

# Towards An SCM Reference Model

by

Ghulam H. Hasnain

## CONTACT INFORMATION

AUTHOR NAME

**GHULAM H. HASNAIN**

AUTHOR'S BUSINESS ORGANIZATION:

**SOFTWARE QUALITY ASSURANCE CONSLUTANTS INC.**

BUSINESS ADDRESS

**9988 APLOMADO DRIVE, SANDY, UT 84092**

VOICE NUMBERS

**Mobile: (801) 671-6709**

E-MAIL ADDRESSES:

**[sqaci@mailcity.com](mailto:sqaci@mailcity.com)**

INTERNET ADDRESS

**[www.sqaci.com](http://www.sqaci.com)**

# **Contents**

**1. Introduction**

**2. Software Configuration Management Definition**

**3. SCM in the Context of Software System Life Cycle Framework**

**4. SCM Procedural Parameters for Software Quality**

**5. SCM Approach and Implementation Strategy – Essential Steps**

**6. Architectural Considerations for Customizing an SCM Approach**

**7. Customizing the Criteria for SCM Tool Selection**

**8. Conclusions**

**9. General References**

**10. Biographical Sketch of the Author**

# 1. Introduction

In this article I will work towards developing a Software Configuration Management (SCM) Reference Model. This model is intended to be a concise and practical guide which brings together in an outline fashion the framework and methodology necessary for planning, customizing, and implementing an SCM approach, independent of specific SCM tool technologies or system platforms.

Current best practices for developing application systems require, adoption not just of a systematic approach inherent in a Development Life Cycle and SCM methodology but also that the methodological approach that is selected has just the right amount of formalism justified in terms of the cost/benefit of the formalism. Furthermore, this approach needs to be appropriately customized to be able to deliver an application system of desired quality and within the given cost and schedule constraints.

To present the SCM Reference Model, I will cover the essential elements of the SCM methodology - the definition of SCM including its contextual framework, SCM procedural parameters that govern software quality, the essential steps for planning an SCM approach and implementation strategy, and the architectural considerations for customizing an SCM approach. I will concentrate on the 'big picture' without going into the details, descriptions, and the transitions, which where necessary, are implied. The key element of this reference model is to facilitate the derivation of a customized SCM approach for the development of the application system at hand.

To begin with, major commonly recognized application development life cycle methodologies and system platforms are identified and the customization guidelines presented corresponding to the life cycle methodology adopted and the platform of the subject application system for which the SCM system is being devised.

To be readily understood and validated, I believe, information is best presented graphically and models particularly so. Therefore, every attempt will be made to present this model in as graphic a manner as possible.

## 2. Software Configuration Management Definition

### SCM Context and Domain

Software configuration management is one of the key qualitative disciplines governing the life cycle of software systems; the other disciplines being quality assurance, project management, and life cycle methodology

Specifically, SCM manages the comprehensive specification of each features of the application system its transformation into software, defect management through the testing cycles, and then migration and release into production and thereafter enhancement change control of the system. SCM is both, influenced by the Capability Maturity Model® (CMM®) as developed by Software Engineering Institute and is an enabler for an information system to comply with Level Two of CMM®.

## SCM Definition

The definition of SCM can be provided alternatively with two scopes – the narrow or the broad scope.

The narrow scope SCM definition covers partial life cycle of software systems. It comes into play at the point of development phase. It pertains to the identification and inventory of code in the form of functional program/module units. This identification and inventory enables the code to be migrated and released into production in a controlled and orderly manner such that the defects are identified and resolved systematically through testing, and the software is migrated as a result of testing certification reviews, migration ready audits, and milestone status reporting. This process is intended to ensure that no code item is missed/lost, misidentified, or overlaid thus avoiding the minimal risk of a few minor errors slipping through or maximum risk of system failure with the corresponding financial, time, and/or quality loss.

