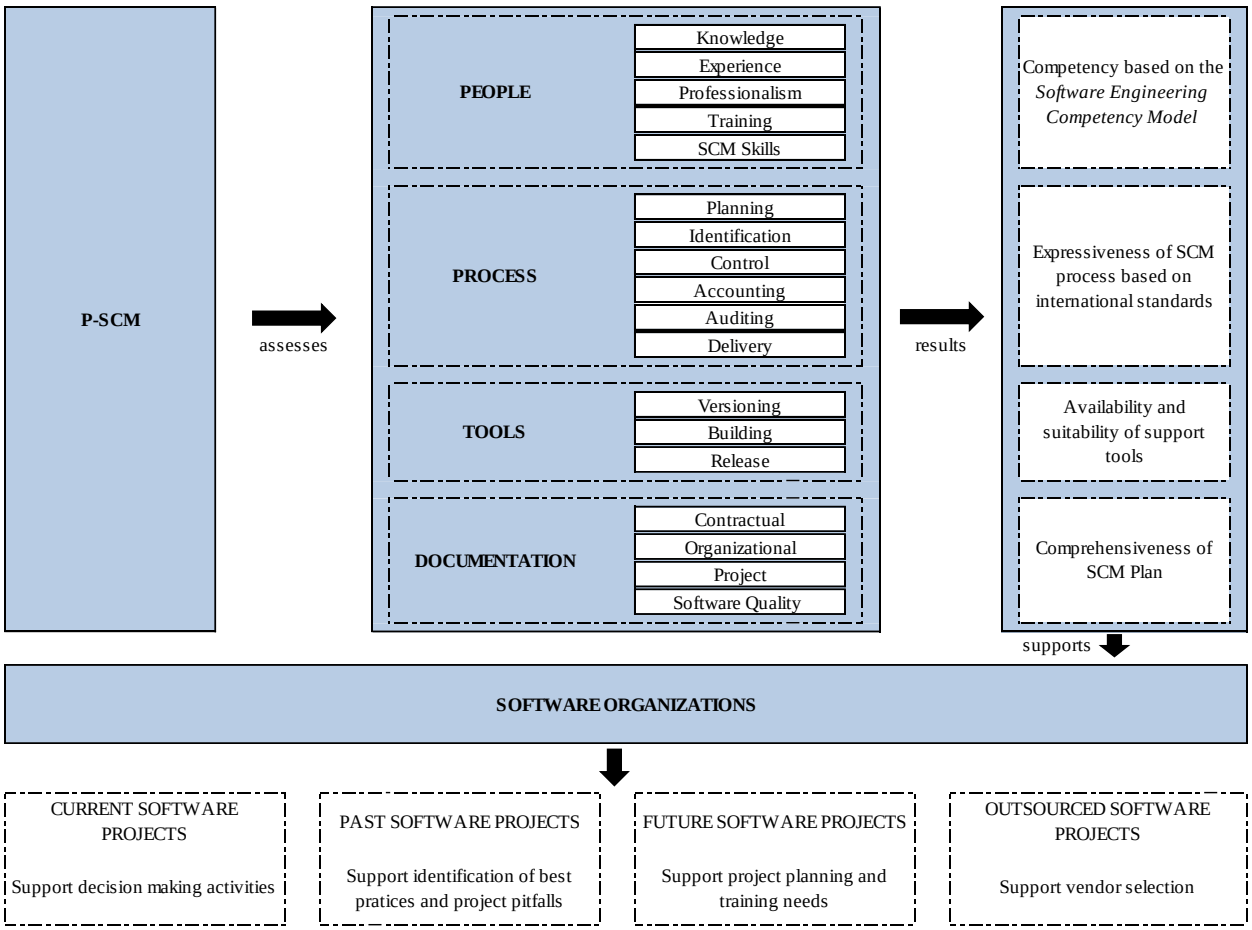

Figure 3: P-SCM Implementation

V. Comparison between Traditional SCM and P-SCM

The distinction between traditional SCM and P-SCM can be viewed from 3 aspects: the People, Software Quality and Documentation (Table 4).

