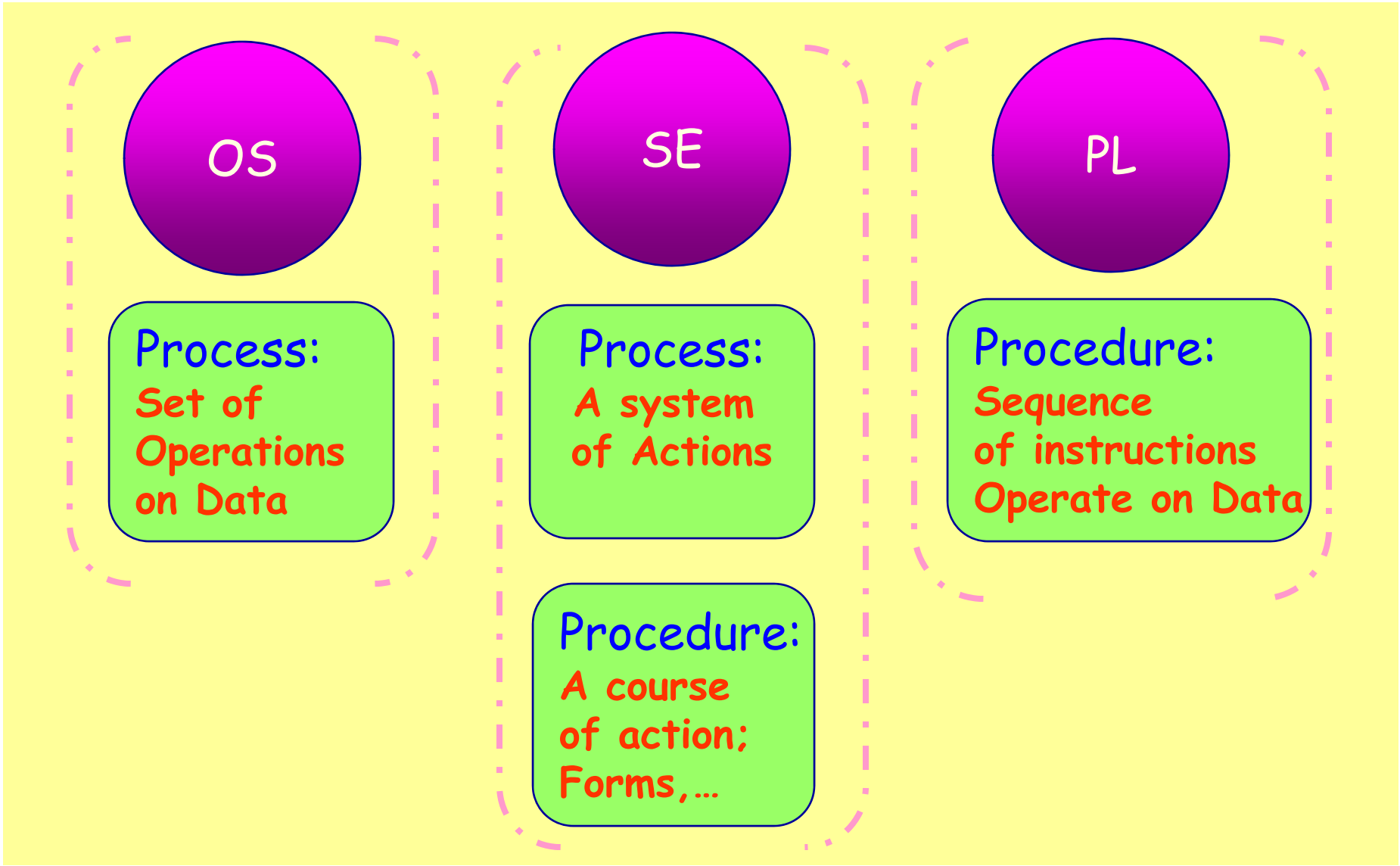
Software Development Process

Process vs. Procedure

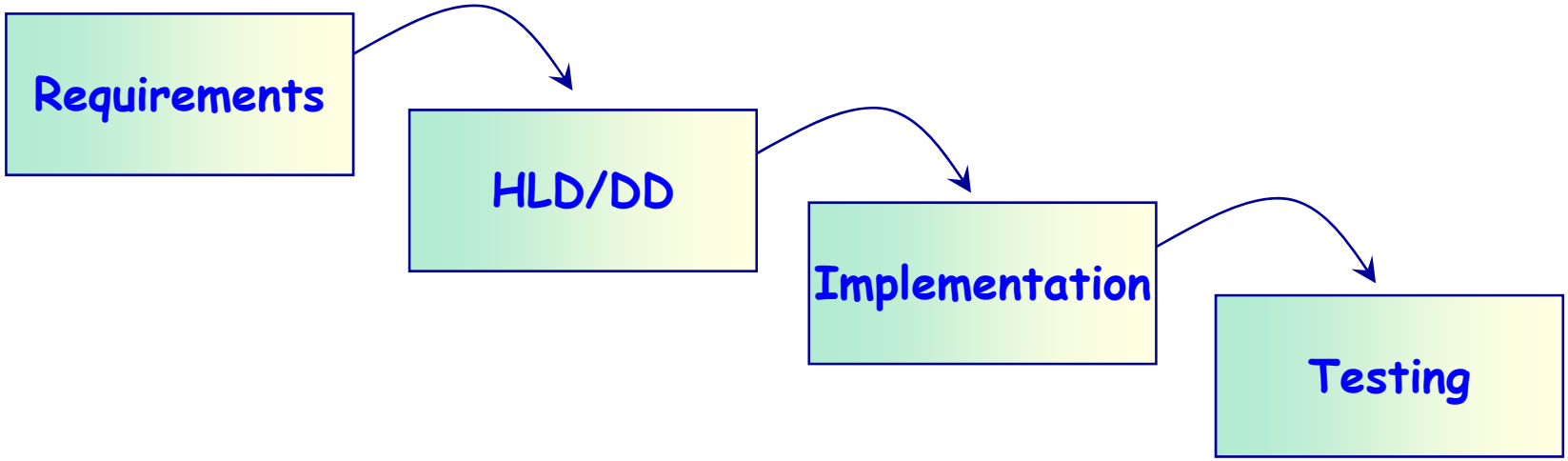


The Process

- A documentation for a sequence of steps that shall be followed when :
  - Starting/Closing the project
  - Writing/review requirement document
  - Writing/review High-level/detailed design document
  - Implementation and Code inspection
  - Writing test plan and Execute test cases