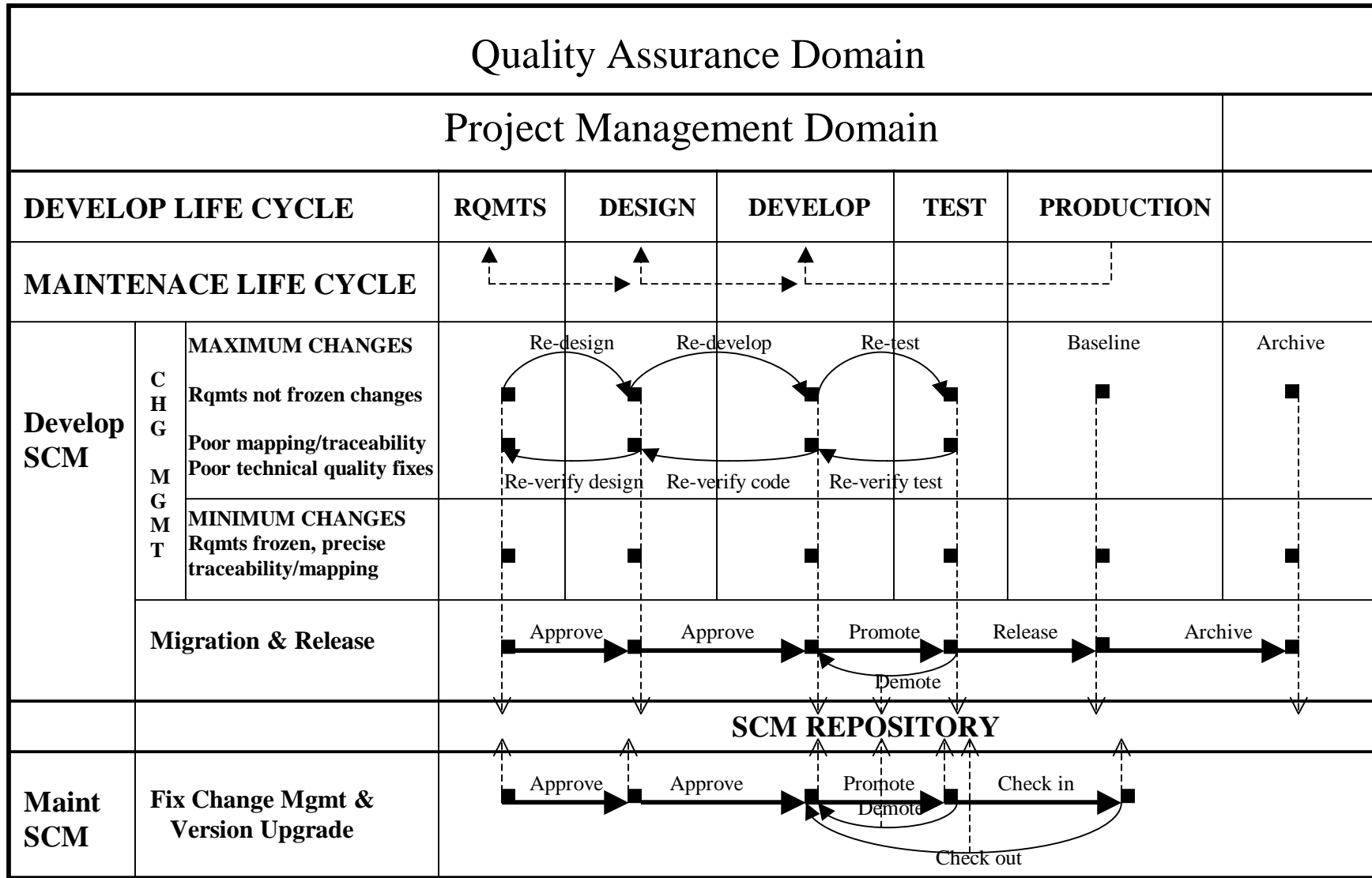
The broad scope – full life cycle – definition of SCM has as its domain the entire software life cycle. It starts with the identification and inventory of software requirements and the tracing of the transformation of those requirements into design, development, test, deployment, and maintenance (instead of starting at the point of code development as for the narrow scope definition) and ensures with testing and migration reviews, audits, and status reporting that the system developed qualitatively and quantitatively delivers the specified functionality and nothing less within the given cost and schedule.

The diagram that follows this section entitled “SCM in the Context of Software System Life Cycle Framework” illustrates the entire domain and flow of the SCM process in a simplified manner. It illustrates both these definitions of SCM. Refer to the Migration and Release row in the diagram. Broad scope SCM definition starts at the point of software requirements where requirement items are uniquely identified and exhaustively inventoried. Next, in the life cycle their transformation into design items takes place and SCM traces and controls exhaustively the transformation of each unique requirement item onto one or more unique design items and then its transformation sequentially into one or more developed software code items. The code item goes through the migration process (promotion and if necessary demotion and then re-promotion) through a number of test phases (unit, integration/system, acceptance etc.) into production. The narrow scope SCM definition starts at the Develop Phase instead of the Requirements Phase – covering partial life cycle at the point when software code has been identified, developed, and inventoried.

