SE-3002
SOFTWARE QUALITY ENGINEERING
RUBAB JAFFAR
RUBAB.JAFFAR@NU.EDU.PK

Part IV-QUANTIFIABLE QUALITY IMPROVEMENT
Quality models and measurements, SCM
Lecture # 37, 38, 39
13, 14, 16 Dec

TODAY'S OUTLINE

- Analysis → Feedback loop→ Follow-up

- Quality models and measurements

  - Types of Quality Assessment Models.

  - Comparing Quality Assessment Models.

  - Data Requirements and Measurement

  - Measurement and Model Selection.

- Software Configuration Management (SCM)

  - Why SCM?

  - SCM concepts and Terminology

  - Software Configuration Management Activities

  - Change Management

  - Build and Release Management

  - System Release

- Revision

QUALITY MEASUREMENT

- The primary purpose of the measurement and analysis activities is to provide feedback and useful information to manage software quality, the quality engineering process, and the overall software development/maintenance process and activities.

- The feedback and information provided are based on the analysis results using various models on the data collected from the quality assurance (QA) and the general development activities.

- In this lecture we examine and classify these models, relate them to the required measurements, compare the different models, and outline a general strategy to select appropriate models and measurements to satisfy specific quality assessment and improvement goals under specific application environments.

QA DATA AND ANALYSIS

- Generic testing process:
  - Test planning and preparation.
  - Execution and measurement.
  - Test data analysis and follow-up.
  - Related data $\Rightarrow$ quality $\Rightarrow$ decisions
- Other QA activities:
  - Similar general process.
  - Data from QA/other sources
  - Models used in analysis and follow-up:
    - provide timely feedback/assessment
    - prediction, anticipating/planning
    - corrective actions $\Rightarrow$ improvement

QA MODELS AND MEASURES

- General approach
  - Adapt GQM-paradigm.
  - Quality: basic concept and ideas.
  - Compare models $\Rightarrow$ taxonomy.
  - Data requirements $\Rightarrow$ measurements.
  - Practical selection steps.
  - Illustrative examples.
- Quality attributes and definitions:
  - Q models: data $\Rightarrow$ quality
  - Correctness vs. other attributes
  - Our definition/restriction: being defect-free or of low-defect
  - Examples: reliability, safety, defect count/density/distribution/etc.

QUALITY ANALYSIS

- Analysis and modeling:

- . Quality models: data $\Rightarrow$ quality

  - a.k.a. quality assessment models or quality evaluation models

- Various models needed for .

  - Assessment, prediction, control

  - Management decisions

  - Problematic areas for actions

  - Process improvement

- Measurement data needed

- Direct quality measurements: success/failure (& defect info)

- Indirect quality measurements:– activities/internal/environmental. Indirect but early quality indicators.

- All described in Chapter 18

QUALITY MODELS

- Practical issues:
    - Applicability vs. appl. environment
    - Goal/Usefulness: information/results?
    - Data: measurement data required
    - Cost of models and related data
- Type of quality models
    - Generalized: averages or trends
    - Product-specific: more customized
    - Relating to issues above

MODELS FOR QUALITY ASSESSMENT

- Quality assessment models are analytical models that provide quantitative assessment of selected quality characteristics or sub-characteristics based on measurement data from software projects.

- Such models can help us obtain an objective assessment of our current product quality, in contrast to the often unreliable subjective assessment based on personal judgment or imprecise qualitative assessment.

- When applied over time, these models can provide us with an accurate prediction of the future quality, which can be used to help us make project scheduling, resource allocation, and other management decisions.

GENERALIZED MODELS: OVERALL

- Model taxonomy:

- Generalized:
  - overall, segmented, and dynamic

- Product-specific:
  - semi-customized: product history
  - observation-based: observations
  - measurement-driven: predictive
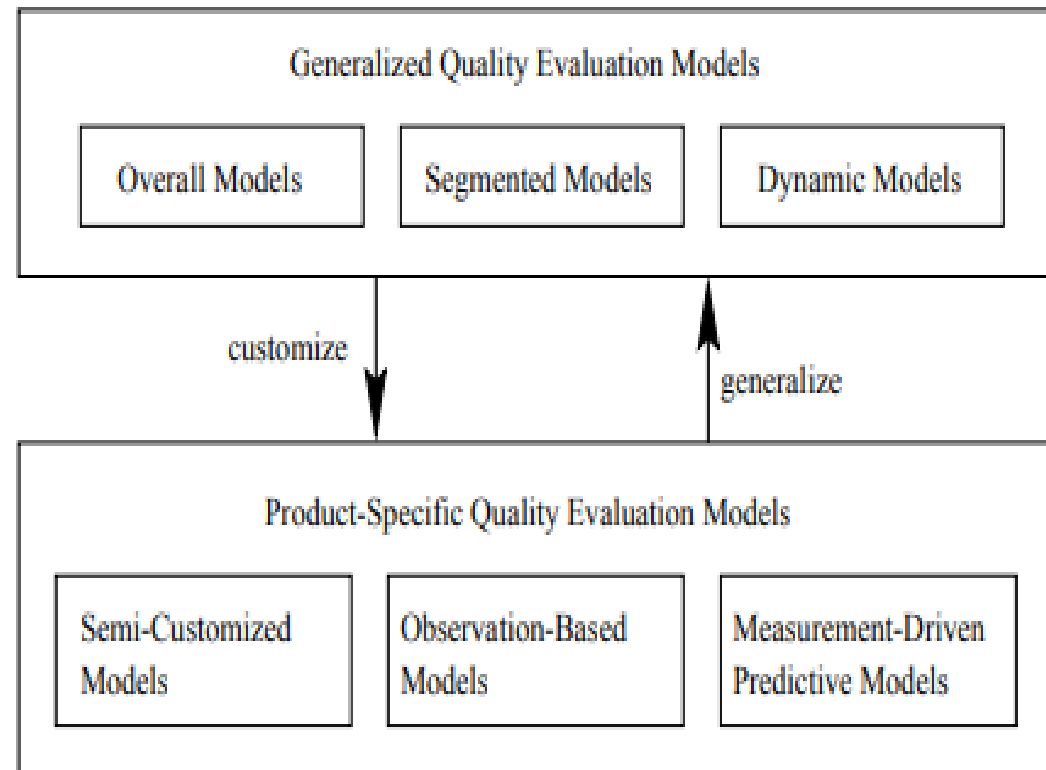
GENERALIZED MODELS: OVERALL

- Key characteristics
  - Industrial averages/patterns: $\Rightarrow$ (single) rough estimate.
  - Most widely applicable.
  - Low cost of use.
- Examples: Defect density.
  - Estimate total defect with sizing model.
  - (counting in-field unique defect only)
- Non-quantitative overall models:
  - As extension to quantitative models.
  - Examples: 80:20 rule, and other general observations.

GENERALIZED MODELS: SEGMENTED

- Key characteristics:
- Estimates via product segmentation.
- Model: segment $\rightarrow$ quality.
- Multiple estimates provided.
- Other applications.
- Commonly used in software estimation.
- Example: COCOMO models.

A segmented model for reliability level estimation

Product Type	Failure Rate (per hour)	Reliability Level
safety-critical software	$< 10^{-7}$	ultra-high
commercial software	10^{-3} to 10^{-7}	moderate
auxiliary software	$> 10^{-3}$	low

GENERALIZED MODELS: DYNAMIC

- Example: Putnam model

- Rayleigh curve for failure rate: $r = 2Bate^{-at2}$

- Overall/average trend over time.

- Often expressed as a mathematical function or an empirical curve.

