Software Configuration Management

Wingsze Seaman
COMP250SA
February 27, 2008

Software Configuration Management
Outline

- CM and SCM Definitions
- SCM History
- CMMI and SCM
- SCM Tools
- SCM/Dynamic Systems
- SCM/Software Architecture
- Resources

Software Configuration Management
Definitions

Configuration Management (CM):

- "The discipline of managing change in large, complex systems" [2]
  - Manage and control numerous corrections, extensions, and adaptations that are applied to a system over its lifetime

Software Configuration Management (SCM):

- "The discipline that enables us to keep evolving software products under control, and thus contributes to satisfying quality and delay constraints." [3]
  - Configuration Management of Software Systems
  - "The objective of SCM is to ensure a systematic and traceable software development process in which all changes are precisely managed, so that a software system is always in a well-defined state at all times." [2]

Software Configuration Management
SCM Tool Responsibilities

- Dart separated CM concerns into 8 categories:
  - Components, Structure, Construction, Auditing, Accounting, Controlling, Process and Team

- SCM can be broken down into three main responsibilities:
  1. Component Repository: Versioning & System Models
  2. Tool Support: Workspace Control & Building
  3. Process control

- How do these main responsibilities cover the concerns:
  - Component Repository:
    - Components, Controlling
  - Tool Support:
    - Construction, Auditing, Controlling
  - Process Control:
    - Accounting, Process, Team

Software Configuration Management

History [2][3]

- 1950s – CM initiated in aerospace industry when production of spacecraft experienced difficulties caused by inadequately documented engineering changes
- 1960s – special ‘correction cards’ were used on the UNIVAC-1100 EXEC-8 operating system
- 1970s – first version control system providing history, delta, multi-user management and merging facilities
  – first version control system
  – appearance of CM tools SCCS, Diff, RCS, Make, and Sablime
- 1980s – debates on the most efficient type of storage and retrieval mechanism
  – resulted in text-based delta algorithms
  – “programming in the large” (versioning, rebuilding, composition)
  – SEI Fellow Watts S. Humphrey created CMM in 1987
  – Some resulting applications:
    – DSEE: only serious commercial product introducing system model concept which was an Architecture Description Language ancestor
    – NSE: workspace and cooperative work control
    – Adele: specialized product model with automatic configuration building
    – Aides de Camp: introducing change set

Software Configuration Management

History [2][3] continued

- 1990s
  - Management of non-textual objects and new algorithms for storing and retrieving objects
  - "programming in the many" (process support, concurrent engineering)
  - "programming in the wide" (web remote engineering)
  - Some resulting applications:
    - ClearCase (DSEE successor): virtual file system
    - Continuus: explicit process support
- 2000
  - inexpensive disk storage, faster CPUs, and more nontextual objects
  - More advanced SCM systems

Software Configuration Management
CMM and CMMI

CMM: Capability Maturity Model

- Developed by Software Engineering Institute (SEI)
- "the quality of a system or product is highly influenced by the quality of the process used to develop and maintain it"

CMMI : Capability Maturity Model Integration

- Result of the evolution of three source models:
  1. The Capability Maturity Model for Software (SW-CMM) v2.0 draft C
  2. The Systems Engineering Capability Model (SECM)
  3. The Integrated Product Development Capability Maturity Model (IPD-CMM) v0.98

WHY CMM?

"Process improvement maturity model for the development of products and services. It consists of best practices that address development and maintenance activities that cover the product lifecycle from conception through delivery and maintenance." [1]

Provides a means for measuring an organization on process maturity

Software Configuration Management
CMM and CMMI [5]