This diagram highlights the fact that as part of software change management not freezing the requirements prior to development may involve redesign, recode, and retest. It also highlights the fact that if the fundamental requirements item identification, transformation, and tracing is not performed with certainty then the resulting software system is of indeterminate quality, may involve re-rewrite of requirements, redesign, recode, and their tracing re-verified.

Finally, notice that the information pertaining to the item identification, inventory, and traced migration, change, and release is captured in the SCM Repository identified in this diagram.

### 3. SCM in the Context of Software System Life Cycle Framework

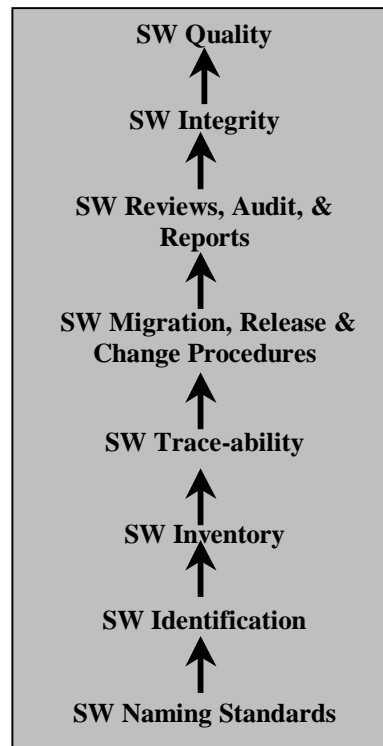


## 4. SCM Procedural Parameters for Software Quality

SCM establishes software quality i.e., the proper use of SCM results in error free or reduced error software and software which approximates contracted requirements. Quality software with few or no errors which meets all the contracted requirements and their functionality is accomplished procedurally by SCM to begin with by the use of naming standards to uniquely (accurately) identify and inventory each and every requirement, design, and code configuration item constituting the system. The unique and exhaustive identification of every requirement and design configuration item (hereon referred to as item) enables SCM to trace the transformation of each requirement item into one or more unique design items and the transformation of each design item into one or more code items.

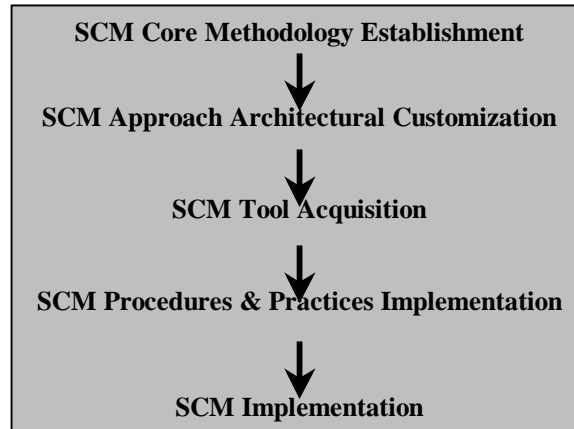
As alluded to earlier the quality assurance mechanisms to enforce adherence to the SCM procedures such as accurate naming of configuration items, their exhaustive inventory, and tracing of their transformation from the antecedent (for example requirement) item to the consequent (design) item during development and then the proper checkout and check-in of code items during maintenance are item migration and system release reviews, audit of inventory, migration approval processes and procedures, and phase item, subsystem and system status reporting imposed on the migration, release, and change management processes. This article will not go into further details of these reviews, audits, and status reporting.

The progressive and hierarchical relationship of the SCM parameters which results in quality software discussed above, is show in the diagram given below:



## 5. SCM Approach and Implementation Strategy - Essential Steps

Consideration must be given to the following successive essential steps shown top down (not an exhaustive list), involved in planning an approach and implementation strategy for a customized SCM system:



The basis for the first step - SCM Core Methodology Establishment - is the standard SCM methodology practiced within the industry, reflected in its literature, and also depicted in the preceding sections of this article.

Once the Core SCM Methodology is identified and established, it must be customized with respect to two critical architectural considerations - the given Application System Platform and the type of Development Life Cycle Methodology adopted for the proposed application system. The central role that the SCM reference model proposed in this article plays is in this customization of the SCM Methodology for the proposed application system. The next section in this article will present the method for customizing the Core SCM Methodology for the proposed application system based on the commonly recognized classification of Application System Platforms and the Development Life Cycle Methodologies.

