EXPERIMENT – 09

Aim: Prepare Version control and change control for software configuration items

Objective: Preparing version control and change control for configuration items

Theory:

The output of software process is information divided into 3 categories.

- Computer Program
- Documents
- Data

The items that comprise all information produced as part of software process are collectively called **software configuration**.

Software Configuration Management is a set of activities that have been developed to manage change throughout the life cycle of computer software. It is software quality assurance activity.

IEEE definition of SCM- 'The process of identifying and defining components in a system, controlling the release and change throughout the life cycle, recording and reporting the status of components and change requests, and verifying the completeness and correctness of system components.'

Goals

- Provide a defined and controlled configuration of the software throughout the lifecycle.
- Provide the ability to consistently replicate the executable object code for software manufacture or to regenerate it in case an investigation or modification is needed.
- Provide control over process inputs and outputs during the lifecycle to insure consistency and repeatability of process activities.
- Provide controls to insure that problems receive attention and that change are recorded, approved and implemented.

- Provide evidence of approval of the software by controlling the outputs of the software process.
- Aid in assessing the software product' compliance with requirements.
- Insure that secure physical archiving, recovery, and control are maintained for configuration items.

STAFFING OF SCM IN SEVERAL WAYS

- A single team performs all software configuration management activities for the whole organization.
- A separate configuration management team is set up for each project.
- All the software configuration management activities are performed by the developers themselves.
- Mixture of all of the above.

CONFIGURATION ITEM

"An aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process".

IEEE defines configuration item as-'an element of configuration item, consisting of selecting the configuration items for a system and recording their functional and physical characteristics in technical documentation.

- Software Configuration items are not only program code segments but all type of documents according to development, e.g
  - all types of code files
  - drivers for tests
  - analysis or design documents
  - user or developer manuals
  - system configurations (e.g. version of compiler used)
- In some systems, not only software but also hardware configuration items (CPUs, bus speed frequencies) exist!
- Even a commercial product used in the system can be a configuration item.

SCM PROCESS

1. It is a software quality assurance activity to control change
2. Any discussion of SCM introduces a complex question

a) How does an organization identify and manage many exiting versions of program in a manner that will enable change to be accommodated efficiency?

b) How does organization control changes before and after release of software to customer?

C) Who has responsibility for approving and ranking changes?

d) How can we ensure that change have been made properly?

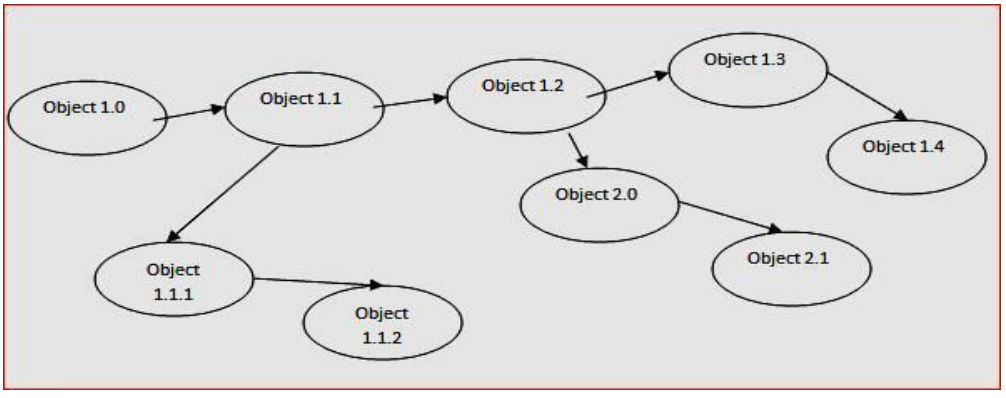
e) What mechanics is used to apprise others of changes that are made?

These questions lead to the definition of five SCM tasks: identification, version control, and change control, configuration auditing, and reporting.

SCM : VERSION CONTROL

An initial release or re-release of a configuration item associated with a complete compilation or recompilation of the item. Different versions have different functionality.

1. Combines procedures and tool to manage difference version of configuration object that are created during software process.
2. One representation of different versions of a system is evaluation graph.
3. Each version of software is a collection of SCI, and each version may be composed of different variants.
4. Object Pool: It is a representation of components of SCI, and each version may be composed of different variants.