Firstly, P-SCM emphasizes the role of People in SCM implementation, covering Process, Tools and Documentation. Traditional SCM relies heavily on the use of tools for implementation where People is mainly tasked to operate these tools.

Secondly, P-SCM emphasizes the importance of software quality where it is identified in Planning, explicitly stated in Control, effects of a proposed change to quality is taken into consideration in Accounting, quality audits are carried out in addition to other project audits, and quality documentation is included in Delivery. Traditional SCM implementation gives little emphasis (if any) to software quality as quality assessment it is usually carried out in other process.

Finally, P-SCM promotes the delivery of project-specific and quality documentations as proof of conformance and to support future enhancements to the software product. Traditional SCM process does not pass these information as part of the product package.

VI. Validation of Model

Two types of validation are carried out to the proposed P-SCM model: theoretical and empirical validations.

A. Theoretical Validation

The theoretical aspects of P-SCM was validated through expert reviews from the industry and higher education institutions. Interview sessions were held with 4 software practitioners discussing the components of P-SCM, selected assessment criteria, and significance/ contribution of P-SCM to software organizations. Results of these reviews established the soundness, plausibility and practicality of P-SCM to software organizations.

B. Empirical Validation

Empirical validation of P-SCM involved comparing P-SCM results to traditional SCM implementation. This validation is on-going and results would be available as early as Q4 of this year, as it is dependent on the completion of active software projects. Comparison data include the number/ percentage of overdue projects, overdue/ undelivered projects due to software quality issues, and undelivered projects. Initial results indicates that P-SCM provides additional input in planning training needs (at the start of a new project/ future projects) and provides additional input in the selection of software vendors (for outsourced projects).

Table 4
Comparison between Traditional SCM and P-SCM

Component	Traditional SCM	P-SCM
People	Mainly tasked with the operation of tools	Dominant factor in SCM implementation
Process		
Planning		
Identify Contractual Req	✓	✓
Identify Org Req	✓	✓
Identify Project Req	-	✓
Identify SWQ Req		
Identification		
Identify CIs	✓	✓
Release Baselines	✓	✓
Control		
Form CCB	✓	✓
Establish CR Procedure	✓	✓
Software quality explicitly stated in change request	-	✓
Accounting		
Track Change Request	✓	✓
Control Changes	✓	✓
Effect of proposed change to quality taken into consideration	-	✓
Auditing		
SCM Audit	✓	✓
Software quality audit carried out as part of the project's requirements	-	✓
Delivery		
Software Building	✓	✓
Software Release	✓	✓
Inclusion of SCM and software quality documentations in delivery	-	✓
Tools	Dominant factor in SCM implementation	Support SCM implementation
Documentation	Mainly acts as project documentation	Guides SCM implementation and passed over to facilitate future product enhancements

VII. Conclusion and Future Work

This paper has proposed model of people-centric software configuration management. P-SCM outlines SCM implementation from the aspects of people, process, tools and documentation.

The P-SCM model is supported with strong theoretical background including international standards coupled with industrial best practices and procedures as evident from the surveys carried out. Critical components and relationship between them were systematically identified to ensure successful implementation by software organizations.

The implementation of P-SCM reveal the competency of the software project team, extensiveness of SCM activities carried out by the organization, availability of supporting tools, and comprehensiveness of the SCM Plan. These in turn, support decision making activities, provide insights to project discrepancies, identify best practices and pitfalls, identification of training needs and tools procurements in the organization.

Both theoretical and empirical validations are carried out to ensure theoretical soundness, plausibility and practicality of P-SCM. Initial results were encouraging in providing additional input in project planning and outsourcing to software organizations.

Future works include a much broader implementation base of P-SCM and the number of software projects/ teams involved.

Acknowledgment

The authors would like to acknowledge the contribution of TATI University College and Universiti Malaysia Terengganu in this project.

References

[1] IEEE 828 Standard for Configuration Management in Systems and Software Engineering, the Institute of Electrical and Electronics Engineers, 2012.

