The International Journal Of Engineering And Science (IJES)
|| Volume || 4 || Issue || 2 || Pages || PP.69-89|| 2015 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Model-Driven Context-Aware Approach to Software Configuration Management: A Focus on Small and Medium Software Development Firms

[1]Davis Nyakemwa Onsomu, Msc, [2]Elisha Ondieki Makori, PhD, [3]Patrick Kinoti, Msc

[1]Faculty of Information Science and Technology, Kisii University
[2]Faculty of Information Science and Technology, Kisii University
[3]Faculty of Information Science and Technology, Kisii University

--------------------ABSTRACT--------------------

Most of the small and medium software development firms do not appreciate and embrace existing software configuration management models due to the bureaucratic nature of the models' design and perceived bias portrayal towards large firms. Software configuration management is a key component in the general software engineering process that leads to the realization of quality produced software and software products. As a result, there is need to address this particular gap by proposing contextualized software configuration management model for small and medium software development firms, especially in developing countries that operate in different policy, regulatory, industry and organizational contexts from the firms in developed countries. Specific objectives of the study included: to establish the approach employed by small and medium software firms in relation to software configuration management; evaluate the effectiveness of the existing software configuration management model employed in small and medium software firms; assess the challenges faced by small and medium software firms in software configuration management practice; propose contextualized SCM model that is relevant and beneficial to small and medium software firms in Kenya and other developing countries and evaluate the effectiveness of the proposed contextualized SCM Model in small and medium software development firms. The model developed drew certain elements of each of the four traditional software configuration management models to come up with an enhanced and improved model. The proposed model capitalized on addressing the weaknesses inherent in the existing models by proposing process modelling approach that includes context into process descriptions, enabling process owners to design processes for change and switch such processes during execution. In construction, the proposed model adopted the ideologies of definition of context and design for change. The study sample was selected from the population of small and medium software development firms within Nairobi city. The study used the survey research and naturalistic observation to collect data. Data collected was coded, analyzed using the Statistical Package for Social Sciences, Microsoft Excel and presented in the form of tables of frequencies, percentages, means, standard deviations and graphs. The study results indicated that most of the firms studied employed traditional models whereas minority did not practice any model. In addition, majority firms did not practice conventional and standard phases of software configuration management across all software projects undertaken. The study identified numerous challenges regarding software configuration management practice in small and medium software development firms. This study established strong indication that no specific contextualized software configuration management model was in existence to address the needs of small and medium software development firms in developing countries including Kenya. The study proposed model that was highly approved and recommended by the respondents, since the model captured the aspirations and needs of the small and medium software firms. The study recommends further development of the proposed model into a software tool to be commercialized.

Keywords:*Software configuration management, contextualization, small and medium software development firms, software engineering.*

Date of Submission: 08-September 2015, ⟺ Date of Accepted : 06-March 2015

I. INTRODUCTION

Background to the Study

Studies indicate that various software configuration management (SCM) models have been designed in developed countries such as China, United States of America, Brazil and Denmark. The models have been developed using different architectures and include component-based software development, POEM, Odyssey-VCS and Ragnarok architectural model (Mei, Zhang & Yang, 2002; Lin & Reiss, 1995; Murta et al, 2007; Christensen, 1999).

Component-based software development model was designed to support software development process together with traditional software configuration management in order to solve issues related to logical software constituents and relationships. In the United States of America, POEM software configuration management model stores large software artifacts such as source code, object code and documents as files in the underlying file system without allowing users to directly access files and directories of the underlying file system. The Brazilian's Odyssey-VCS design, is the integrated software configuration management model for unified modelling language. This model composes of version control system and two complementary components of customizable change control system and traceability link detection tool that uses data mining to discover change traces among versioned unified modelling language (UML) model elements, and provides the rationale of change traces, automatically collected from the integrated SCM infrastructure. This model is focused towards software configuration management on software developed using fine-grained UML model elements (Murta et al, 2007).

In Denmark, Ragnarok architectural model allows tight version control and configuration management of the architecture of the software system. The model takes the logical software architecture as the starting point and uses this structure to drive the version-and-configuration control process. Ragnarok places strong emphasis on reproducibility of configurations and architectural changes. In addition, this model emphasizes the application to the handling of software with evolving architecture tendency (Christensen, 1999).

Findings reveal that most of the software configuration management models in existence in the world today, evolved completely independent of each other based on the needs of the unique platforms design, and perceived ways in which the software was developed in respective environments.

Many companies created home-grown SCM models to meet own specific needs while software vendors responded with plethora of designs with bias towards single platform or context (Cravino et al, 2009). In some developing countries including Kenya, different studies show that there is lack of software configuration management model that specifically addresses the needs of small software development firms (Pino, Garcia & Piattni, 2009; Mohan et al, 2008; Er & Erbas, 2010; Kogel, 2008). Small and medium software development firms in developing countries operate in different policy, regulatory, industry and organizational contexts. Additionally, the applicability of the models designed for developed countries is not always relevant to small and medium software firms in developing countries. There is lack of software configuration management model that looks at the SCM practice from process-centered functionality area view of configuration management functionality requirements with the aim of providing contextualized approach for the firms in addressing pertinent issues, problems and weaknesses inherent in existing models and even systems (Humble & Farley, 2010; Balamuralidhar & Prasad, 2011; Ochuodho & Brown,1991; Hong et al,2002; Rosenblum & Krishnamurthy,1991).

There are four traditional standard SCM models in existence from which SCM models are accepted and applied in business organizations (Feiler, 2010; Sovran et al, 2011; Dix & Gongora, 2011; Rodriguez et al, 2011; Rubin et al, 2008; Zhu et al, 2011; Kaur & Singh, 2011). These include the check-out/check-in, composition, long transaction and change set. The classification is based on certain patterns observed in support of the repository which is the centralized library, that consists of objects under configuration management control. Most SCM systems today are essentially based on any one of these models. Despite this, this study has identified paramount weaknesses in these four models and hence shall capitalize on the same to yield contextualized software configuration management model that is specific to the needs of small and medium software development firms in developing countries.

Statement of the Problem

Small and medium software development companies form large population out of the total number of software companies in the world. Start-up companies play significant role in the booming software economy, although literature discussing the issues of small and medium software development firms in terms of software configuration management process or methods is virtually non-existent. One of the greatest difficulties in applying software configuration management in small companies is the unawareness of the importance of that activity and, sometimes, the idea that the task is bureaucratic service that only produces delays.

There is evidence that majority of small and medium software firms are not adopting existing standards, perceiving the standards as being oriented towards large organizations. Existing standards and models are more complex for small enterprises to comprehend owing to inadequate availability of skills and resources. Studies show that small and medium firms' negative perceptions of process model standards are primarily driven by negative views of cost, documentation and bureaucracy. Different studies indicate lack of software configuration management model that specifically addresses the needs of small and medium software development firms especially in developing countries (Pino, Garcia & Piattni, 2009; Mohan et al, 2008; Er & Erbas, 2010; Kogel, 2008).

Small and medium software development firms in developing countries operate in different business environment that is not always conducive for the applicability of the models designed for developed countries. Additionally, studies indicate that there is lack of software configuration management model that looks at the practice from process-centered functionality view, which is integral part of configuration management functionality requirements, in relation to providing contextualized approach for small and medium software development firms in addressing pertinent issues, problems and weaknesses inherent in existing models and even systems (Humble & Farley,2010; Balamuralidhar & Prasad,2011; Ochuodho & Brown,1991; Hong et al, 2002 & Rosenblum & Krishnamurthy,1991).

Purpose of the Study

The main objective of this study was to propose software configuration management model that is suited to the needs and captures the aspirations of small and medium software development firms in Kenya and other developing countries.

Objectives of the Study

The specific objectives of the study include to:

i. Establish the approach employed by small and medium software firms in relation to SCM.
ii. Evaluate the effectiveness of the existing SCM model employed in small and medium software firms.
iii. Assess the challenges faced by small and medium software firms regarding SCM practice.
iv. Propose contextualized SCM model that is relevant and beneficial to small and medium software firms in Kenya and other developing countries.
v. Evaluate the effectiveness of the proposed contextualized SCM model in small and medium software firms.

Research Questions of the Study

i. What approaches are employed by small and medium software development firms in relation to SCM practice?
ii. What is the effectiveness of the existing SCM model employed in small and medium software firms?
iii. What are the challenges faced by small and medium software firms regarding SCM practice?
iv. To what extent is the proposed SCM model relevant and beneficial to small and medium software firms in Kenya and other developing countries?
v. To what extent is the proposed SCM model effective to small and medium software firms?

Assumptions of the Study

This study was based on the following assumptions:

i. Small and medium software development firms in Nairobi, Kenya practice software configuration management in the firms' day to day operations.
ii. Small and medium software development firms in Nairobi, Kenya have the capacity and willingness to perform software configuration management at the level suggested by this proposed SCM model.

Scope of the Study

The study focused on small and medium software development firms in the city of Nairobi, Kenya. In addition, the study focused on the software configuration management aspect of the software engineering process. This involved change management in the software engineering industry with a view of proposing a model tailored for the needs of small and medium software development firms in Kenyan context. Small and medium enterprises include business enterprises with number of workers not exceeding 50.

Limitations of the Study

This study envisioned the following limitations:

i. The study aimed at developing software configuration management model, however the practical development of the proposed solution is left to interested persons who would wish to carry on with the study.

II. LITERATURE SURVEY

Research Gaps in Existing SCM Models

Existing models have numerous challenges or research gaps as noted by various authors grouped as process functionality, auditing functionality, accounting functionality and controlling functionality.

2.3.1 Process Functionality

Process functionality involves a number of aspects such as clear definition of processes, support and indiscipline indistinction, invalidated effectiveness of life cycle support, unclear task management process, informational indecisions in tool use and invalidation of automated workflow systems.