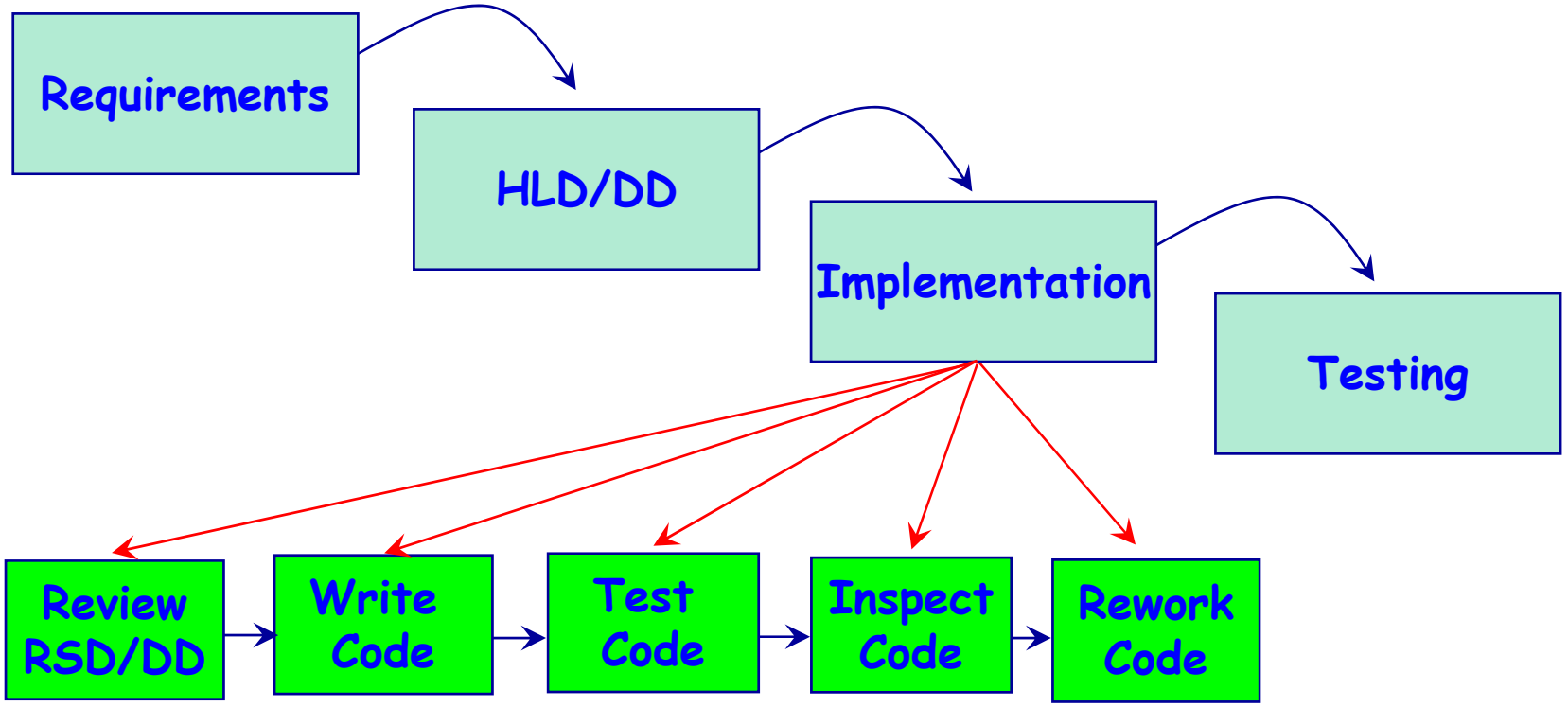
The Detailed Process

- For each phase in the software development, we need to have a detailed process that has been tailored mainly to that phase