- Combined models possible,

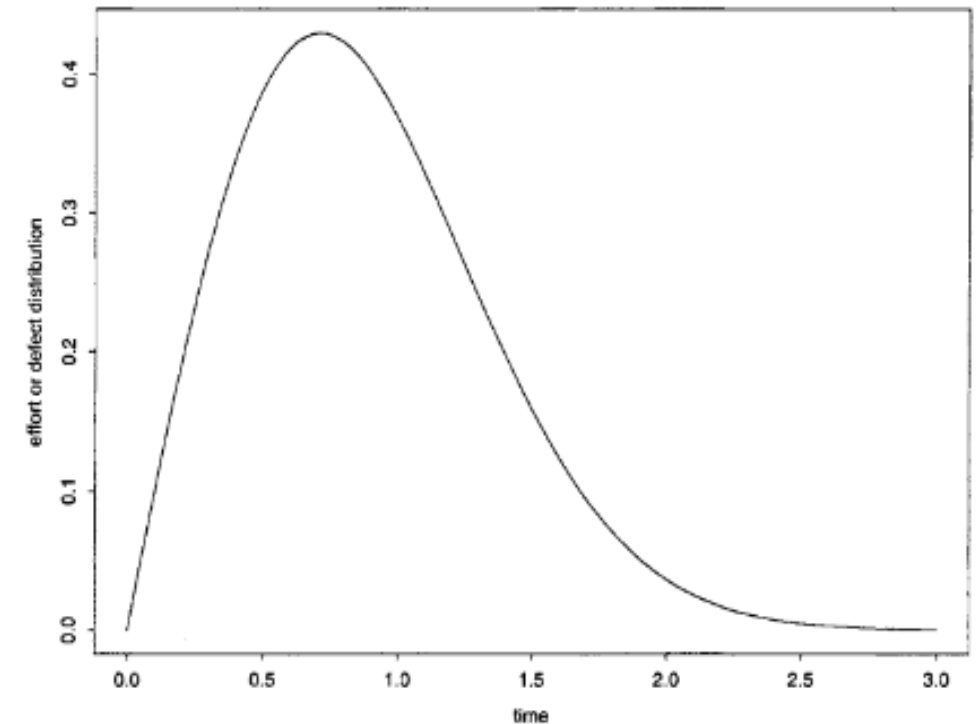- e.g., segmented dynamic models.

Figure 19.2 Effort or defect profile in the Putnam Model

PRODUCT-SPECIFIC MODELS (PSM)

- Product-specific info. Used (vs. none used in generalized models)

- Better accuracy/usefulness at cost ↑

- Three types:

  - semi-customized

  - observation-based

  - measurement-driven predictive

- Connection to generalized models (GMs):

  - Customize GMs to PSMs with new/refined models and additional data.

  - Generalize PSMs to GMs with empirical evidence and general patterns.

SEMI-CUSTOMIZED MODELS

- Semi-customized models:

  - Project level model based on history.

  - Data captured by phase.

  - Both projections and actual.

  - Linear extrapolation.

- The information from these semi-customized models can be directly used to predict defect distribution for the current release.

Table 19.2   DRM (defect removal model): defect distribution for previous releases of a product

Requirement	Design	Coding	Testing	Support
5%	10%	35%	40%	10%

PSM: OBSERVATION-BASED

- Observation-based models relate observations of the software system behavior to information about related activities to provide more precise quality assessments.

- Examples of such models include various software reliability growth models (SRGMs) where observed failures and associated time intervals are fitted to SRGMs to evaluate product reliability.

- Focus on the effect/observations

- Assumptions about the causes

- Assessment-centric

- Example: Goel-Okumoto SRGM
  - functional relation: $m(t) = N(1 - e^{-bt})$
  - observed failures over time
  - curve fitting
  - reliability assessment/prediction
  - management decisions: exit criteria

PSM: PREDICTIVE

- Measurement-driven predictive models

  - Establish predictive relations between quality and other measurement on the historical data

  - Provide early prediction of quality , and identify problems early so that timely actions can be taken to improve product quality.

  - Modeling techniques: regression, TBM etc.

  - Risk assessment and management

- Model characteristics:

  - Response: chief concern

  - Predictors: observable/controllable

  - Linkage quantification

Table 19.3 High-defect modules for two products identified by tree-based

Product	Subset	#Modules	Mean-DF
LS	lrrr	16	9.81
	rlr	53	10.74
	rr	17	22.18
	whole product	1296	1.8
NS	rlll	8	55.0
	rr	5	77.0
	whole product	995	7.9

MODEL COMPARISON AND INTERCONNECTIONS

- Different types of quality assessment models and their relations can be compared by looking at their ability to provide useful information, their applicability to different project environments, and their inter-connections.

- we can compare the following:

  - Usefulness of the modeling results, in terms of how accurate the quality estimates are and the applicability of the models to different environments.

  - Model inter-connections, which can be examined in two opposite directions:

    - Customizution of generalized quality models to provide better quality estimates when product-specific information is available.

    - Generalization of product-specific models when enough empirical evidence from different products or projects is accumulated.