In the aspect of clear definition of processes**,** although every SCM system comes with built-in process in the small (check-out/ check-in cycle and long transactions), the degree to which large scale processes are supported varies. Professional experience advises that the big leap forward is the clear definition of software processes. Use of tools is beneficial only if the tools are really supportive although such tools take the role of bureaucrats increasing the number of required interactions for the developers. SCM systems that are too rigid in enforcing the process are cursed by developers and reduce effectiveness (Fruhauf & Zeller, 1999; Loumos et al, 2010; Aiello & Sachs, 2010; Berzisa & Grabis, 2011).Resource implications – particularly that of management time – mean that the implementation process is markedly more taxing for small and medium enterprises than large companies. Consequently, well-designed development process, with clear focus and effective process management improves efficiency and the likelihood of success, (Hudson et al, 2001).

In relation to support and discipline indistinction, the distinction between support (use of tools) and discipline (use of standard) remains to be validated in existing SCM models (Schmidt, 2012).The SCM automated tools used for the project and described in the software configuration management plan need to be compatible with the software engineering environment development or maintenance occurs. SCM tools offer wide range of capabilities, and the most useful tool set for supporting the engineering and management environment has to be chosen from among other available tool sets (IEEE Standard for SCM Plans, 1990).

In the context of quality, SMEs are finding it hard to distinguish between use of tools and use of standards as the requirement for marketing rather than for quality reasons. As a result SMEs in particular are not benefiting sufficiently from the quality industry, and thus, affecting the quality of products and services, confusing the system and displaying alarming lack of appreciation (Jones et al, 2010; Schmidt, 2012; European Telecommunications Standards Institute, 2011).

In invalidated effectiveness of life cycle support, one failure of existing SCM models is that the effectiveness of life cycle support has not been validated. The distinction between support and discipline, and thus, the effectiveness of life cycle support remains to be validated (Fruhauf & Zeller, 1999; Chen et al, 2011; Weinreich & Buchgeher, 2012; Crowston et al, 2012). Software SMEs view life-cycle support as being infeasible (overly time-consuming or costly to implement) rather than non-beneficial. Unlike the high-process focus in life cycle support, SMEs often adopt low process focus electing only to implement process improvements in response to negative business events (Clarke et al, 2010; Baddoo & Hall, 2010; Clarke et al, 2011).

In the aspect of unclear task management process, rather than enforcing activities, more advanced SCM systems offer means to track current and pending processes. Task management is the area overlapping with (project) management. If tools are used then there is need to carefully decide the type of information to be kept in the SCM model and the project management tool. Tight coupling of work activities with the state control of the work results leads to sluggish SCM systems (Fruhauf & Zeller, 1999; Klosterboer, 2010; Sarma & Hoek, 2008); therefore, in the existing models, task management is not clear. Considering evidence of important software process improvement occurring to the system life cycle, SMEs find it difficult to distinguish between task management and project management. This can be the case where there is SCM-specific process that has corresponding parent project level process, for example, the configuration identification and the software configuration identification process. There is strong overlap between task management and project management processes (Clarke et al, 2010; Baddoo & Hall, 2010; Clarke et al, 2011).

Regarding informational indecisions in tool use, task management is the area overlapping with (project) management. If tools are used, then there is need to carefully decide the type of information to be kept in the SCM model and project management tool. Failure of existing SCM models involves where the SCM tools used, is not carefully decided which type of information is kept in the model and project management tool (Fruhauf & Zeller, 1999; Heer et al, 2010; Dabbish et al,2010).

Most SMEs share characteristics that distinguish them from large enterprises. In contradiction, such characteristics may also impose restrictions on such firms' economic, human and technological aspects such as technology adoption (Rivas et al, 2010).

In invalidation of automated workflow systems, ultimate process support is achieved with automated workflow systems. To the contrary, such systems are not yet validated raising queries on how such systems handle workflow automatically. In practice, work flow is typically organized by informal communication. Most SCM systems support triggers that are associated with specific events like automatic notification by e-mail whenever change occurred. These communication features are well-understood, cheap and effective means for simple work flow support (Wang et al, 2012; Elmroth et al, 2010; Fruhauf & Zeller, 1999).One weakness or failure of existing SCM models is that automated workflow systems that achieve ultimate process support need to be validated. Workflow system that achieves process support in software SMEs is evidently deficient giving room for non-validation of processes within business operations that may hamper process improvement initiatives (Yahaya et al, 2012 & Ozcelik, 2010).

2.3.2 Auditing Functionality

Auditing functionality involves the aspect of traceability of related documents that is lacking in existing SCM models. Queries are raised on how changes during implementation can be traced back to the design phase and the requirements phase. Further queries have been raised regarding the relationship between changes in implementation and in documentation.

Every SCM system provides mature and widely used features to inquire about the change history of specific configuration items. In contrast, the unsolved problem is the traceability of related documents although change-based versioning or activity-based SCM (Micallef & Clemm, 1996), allows these changes to be associated with each other. There is still room for improvement in this particular aspect (Anquetil et al, 2010; Mader et al, 2012; Fruhauf & Zeller, 1999).Software configuration status accounting is the record keeping and reporting activity performed by the configuration librarian to maintain the traceability of changes and product versions. This may not be applicable in majority of software SMEs since such firms tend to view the procedure as overly bureaucratic and time-consuming (Habra et al, 2011).

2.3.3 Accounting Functionality

Accounting functionality involves the aspect of deficiencies in tagging. Accounting facilities let users (and managers) inquire about the status of the product. SCM systems at least allow classifying components and versions according to specific properties (experimental, proposed or stable). Consequently, existing SCM models are facing pending problems in the simple tagging method used to facilitate the classification of components and versions according to specific properties (experimental, proposal or stable) (Fruhauf & Zeller, 1999; Treude & Storey, 2009; Kim & Youn, 2010).Software SMEs disregard the techniques and procedures that guarantee proper tagging used to facilitate classification of versions and components during software status accounting of the SCM. This results in misclassification of versions that undermines version and component traceability (Habra et al, 2011; Ozcelik, 2010; Yahaya et al, 2012).

2.3.4 Controlling Functionality

Controlling functionality involves the aspect of failed control processes. Tracking of change requests and defect reports is at the heart of the maintenance process, starting as soon as independent testing begins. The process of handling these, especially responsibility for decisions and definitions of records to be kept, determines the responsiveness of the organization on user needs. In small organizations, simple Excel sheet provides enough support, however, bigger organizations require elaborated database with dedicated queries, failure in existing SCM models. Tracking of product defects is significant SCM topic that provides immediate insight on the current product quality. Bug-tracking tools frequently come as standalone tools, from the freely available GNATS system to elaborated commercial systems. On the contrary, the integration with SCM repositories as well as automated testing facilities still leaves a lot to be desired, raising challenges for SCM vendors and researchers (Rupareila, 2010; Chen & Chen, 2009; Fruhauf & Zeller, 1999).Software SMEs are evidently noted for casually handling the issue of change request tracking and this undermines the quality of the final software product considerably (Rivas et al, 2010; Mader & Gotel, 2012; Loumos et al, 2010).

2.3.5 Other Research Gaps in Existing SCM Models

Other challenges or research gaps identified in SCM Models are explained as follows: In the aspect of mismanagement of change requests, advanced SCM systems (Whitgift, 2001) offer elaborated management of change requests. The effectiveness of the process remains to be validated, although improvements are more likely to come from SCM vendors than from SCM researchers (Hadden, 1998 & Fruhauf & Zeller, 1999). The effectiveness of the elaborated management of change requests whereby the whole development process is organized along the processing of change requests as depicted in the LIFESPAN SCM system/ model needs to be validated. Software SMEs are evidently noted for casually handling the issue of change request tracking and this undermines the quality of the final software product considerably (Rivas et al, 2010; Mader & Gotel, 2012; Loumos et al, 2010).

Disintegration of interfacing processes is where advanced SCM models like LIFESPAN offer elaborated management of change requests where the whole development process is organized along the processing of change requests. The effectiveness of the process remains to be validated by existing SCM models. Tracking of product defects provides immediate insight on the current product quality, however, the integration with SCM repositories as well as automated testing facilities still leaves a lot to be desired which is failure on the part of existing SCM models (Biffl & Schatten, 2009; Fruhauf & Zeller,1999; Bose et al, 2008).

Integration of product defects tracking, SCM repositories and testing facilities is an area of concern in software SMEs that hampers collaborative software development when in absence, more so in distributed environment like that of small scale offshore software development projects (Boden et al, 2008; Katchow et al, 2011; Duhan et al, 2012).Existing SCM models have not been integrated with the organization's business process (especially the software development process) and this is the failure on the part of the existing SCM models (Aiello & Sachs, 2010 & Moser et al, 2010). SCM systems and the business process are regarded as two different entities more so in small and medium software SMEs. This may lead to the SCM process that does not bear relevance to the SME's business agenda leading to the subsequent withdrawal from business operations. This may undermine the quality of the final software product (Clarke et al, 2010; Loumos et al, 2010; Clarke et al, 2011).

In inflexibility of SCM models, the software organizations should employ various software tools for completing projects properly (in terms of budget, schedule and quality) according to defined software process. The necessity of using tools for software development is increasing steadily due to cost and schedule pressures on software projects and increasing complexity of projects in terms of management and technical aspects. Indeed, it is impossible to perform most of the tasks without the use of corresponding tools. As the use and importance of these tools is increasing, the integration of tools becomes an issue under consideration.

The integration of such tools enabling the streamlining of individual tools by providing sharing of data and methods among applications (Nalbant, 2004).There exist studies regarding the integration of these tools, although these studies are not in the desired level (Forte, 1989 & Sharon & Bell, 2000). These studies focus on the achievement of collaborative working of tools with each other. On the contrary, the need for the integration to collect and unify high-level operational information in order to enable quantitative management (planning, execution, monitoring) of software projects, remains uncovered (Nalbant, 2004).

The existing SCM systems/models were initially designed for bigger structures. The cost of evaluation process and its duration is disproportional to the available resources. The number of actors involved in the SCM process is very small and usually, one actor plays many roles. These factors compound to make flexibility of the SCM systems/models to blend with the software SME business process almost impossible (Aggarwal, 2012; Jimenez et al, 2010; Habra et al, 2011).