[2] IEEE 15939 Standard Adoption of ISO/IEC 15939:2007 - Systems and Software Engineering - Measurement Process, the Institute of Electrical and Electronics Engineers, 2009.

[3] ISO 10007 Quality Management - Guidelines for Configuration Management, International Organization for Standardization, 2009.

[4] ISO/IEC 12207 Standard for Systems and Software Engineering - Software Life Cycle Processes, International Organization for Standardization, 2008.

[5] ISO/IEC 15288 - Systems and Software Engineering - System Life Cycle Processes, International Organization for Standardization, 2008.

[6] ISO/IEC 15939 Systems and Software Engineering - Measurement Process, International Organization for Standardization, 2007.

[7] ISO/IEC/IEEE 24765 Systems and Software Engineering - Vocabulary, International Organization for Standardization, 2010.

[8] S. I. Feldman, "Make - A program for maintaining computer programs," *Software, Practice and Experience*, vol. 9, no. 3, pp. 255–265, 1979.

[9] W. F. Tichy, "RCS - A system for version control," *Software - Practice and Experience*, vol. 15, no. 7, pp. 637-654, July 1985.

[10] A. Wright, *Requirements for a Modern CM System*. CaseWare, Inc., 1990.

[11] Sun Teamware Product Documentation. Sun MicroSystems Inc., 2000.

[12] S. P. De Rosso, and D. Jackson, "What's wrong with git? A conceptual design analysis," in *Proc. 2013 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software*, New York, 2013, pp. 37-52.

[13] L. McVoy, "Preliminary product line support in bitkeeper," in *Proc. 19th International Conference on Software Product Line*, New York, 2015, pp. 245-252.

[14] T. Buchmann, A. Dotor, and B. Westfechtel, "MOD2-SCM: A model-driven product line for software configuration management systems," *Information and Software Technology*, vol. 55, no. 3, pp. 630-650, March 2013.

[15] P. Kaur, and H. Singh, "A layered structure for uniform version management in component based systems," *SIGSOFT Software Engineering Notes*, vol. 34, no. 6, pp. 1-7, Dec. 2009.

[16] Y. Ki, and M. Song, "An open source-based approach to software development infrastructures," in *Proc. IEEE/ACM International Conference on Automated Software Engineering*, Washington, 2009, pp. 525-529.

[17] T. Mikkonen, and A. Nieminen, "Elements for a cloud-based development environment: online collaboration, revision control, and continuous integration," in *Proc. WICSA/ECSA 2012 Companion Volume*, New York, 2012, pp. 14-20.

[18] E. H. Bersoff, V. D. Henderson, and S. G. Siegel, *Software Configuration Management: An Investment in Product Integrity*. Prentice-Hall, Inc., 1980.

[19] F. J. Buckley, *Implementing Configuration Management: Hardware, Software, and Firmware*. IEEE Computer Society Press, 1993.

[20] R. Conradi, and B. Westfechtel, "Version models for software configuration management", *ACM Computing Surveys*, vol. 30, no. 2, pp. 232-282, 1998.

[21] M. C. Chu-Carroll, J. Wright, and D. Shields, "Supporting aggregation in fine grained software configuration management", in *Proc. 10th ACM SIGSOFT Symposium on Foundations of Software Engineering*, New York, 2002, pp. 99-108.

[22] R. Premraj, A. Tang, N. Linssen, H. Geraats, and H. van Vliet, "To branch or not to branch?", in *Proc. 2011 International Conference on Software and Systems Process*, New York, 2011, pp. 81-90.
[23] A. Deraman, J. Yahya, S. Fahmy, and Z. Mohamad, "Technical Report 59297-2. Survey on software configuration management approaches in Malaysia," unpublished.
[24] S. Fahmy, A. Deraman, and J. Yahaya, "Software Configuration Management Practice In Malaysia," *Journal of Theoretical and Applied Information Technology*, vol. 94, no. 1, pp. 1-17, 2016.
[25] A. Deraman, J. Yahya, Z. Mohamad, and S. Fahmy, "Technical Report 59297-3. Software configuration management in public higher education institutions: an observation", unpublished.