SUMMARY OF QUALITY ASSESSMENT MODELS AND THEIR APPLICATIONS

The usefulness of a model mainly depends on two factors:

- Is it applicable to the specific development environment and the specific product under development or maintenance?

- If so, how accurate the model is in its quality estimates?

Table 19.4 Summary of quality assessment models and their applications

Model Type	Sub-Type	Primary Result	Applicability
generalized models		rough quality estimates	all or by industry
	overall	overall product quality	across industries
	segmented	industry-specific quality	within an industry
	dynamic	quality trend over time	trend in all
product-specific quality models		better quality estimates	specific product
	semi-customized	quality extrapolation	prev→cur release
	observation-based	quality assessments	current product
	measurement-driven	quality predictions	both above

MODEL COMPARISON AND INTERCONNECTIONS

- This usefulness can be weighted against its cost, in particular, the cost of collecting the required measurement data, which is typically the dominant part of the modeling cost.

- Generalized models provide rough quality estimates based on empirical data from industry.

- Product-specific models provide more precise quality assessments using product-specific measurements.

- However, generalized models are more widely applicable and less expensive to use than product-specific models, because they do not require product-specific measurements.

- Consequently, generalized models may be more useful in the product planning stage, and in the early phases of product development, when most product-specific data are unavailable.

MODEL COMPARISON AND INTERCONNECTIONS

- One exception to this general rule is when there exist historical data for the previous releases of the current product. Semi-customized models can be used to provide better estimates under this situation.

- As development or maintenance activities progress, more measurement data can be collected and various detailed quality models in the category of product-specific models, such as observation-based models and measurement-based predictive models, can be used to better manage the QA activities as well as the overall software development or maintenance processes.

SOFTWARE TESTING METRICS

- **Software Testing Metrics** are the quantitative measures used to estimate the progress, quality, productivity and health of the software testing process.

- Metric defines in quantitative terms the degree to which a system, system component, or process possesses a given attribute. E.g. Total number of defects

- **Types of Test Metrics**

- **Process Metrics:** It can bae used to improve the process efficiency of the SDLC (Software Development Life Cycle)

- **Product Metrics:** It deals with the quality of the software product

- **Project Metrics:** It can be used to measure the efficiency of a project team or any testing tools being used by the team members

Process Metrics | Product Metrics | Project Metrics

TEST METRICS GLOSSARY

- **Rework Effort Ratio =** (Actual rework efforts spent in that phase/ total actual efforts spent in that phase) X 100

- **Requirement Creep =** (Total number of requirements added/No of initial requirements)X100

- **Schedule Variance =** (Actual Date of Delivery – Planned Date of Delivery)

- **Cost of finding a defect in testing =** (Total effort spent on testing/ defects found in testing)

- **Schedule slippage =** (Actual end date – Estimated end date) / (Planned End Date – Planned Start Date) X 100

- **Passed Test Cases Percentage** = (Number of Passed Tests/Total number of tests executed) X 100

- **Failed Test Cases Percentage** = (Number of Failed Tests/Total number of tests executed) X 100

- **Blocked Test Cases Percentage** = (Number of Blocked Tests/Total number of tests executed) X 100

- **Fixed Defects Percentage** = (Defects Fixed/Defects Reported) X 100

- **Accepted Defects Percentage** = (Defects Accepted as Valid by Dev Team /Total Defects Reported) X 100

- **Defects Deferred Percentage** = (Defects deferred for future releases /Total Defects Reported) X 100

- **Critical Defects Percentage** = (Critical Defects / Total Defects Reported) X 100

- **Average time for a development team to repair defects** = (Total time taken for bugfixes/Number of bugs)

- **Number of tests run per time period** = Number of tests run/Total time

- **Test design efficiency** = Number of tests designed /Total time

- **Test review efficiency** = Number of tests reviewed /Total time

- **Bug find rote or Number of defects per test hour** = Total number of defects/Total number of test hours

SOFTWARE QUALITY MODELS

- Software Quality Models are a standardised way of measuring a software product.

- With the increasing trend in software industry, new applications are planned and developed everyday.

- This eventually gives rise to the need for reassuring that the product so built meets at least the expected standards.

- During this course, we have discussed the following main categories of software quality models:

- Quality Definition Models:

  - McCall

  - FURPS

  - ISO 9126

  - GQM

  - CMMI

- Your task is to compare the above models and provide a critical analysis based on comparisons.