In the aspect of double maintenance, the problem occurs when the same version of a program, component or file has to be maintained in different places. With the growing maturity and increasingly powerful functionality of SCM systems, parallel development has become a norm rather than an exception. It is rare to find project in which locking is practiced (Sarma et al, 2007). Double maintenance is form of direct conflict and according to (Sarma et al, 2007), direct conflicts are caused by concurrent changes to the same artifact. Double maintenance is a problem to software SMEs as it leads to problematic issues of coordination and communication thus affecting productivity and product quality (Jimenez et al, 2010; Aggarwal, 2012; Duhan, 2012).

Simultaneous update is whereby the problem occurs when two developers check-out a component (Shamsaie & Habibi, 2011).The first developers commits modifications, while the second one checks-in the same, erasing the ones made by the first one. Simultaneous update occurs when two or more developers take the copy of the configuration item and make changes.

When the developer returns the modified configuration item to the master library, modifications made by developers who have returned own configuration item earlier are lost. Charge-out/charge-in or locking mechanism is required to prevent simultaneous update (ESA Board for Software Standardization and Control, 1995). Simultaneous update in software SMEs leads to substantial loss of time and computing resources as the work in question has to be re-done. This strains the already limited resources of the software SME and may affect productivity and product quality in the long run (Ghobakhloo et al, 2011; Alzaga & Martin, 2010; Jimenez et al, 2010).

In logical conflict, the problem occurs when changes are committed while the component or part of the program that has not been modified stops the changes from working (Priedhorsky & Terveen, 2011). The authors add that logical conflict may hamper the development of products in software SMEs leading to vast resource-consumption in solving the subsequent problems encountered under such situations.

In the aspect of bad branching strategy, the problem is manifested when the complex branching strategy applied creates difficulties in knowing the purpose of each branch or how the branches should be merged. In addition, this also can lead to merge problems. In relation to studies conducted by Shihab et al (2012), branching plays major role in the development process of large software. Branches provide isolation so that multiple pieces of the software system can be modified in parallel without affecting each other during times of instability. The need to move code across branches introduces additional overhead whereby the branch in use can lead to integration failures due to conflicts or unseen dependencies. Branches are used extensively in commercial and open source development projects, however, the effects that different branch strategies have on software quality are not well understood. Merge problems as a result of bad branching strategy are common in software SMEs as these firms do not have clearly established structures to manage branching during the firms' software development process. This may lead to problems in productivity and product quality (Anquetil et al, 2010; Kaur & Singh, 2011; Ruparelia, 2010).

Users need to better understand configuration management processes in order to be able to demand better supportive implementations for such processes. This requires detailed definition of CM processes; understanding of how much control is to be enforced compared to how much guidance is to be given by the process manager; adequate implementations; and monitoring of how well the process is followed and where implementations can be made. Better understanding and implementation of process enables improved support for users in attaining higher quality of product, more time for being productive on creative tasks and better forecasting of software costs (Loumos et al, 2010; Aiello & Sachs, 2010; Berzisa & Grabis, 2011).Certain steps must be carried out in logical or orderly manner, but there is little automated guidance as to which steps should be done when. The order of commands in the menu suggests the command order, but this is really a simple guide. At any point in time user cannot immediately know the next step. Furthermore, to implement the process, more than step sequences (control flow) are needed and some semantic context required too. The configuration and change control (CCC) turnkey system keeps audit trail of the CCC commands that the user issues. On the spotlight is the fact that the audit trail for emergency fixes gives no indication whether any file was checked out and changed. Consequently, there is no data associated with the audit trail, only some logging of actions. This information may be insufficient for particular organization where simple mechanism for the audit trail is provided as customers may want semantic content in the audit trail. In regard to this, the process implementation involving control flow of commands and avoiding capturing of data state is likely to be insufficient for the customer (Loumos et al, 2010; Aiello & Sachs, 2010; Berzisa & Grabis, 2011).

2.4 Research Gaps this Study Addressed

This study concentrated on the aspect of clear definition of processes in the process functionality requirement of SCM models. The following areas of weakness under this aspect were of major concern to this study, simultaneous update, logical conflict and tracking of change requests and defect reports.

In simultaneous update, charge-out/charge-in or locking mechanism is required to prevent simultaneous update. This study proposed the use of elements of the check-out/check-in model to prevent simultaneous update. Activities that use repository have long duration, while treating the entire activity as one transaction is impractical. Systems crashing during such an activity results in loss of days of work. As a result, the repository manager must support check-out and check-in of objects. The check-out operation copies the object from the shared repository into the user's private workspace. After working on the object, the user issues the check-in operation, which copies the object from the private workspace into the shared repository. Check-out and check-in execute as (separate) short transactions. Essentially, check-out sets a persistent lock on the object, which is released by check-in. Check-out should support shared and exclusive modes (Ghobakhloo et al, 2011; Alzaga & Martin, 2010; Jimenez et al 2010).

In logical conflict, this study proposed the embracement of elements of the long transaction model in order to address the problem of logical conflict. Transaction is started when making the change. The change is made in the workspace, which represents the working context and provides local data storage visible only within the scope of the workspace. This (workspace) may be mapped into the file system allowing transparent access to the repository for the development tools. The workspace consists of working configuration that are frozen states of previous working configurations. The working space originates from bound configuration in the repository or preserved configuration of enclosing workspace. When the changes are finished, the transaction is committed, which effectively creates new version of the configuration in the repository or enclosing workspace and makes the changes visible outside the workspace. Finally, the workspace may be deleted or used for further changes. If the workspace originates from another workspace, the results is hierarchy of workspaces. The different levels in the hierarchy represent different levels of visibility. The bottom workspaces belong to the individual developers, one level up is the workspace for the team and the next level may be visible to the testing team and until the hierarchy ends to the repository (Priedhorsky & Terveen, 2011).

In the aspect of tracking of change requests (CRs) and defect reports (DRs), the change process begins when the need for the change occurs. The proposer of the change fills the change request form describing the change, reason, items and versions to be worked on. Each change request should also get an identification number. CRs go through the whole change process and shall be complemented with more information in each stage. After the CR has been initiated, it is evaluated and either approved or rejected by the configuration control board. After the evaluation, the configuration control board (CCB) may reject the CR and include the reason to the change request. If the CR is approved, it is delivered further for implementation. During the implementation of the change request, this study proposed that the change set model shall be applied, after which the process shall be verified (Rupareila, 2010; Chen & Chen, 2009; Fruhauf & Zeller, 1999).

The main concept in the change set model is the change set, which represents the set of modifications to different components making up the logical change. Typically when implementing the requested change to software requires modifications to several components. Change sets involve several aspects. Developers can work with groups of components belonging to the same logical change instead of dealing with each component separately. Change requests, which are descriptions of the changes to be made, may be easily linked to the actual changes made to the components. Queries on the dependencies between logical changes, changed components, and versions of configurations can be made (Rupareila, 2010; Chen & Chen, 2009; Fruhauf & Zeller, 1999).
These queries include determining which:

i. Component has been modified as part of the logical change
ii. Change sets are included in the particular configuration
iii. Configurations include the particular change

During the stage of change request, who is responsible for decisions and the definition of the records to be kept is determined. The next stage is to determine why the change request has been made. This involves two aspects – enhancements and error corrections. If the change request has been to correct errors, the next level shall be product defect tracking. The product defects tracking is integrated with two levels:

i. SCM repositories under which the check-out/check-in model shall be applied
ii. Automated testing facilities

The next stage after product defect tracking is "investigation to ascertain the cause of the error." At this stage, the cause of the error is determined. The next and final stage shall be "proposal to fix error and cost estimation to fix the error." To document the product knowledge, this study proposed the use of the SCM repository (Rupareila, 2010; Chen & Chen, 2009; Fruhauf & Zeller, 1999).

Design and Description of the Contextualized SCM Model

To address these challenges, this study proposed process modeling approach that includes the context into process descriptions, enabling process owners to design processes that can be changed and switched during execution. In this approach, the firm is viewed as the value producing mechanism with modular capabilities and flexible organization design for action. Change should be regarded as the switching of context of the software project using the process. The proposed process model integrates the context into software processes, enabling the process owners to "design processes for change" resulting in adaptive and modular processes. Moreover, the process designs can be reused by different projects with similar context and switched during process execution if the given context changes. The proposed model adopts two ideologies to the processes, the definition of context and the designing for change.

In the former ideology, the context includes the reason for being and restraints of the projects set by the environment or software development firm whereas, in the latter ideology, instead of inert process descriptions, the modular and ready-to-change design is suggested by the model.

The structural and logical elements of the proposed model are process sequence, process abstraction, context and solid processes. In process sequence, the ISO/IEC 12207 standard is adopted to decompose the software configuration management process. In the aspect of process abstraction, the unified modelling language is applied to define the operations performed, the inputs and outputs of the processes are portrayed together with the attributes. In context definition, the context is defined in terms of the strategy of the software firm, whether the firm focuses on research and development, market or client. In solid process, each context above is handled with the same abstract process with the same operations, but the way of accomplishment varies according to the context.