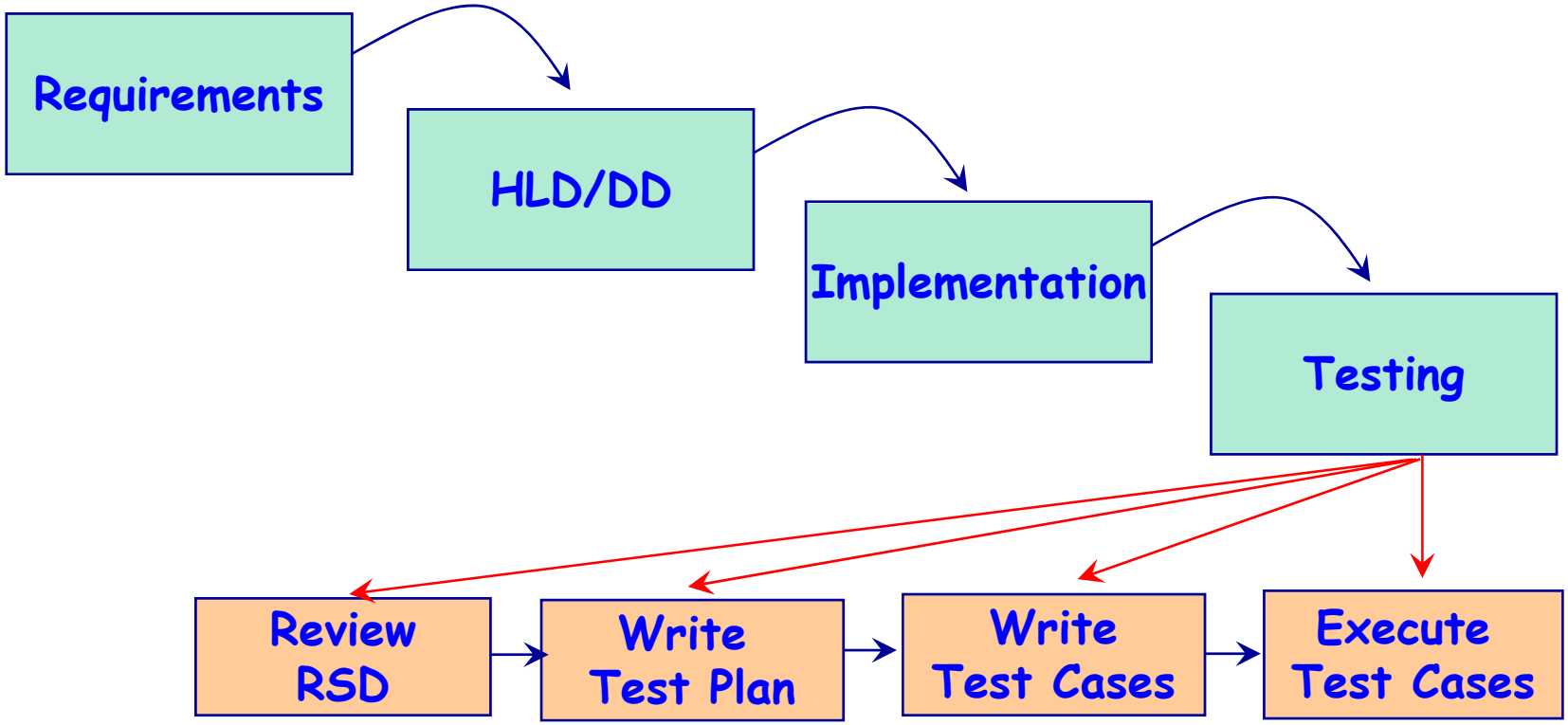
The Implementation Phase Process

- Here is the tailored process for the implementation phase:



The Testing Phase Process

- Here is the tailored process for the testing phase:



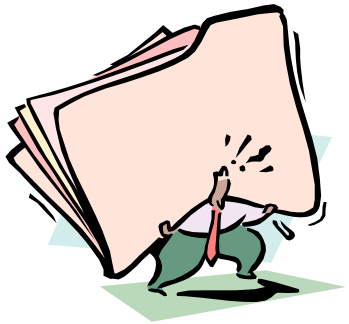
Tailoring the Process

How to tailor a process?

- First, each process shall have an owner.
- Adopt a standard process or borrow an existing one
- Add/Delete tasks or artifacts as you see appropriate but remember there are always risks for deleting an artifact and there is an overhead for adding one.
  - For example, you may not have formal requirement review or a test plan document

The Risk to Tailor a Process

- A process shall be documented, however to tailor a process may entail some risks; therefore the software organization shall provide general guidelines for process changes:



  - What are the artifacts or tasks
  - Who relies on these artifacts
  - What are the risks of deleting or modifying a task or an artifact
  - Guidelines for tailoring the process to certain domains
  - Document the rational behind the changes

Software Development Process Artifacts

- What is the artifact?

- The software artifact is an item produced as a result of the process execution
- The artifact could be a document, code, test results, …
- Examples:
  - Source Code and executable image
  - Test Results
  - RSD
  - HLD, class-diagram, sequence-diagram
  - Test Plan

Documentation Plan

- Documentation plan is an essential part of the SW Development Plan and it provides answers to the following questions:

- What are the artifacts to be produced?
- Who produces these artifacts?
- Who uses these artifacts?
- What is the format of the artifact?
- Who is the owner of the artifact?

Documentation Release

- Like software, documentation shall be kept under configuration management.

- Ideally, the documentation and the software releases shall be kept synchronized

Artifact dependencies and Ownership

Artifact	Produced By	Used By	Owner
Requirement Document	Customer	System Engineer Developer Tester	Customer
HLD/DD Document	System Engineer	Developer	System Engineer Process owner
Source Code	Developer	Developer Tester	Developer Process owner
Testing Plan	Tester	Tester	Tester Process Owner
⋮	⋮	⋮	⋮

Common Artifacts in the SW Development

Artifact	Description
Statement of Work	•The requirements for the project and the process •What is the outcome of the project
Requirement Specifications	•Technical specifications (functions, performance, standards met, etc.) of the final product
High-Level and Detailed Design Specification	•Modules and components of the system and how they interact •What are the software development files that will be implemented
Implementation	•What are the files that will be produced •Procedure for the build •End User Documentation •Maintenance Documentation •Installation Guide
Testing	•Test Plan •Test Cases •Test Results

How to honor the Software Process?

