Basic Activities of Software configuration Management

Agha Salman Haider
Sr. Lecturer, Jazan University, Saudi Arabia

Abstract:
SCM is a critical element of software engineering (Feiler 1990). According to Leon (2000), it is needed because of the increased complexity of software systems, increased demand for software and the changing nature of software. Leon also states that SCM can be used as a strategic weapon that will give the organization an edge over those who are not using SCM or using it less effectively. When used effectively during a product's whole life-cycle, SCM identifies software items to be developed, avoids chaos when changes to software occur, provides needed information about the state of development, and assists the audit of both the software and the SCM processes. Therefore, its purposes are to support software development and to achieve better software quality.

Keywords: SCM, Activities, SCM Plan

Introduction:
According to the IEEE's (IEEE Std. 828-1990) traditional definition of SCM, the following four activities are included: configuration identification, configuration control, configuration status accounting and configuration audits. Successful SCM implementation also requires careful planning (Abran & Moore 2001). SCM planning produces a document called SCM plan, in which SCM activities and other practices of SCM are described carefully (IEEE Std. 828-1990).

Basic Activities of SCM

Configuration Management Planning

According to Leon (2000), configuration identification is a process where a system is divided into uniquely identifiable components for the purpose of software configuration management.
These components are called computer software configuration items (CSCIs) or shorter and more generally just configuration items (CIs). A CI can be a unit or a collection of lower level items (Rahikkala 2000). IEEE (IEEE Std. 610.12-1990) defines configuration identification as an element of SCM, consisting of selecting the CIs and recording their functional and physical characteristics in technical documentation. Each CI must be named and versioned uniquely to distinguish it from the other CIs and from other versions of CIs (Whitgift 1991). Examples of CIs are project plan, specifications, design documents, source codes, test plans and test data, executables, make files, tools, and SCM plan. Whitgift (1991) also states that every source item should have a status attribute which defines the level of approval that the item has achieved. An example of the range of status values for an element code is: untested, module tested and integration tested. Accordingly, a document can have such values as draft, proposed and approved. In the configuration identification phase, a project's baselines and their contents are also identified. A baseline is a software configuration management concept that helps us to control change (Leon 2000). It is a document or product that has been formally reviewed and that thereafter serves as a basis for further development. It can also be an assembly of CIs, an accepted configuration (Taramaa 1998) Perforce.

Configuration control

As stated earlier, software can change very quickly and easily, and uncontrolled changes can lead to chaos. Therefore, after the configuration items of the system have been identified, the next step is to control the changes to the software.

Controlling changes during software development has been defined as a task for SCM (Pressman 1997). According to Leon (2000), baselines have a very important role in managing change. According to (IEEE Std. 610.12-1990), baselines can be changed only through formal change control procedures including the following steps: evaluation, coordination, approval or disapproval and implementation of changes to configuration items.

A change request can result from many things. For example, new features, enhancements of existing features or defects can lead to change requests (Leon 2000.) Figure 4 above presents a
traditional change control process. The process starts, when a need for change is noticed. A properly completed change request form is sent to the configuration control board (CCB), whose main function is to evaluate and to approve or disapprove change requests (Leon 2000). According to Leon (2000), change requests can also be deferred when they and their associated documentation are filed for later resolution. If a change request is approved, proposed changes are assigned to developers to be implemented. After implementation, the changes are verified in various testing procedures to ensure that they have been implemented as agreed. Change is one of the most fundamental characteristics in any software development process (Leon 2000). Lehman (1980) also suggests that change is intrinsic in software and must be accepted as a fact of life. Making changes to software is easy, but if it is done at will, chaos will result (Leon 2000).

According to McConnell (1998), effective projects control changes, whereas ineffective projects allow changes to control them. However, Whitgift (1991) states that the level and formality of control required varies considerably; large teams need strict and formal change control, but small teams can rely on much less formal control.

**Configuration status accounting**

Software development produces lots of information that should be recordable and reportable whenever needed. According to IEEE (IEEE Std. 610.12-1990), configuration status accounting consists of the recording and reporting of information needed to manage a configuration effectively, including a listing of the approved configuration identification, the status of proposed changes to the configuration and the implementation status of approved changes. All this and other information related to CIs and activities concerned with them are thus available for the people involved in the project. Status accounting reports include change logs, progress reports, CI status reports and transaction logs (Leon 2000).
Configuration audits

According to Leon (2000), the purpose of configuration audits is to ensure that the software product has been built according to specified requirements (Functional Configuration Audit, FCA), to determine whether all the items identified as a part of CI are present in the product baseline (Physical Configuration Audit, PCA), and whether defined SCM activities are being properly applied and controlled (SCM system audit or in-process audit). A representative from management, the QA department, or the customer usually performs such audits. The auditor should have competent knowledge both of SCM activities and of the project (Leon 2000).

SCM planning

According to Abran & Moore (2001), a successful SCM implementation requires careful planning and management. All of the SCM activities introduced above are described in the SCM plan. The main purpose of the SCM plan is to answer such questions as: who is going to do what, when, where, and how (Buckley 1996). Thus, the SCM plan serves as a guideline for the people working with software configuration management. According to Buckley (1996), a configuration management plan is written for each project. However, an organization may use a generic SCM plan template that can be tailored to each particular project.

Conclusion:

The purpose of this perspective was to find out what the agile methods define as SCM planning. The results show that agile methods literature does not mention SCM planning at all. In software configuration management literature, SCM planning has been described as a basis for successful SCM implementation. Therefore, agile methods should also place emphasis on SCM planning.
References:


- Pilato, Fitzpatrick and Sussman (2009), Subversion 1.6 Official Guide, Fultus Corporation, p 27.


- Taylor G R (2005), Integrating Quantitative and Qualitative methods in research, University Press of America, USA.


- Kirsch G (1992), Methods and Methodology in composition research, Illinois University, USA.