In adopting these four structural elements that are in line with the two ideologies aforementioned, the "element of contextualization from the study" aspect of the proposed model is realized.
The proposed model then focuses on three of the most significant weaknesses/challenges identified in the existing four standard SCM models to realize the contextualized model that is suitable for small and medium software development firms within the Kenyan context. The proposed model adopts elements of the check-out/ check-in model to solve the challenge of simultaneous update, long transaction model to address the issue of logical conflict and change set model to solve the problem of tracking of change requests and defect reports.
The process chain of the SCM process is shown below: The processes are broken down until the level where the process is performed by a single owner. At the end of the breaking-down, the process abstraction and the solid processes summarizing the differences in contexts of different projects are illustrated:

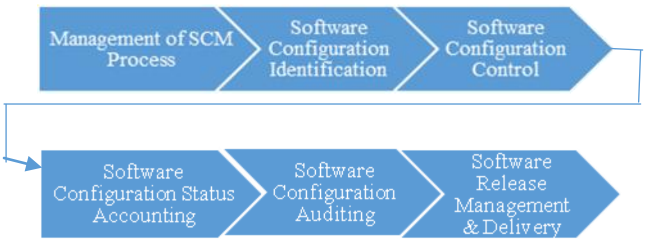


Figure 1: Software Configuration Management Process (adopted from ISO/IEC 12207:2008 Standards)

The element of contextualization from the study can be structured into four main structural elements, namely: process sequence, process abstraction, context definition and solid process. Firstly, in process sequence, the ISO/IEC 12207 Standard is adopted to decompose the software configuration management process. The software configuration management process is decomposed into the following sub-processes that occur in a successive definite chain: software configuration identification, software configuration control, software configuration status accounting, software configuration auditing and software release and delivery as shown in Figure 1 above.

This provides the model with a process-centric approach that enables it to be easier to apply, easier to adopt, easier to contextualize, easier to adapt to varying scenarios and contexts, easier to debug in case of occurrence of errors and easier to implement due to it modularity.
The illustrated "software configuration management process" is decomposed using ISO/IEC 12207 standard, and the process sequence is shown in figure 2 below:

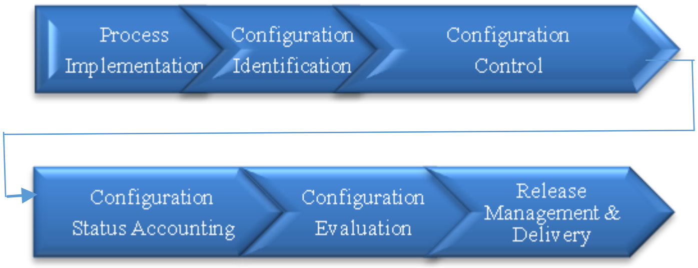


Figure 2:"Software configuration management" sequence (adopted from ISO/IEC 12207:2008 Standards)

Secondly, in process abstraction, assuming that "software configuration management" is performed by the single role in the firm (the process owner can be the person or group of people), breakdown is deemed to be complete and definition of the process abstraction starts. The portrayal of the abstraction is undertaken by the process owners. The process abstraction is depicted by notation derived from the unified modeling language. The operations performed, the inputs and outputs of the process are represented together with the attributes. The inputs of the process are produced by supplier processes and used as inputs to the consumer processes. The supplier processes in this case are the preceding procedures to the current stage in progress. The consumer processes are the procedures being fed by the preceding stage. The attributes are the individualities of the process that determine the changing behavior of the operations. For the "software configuration management process", the operation is "developing software configuration management processes" and the inputs are process implementation, configuration identification, configuration control, configuration status accounting, configuration evaluation and release management and delivery.The outputs are the "software configuration management processes". The attributes are the participation of end-users/customers, variety of project features, and diversity of SCM processes. There can be as many operations and attributes as desired.

The process abstraction provides the same interface to all projects using the "software configuration management" process. The changes in the context are summarized in solid processes which are determined in accordance with the context. In the next section, determination of context is portrayed.

Thirdly, after process abstraction, the next step in the model is context definition in alignment with the strategy of the software development firm. The firm strategy pervades in the portfolio of projects with different conditions through the context. Since the conditions and limitations can be different for singular projects, several contexts need to be defined. The reason to exist and the restraints of the projects are portrayed through the context. To illustrate the model, the three different possible contexts are involved as highlighted below:

Context 1: Research and development (R&D) focused context where types of projects exist to develop software for the purposes of gaining technical capability in certain domain. The restraints are conformance to certain standards, minimum profit level, and given level of client satisfaction. There are no or few clients at the time of development.
Context 2: Market-focused context whose reason for existence is profit. The restraints are conformance to capability maturity model integration (CMMI) level 5, and given level of client satisfaction. The number of clients and/or end-users is high in this kind of projects.
Context 3: Client-focused context: The type of project aims to fulfill the client's requirements. The restraints are conformance to capability maturity model integration (CMMI) level 5, minimum profit level and strict adherence to client requirements. These are usually client-specific projects developed with the participation of the client.

For context 1, high technology requirements may exist, whereas for context 2, use of familiar technologies and similarity to previous projects is of importance. For the third context, adherence to client requirements takes priority and the operations need to be carried out accordingly. Integrating the context in the process model enables the firm to act dynamically in response to changes in the environment. Afterwards, the solid processes are described for each context.

In context definition, determination of context is portrayed. Context is defined in terms of the strategy of the software firm whether the firm is focused on research and development, market or client. In solid process, each context above is handled with the same abstract process with the same operations, but the way of accomplishment varies according to the context.

Fourthly, in solid process, each context is addressed by the solid process as shown in Figure 4 below. Each solid process describes the same abstract process with the same operations, but the way of accomplishment varies according to the context. The number of solid process portrayals depends on the number of different portfolios in the software firms, and new contexts can be added to respond to the changes in the environment.

Moreover, projects having different contexts can use solid process from the repository suitable to the specific context, by switching the solid process being used. The process models are organized into a library of abstract and solid processes.

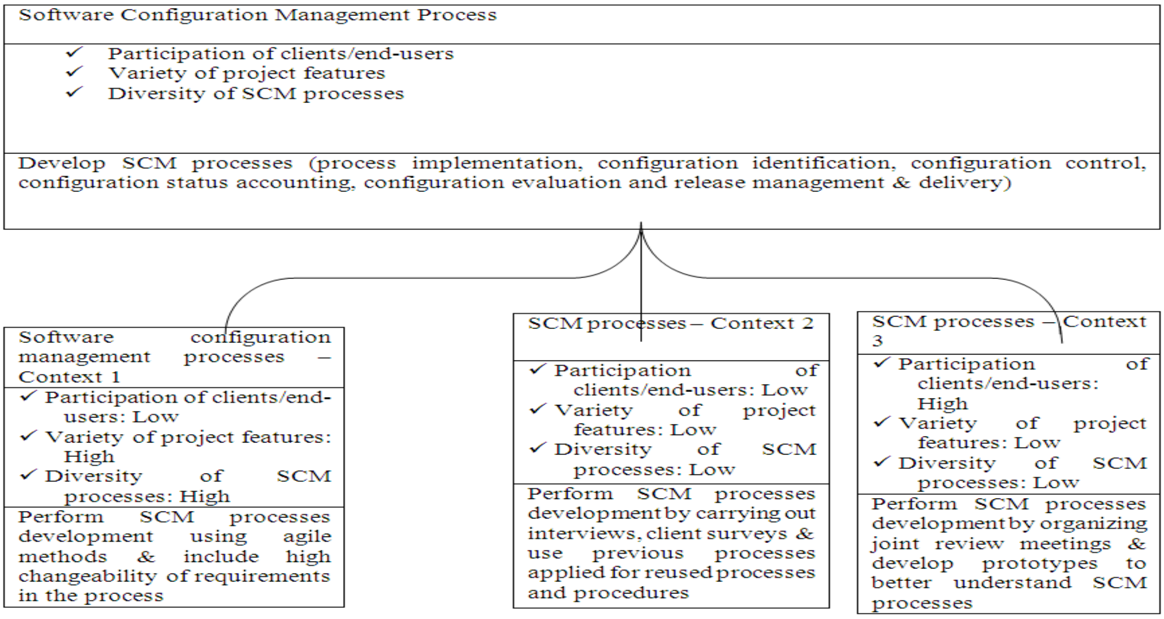
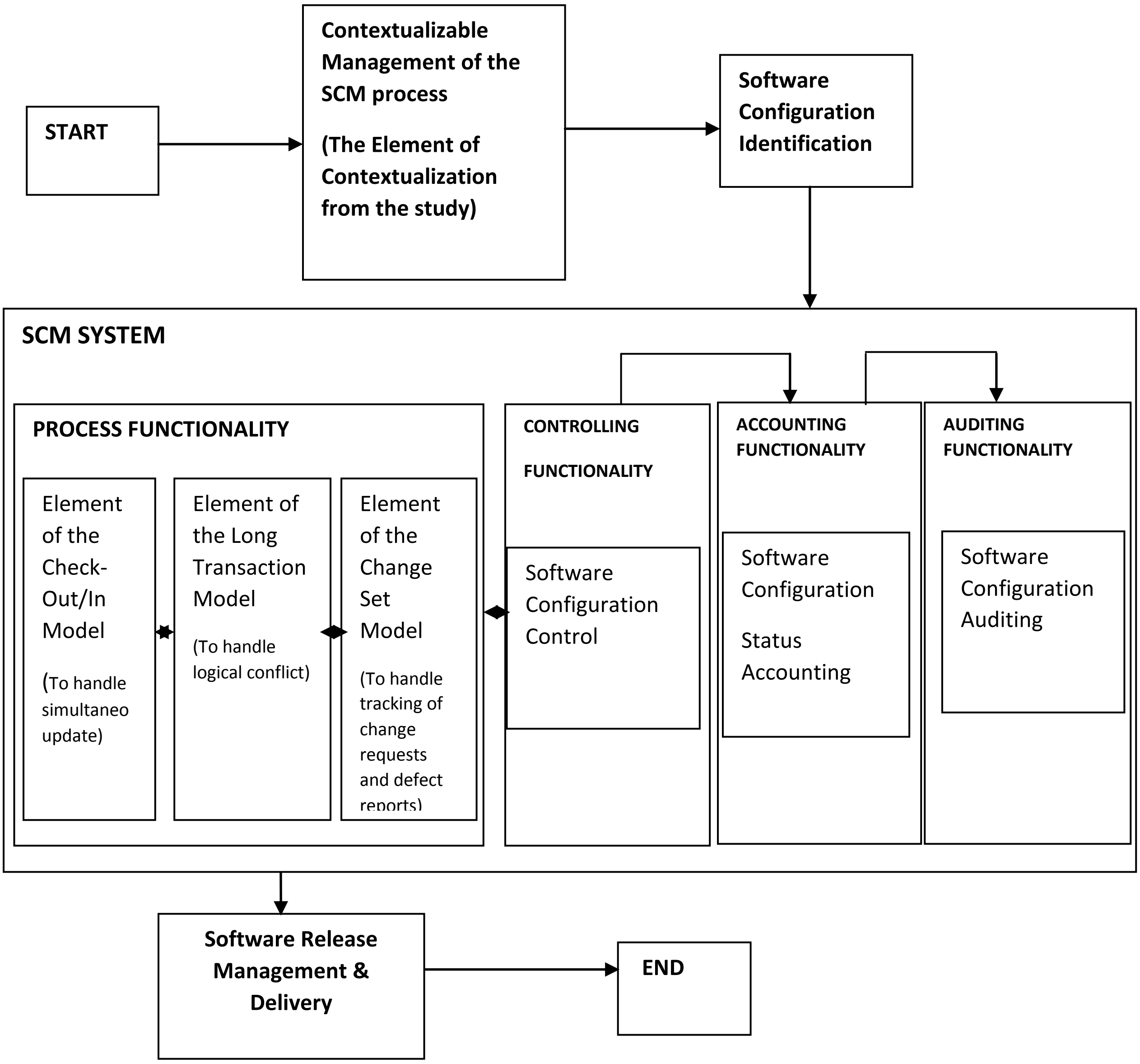