Keeping in mind cost considerations, an SCM tool which maximally automates the customized SCM approach, is selected next. It is critical that the criteria for selecting the SCM tool also be customized based on the Approach to Managing Application System Code Development adopted and the type of Application System Maintenance and Release Cycle inherent in the proposed application system as described in the section 7.

Finally, SCM Procedures and Practices are developed based on the customized SCM methodology approach adopted and the SCM tool selected for the development of the proposed application system. SCM Procedures and Practices and SCM Implementation are not discussed further in this article.

## 6. Architectural Considerations for Customizing an SCM Approach

Major architectural considerations for customizing an SCM approach for an application system depend on the parameters applicable to the type of platform technology of the application system being developed, and the type of development life cycle methodology and the corresponding SCM approach adopted for it.

First, let us consider the system platform dictated by the application system and its impact on the approach to SCM for that application system. I will keep this discussion of the system platform with respect to SCM simple by only considering the major categories of platforms and by not getting into their variations and details.

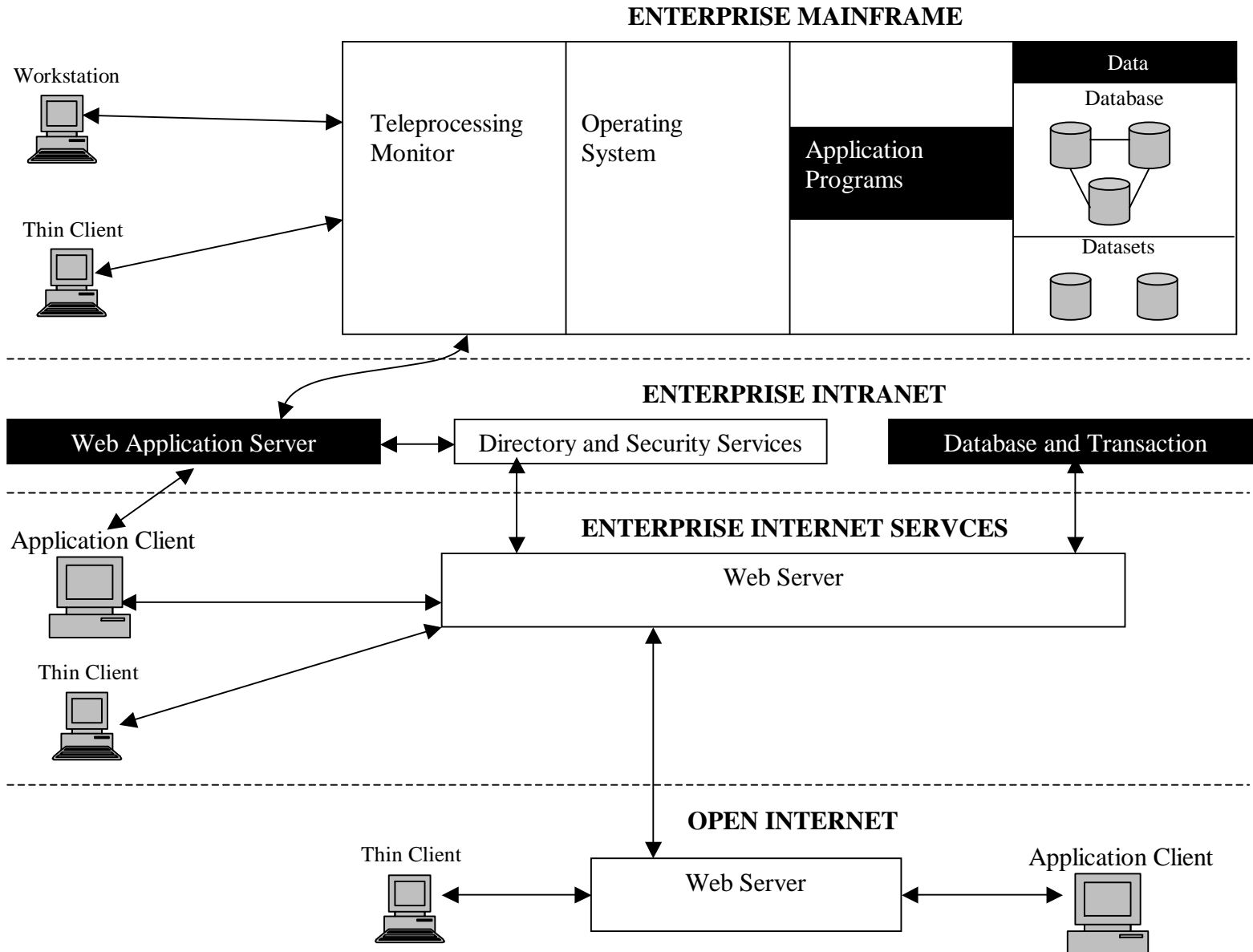
The three categories of system platform that I will concern myself with are:

- **Homogeneous Mainframe Platform in a Local Configuration**
- **Homogeneous Client/Server Platform in a Local Configuration**
- **Hybrid Heterogeneous Platform as a result of combining the Client/Server with the Mainframe Platform (typically in an Internet configuration).**

I will regard the Distributed Platform as a variation of the hybrid heterogeneous category.

The diagram System Platform Framework given below illustrates instances of each of these platform category configurations and its major elements in the overall context of the Hybrid Heterogeneous Platform. In this diagram the Local Mainframe Platform labeled as the Enterprise Mainframe shows as its major components the Teleprocessing Monitor, the Operating System, the Application Programs, and the Data. Although, all of these components can be subject to software configuration management, typically the Application Programs and the Data are the only elements that are subjected to it. The Homogeneous Client/Server Platform in a local configuration is illustrated by the configuration titled Enterprise Intranet with its components being the Web Application Server (which should in this case be labeled more appropriately as just Application Server), the Directory and Security Services, and the Database and Transaction Services. Again each of these components can be subject to SCM, however typically only the Server Applications and the Database and Transaction elements are. The elements of the Hybrid Heterogeneous Platform combining the Mainframe with the Client/Server are shown by the overall diagram. In this configuration the mainframe Application Programs and the Data together with the Web applications and the Web database and transaction elements are minimally the elements typically subject to SCM.

# SYSTEM PLATFORM FRAMEWORK




The chart given below correlates each type of platform to the Development Life Cycle associated with it and identifies that platform as being either homogeneous Mainframe or homogeneous Client/Server or heterogeneous hybrid combination of both.

Correlation of Type of System Platform to Type of Development Life Cycle						
SYSTEM PLATFORM COMPONENTS			PLATFORM	TYPE of LIFE		
	Homogenous		Client/Server (Local)		Minimal Life Cycle	
Heterogeneous			INTERNET	Both Life Cycles		
	Homogeneous		MAINFRAME (Local)		Monolithic Life Cycle	

**Types of Development Life Cycle Methodologies determining corresponding SCM approach:**




- **Full Life Cycle SCM approach is appropriate for large scale complex application systems utilizing Monolithic Life Cycle with requirements that do not lend themselves to partitioning into a small number of development segments or phases represented by:**
  - **Phase cascading/waterfall development life cycle model for monolithic complex systems assuming simultaneous development of all requirement components.**
  - **Spiral development life cycle model with evolving requirements with known multiple progressive development phases evolving into a large complex system. The development of each phase incorporates the contents of the previous phases plus new phase components to be developed to constitute the system. In this way this version of the spiral model resembles the monolithic model as it anticipates and plans for the phases to follow the initial foundation phase.**

The charts below displays the application and unique Full Life Cycle SCM (lateral) progression in the Monolithic Life Cycle framework:

Monolithic System Development with Full Life Cycle SCM					
Component 1	RQMTS	DESIGN	DEVELOP	TEST	PRODUCTION
-	RQMTS	DESIGN	DEVELOP	TEST	PRODUCTION
Component n	RQMTS	DESIGN	DEVELOP	TEST	PRODUCTION
SCM	: : : : : Progression of Application Development and SCM : : : : : 				

- **Make/Build SCM approach for systems utilizing Minimal Life Cycle with requirements which lend themselves to being segmented into a limited number of development phases represented by:**
  - **Incremental (Extreme Programming/Rapid Application Development) development life cycle model with requirements that can be segmented and lend themselves for development as functional units independent of each other.**
  - **Spiral development life cycle model with evolving requirements but with known limited progressive development phases resulting in system of relatively simple size and complexity.**

The charts below displays the application and unique Make/Build SCM (vertical) progression in the Minimal Life Cycle framework:

Minimal System Development & Application of SCM					
COMPONENT 1	SCM	• • •	SCM	COMPONENT n	SCM
Requirements	: : : : : Progression of Development and Application of SCM : : : : : 	Requirements	: : : : : Progression of Development and Application of SCM : : : : : 	Requirements	: : : : : Progression of Development and Application of SCM : : : : : 
Design		Design		Design	
Implement		Implement		Implement	

