CALIFORNIA STATE UNIVERSITY
FULLERTON™

COLLEGE OF ENGINEERING
AND COMPUTER SCIENCE

Advanced Software Process

Part III: The Defined Process

12. Software Configuration Management (II)

Dave Garcia-Gomez
Faculty / Lecturer
Department of Computer Science

Course Roadmap

Part I: Software Process Maturity

- 1. A Software Maturity Framework
- 2. The Principles of Software Process Change
- 3. Software Process Assessment
- 4. The Initial Process

Part II: The Repeatable Process

- 5. Managing Software Organizations
- 6. The Project Plan
- 7. Software Configuration Management (Part I)
- 8. Software Quality Assurance

Part III: Defined Process

- 9. Software Standards
- 10. Software Inspections
- 11. Software Testing
- 12. Software Configuration Management (continued)
- 13. Defining the Software Process
- 14. The Software Engineering Process Group

Part IV: The Managed Process

- 15. Data Gathering and Analysis
- 16. Managing Software Quality

Part V: The Optimizing Process

- 17. Defect Prevention
- 18. Automating the Software Process
- *19. Contracting for Software*
- *20. Conclusion*

Software Configuration Management (Continued)

- The Software Configuration Management Plan
- Software Configuration Management Questions
- SCM Support Functions
- The Requirements Phase
- Design Control
- The Implementation Phase
- Operational Data
- The Test Phase
- SCM for Tools
- Configuration Accounting
- The Software Configuration Audit

Software Configuration Management

- To preserve the integrity of a software design throughout its useful life, it must be maintained under SCM control.
  - Most programs have long lives, and many changes are made to correct errors and to make enhancements.
  - When the design is not maintained, these subsequent changes must be made with an incomplete knowledge of the design or the logic behind it.
  - It is essential to update the design at the same time that the code is changed.

Software Configuration Management

To make design changes and to provide for their orderly implementation and test, control must be maintained over the following items:

- Operational concept
- Requirements
- Specifications
- Design documents
- Source code
- Object code
- Test plans
- Test cases, test configurations, and test results
- Maintenance and development tools
- User manuals
- Maintenance manuals
- Interface control documents

Software Configuration Management

- Example Baseline Contents [Table 12.1]
  - To control items and not to constrain early development excessively, design baselines are established at appropriate points.
  - While the items are relevant to the initial software development project, many of them are needed for the subsequent repair and enhancement effort as well.

The Software Configuration Management Plan

- The first step in establishing an SCM system is to develop a Software Configuration Management Plan.
- SCM Plan includes objectives, responsibilities, and the approach and methods to be used.
- Example SCMP Contents [Table 12.2]
- SCM Plan Checklist [Table 12.3]

Example SCMP Contents [Table 12.2]

- Overview
- SCM Organization
- SCM Methods
- SCM Procedures
- SCM Implementation

SCMP Checklist [Table 12.3]

- General
- Organization
- Identification
- Control
- Status Accounting
- Audits
- Program Phasing
- Subcontractor/Vendor
- Interface Control

Software Configuration Management Questions

- For the requirements phase:
  - – Where is the official requirement statement?
  - – What changes have been made, and have they been approved?
  - – How much change activity has there been, and what is the impact on design and/or development?
  - – How many requirements changes have been made without changes in the contract scope?

Software Configuration Management Questions

- For the design phase:
  - Where is this particular requirement covered by the design?
  - What is the particular requirement that this design element satisfies?
  - What is the current approved specification for this interface?
  - Has the format for this data item been specified, where, by whom, and who is using it?
  - What is the design impact of this requirements change?
  - What tools and methods were used to develop this design?

Software Configuration Management Questions

- For implementation:
  - How has this particular function been allocated to the various implementation areas, and why?
  - What is the design logic for this particular code element?
  - How is the design affected by this code change?
  - What is the implementation impact of this requirements change?
  - What compiler version was used to produce this code?

Software Configuration Management Questions

- For the testing phase:
  - Where are the tests that verify this functional requirement?
  - Where is the test data for use with these tests?
  - Where is the documentation that explains the results to be expected from these tests?
  - Has this test bucket been updated to reflect the functional content of the latest release or system build?
  - What tests have been run on this program version, and what were the results?
  - What test tools and support facilities were used for these tests?

SCM Support Functions

- To support a full range of SW engineering activities, a fairly sophisticated set of SCM functions is required.
- While those functions may be used in varying degrees during requirements, design, implementation, test, and maintenance, they are all required to provide adequate support and effective control.

SCM Support Functions

- A protected baseline for the operational concept, specifications, design, implementation, test, and tools
- A protected file with a description of all changes and revisions
- Means for each software engineer to read any unlocked element in the baseline
- A private workspace where charged-out designs or implementations can be modified

SCM Support Functions

- Templates that assist in the preparation of new design, implementation, or test descriptions
- A procedure for making approved charge-outs that permits SW engineers to obtain any available baseline element for their private workspace and lock out the SCM copy function to prevent anyone else from making simultaneous changes
- A charge-in procedure

SCM Support Functions

- A procedure for making approved deletions of defunct elements
- A way to collect, format, and produce consolidated system documents containing the key element descriptions for any given baseline
- A way to check that all elements and relevant descriptions have been carried over between baselines

SCM Support Functions

- A centralized data dictionary containing the official record of all named items and their formats
- A where-used record of every use of every interface and data item in the system

The Requirement Phase

- The essential role of the requirements phase is to ensure that the users' needs are properly understood before designing and implementing a system to meet them.
- The requirements also provide part of the basis for system and acceptance testing.