Figure 3: Solid Process Representation

After the contextualizable management of the SCM process, the next process is the software configuration identification, which leads to the software configuration system comprising of the process-centered functionalities of practice that occur sequentially as process functionality, controlling functionality, accounting functionality and auditing functionality. To address the challenge of simultaneous update, the element of check-out/check-in element is adopted. In relation to the challenge of handling logical conflict, the element of the long transaction model is adopted. Regarding the challenge of handling the tracking of change requests and defect reports, the element of the change set model is adopted. This entire chain of sub-processes forms the SCM system that supplies the final process of software release management and delivery eventually realizing quality software that has undergone all the necessary rigours, checks and balances of effective and standard model. The achieved SCM model has process-centered functionality view of the software configuration management process. This approach includes the context into process descriptions, enabling process owners to design processes with future change in consideration and switch processes during execution resulting in adaptive and modular processes. Moreover, the process designs can be reused by different projects with similar context and be switched during process execution if the given project's context changes. This contributes significantly to addressing pertinent issues, problems and weaknesses inherent in existing SCM models and systems that have adopted these designs with such functionalities. Figure 5 below illustrates the proposed SCM model.



RESEARCH ELABORATIONS

Research Design

This is both a qualitative and quantitative study that was confined to selected sample of small and medium software development firms in Nairobi, Kenya. In addition, expert opinions and ideas from software engineering professionals more so lead developers and developers were purposively selected and utilized.

Population, Sampling Strategy and Techniques

The unit of analysis for the study was any small and medium software development firm. The target population included all small and medium software development firms within the city of Nairobi, Kenya, which develop software for sale as well as in-house development groups within organizations. In this study, the small and medium firms targeted were the firms with employees not exceeding 50 in number. The number of small and medium software development firms in Nairobi is enormous as the influx of new small and medium firms is estimated at 200 - 250 per annum which stood at 1850 firms as at 2013 (Kenya Companies Registry, 2014). The listing of these companies was acquired from the authenticated listing source of Kenyan software development firms at the Government Registrar of Companies Department.

The sample of this study comprised of small and medium software development firms drawn from five distinct strata of the city of Nairobi, namely Nairobi central business district, Eastlands, Westlands, Upper Nairobi and Southlands.

In this study, it is clearly indicated which sort of firms fall under the category of small and medium software development firms. The study's long-term intentions are for the proposed software configuration management model to be internationally acceptable and adopted. The study proposed to use the sample population of small and medium software development firms within the city of Nairobi, Kenya as the yardstick to test the practicability and adoptability of the proposed software configuration management model to the firms. To determine the sample size for the study, Fisher's formula was employed as follows:

$n=Z^2pq/d^2$
Where, n= desired sample size
Z= standard normal deviation, which is set at 1.96 (95% confidence level)
P= proportion of the targeted population that have the characteristic focused in the study, which is estimated at 85% (0.85).
q=1-p
d= degree of accuracy, which is set at 5%. The degree of proportion of error that should be accepted in the study is 0.05, since the study has 95% confidence level.
Therefore, Desired Sample (n) = $\{1.96^2*(0.85*(1-0.85)\}/0.05^2$
n= 196
Since the total population for each region is less than 10,000, the researcher applied the finite correction formulae (nf). This is applied together with the Fisher's formulae in successive steps as indicated:

$$nf = \frac{n}{1 + n/N},$$

N = 1850, n = 196
nf= 196/(1+196/1850) = 177

Crucial aspect of the sampling technique is determining the unit or level of analysis. This study recognized that research work is often couched in social setting and identified ten different levels (units) of analysis, namely society, profession, external business context, organizational context, project, group team, individual, system, computing element (program) and abstract concept. The unit of analysis for this study is the organization, which is, small and medium software development firms. In the sampling of the population, the study used the cluster sampling technique. The rationale for the sample cluster sampling is where the population is divided into units or groups called strata (usually there are units or areas in which the population has been divided in), which should be as representative as possible for the population, representing the heterogeneity of the population being studied and the homogeneity within each of the strata. The sample of this study was selected from the population of small and medium software development firms within Nairobi city. In sampling of the population, the study area was divided into five distinct strata - Nairobi Central Business District, Eastlands, Westlands, Upper Nairobi and Southlands. Each of these strata represented the heterogeneity of the population being studied and the homogeneity within each of the strata as justified by the fact that the firms are located in the same geographical zone.

The preferred sample size selected for this study was 177 small and medium software development firms. From each software development firm, 2 software developers were selected to participate in the study. These were preferably the software lead developers and one of the developers, who was selected through the use of simple random method from the other developers/employees. This made total of 354 respondents for the study as tabulated in Table 1.

TABLE 1: SAMPLE DISTRIBUTION

STRATA	NUMBER OF FIRMS	TOTAL PARTICIPANTS	PERCENTAGE
Nairobi CBD	37	74	20.8
Eastlands	35	70	19.8
Westlands	35	70	19.8
Upper Nairobi	35	70	19.8
Southlands	35	70	19.8
TOTAL	**177**	**354**	**100**

Data Collection Methods and Approaches

In this study, only primary data was collected. The data collected was both qualitative and quantitative in nature. The data collection procedures or methods employed were questionnaires for the software developers and interviews for the lead developers. The questionnaire comprised of four sections each based on the objectives of the study. The questions were both open and closed ended and structured in such a manner that all objectives of the study were captured. The questionnaire tool was used to collect data from the software developers. This was through drop-and-pick method for the sake of the respondents' convenience. Data from the lead developers was collected using the interview method. Questions in the interview were designed to acquire both qualitative and quantitative data. The interview questions consisted of four sections each based on the study objectives. The questions captured various themes and sub-themes based on the study's objectives. The face to face interviews were conducted with the aid of the interview guide that was designed to capture relevant information in line with the objectives of the study.

Data Analysis and Presentation

Data analysis involved the systematic application of statistical and/or logical techniques to turn raw data into information that was used in making decisions. The questionnaires were coded and edited for analysis in Statistical Package for the Social Sciences (SPSS) and the quantitative data analysis was used to give descriptive statistics such as mean and standard deviation that were then presented in form of tables and figures for easy understanding and interpretation. Thematic representations were employed to present the qualitative data obtained from the interviews as well from the questionnaires.

RESULT AND DISCUSSION

Approach Employed to SCM

The first objective of this study was to establish the approach employed by small and medium software firms in relation to SCM. In establishing the SCM approach employed by small and medium software development firms in Nairobi, Kenya, the study findings revealed the information indicating that, majority of the small and medium software development firms (80%), employ one of the existing traditional standard four SCM models. With the numerous challenges identified by this study regarding the existing traditional standard four SCM models, this study robustly questioned the effectiveness of these models. Interestingly, according to the findings from this study, overwhelming majority (100%) of the small and medium software development firms were found to be facing challenges in relation to SCM practice, significant majority of which were explicitly identified as challenges of the existing traditional standard models. The study revealed that majority of the small and medium software development firms do not practice the conventional and standard phases of software configuration management. Only 40% of the respondents indicated that the firms partially employ conventional and standard phases of software configuration management in software development projects. This is strong indication that significant majority of the small and medium software development firms in Nairobi, Kenya apply ineffective approaches to SCM.

Effectiveness of Existing SCM model Employed

The second objective of this study was to evaluate the effectiveness of the existing SCM model employed in small and medium software firms. The findings illustrate clearly that majority of the software developers and software lead developers in the small and medium software development firms in Kenya are not knowledgeable about the SCM process, and have low understanding of the activities and processes involved in the practice.

Majority of the small and medium software development firms employed the practice of SCM in certain circumstances. The firms also practiced SCM only when enough software development time was available to them. Findings strongly reveal that significant majority of the small and medium software development firms in Nairobi, Kenya do not employ any substantial and effective approach to software configuration management during software development activities. The study established the practice and interest of improving SCM indicating that the existing SCM models applied in small and medium software development firms in Kenya are ineffective and hence the much generated interest of 80% of the respondents, in improving the firms' approach after the proposed contextualized model was demonstrated to them.

Challenges Encountered

The third objective of this study was to establish the challenges faced by small and medium software firms in SCM practice. On evaluating the challenges faced by both software developers and software lead developers in the SCM practice, the study found out that, several factors limited the firms' ability to apply the SCM process in software development projects. These include SCM being bureaucratic and hence time consuming to implement; limited skilled manpower to handle SCM; SCM being time-intensive and therefore time consuming; frequently changing demands from clients hindering application of SCM; SCM is cost intensive and therefore uneconomical to practice; SCM is labour-intensive and therefore leads to schedule delays; firms find the process of handling the tracking of change requests and defect reports difficult to manage; firms have challenges when it comes to simultaneous update of changes made by different developers; firms have challenges of logical conflict whereby when changes are committed, component of the program that has not been modified leads to the generation of software errors when the software or program is run, and challenges of smoothly managing the various sub-processes involved when practicing SCM.