Each node on graph is an aggregate object i.e. a complete version of software.

SCM : CHANGE CONTROL

Change request, change report and engineering change order are generated as the part of the configuration control activity within the SCM Process. These documents are often represented as printed or electronic forms .the outline that follows is intended to provide an indication of the content of each. The software change report (SCR) is completed as consequence of a formal request for change of a base –lined SCI. The SCR is completed after the change request has been evaluated by software engineering

Parts of Software Change Report

- **Name , Identification and Description of SCI:** The name version/control Numbers of the SCI is specified, including page numbers if a document is involved.
- **Requester**: The name of the person requesting the change
- **Evaluator/Analyzer** : The name of the person who has evaluated the change request
- **Contact /information**: How to contact the requester and the Evaluator
- **Date/Location & Time**: When and where the change report was generated

Change Request Form	
Project:	**Number:**
Change Requester:	**Date:**
Requested Change:	
Change Analyzer :	
Components Affected:	
Associated Components:	
Change Assessments:	
Change Priority:	
Change Implementation:	
Estimated Effort:	
Date to CCB:	**CCB decision date:**1/2/03
CCB Decision:	
Change Implementer:	**Date of Change:**
Date Submitted to QA:	**QA decision:**
Date Submitted to CM:	
Comments	

Example: Version control and Change control for Attendance Maintenance System.

Change Request Form	
Project: Attendance Maintenance system	**Number:** 19-10
Change Requester: SRINADH SWAMY	**Date:** 11/11/15
Requested Change: When Monthly attendance component is selected from the structure, display the name of the file where it is stored	
Components Affected: Display-Icon. Select, Display-Icon. Display	
Associated Components: File Table where attendance is stored	
Change Assessments: Relatively simple to implement as a file name table is available. Requires the design and implementation of a display field. No changes to associated components are required.	
Change Implementation: Based on Analyzer Comments and specifications	
Estimated Effort: 0.5 days	
Date to CCB: 1/12/15	**CCB decision date:**16/01/16
CCB Decision: Accept change to be implemented in Release 2.1	
Change Implementer: Srikanth Varma	**Date of Change:** 20/01/16
Date Submitted to QA: 22/01/16	**QA decision:** Yes
Date Submitted to CM: 24/01/16	
Comments: The change was successfully made in the software	

Add-on Experiment:

Derive LOC based estimation for size oriented metrics.

Overview

Software process and project metrics are quantitative measures that enable software engineers to gain insight into the efficiency of the software process and the projects conducted using the process framework. In software project management, we are primarily concerned with productivity and quality metrics. There are four reasons for measuring software processes, products, and resources (to characterize, to evaluate, to predict, and to improve).

Process and Project Metrics

- Metrics should be collected so that process and product indicators can be ascertained
- *Process metrics* used to provide indictors that lead to long term process improvement
- *Project metrics* enable project manager to
  - Assess status of ongoing project
  - Track potential risks
  - Uncover problem are before they go critical
  - Adjust work flow or tasks
  - Evaluate the project team's ability to control quality of software wrok products

Process Metrics

- Private process metrics (e.g. defect rates by individual or module) are only known to by the individual or team concerned.
- Public process metrics enable organizations to make strategic changes to improve the software process.
- Metrics should not be used to evaluate the performance of individuals.
- Statistical software process improvement helps and organization to discover where they are strong and where are week.

Statistical Process Control

1. Errors are categorized by their origin
2. Record cost to correct each error and defect
3. Count number of errors and defects in each category
4. Overall cost of errors and defects computed for each category
5. Identify category with greatest cost to organization
6. Develop plans to eliminate the most costly class of errors and defects or at least reduce their frequency

Project Metrics

- A software team can use software project metrics to adapt project workflow and technical activities.
- Project metrics are used to avoid development schedule delays, to mitigate potential risks, and to assess product quality on an on-going basis.
- Every project should measure its inputs (resources), outputs (deliverables), and results (effectiveness of deliverables).