SOFTWARE CONFIGURATION MANAGEMENT

- **Configuration Management is the *control* of the evolution of complex software.**

- **SCM is the discipline that enable us to keep evolving software products under control, and thus contributes to satisfying quality and delay constraints.**

- **There is a need for storing the different components of a software product (artifacts) and all their versions safely.**

25

SCM DEFINITION

- A discipline applying technical and administrative direction and surveillance to:

  - **Identify and document the functional and physical characteristics of a configuration item**

  - **Control changes to those characteristics**

  - **Record and report change processing and implementation status**

  - **Verify compliance with specified requirements**

Definition from IEEE

SOFTWARE CONFIGURATION MANAGEMENT

- **In software engineering, software configuration management (SCM) is the task of *tracking* and *controlling* changes in the software.**

- **SCM practices include *revision control* and the establishment of *baselines.***

- **If something goes wrong, SCM can determine *what was* changed and *who* changed it**

VERSION CONTROL

- **A component of software configuration management, version control, also known as <u>revision control or source control</u>, is the <u>management of changes</u> to documents, computer programs, large web sites, and other collections of information.**

- **Changes are usually identified by a number termed the "revision number", "revision level", or simply "revision".**

- **For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on.**

- **Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged.**

SOFTWARE CONFIGURATION MANAGEMENT

- **When you develop software, change is inevitable.**

- **Change increase the level of confusion among development team.**

- **Confusion arise when changes are not analyzed before they made, recorded before they are implemented.**

Software   Engineering

SOFTWARE CONFIGURATION MANAGEMENT

Why client is running the wrong version of the software …

Oops! a bug that was fixed in the software suddenly reappears …

I don't know why a developed and tested functionality is missing …

SCM CONCEPTS

- **Configuration**

- **Configuration Control**

- **Software Configuration Item (SCI)**

- **Baselines**

- **Version (or variants)**

Software   Engineering

SCM CONCEPTS

Configuration

- **The functional and *physical characteristics* of hardware or software as set forth in technical documentation or achieved in a product.** **(ANSI/IEEE 828, 1990)**

SCM CONCEPTS

Configuration Control

- **An element of configuration management, consisting of <u>evaluation, coordination, approval or disapproval and implementation of changes</u> to configuration items *after formal establishment* of their configuration identifications.**

(ANSI/IEEE 828,1990)

Software   Engineering

SCM CONCEPTS

- **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."

- **Software configuration items are not only program code segments but all type of documents according to development, e.g**
  - all type 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!**

34

SCM CONCEPTS

Baseline

- **Baseline is a specification or product that has been formally *reviewed* and *agreed* upon, that thereafter serves as the basis for further development and that can be changed only through formal change procedures.**

- **e.g. SRS, Design Doc, Source Code etc**

Software   Engineering

SOFTWARE CONFIGURATION MANAGEMENT - WHY

Management

- **New versions of software systems are created as they change**

  - **For different machines/OS**

  - **Offering different functionality**

  - **Tailored for particular user requirements**

- **Configuration management is concerned with managing evolving software systems**

  - **System change is a team activity**

  - **CM aims to control the costs and effort involved in making changes to a system**

SOFTWARE CONFIGURATION MANAGEMENT – WHY?

- **Involves the development and application of procedures and standards to manage an evolving software product**

- **May be seen as part of a more general quality management process**

- **When released to CM, software systems are sometimes called baselines as they are a starting point for further development**