Proposed Contextualized SCM Model

The fourth objective of this study was to propose contextualized SCM model that is relevant and beneficial to small and medium software firms in Kenya and other developing countries. The study results indicated that, the proposed contextualized SCM model meets the SCM requirements of significant majority of the small and medium software development firms in Kenya in terms of the approach employed; effectiveness of SCM model; ability to efficiently address the challenges in relation to SCM process; structurally-inherent nature of being adaptive, contextualizable, relevant and beneficial to the firm in question regardless of the context of operation if adopted and process-oriented approach qualifying it to be faster to use, less tedious to apply, less bureaucratic to implement and overally easier to understand compared to the existing traditional standard models.

The study findings indicate that the proposed SCM model is highly applicable in SCM practice. This is strong indication of the high capability of the proposed contextualized SCM model to meet the needs and requirements of small and medium software development firms in Kenya. This also confirms the proposed model's effective approach and ability to address the numerous challenges faced by small and medium software development firms in Kenya. In addition, commercialization and customization of the proposed SCM model is clearly demonstrated.This according to the study participants, qualified the model as SCM tool offering precious solution to the numerous challenges currently faced by such software development firms. Majority of the respondents looked forward to the proposed SCM model being developed into software tool that can be commercialized and customized to the needs of individual firms.

Effectiveness of Proposed Contextualized SCM Model

The fifth objective of this study was to evaluate the effectiveness of the proposed contextualized SCM model in small and medium software development firms. The study results indicate that majority of the respondents had high perception towards the proposed SCM model (47.2%) while 38.6% of the respondents had moderate perception towards the proposed SCM model. This is indication towards the fact that the proposed SCM model was effective in the sense that majority of the respondents had high level of perception (47.2%) and moderate level of perception (38.6%) towards its functionalities and application. This enables this particular proposed SCM model to be effective towards addressing pertinent SCM issues faced by the small and medium software development firms.

The study results as indicated by the mean value in the range of 4.0 – 4.9, indicate that the respondents strongly agreed that the proposed SCM model effectively addresses pertinent issues of SCM such as the clear definition of processes through the process modeling approach used; the challenge of simultaneous update; the challenge of logical conflict; the challenge of tracking of change requests and defect reports; and the general challenges faced by small and medium software development firms in their SCM application. This is explicit indication that the proposed SCM model is effective as evidenced by the fact that it addresses the pertinent challenges in existing SCM models identified previously in this study.

Findings from the study indicate that majority (81.2%) of the respondents are of the view that the proposed SCM model is effective in improving how their firms apply the SCM process. This is significantly positive indication of the effectiveness of the proposed SCM model in addressing the needs of small and medium software development firms.

The study results indicate strong level of agreement by the respondents as shown by mean value range of 4.0 – 4.9. The proposed SCM model has adopted elements of the check-out/check-in model to address the challenge of simultaneous update. According to the results in Table 20, the check-out/check-in model has mean score of 4.8721 which is higher compared to the composition model (3.7240), the long transaction model (4.1104) and the change set model (3.9281). This is indication that by adopting elements of the check-out/check-in model, the proposed SCM model is better placed in addressing the challenge of simultaneous update. The proposed SCM model in addition, as adopted elements of the long transaction model to address the challenge of logical conflict. Based on the results in Table 20 pg.64, the long transaction model has mean score of 4.9321 which is higher as compared to the check-out/check-in model (4.7149), the composition model (4.0381) and the change set model (3.9926). This is an indication that by adopting elements of the long transaction model, the proposed SCM model is better placed in addressing the challenge of logical conflict. The proposed model has adopted elements of the change set model to handle the challenge of tracking of change requests and defect reports. The results in Table 20 pg.64 show that the change set model has mean score of 4.8296 which is higher as compared to the check-out/check-in model (3.9999), the composition model (3.5392) and the long transaction model (4.3018). This is indication that by adopting elements of the change set model to handle the challenge of tracking of change requests and defect reports, the proposed SCM model is better placed in addressing the challenge of tracking of change requests and defect reports.

The study results show that the proposed contextualized SCM model has higher superiority level as indicated by the mean score of 4.9327 as compared to other existing SCM models which score: check-out/check-in model (2.7141), composition model (2.8164), long transaction model (2.7219) and change set model (2.8719). This is explicit indication that the proposed contextualized SCM model is superior in terms of collectively addressing SCM issues in small and medium software development firms.

CONCLUSION

Based on the findings and discussions presented in the preceding sections, this study makes the following conclusion:

Findings reveal that majority of the software developers and lead developers have low level of perception of the SCM practice while only minority of the software developers and lead developers have high perception of the SCM practice.

SCM being key and paramount composition of quality software engineering practice, this study raises pertinent and important questions regarding the quality of software developed by majority of the small and medium software development firms in Kenya. SCM is explicitly neglected by majority of the firms leading to doubts regarding the quality of software produced by these firms.

Small and medium software development firms employ the practice in select and biased circumstances based on the various factors such as when need arises, when there is availability of software engineers well conversant with SCM, and when the client demands changes to already developed software. Majority of the small and medium software development firms in Kenya do not apply SCM practice in software projects raising issues regarding to what extent is the produced software assured of meeting the stipulated global standard quality levels of software.

There are a number of significant challenges existing in the software industry as regards the application of SCM in the software engineering discipline. Small and medium software development firms have experienced the impact of these challenges. Firms of these stature especially in the developing countries are at much more vulnerability level of being adversely and negatively affected by these challenges owing to reasons such as bureaucratic nature of existing SCM standards and models, time consuming to implement nature of existing standards and models and limited skilled manpower to handle SCM.

These challenges hinder the adoption and application of SCM practice among such firms which form significant population of the software development firms in the developing countries.

The proposed contextualized SCM model meets the SCM requirements of significant majority of the small and medium software development firms in terms of the approach employed, effectiveness, efficiency, structurally-inherent adaptive nature, contextualizable nature, relevancy and beneficial nature to the firm in question. This is especially significant for the small and medium software development firms that operate in different policy, regulatory, industry and organizational contexts. The applicability of the models designed for developed countries is not always relevant to small and medium software development firms in developing countries. The proposed SCM model approach includes the context into process descriptions, enabling process owners to design own processes for change and switch processes during execution resulting in adaptive and modular processes.

Studies strongly advocate for the development of research works about new models that are focused towards tailoring and adaptation of software processes improvements such as SCM in SMEs. This is a result of lack of existing standards suitable for SMEs especially in software engineering (Pino et al, 2008, Hareton & Terence, 2001 and Johnson & Brodman, 1999). This study consequently as achieved contextualized SCM model that is relevant and adaptable for use by small and medium software development firms. The proposed model addresses a number of key challenges identified in this study that are faced by such firms in their application of the SCM process.

By adopting the proposed contextualized SCM model, the concerned firm amicably addresses the challenges of simultaneous update, logical conflict, tracking of change requests and defect reports and clear definition of SCM processes. This is major contribution that the proposed SCM model has made with a view and intention of addressing pertinent and key research gaps and challenges identified in the existing four standard SCM models. An implication of this proposed contextualized SCM model is the more conventional, standard, accountable, relevant and auditable manner of applying the SCM process more so in small and medium software development firms. This shall improve significantly the quality of produced software by such firms in addition to reducing the complexity associated with the existing SCM standards.

There is need to commercialize and customize the proposed SCM model to the specific needs of each of the software development firms in order for maximum benefits to be derived in relation to SCM practice and general software engineering activities. This qualifies the model as SCM tool offering precious solution to the numerous challenges currently faced by small and medium software development firms.

The proposed contextualized SCM model is highly understood in its functionalities and application in the SCM process. The proposed model addresses the challenges faced by small and medium software development firms effectively. Use of this model contributes positively to the improvement of the SCM process in concerned firms. By adopting elements of the check-out/check-in model, long transaction model and change set model, the proposed SCM model effectively addresses pertinent SCM issues faced by small and medium software development firms in their application of the SCM process. Compared to the existing standard four SCM models, the proposed contextualized SCM model is superior in its execution and operation of the SCM process.

RECOMMENDATIONS

Based on the study findings, recommendations are made to address the various challenges faced by small and medium software development firms, to ensure that amicable and practical solutions that capture the aspirations of developers and lead developers in practice of SCM are provided.

Approach Employed to SCM

This study recommends the adoption of the proposed contextualized SCM model to meet the required SCM standards of practicing SCM in addition to meeting the particular firm’s context needs. Findings reveal that, majority of the small and medium software development firms are using ineffective approaches to practicing SCM and this often leads to numerous challenges to the firms in managing the practice. In addition, due to adoption of ineffective SCM approaches, majority of the firms do not practice SCM altogether as majority perceive it to be tedious, expensive and time-consuming to such firms’ operations. According to Pino et al (2008), existing standards are not suitable for small and medium software organizations. In Staples et al (2012), the reasons why such standards are not adopted is explored. This reinforces the need to have available strategies for process improvement which are tailored to small companies' characteristics. These strategies must be aligned with the widely recognized standards (for large firms), in order to enable small companies establish a solid base for process improvement (Pino et al, 2008). By adopting this proposed contextualized SCM model, the firms shall positively take up the practice and this shall lead to better quality software being produced.

Effectiveness of Existing SCM Model

This study recommends the development of this proposed model into software tool so that interested firms can adopt and use the model in software development activities. This ensures that the practice of the SCM process is more effective, since the tool structure is designed from highly recommended model. This shall facilitate guidance of the implementation of SCM as described in the proposed contextualized model.

Challenges Encountered

To address the challenges incurred by majority of small and medium software development firms as evidenced by this study's findings, this study recommends that the concerned firms adopt this model in all software development life cycles. The proposed contextualized SCM model effectively addresses the challenges revealed in this study. By adopting this model, small and medium software development firms shall overcome most of the challenges faced in SCM practice and the general software engineering process.