Requirements Changes

- The early design will likely be changed many times.
- If the requirements cannot be completely specified at the beginning of the project, they must be gradually evolved.
  - Throughout the requirements definition and feedback process, change is constant, and means are needed to control it.
  - As soon as initial requirements agreement is reached, a baseline is established as a design foundation, as a basis for change negotiations, and as a reference point for acceptance testing.

Requirements Baselines and Change Control

- Control of changes to the requirements baseline is handled in much the same way as described early (in Ch.7).
- Each change is submitted on a design change request (DCR) form and reviewed, approved, and tracked with the same discipline as design and code changes.
  - All requirements changes are handled in this way, whether submitted by the customer, systems design, development, or test.
  - Without such a discipline, baseline integrity for the entire project is exposed.

Requirements Baselines and Change Control

- It is also essential to establish project naming conventions.
- The Software Configuration Identification (SCI) system uniquely identifies every project development item.
- The SCI definitions are kept under configuration control and expanded as more is learned about the product and its structure.
- SCI Tree Structure [Figure 12.1]
- Design Change Request Form [Table 12.4]

External Specifications

- As the project proceeds, detailed specifications are developed for each program component.
- While these specification may all be produced at one time, generally some of the component specs are evolved in concert with development.

Design Control

- During design, the Change Control Board (CCB) controls the design changes, and SCM maintains complete records of every change and its rationale.
- The Software Configuration Identification then connects each design element back to the specifications.

Design Control

- A good design requires a clear problem statement.
- The design is the template against which the implementers produce the code; it must be clear, unambiguous, and absolutely current.
- A permanent design reference is required for later repair or product extension.
- A permanent record is needed of the design changes, why they were made, and how they were implemented.
- To maintain control as the design is subdivided into modules, an SCM procedure is required to check the self-consistency of the modules and interface descriptions in each design baseline.

Design Control

- The design phase continues until the smallest structural elements (e.g., modules, packages) of the systems are defined and specified.
- The detailed design phase then starts with these module specifications.
- Example Module Specification [Table 12.5]
- Additional SCM facilities needed for detailed design:
  - Template for module specifications and design skeletons, and etc. (p.238)

The Implementation Phase

- While the SCM tasks for the implementation phase were largely covered in Chapter 7, some additional design-related capabilities are also needed.
  - First, with a fully controlled design, implementation can start on a solid foundation.
  - As implementation proceeds, design problems will be found and design changes will be needed.
  - When such changes are made, the design documentation is updated.

Operational Data

- In addition to code and documentation, many software systems also require a considerable amount of operational data.
  – For applications such as payroll, for example, this would include the appropriate employee records and tax data.
  – Most large-scale software systems generally require large amounts of specialized data that must be specified, produced, tested, documented, and controlled.
  – It is thus essential that SCM provisions be made to handle this data.

The Test Phase

- Once a newly developed program is put in a baseline, all further changes must be controlled.
- The major concerns are problem incidents, temporary patches, permanent bug fixes, and enhancements.
- In addition, precise and complete records must be maintained of every test run, the data and test case used, the configurations on which they were run, and the results.

The Test Phase

- Once a newly developed program is put in a baseline, all further changes must be controlled.
- The major concerns are problem incidents, temporary patches, permanent bug fixes, and enhancements.
- In addition, precise and complete records must be maintained of every test run, the data and test case used, the configurations on which they were run, and the results.

The Test Phase

- Implementation Change Control Procedure [Figure 12.2]

SCM for Tools

- One important SCM function that is often overlooked is the need to maintain baseline control over the complete set of tools used to specify, design, implement, and test the software.
  - One approach is to maintain a standard tool set under configuration control and to establish a new tool baseline whenever any change is made.
  - Each module specification, design, and code record then includes a notation of the tool baseline used in its production.
  - A summary of this data should also be retained with the module revision history.

Configuration Accounting

- The purpose of software configuration status accounting is to maintain a continuous record of the status of all baselined items.
- This record is not only a useful management tool, but, if old records are routinely archived, it also provides valuable insurance against disaster.

Configuration Accounting

- The information required for SC status accounting:
  - The time at which each baseline was established
  - When each SW configuration item and change was included in the baseline
  - A description of each configuration item
  - The status of each SW-related engineering change
  - The description of each SW change
  - The status of each SW change
  - The documentation status for each baseline
  - The changes planned for each identified future baseline

The Software Configuration Audit

- A software configuration audit should periodically be performed to ensure that the SCM practices and procedures are rigorously followed.
- This audit can be performed directly by the SCM staff or by some independent assurance function.
- It is generally advisable to have SCM conduct frequent self-audits with an independent party occasionally performing spot checks.

References

Humphrey, Watts S., *Managing the Software Process*, The SEI Series in Software Engineering, Addison-Wesley, 1989. (29th Printing, May 2003) (ISBN 0-201-18095-2)