CONFIGURATION MANAGEMENT LIBRARY

- **The CM Library describes the central storage facility for all CI's.**

- **The CM Library contains the;**

  - **current baseline project,**

  - **approved documentation,**

  - **artifact files to keep the project running smoothly**
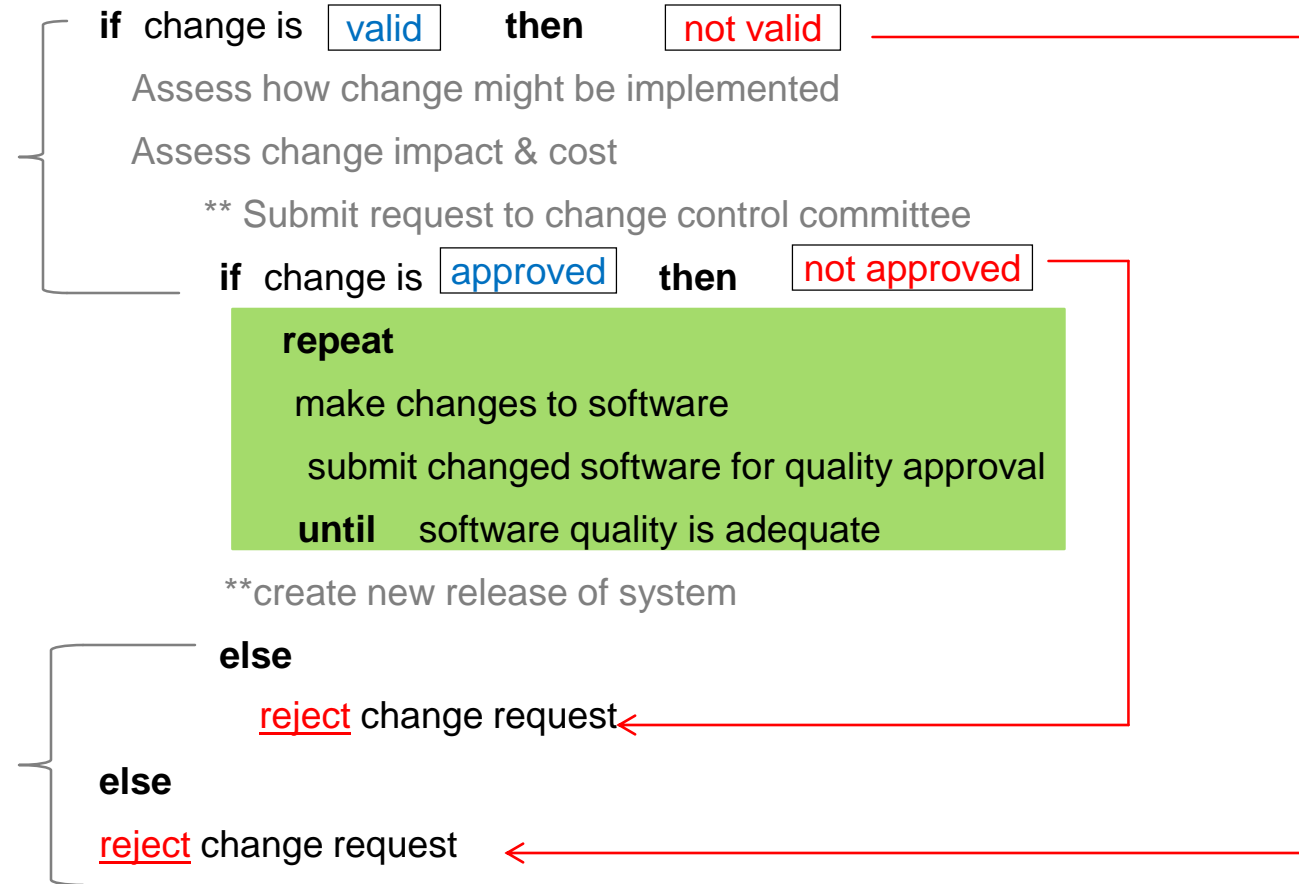
SCM CONCEPT
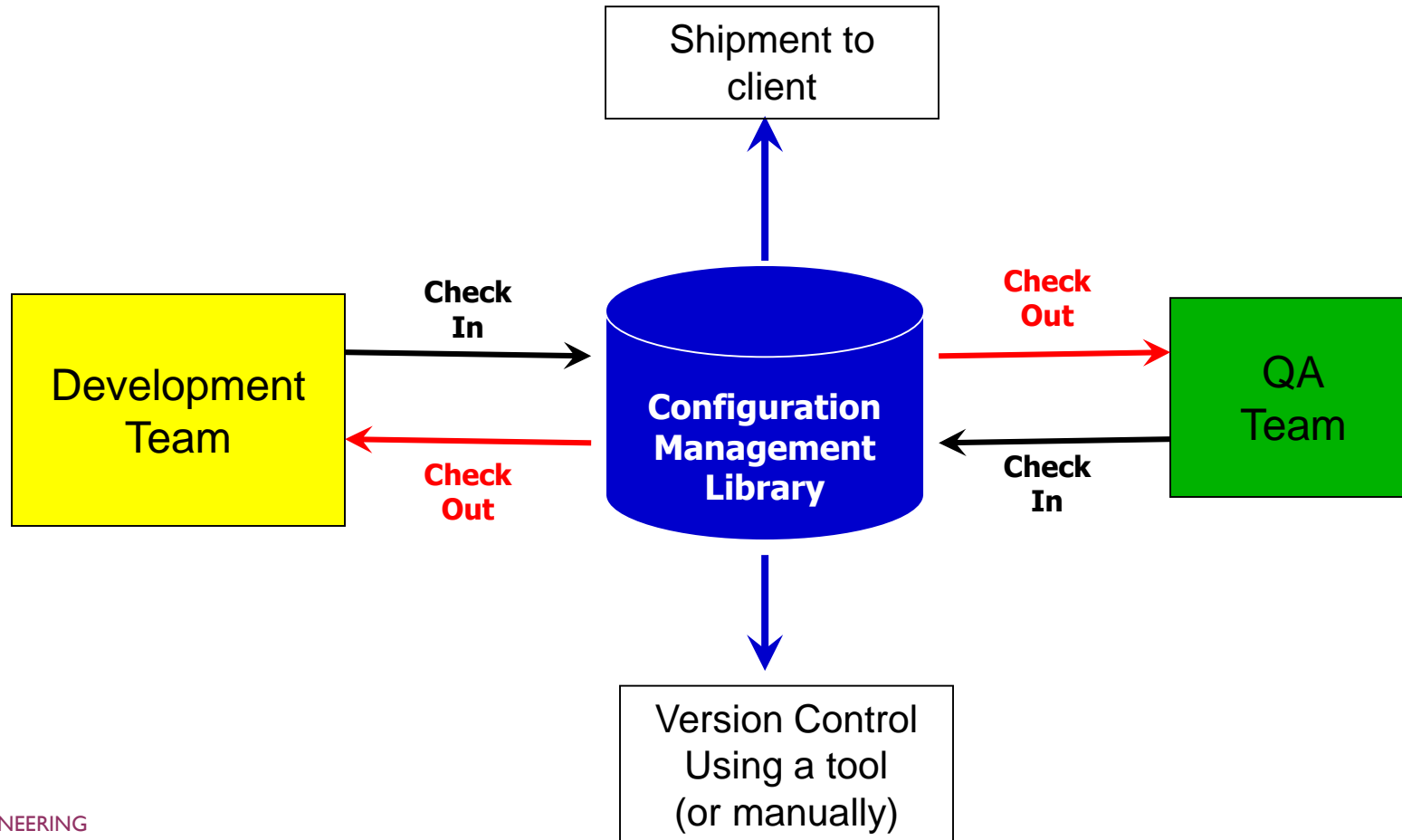
SCM ACTIVITIES