Proposed Contextualized SCM Model

This study recommends the adoption of the proposed contextualized SCM model to ensure that the concerned firms follow the practice in conventional, standard, accountable, relevant and auditable manner. The proposed SCM model ensures that firms are able to practice the process in a manner that is relevant to such firms' environment of operation and in doing so, end up reaping maximum benefits from the process. Studies such as Hareton and Terence (2001), Johnson and Brodman (1999) and Saiedian & Carr (1997) show that the models from the Software Engineering Institute or International Standards Organization are difficult for small and medium organizations to apply. This because of the complexity of their recommendations and the consequential large investment in terms of time and resources. It is thus important to consider these models as the reference to develop research works about new models related with the tailoring and adaptation of software processes improvement in SMEs. Below is diagrammatic representation of the proposed contextualized software configuration management model for small and medium software development firms in Kenya based on the four main structural elements of process sequence, process abstraction, context definition and solid process.

Effectiveness of Proposed Contextualized SCM Model

This study recommends the adoption of this proposed contextualized SCM model in small and medium software development firms owing to its established effectiveness among studied firms. The proposed model has been found out to be easily understood by developers; effective in addressing pertinent SCM issues and challenges faced by small and medium software development firms; effective in improving how firms apply the SCM process; effectiveness in adopting elements of the existing standard SCM models to address pertinent SCM challenges and high superiority as compared to the existing standard SCM models in addressing pertinent SCM issues in small and medium software development firms.

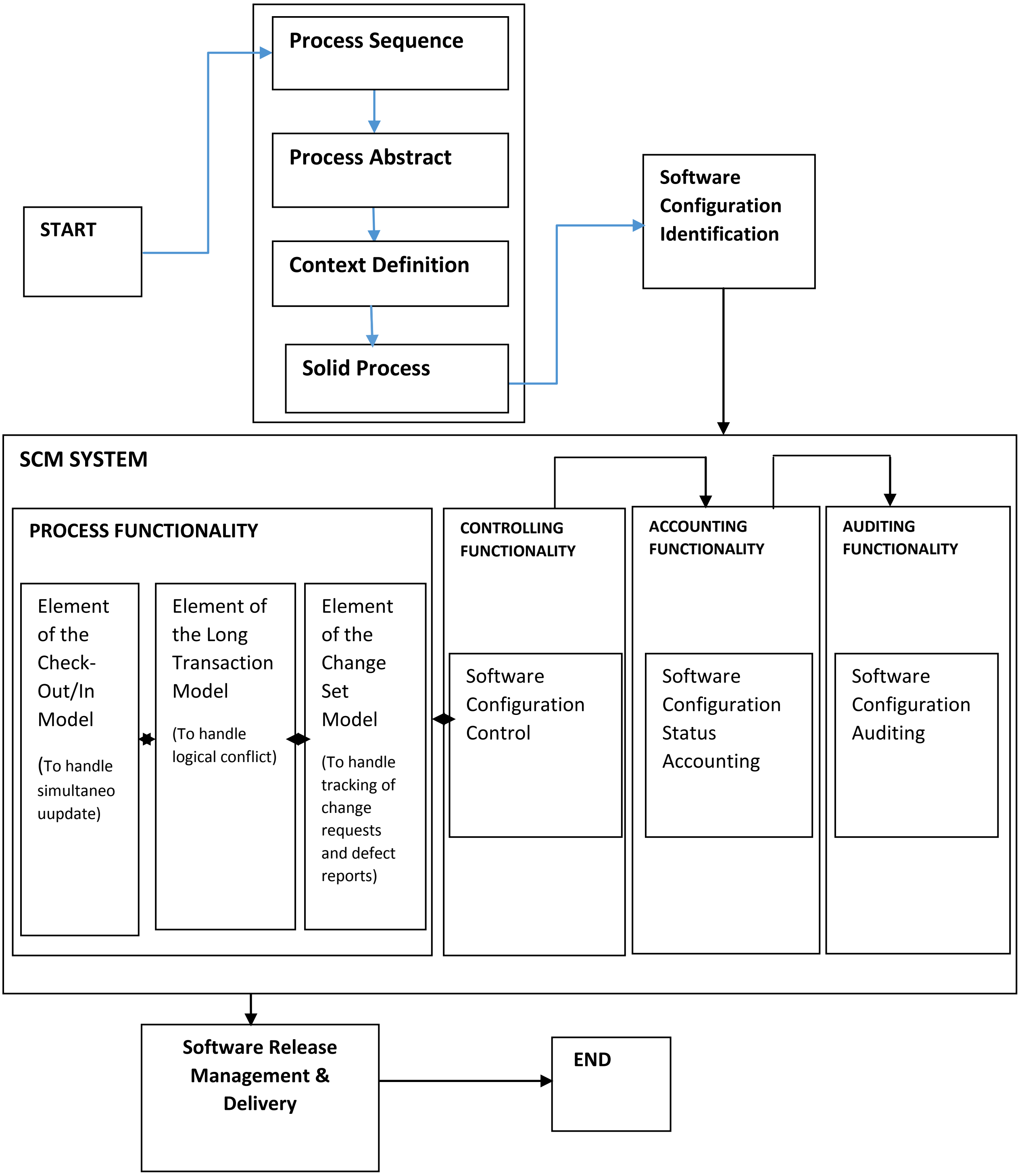


Figure 8: Proposed Contextualized SCM Model for Small and Medium Software Development Firms

REFERENCES

[1]. Aggarwal, H. (2012). Identification of Effective Key Processes in Software Process Improvement Models for SMEs. *International Journal of Research in Engineering & Applied Sciences*, 2(2):

[2]. Aiello, R., and Sachs, L. (2010). *Configuration Management Best Practices: Practical Methods that work in the Real World, 1st edition*. Addisson-Wesley Professional.

[3]. Alzaga, A., and Martin, J. (2010). *A Design Process Model to Support Concurrent Project Development in Networks of SMEs*. Foundation TEKNIKER, Eibar, Spain.

[4]. Anquetil, N., Kulesza, U., Mitschke, R., Moreira, A., Royer, J., Rummler, A., and Sousa, A. (2010). *A model-driven traceability framework for software product lines*. New York: Springer-Verlag.

[5]. Baddoo, N., and Hall, T. (2010). De-Motivators for Software Process Improvement: An Analysis of Practitioners' views. *Journal of Systems and Software*, 66(1), 23-33.

[6]. Balamuralidhar, P., and Prasad, R. (2011). Self-configuration and Optimization for cognitive networked devices. *Wireless Personal Communications: An International Journal*, 59(3), Kluwer Academic Publishers.

[7]. Berzisa, S.,and Grabis, J. (2011). Combining project requirements and knowledge in configuration of project management information systems. *Profes '11: Proceedings of the 12th International Conference on product focused software development and process improvement*. ACM.

[8]. Biffl, S., and Schatten, A. (2009). A platform for service-oriented integration of software engineering environments. *Proceedings of the 2009 Conference on New Trends in Software Methodologies, Tools and Techniques*. IOS Press.

[9]. Boas, G.V., Cavalcanti, A.R., and Amaral, M.P. (2010). An Approach to implement Software Process Improvement in Small and Mid-Sized Organizations. *Proceedings of the 2010 Seventh International Conference on the Quality of Information and Communications Technology*, IEEE Computer Society.

[10]. Boden, A., Muller, C., and Nett, B. (2011). Conducting a Business Ethnography in Global Software Development Projects of Small German Enterprises. *Information and Software Technology*, 53(9), Butterworth-Heinemann.

[11]. Bose, I., Pal, R., and Ye, A. (2008). ERP and SCM systems integration: The case of a valve manufacturer in China. *Information and Management*, 45(4). Elsevier Science Publishers.

[12]. B.V.*Capability Maturity Model Integration (CMMI) Overview*. Carnegie Mellon University: Software Engineering Institute.

[13]. Chen, C.Y., and Chen, P.C. (2009). A Holistic approach to managing software change impact. *Journal of Systems and Software*, 82(12). Elsevier Science Inc.

[14]. Chen, N., Hoi, S., & Xiao, X. (2011). Software process evaluation: A machine learning approach. *ASE '11: Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering.* IEEE Computer Society.

[15]. Christensen, H.B. (1999). The Ragnarok Architectural Software Configuration Management Model. *Proceedings of the 32nd Hawaii International Conference on System Sciences.*

[16]. Clarke, P., O'Connor, R. (2011). An Approach to evaluating Software Process Adaptation. In: *Proceedings of the 11th International Conference on Software Process Improvement and Capability Determination,* pp.28-41. Springer-Verlag, Hiedelberg/Berlin, Germany.

[17]. Clarke, P., O'Connor, R., Yilmaz, M. (2010). *A hierarchy of SPI activities for software SMEs: results from ISO/IEC 12207-based SPI assessments.* Dublin City University, Ireland.

[18]. Cravino, P., Lawrence, D., Lopez, A., Onorato, B., and Shen, Z. (2009).*Enterprise Software Configuration Management Solutions for Distributed and System Z.* International Business Machines (IBM).

[19]. Crowston, K., Wei, K., Howison, J., and Wiggins, A. (2012). Free/Libre open-source software development: What we know and what we do not know. *Computing Surveys (CSUR)*, 4(2). ACM.

[20]. Dabbish, L.A., Wagstrom, P.,Sarma,A., and Herbsleb, J.D. (2010). Coordination in innovative design and engineering: observations from a lunar robotics project. *Group '10: Proceedings of the16th ACM international conference on supporting group work.* ACM.

[21]. Dix, A., & Gongora, L. (2011). Externalisation and Design. *DESIRE '11: Proceedings ofthe second conference on creativity and innovation in design.* ACM.

[22]. Duhan, S., Levy, M., and Powell, P. (2012). *Is Strategy in SMEs using Organizational Capabilities: The CPX Framework.* United Kingdom.