Software Measurement

- *Direct process measures* include cost and effort.
- *Direct process measures* include lines of code (LOC), execution speed, memory size, defects rep orted over some time period.
- *Indirect product measures* examine the quality of the software product itself (e.g. functionality, complexity, efficiency, reliability, maintainability).

Size-Oriented Metrics

- Derived by normalizing (dividing) any direct measure (e.g. defects or human effort) associated with the product or project by LOC.
- Size oriented metrics are widely used but their validity and applicability is widely debated.

Function-Oriented Metrics

- Function points are computed from direct measures of the information domain of a business software application and assessment of its complexity.
- Once computed function points are used like LOC to normalize measures for software productivity, quality, and other attributes.
- The relationship of LOC and function points depends on the language used to implement the software.

Reconciling LOC and FP Metrics

- The relationship between lines of code and function points depends upon the programming language that is used to implement the software and the quality of the design
- Function points and LOC-based metrics have been found to be relatively accurate predictors of software development effort and cost
- Using LOC and FP for estimation a historical baseline of information must be established.

Object-Oriented Metrics

- Number of scenario scripts (NSS)
- Number of key classes (NKC)
- Number of support classes (e.g. UI classes, database access classes, computations classes, etc.)

- Average number of support classes per key class
- Number of subsystems (NSUB)

Use Case-Oriented Metrics

- Describe (indirectly) user-visible functions and features in language independent manner
- Number of use case is directly proportional to LOC size of application and number of test cases needed
- However use cases do not come in standard sizes and use as a normalization measure is suspect
- Use case points have been suggested as a mechanism for estimating effort

WebApp Project Metrics

- Number of static Web pages (N_{sp})
- Number of dynamic Web pages (N_{dp})
- Customization index: $C = N_{sp} / (N_{dp} + N_{sp})$
- Number of internal page links
- Number of persistent data objects
- Number of external systems interfaced
- Number of static content objects
- Number of dynamic content objects
- Number of executable functions

Software Quality Metrics

- Factors assessing software quality come from three distinct points of view (product operation, product revision, product modification).
- Software quality factors requiring measures include
  - correctness (defects per KLOC)
  - maintainability (mean time to change)
  - integrity (threat and security)
  - usability (easy to learn, easy to use, productivity increase, user attitude)
- Defect removal efficiency (DRE) is a measure of the filtering ability of the quality assurance and control activities as they are applied through out the process framework

  DRE = E / (E + D)

  E = number of errors found before delivery of work product

  D= number of defects found after work product delivery

Integrating Metrics with Software Process

- Many software developers do not collect measures.
- Without measurement it is impossible to determine whether a process is improving or not.
- Baseline metrics data should be collected from a large, representative sampling of past software projects.
- Getting this historic project data is very difficult, if the previous developers did not collect data in an on-going manner.

Arguments for Software Metrics

- If you don't measure you have no way of determining any improvement
- By requesting and evaluating productivity and quality measures software teams can establish meaningful goals for process improvement
- Software project managers are concerned with developing project estimates, producing high quality systems, and delivering product on time
- Using measurement to establish a project baseline helps to make project managers tasks possible

Baselines

- Establishing a metrics baseline can benefit portions of the process, project, and product levels
- Baseline data must often be collected by historical investigation of past project (better to collect while projects are on-going)
- To be effective the baseline data needs to have the following attributes:
  - data must be reasonably accurate, not guesstimates
  - data should be collected for as many projects as possible
  - measures must be consistent
  - applications should be similar to work that is to be estimated

Metrics for Small Organizations

- Most software organizations have fewer than 20 software engineers.
- Best advice is to choose simple metrics that provide value to the organization and don't require a lot of effort to collect.
- Even small groups can expect a significant return on the investment required to collect metrics, if this activity leads to process improvement.

Establishing a Software Metrics Program

1. Identify business goal
2. Identify what you want to know
3. Identify subgoals
4. Identify subgoal entities and attributes
5. Formalize measurement goals
6. Identify quantifiable questions and indicators related to subgoals
7. Identify data elements needed to be collected to construct the indicators
8. Define measures to be used and create operational definitions for them
9. Identify actions needed to implement the measures
10. Prepare a plan to implement the measures