- **The process has to be practiced in order to ensure the quality control and quality assurance for the software system.**

- Reviews and Audits are the means by which we ensure the integrity and health of the process.
- Reviews are conducted by peers and managers
- Audits are conducted by independent organization

Reviews and Audits

Why we need reviews?

- **We can check a program for quality by testing, …**

- But:
  - How do we check that the test plan being used for testing has the right test cases?
  - How we check the requirement specification documents for defects?
  - How we check the design document for design errors?

Reviews

- **Reviews are the most effective method to improve quality by identifying defects.**

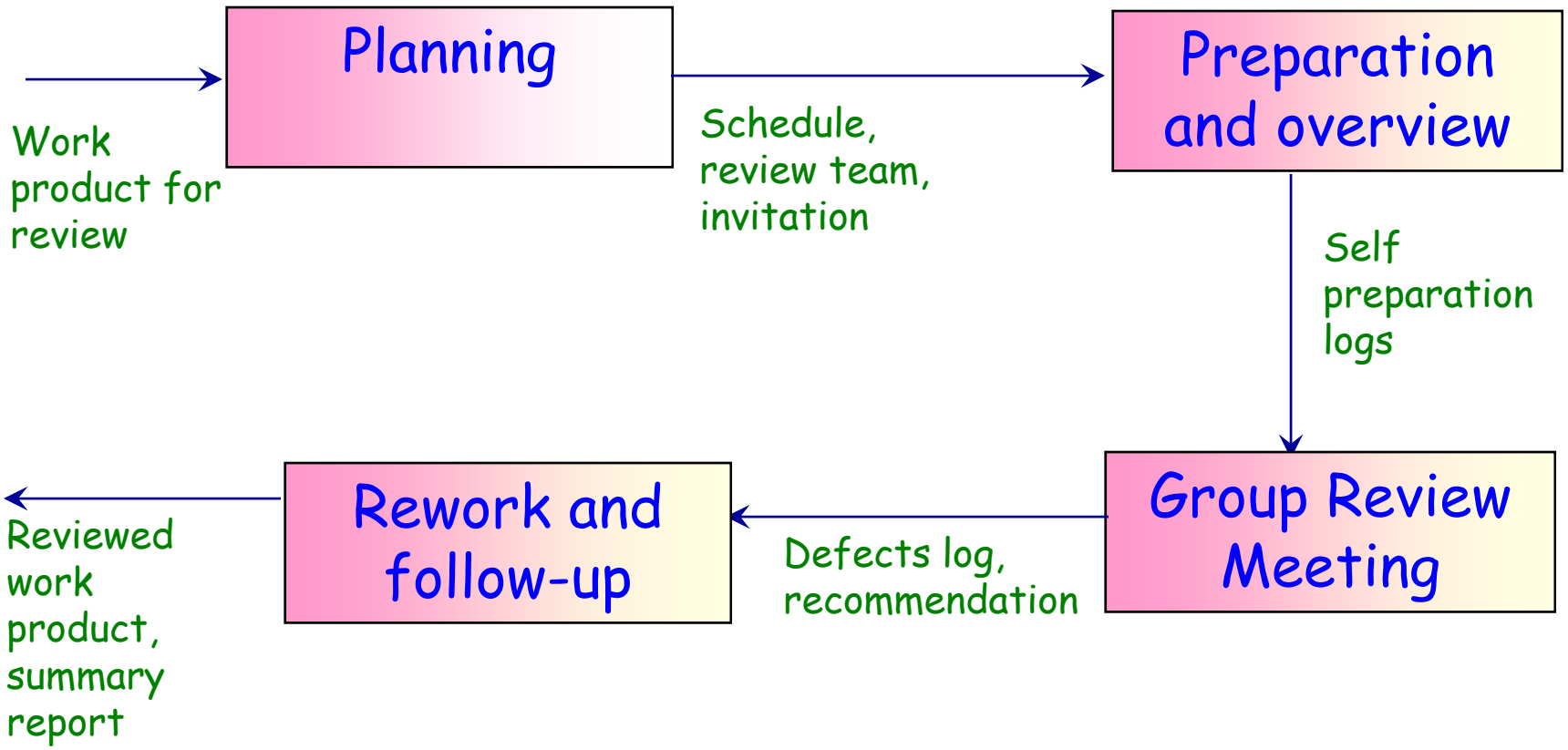
- Applied in
  - Design document
  - Test plan
  - Code "inspection"