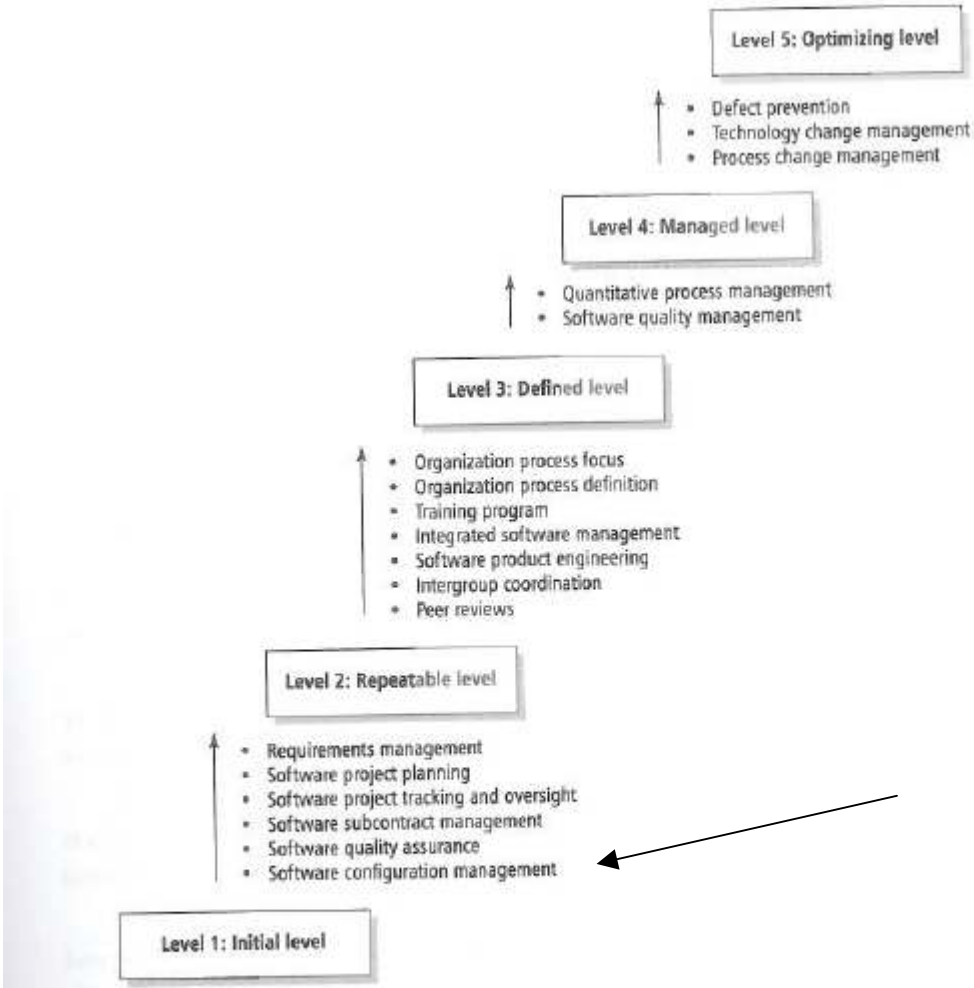


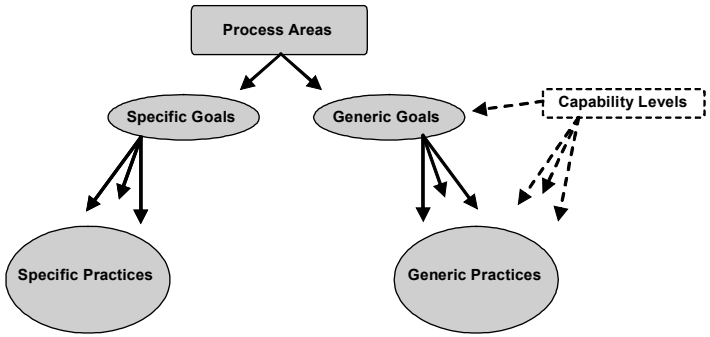
Figure 6.13 Maturity levels and associated key process areas (*Source: M.C. Paulk et al.*, The Capability Maturity Model, ©*Addison-Wesley Longman 1995. Reproduced with permission*)

SCM part of Maturity Level 2 of CMM

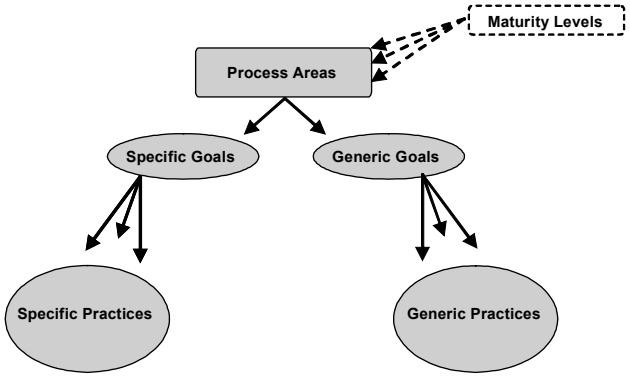
Software Configuration Management

CMM Representations Capability vs Maturity Levels [1]

Continuous Representation



Staged Representation



Level	*Continuous Representation Capability Levels*	*Staged Representation Maturity Levels*
Level 0	Incomplete	N/A
Level 1	Performed	Initial
Level 2	Managed	Managed
Level 3	Defined	Defined
Level 4	Quantitatively Managed	Quantitatively Managed
Level 5	Optimizing	Optimizing

Software Configuration Management
CMMI Process Areas

There are 22 process areas [1]:

- Causal Analysis and Resolution (CAR)
- Configuration Management (CM)
- Decision Analysis and Resolution (DAR)
- Integrated Project Management +Integrated Product and Process Development (IPM+IPPD)
- Measurement and Analysis (MA)
- Organizational Innovation and Deployment (OID)
- Organizational Process Definition +IPPD (OPD+IPPD)
- Organizational Process Focus (OPF)
- Organizational Process Performance (OPP)
- Organizational Training (OT)
- Product Integration (PI)
- Project Monitoring and Control (PMC)
- Project Planning (PP)
- Process and Product Quality Assurance (PPQA)
- Quantitative Project Management (QPM)
- Requirements Development (RD)
- Requirements Management (REQM)
- Risk Management (RSKM)
- Supplier Agreement Management (SAM)
- Technical Solution (TS)
- Validation (VAL)
- Verification (VER)

Software Configuration Management
CMMI Process Areas - Categories

These 22 process areas can be grouped into four categories:

- Process Management
- Project Management
- Engineering
- Support

Configuration Management is considered part of the Support category. It's purpose is to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits.

Specific Goal and Practice Summary:

SG1 Establish Baselines
- SP 1.1 Identify Configuration Items
- SP 1.2 Establish a Configuration Management System
- SP 1.3 Create or Release Baselines

SG2 Track and Control Changes
- SP 2.1 Track Change Requests
- SP 2.2 Control Configuration Items

SG3 Establish Integrity
- SP 3.1 Establish Configuration Management Records
- SP 3.2 Perform Configuration Audits

Software Configuration Management

Configuration Management Plan

Configuration Management Plan

- CM Plan document describes methods to identify configuration items, to control change requests, and to document the implementation of those change requests.
- Contents of plan can be found in IEEE Standard for Software Configuration Management Plans, IEEE Std 828-1990. [IEE90b]
- Configuration or Change Control Board (CCB)
- Main sections [5]:
  - Management: This section describes how the project is being organized. Particular attention is paid to responsibilities which directly affect configuration management; how are change requests being handled, how are development phases closed, how is the status of the system maintained, how are interfaces between components identified? Also, the relationship with other functional organizations, such as software development and quality assurance, is delineated.
  - Activities: this section describes how a configuration will be identified and controlled and how its status will be accounted and reported. A configuration is identified by a baseline: a description of the constituents of that configuration. Such a configuration must be formally approved by the parties involved.

Software Configuration Management

Example of Configuration Management Plan [5]

1 Introduction
   a. Purpose
   b. Scope
   c. Definitions and acronyms
   d. References
   e. Tailoring
2 SCM management
   a. Organization
   b. SCM responsibilities
   c. Applicable policies, directives and procedures
3 SCM Activities
   a. Configuration Identification
   b. Configuration Control
   c. Configuration Status Accounting
   d. Configuration Auditing and Review
   e. Interface Control
   f Subcontractor/vendor control
4 SCM Milestones
5 SCM Resources/Training
6 SCM Plan Maintenance

Software Configuration Management
SCM Tools/Products

- IBM Rational ClearCase
- Perforce
- PureCM
- Microsoft Visual SourceSafe
- SourceGear Vault
- Telelogic Synergy/CM
- Concurrent Versions Systems (CVS):
- Subversion (SVN)

Software Configuration Management
SCM & Dynamic Systems

- Dynamic Change Management:
  - Separating structural concerns from component application concerns
  - Managing change without knowing what could change in the future as the system is extended (new functions, updated functions)

  - Objectives of an application independent configuration management facility [4]:
    - Changes should be specified in terms of the system structure
    - Change specifications should be declarative
    - Change specifications should be independent of the algorithms, protocols, and states of the application
    - Changes should leave the system in a consistent state
    - Changes should minimize the disruption to the application system

Software Configuration Management
SCM/Software Architecture

- Involvement of stakeholders
- Components need to be identified
- Relationships amongst components must be addressed
- Process to set up SCM and Software Architecture
- Control of SCM and Architecture
- Way to resolve change (CCB teams)

Software Configuration Management
References

[1] M. Chrissis, M. Konrad, S. Shrum. CMMI: Guidelines for Process Integration and Product Improvement. 2nd Edition, Addison-Wesley. 2007

[2] J. Estublier, G. Clemm, D. LeBlanc, W. Tichy, A. van der Hoek, D. Wiborg-Weber, R. Conradi. "Impact of Software Engineering Research on the Practice of Software Configuration Management," ACM Transactions on Software Engineering and Methodology, Vol. 14, No. 4, October 2005.

[3] J. Estublier. “Software configuration management: a roadmap,” Proceedings of the Conference on The Future of Software Engineering, May 2000.

[4] J. Kramer and J. Magee. "The Evolving Philosophers Problem: Dynamic Change Management”, IEEE Transactions on Software Engineering, 16 (11) November 1990.

[5] H. van Vliet. Software Engineering: Principles & Practice, 2nd Edition, Wiley, 2002.

[6] J. Charvat. Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects, Wiley, 2003.

[7] R. Raygan. “Software Configuration Management Applied to Service Oriented Architecture,” SoftCOM, 2007.