[23]. Elmroth, E., Hernandez, F., and Tordsson, J. (2010). Three fundamental dimensions of scientific workflow interoperability: model of computation, language, and execution environment. *Future Generation Computer Systems*, 26(2). Elsevier Science Publishers B.V.

[24]. Er, N.P., and Erbas, C. (2010). Aligning software configuration management with governance structures. *SDC '10: Proceedings of the 2010 ICSE Workshop on Software Development Governance*, ACM.

[25]. European Telecommunications Standards Institute. (2011). *Small and Medium-sized Enterprises (SMEs) in Standardization; Understanding and Supporting SME involvement in ICT standardization.* Sophia AntipolisCedex- France.

[26]. Feiler, P.H. (2010). *Configuration Management models in commercial Environments.*Tech Rept. CMU/SEI-91-TR-7, ADA 235782, Software Engineering Institute, Carnegie Mellon University.

[27]. Forte, G. (1989). In search of the Integrated Environment. *CASE Outlook*. Retrieved on July19, 2012, from journals.tubitak.gov.tr.

[28]. Fruhauf, K., and Zeller, A. (1999).*Software Configuration Management: State of the Art, State of the Practice.*

[29]. Ghobakhloo, M., Sabouri, M.S., Hong, T.S., and Zulkifli, N. (2011). Information Technology Adoption in Small and Medium-Sized Enterprises; An Appraisal of Two Decades Literature. *Interdisciplinary Journal of Research in Business,* 1(7), Pp.53-80.

[30]. Habra, N., Niyitugabira, E., Lamblin, A., and Renault, A. (2011). *Software Process Improvement in Small Organizations using Gradual Evaluation Schema*. University of Namur.

[31]. Hadden, R. (1998). "Key Practices to the CMM: Inappropriate for Small Projects Panel". In: *Proceedings of the Software Engineering Process Group Conference,* Chicago.

[32]. Heer, T., Heller, M., Westfechtel, B., and Worzberger, R. (2010). *Tool Support for dynamic development processes*. Springer-Verlag.

[33]. Hong, M., Zhang, L.,and Fuqing, Y. (2002). A Component-based software configuration management model and its supporting system. *Journal of Computer Science and Technology,* 17(4). Institute of Computing Technology.

[34]. Hudson, M., Smart, A., and Bourne, M. (2001). Theory and Practice in SME performance measurement systems. *International Journal of Operations & Product Management,* 21(8), pp. 1096-1115.

[35]. Humble, J., and Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, 1st edition*. Addison-Wesley Professional.

[36]. Jimenez, M., Vizcaino, A., and Piattini, M. (2010). Improving Distributed Software Development in Small and Medium Enterprises. *The Open Software Engineering Journal,* 4, pp.26-37.

[37]. Jones, R., Thomas, P., and Thomas, K. (2010). Quality Management Tools and Techniques: Profiling SME use & Customer Expectations. *The International Journal for Quality and Standards.*

[38]. Katchow, R., Weerd, I., Brinkkemper, S., and Rooswinkel, A. (2011). *Software Product Manager: A Mechanism to manage Software Products in Small and Medium ISVs*. Utrecht University, The Netherlands.

[39]. Kaur, P., and Singh, H. (2011). A model for versioning control mechanism in component-Based systems. *SIGSOFT Software Engineering Notes*, 36(5). ACM.

[40]. Kim, D., and Youn, C. (2010). Traceability Enhancement Technique through the integration of software configuration management and individual working environment. *SSIRI '10: Proceedings of the 2010 Fourth International Conference on secure software integration and reliability improvement.* IEEE Computer Society.

[41]. Klosterboer, L. (2010). *Implementing ITIL Configuration Management, 2nd edition*. IBM Press.

[42]. Kogel, M. (2008). Towards Software Configuration Management for Unified Models. CVSM' 08: *Proceedings of the 2008 international workshop on comparison and versioning of software models*, ACM.

[43]. Lin, Y.J., and Reiss, S.P. (1995). Configuration Management in terms of modules. *Proceedings of the 5th International Workshop on Software Configuration Management*, pp.17-26.

[44]. Loumos, V., Christonakis, G., Mpardis, G., and Tziova, P. (2010). Change Management and Quality of Service through Business Process Modelling: The N-VIS, a Public Sector Project. *ITNG '10: Proceedings of the 2010 Seventh International Conference on Information Technology: New Generation*. IEEE Computer Society.

[45]. Mader, P., and Gotel, O. (2012). Controversy Corner: Towards automated traceability maintenance. *Journal of Systems and Software*, 85(10). Elsevier Science Inc.

[46]. Mei, H., Zhang, L., and Yang, F. (2002). *A Software Configuration Management Model for Supporting Component-Based Software Development*. Software Engineering Notes, 26(2), 53.

[47]. Micallef, J., and Clemm, G.M. (1996). The Asgard System: Activity-Based Configuration Management. In *SCM-6 Workshop, March 1996* (pp.175-187). Berlin, Germany: Springer Verlag LNCS1167.

[48]. Mohan, K., Xu, P., Cao, L., and Ramesh, B. (2008). Improving change management in software development: Integrating traceability and software configuration management. *Decision Support systems*, 45(4).

[49]. Moser, T., Mordinyi, R., and Biffl, S. (2010). An ontology-based methodology for supporting knowledge-intensive multi-discipline engineering processes. *ODiSE '10: Ontology-Driven Software Engineering*. ACM.

[50]. Murta, L., Dantas, C., Oliveira, H., Lopes, L., and Werner, C. (2007). An Integrated Software Configuration Management Infrastructure for UML models. *Elsevier, Science of Computer Programming*, 65(3).

[51]. Nalbant, S. (2004). An Information System for Streamlining Software Development process. *Turk J ElecEngin*, 12(2). Retrieved on July 19, 2012, from journals.tubitak.gov.tr.

[52]. Ochuodho, S.J., and Brown, A.W. (1991). A Process-oriented version and configuration management model for communications software. SCM '91: *Proceedings of the 3rdinternational workshop on software configuration management*. ACM.

[53]. Ozcelik,Y. (2010). Do business process re-engineering projects payoff? Evidence from the United States. *International Journal of Project Management*, 28, pp.7-13.

[54]. Pino, F.J., Garcia, F., and Piattni, M. (2009). Key Processes to start software process improvement in small companies. SAC' 09: *Proceedings of the 2009 ACM Symposium on Applied Computing, ACM.*

[55]. Priedhorsky,R. and Terveen,L.(2011). Wiki grows up: arbitrary data models, access control, and beyond. *WikiSym '11: Proceedings of the 7th International Symposium on Wikis and Open Collaboration.* ACM

[56]. Rivas, L., Perez, M., Mendoza, L., and Griman, A. (2010). Tools Selection Criteria in Software-developing Small and Medium Enterprises. *JCS&T*, 10(1).

[57]. Rodriguez, C., Sanchez, M., and Villalobos, J. (2011). Executable model composition: a multilevel approach. *SAC '11: Proceedings of the 2011 ACM Symposium on Applied Computing*. ACM.

[58]. Rosenblum, D.S., and Krishnamurthy, B. (1991). An event-based model of software configuration management. SCM '91: *Proceedings of the 3rdinternationalworkshop on software configuration management*. ACM.

[59]. Rubin, J., Chechik, M., and Easterbrook, S.M. (2008). Declarative approach for Model composition. *MiSE '08: Proceedings of the 2008 international workshop on models in software engineering*. ACM.

[60]. Ruparelia, N.B. (2010). The history of version control. *SIGSOFT Software Engineering Notes*, 35(1). ACM.

[61]. Sarma, A., Bortis, G., and Hoek, A. (2007). *Towards Supporting Awareness of Indirect Conflicts Across Software Configuration Management Workspaces.* University of California. USA: Irvine.

[62]. Sarma, A., and Hoek, A.V. (2008). *Palantir: enhancing configuration management systems with workspace awareness to detect and resolve emerging conflicts*. Long Beach: California State University

[63]. Schimdt, C. (2012). SMEs: Using CSR to Achieve Sustainability. ECOLOGIA.

[64]. Sharon, D., & Bell, R. (2000). Tools that Bind: Creating Integrated Environments. *IEEE Software.*

[65]. Shamsaie, A., and Habibi, J. (2011). Planning updates in multi-application wireless sensor Networks. *ISCC '11: Proceedings of the 2011 IEEE Symposium on Computers and Communications*. IEEE Computer Society.

[66]. Shihab, E., Bird, C., and Zimmermann, T. (2012). *The Effect of Branching Strategies on Software Quality*. Software Analysis and Intelligence Lab (SAIL). Queens University, Canada.

[67]. Sovran, Y., Power, R., Aguilera, M.K., and Li, J. (2011). Transactional storage for Geo-replicated systems. *SOSP '11: Proceedings of the Twenty-Third ACM Symposium on Operating Systems principles*. ACM.

[68]. Treude, C., and Storey, M.A. (2009). *How tagging helps bridge the gap between social and Technical aspects in software development*. Canada: University of Victoria.

[69]. Wang, Y., Yang, J., Zhao, W., and Su, J. (2012). Change impact analysis in service-based Business processes. *Service Oriented Computing and Applications*, 6(2).NewYork: Springer-Verlag.

[70]. Weinreich, R., and Buchgeher, G. (2012). Towards supporting the software architecture life cycle. *Journal of Systems and Software*, 85(3). Elsevier Science Inc.

[71]. Whitgift, D. (2001). *Methods and Tools for Software Configuration Management*. John Wiley and Sons, UK: Chichester.

[72]. Yahaya, J., Fithri, S., and Deraman, A. (2012). An Enhanced Workflow Reengineering Methodology for SMEs. *International Journal of Digital Information and Wireless Communications*, 2(1).

[73]. Zhu, Y., Tang, F., You, I., Lou, L., Guo, M., and Shen, Y. (2011). PPMLT: A Pipeline Based Processing Model of Long Transactions. *AINA '11: Proceedings of the 2011 IEEE International Conference on Advanced Information Networki*ng and Applications. IEEE Computer Society.