- Can be used to
  - Track progress
  - Prevent defects discovered by Customer
  - Improve productivity by finding defects in effective ways

Reviews

- **Reviews can come in different forms:**
  - Formal Group Review or inspection
  - Desk Review, one person involved in the review
- **Reviews are conducted by**
  - Technical people for technical people
- **Reviews intent**
  - To identify problems NOT to resolve them

The Review Process



The Planning Phase of the Review Process

- **Objective:**
  - Prepare for the review by selecting the group review team and schedule the review
- **Participants:**
  - Author
  - Project Manager
  - Moderator
  - Reviewers

The Overview and Preparation Phase

- **Objective:**
  - Deliver and explain the work package to the reviewers
- **Work package includes:**
  - **Work product**
  - **Specifications**
  - **Checklists**
  - **Standards**
- **Outcome:**
  - The self-review forms are completed; defects, actual time spent are recorded

The Group Review Meeting Phase

- **Objective:**
  - Come up with a final defect list that agreed upon by the whole group
- **Logistics**
  - **Moderator is controlling the Review Report**
  - **Moderator logs actual time spent for preparation**
  - **Start/End time of the session**
  - **Concerns/issues from team members**
- **Two Roles have to be filled from the reviewers:**
  - Scribe
  - Reader

The Rework and Follow-Up Phase

- **Objective:**
  - The author performs the rework to fix the defects raised during group review meeting
- **Moderator**
  - Has to decide on fix reworks
  - Has to decide whether a re-review meeting is required to go over the fixes.
  - Has to ensure that all data and review results are recorded and must be submitted along with group review summary to the project leader

Guidelines for Reviews in Projects

- Not all products shall go under a formal group review because :

  - Reviews are expensive
  - Reviews are overhead activities

Review Types

- **What are the types of the reviews?**

1. Technical
   - Peer Reviews
2. Managerial
   - Status Reviews
   - Quality Gates Reviews

Guidelines for RSD Reviews

Work Product	*Focus*	*Entry Criteria*	*Participants*
Requirement Specification Document	• **Requirements meet customer needs** • **Requirements are implementable** • **Omissions, inconsistencies, and ambiguities in the requirements**	• **The Document conforms to the standards**	• **Customer** • **Designers** • **Testers** • **Installation team members** **User documentation author**

Guidelines for High-Level Design Reviews

Work Product	*Focus*	*Entry Criteria*	*Participants*
High-Level Design	• **High-level design implements the requirements** • **The design is implementable** • **Omissions and other defects in the design**	• **The document conforms to standards** • **The requirements have been reviewed and finalized**	• **Requirements Author** • **Detailed Design author** • **Developer**

Guidelines for Code Reviews

Work Product	*Focus*	*Entry Criteria*	*Participants*
Code	• Code implements the design • Code is complete and correct • Defects in code	• The code compiles and passes style and other norms	• Designer • Tester • Developer

Guidelines for Test Cases Reviews

Work Product	*Focus*	*Entry Criteria*	*Participants*
System Test Cases	• The set of test cases checks all conditions in the requirements • System test cases are correct • Test cases are executable	• Requirements have been base lined • System test plans is consistent with the standards	• Requirements author • Tester • Project Leader

Guidelines for High-Level Design Reviews

Work Product	*Focus*	*Entry Criteria*	*Participants*
Project Management Plan	**PProject management plan meets project management and control needs** **• Completeness** **• Project management plan is implementable** **• Omissions and ambiguities**	**• The project management plan follows the standard template**	**• Project leader** **• Another Project Leader**