1. **Identify** the changes

2. **Control** the change

3. **Ensure that change is** properly implemented

4. **Report changes** to others involved in the project

CHANGE MANAGEMENT

- **Software systems are subject to continual _change requests_**
  - **From users**
  - **From developers**
  - **From market forces**

- **Change management** **is concerned with keeping track of these changes and ensuring that they are implemented in the most cost-effective way**

Software   Engineering

CHANGE MANAGEMENT PROCESS

if change is [valid] **then** [not valid]

Assess how change might be implemented

Assess change impact & cost

** Submit request to change control committee

if change is [approved] **then** [not approved]

> **repeat**
>
> make changes to software
>
> submit changed software for quality approval
>
> **until** software quality is adequate

**create new release of system

else

reject change request

else

reject change request

42

SOFTWARE QUALITY ENGINEERING

SOFTWARE CHANGE STRATEGIES

- **The software change strategies that could be applied separately or together are:**

- **Software maintenance:** The changes are made in the software due to requirements.
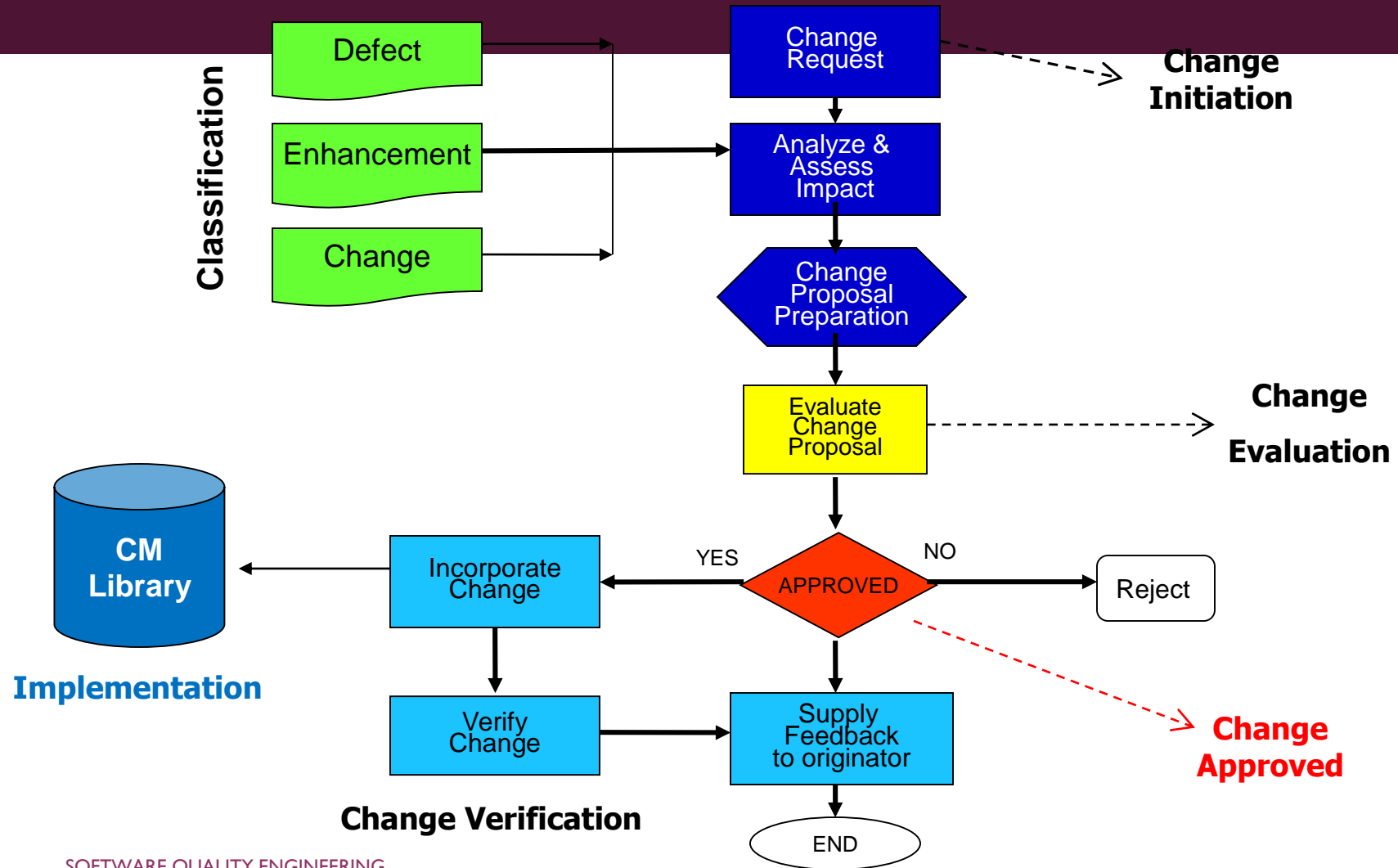
   Types of maintenance?

- **Architectural transformation:** It is the process of changing one architecture into another form.

- **Software re-engineering:** New feature can be added to existing system and then the system is reconstructed for better use of it in future.

TYPES OF MAINTENANCE

- Maintenance is defined as the process in which changes are implemented by either modifying the existing system's architecture or by adding new components to the system.

- **Corrective Maintenance**: Means the maintenance for correcting the software faults.

- **Adaptive maintenance**; Means maintenance for adapting the change in environment.

- **Perfective maintenance**: Means modifying or enhancing the system to meet the new requirements.

- **Preventive maintenance**: Means changes made to improve future maintainability.

CHANGE CONTROL

That is all