## 7. Customizing the Criteria for SCM Tool Selection

Once a customized SCM approach has been developed, an SCM tool is typically acquired to maximally automate the features of this approach. Besides the cost/benefit considerations and the fact that the tool reflect the customized SCM approach that has been adopted as discussed in the previous sections, there are three major additional considerations that affect the selection of an SCM tool:

- **If the proposed application system involves a hybrid homogeneous platform with a mix of the client/server with the mainframe then it is likely that more than one SCM tool may be needed. So far, SCM tool vendors have tended to specialize in either the client/server or the mainframe platform for their tool. However, as we speak vendors are releasing versions of SCM tools which they claim cover both these platforms simultaneously. That remains to be seen. If true and with all other conditions being met, such an SCM tool would theoretically be ideal in that it would eliminate the problem of integrating the two separate SCM tools that would be used otherwise for each type of platform.**
- **Approach to Managing Application System Code Development impacting SCM Tool selection and SCM Procedures and Practices. The application system development Project Management based on complexity and size of the system and the pressing cost and schedule constraints may choose to have the software development conducted in one of the following manners with the corresponding implications:**
  - **Exclusive development of a functional unit assigned to a single developer. This approach would not have any special significance for the tool.**
  - **Parallel and simultaneous development of features of a functional unit assigned to multiple developers. This development approach would require functional unit feature code merge capability in the SCM tool selected.**
- **Types of Application System Maintenance and Release Cycle inherent in the proposed application system with high impact on SCM Tool selection and SCM Procedures and Practices:**
  - **Single release during a significant time span such a year or longer with high maintenance typically for an in-house developed production application system.**
  - **Multi release during a significant time span such a year with low maintenance typical example is a software vendor product.**

## 8. Conclusions

This SCM Reference Model is intended to be simultaneously a practical and theoretically sound reference guide to planning and implementing an appropriate SCM system for the development and maintenance of an information application system. It is designed to enable an SCM system to be customized based on the platform of the proposed application system and the development life cycle approach necessitated by that platform.

Although presented in an outline form, the scope of this reference model is comprehensive of parametric, architectural, and procedural considerations and its format is primarily graphic to facilitate its validation for correctness and completeness.

Bearing in mind that this is a model and not a complete manual, it can provide a starting point and a reference to planning, developing, and implementing an SCM system.

## 9. General References

Listed below are seminal general references that either relate to the foundations of the SCM discipline or are bibliographical resource lists, including their web addresses, where other SCM resources can be found. This list does not attempt to be exhaustive by any means.

1. **Airtime, *Index for the Project Management System - Software Configuration Management Plan*, [http://sparc.airtime.co.uk/users/wysywig/scmp\\_1.htm](http://sparc.airtime.co.uk/users/wysywig/scmp_1.htm)**
2. **Brad Appleton's Software Configuration Management Links, [www.enteract.com/~bradapp/links/scm-links.html#CM\\_Guides\\_Tut](http://www.enteract.com/~bradapp/links/scm-links.html#CM_Guides_Tut)**
3. **IEEE Computer Society, *828-1998 IEEE Standard for Software Configuration Management Plans – standards*, [www.ieee.org/catalog/software2.html](http://www.ieee.org/catalog/software2.html)**
4. **Institute of Configuration Management, The CMII Model, CMII Awareness Course, [www.icmhq.com](http://www.icmhq.com)**
5. **Questcon Technologies' *Software Quality Assurance Links page*, [www.questcon.com/quality/sites.html](http://www.questcon.com/quality/sites.html)**
6. **Software Engineering Institute, Carnegie-Mellon University, *Software Configuration Management Publications*, [www.sei.cmu.edu/legacy/scm/](http://www.sei.cmu.edu/legacy/scm/)**
7. **Systems, Standards & Technology Council of Government Electronic and Information Technology Association, *ANSI/EIA-649-1998 National Consensus Standard for Configuration Management Plan*, [www.geia.org/sstc/prod01.htm](http://www.geia.org/sstc/prod01.htm)**

## 10. Biographical Sketch of the Author

Ghulam H. Hasnain, M.B.A. has over 20 years of development, project management, and consulting experience in the software industry specializing in software configuration management and quality assurance. Ghulam is an

acknowledged professional within the software industry and has held positions at Fortune 500 corporations such as Boeing, Manpower, American Stores and General Electric Information Service. Since 2001 he has been a valuable part of the Software Configuration Solutions, Inc. team working as a Software Configuration Specialist with corporations around the globe.