Data Collection During Reviews

- **Reviews are mainly human processes, therefore:**
  - We need to record the data
- **What to record ?**
  - Effort Data
  - Defect Data
- **Why to record ?**
  - Analyze the effectiveness of the reviews
  - Construct the Review Capability Baseline

Data Collection During Reviews

- **Reviews are mainly human processes, therefore:**
  - We need to record the data
- **What to record ?**
  - Effort Data
  - Defect Data
- **Why to record ?**
  - Analyze the effectiveness of the reviews
  - Construct the Review Capability Baseline

Data Collection Forms in Technical Reviews

- Forms used for data collection during technical reviews:

  - Self-Preparation log
  - Group Review Meeting Log
  - Group Review Summary report

Self-Preparation Log

- **Objective :**
  - **Record all defects**
  - **Record Effort Spent**
- **The Form:**

Project Code:
Work Product ID:
Reviewer Name
Effort Spent for Preparations (hours):
Issue list:

SI	Location	Description	Criticality

Group Review Meeting Log

- **Objective :**
  - **Record defects agreed upon by the team**
  - **Record Effort Spent by the team**
- **The Form:**

Project Code:
Moderator:
Author:
Date:
Effort Spent on review meeting (hours):
Defects to be closed by (date):
Defect List:

Meeting Type:
Scribe:
Reviewers:
Observers:
Work Product ID:

SI	Location	Description	Reviewer	Criticality

Group Review Summary Report

- Objective :
  - To analyze the effectiveness of the review
- The Form:

Group Review Summary Report

- The Form:

Project Work Product Type Size of Product Moderator Reviewers Author	
Effort (Person-Hours) Overview meeting Preparation Group review meeting	
Defects Number of critical defects Number of major defects Number of minor defects Number of defects found during preparation Number of defects found during group review meeting	
Result	***Moderator reexamination***
Recommendations for next phase	
Comments (Moderator)	
Prepared by:	***Date:***

The Review Capability Baseline

- The effectiveness of the review process depends on how well the process has been executed
- How does the project manager or the moderator evaluate whether a review has been effective?

- The statistical process control (SPC) can be used to monitor and control the reviews

The Review Capability Baseline

- How can the SPC be applied to monitor the reviews?
  - Project managers must
    1. Identify Performance parameters
    2. Control limits for performance parameters
    3. Monitor actual performance
    4. Determine effectiveness through
       a) Control charts (plot performance parameters)
       b) Control limits (see whether parameters within the ranges); EASY to APPLY

The Review Capability Baseline

- Examples of Performance Parameters
  - Coverage rate during preparation
  - Coverage rate during group review meeting
  - Defect density for minor defects
  - Defect density for major defects
- Control limits are determined from past data and group review capability baseline

The Review Capability Baseline

Review Item	Preparation Coverage Rate	Group Review Coverage Rate	Minor Defect Density	Major Defect Density
Requirements	5-7 pages/hour	0.5-1.5 defects/page	0.1-0.3 defects/page	
High-Level design	4-5 pages/hour (200-250 specification statements/hour)	0.5-1.5 defects/page	0.2-0.6 defects/page	
Detailed design	3-4 pages/hour (70-100 specification statements/hour)	0.5-1.5 defects/page	0.2-0.6 defects/page	
Code	160-200 LOC/hour	110-150 LOC/hour	0.01-0.06 defects/LOC	0.01-0.06 defects/hour
Integration test plan	5-7 pages/hour	0.5-1.5 defects/page	0.1-0.3 defects/page	
Integration test cases	3-4 pages/hour			
System test plan	5-7 pages/hour	0.5-1.5 defects/page	0.1-0.3 defects/page	
System test cases	3-4 pages/hours			
Project management and configuration management plan	4-6 pages/hour	2-4 pages/hour	0.6-1.8 defects/page	0.1-0.3 defects/page

Charting the Reviews

- Monitoring the reviews using the SPC Charting tool:
  - The SPC tool is a spreadsheet that has the review capability baseline built into it
  - If process changes, then we need to change the control limits that affect the control charts
  - Change the control limits based on the past 10-15 performance data points

Analysis Guidelines for the Reviews

- If the number of the defects found during the review is within the range given in the baseline, the review is considered effective, and the exit criteria is satisfied, ELSE :
  - The moderator or the project leader needs to determine the cause and take preventive and corrective actions
  - There are two sets of guidelines, one if the defect density below the limit and another one when defect density above the limit

Analysis Guidelines for the Reviews

- **If defects found are less than norms:**

Possible Reason	Actions to Consider
Work product was very simple	•Convert group review of similar work product to one person review (desk review) •Combine reviews
Reviews may not be thorough	Check coverage rate; if too low, reschedule a review, perhaps with a different team
Reviewers do not have sufficient training on group reviews or experience with the reviewed material	•Schedule or conduct group review training •Re-review with a different team
Work product of very good quality	•Confirm this fact by coverage rate, experience of the author, reviewers, and so on; see if this quality can be duplicated in other parts of the project. •Revise defect prediction in downstream activities; see if there are general process improvement lessons

Analysis Guidelines for the Reviews

- **If defects found are more than norms:**

Possible Reason	Actions to Consider
Work product is of low quality	•Examine training needs for author •Have the work product redone •Consider reassigning future tasks (easier tasks to the author)
Work product is very complex	•Ensure good review or testing downstream •Increase estimates for system testing •Break the work product into smaller components
There are too many minor defects and two few major defects	•Identify causes of minor defects; correct in the future by suitably enhancing checklists and making authors aware of the common causes •Reviewer may have insufficient understanding of the work product. If so, hold an overview meeting or have another review with different reviewers
Reference document against which review was done is not precise and clear	•Get the reference document reviewed and approved
Reviewed modules are the first ones in the project	•Analyze the defects, update the review checklist, and inform developers. Schedule trainging.

The NIH and NAH Software Syndromes

- In the old days, the rule for software tool usage
  - NIH : Not Invented Here
- In the current days, the rule for software process and review practices
  - NAH : Not Applicable Here

Status Reviews

- This review focuses on the project progress
- Planned vs. actual progress
- Status on issues raised in previous status review meeting
- Identify and address concerns that may have negative impact on the project
    1. Schedule
    2. Cost
    3. Quality

Conducting the Audit

- What to audit and why?
  - The auditors focus on whether the defined process is being followed in the project
  - It discovers problems but doesn't solve them
- Who execute it?
  - EXPENSIVE Independent Auditors

Conducting the Audit

- When to audit?
  - Regularly
  - Audits are sign of healthy projects and organizations
  - Audits shall be thought of as preventive measures rather than reactive measures

Conducting the Audit

- The audit methodology:
  - Verify stakeholders are practicing their processes
  - Look how an activity is done
  - Verify the output (artifacts) of the activity
  - Number of questions are asked from the audit checklist

The Audit Checklist

1. Is the project plan documented in the standard project plan template?
2. Has the project plan been group reviewed?
3. Has the project plan been approved and baselined, and is it under configuration management?
4. Is there a signed contract?
5. Have commitments to customer or other group been approved?
6. Is there an estimated effort for the project that is based on historical data?
7. Have the effort estimates and the schedule been reviewed?

The Audit Checklist

8. Is the quality plan complete, and has it been reviewed?
9. Is the life cycle in the project identified and documented?
10. Are personnel identified and the responsibility for each work element defined and tracked?
11. Are deliverables to the customer, including the documentation, clearly identified?
12. Are risks and risks mitigation plans identified and properly documented?
13. Are reviews, progress reporting, tracking, and approval mechanisms identified?

The Audit Follow-Up

- The Audit intent is to Audit the applied process rather auditing the people
- If evidence suggests that the approved processes are not followed, a noncompliance report(NCR) will be issued
- People involved with this process shall not be punished because of the NCR report, rather it shall be an opportunity to take corrective actions.
- Identifying the noncompliance is the goal of these audits

Software Configuration Management

Software Configuration Management

What is CM?

Software Configuration Management is the process of identifying, organizing and managing modifications to software

Software Configuration Management

Configuration Baseline:

- ✓A configuration baseline is a named collection of software components and supporting documents that is subject to change management and is upgraded, maintained, tested as a unit

Software Configuration Management

Two classes for Baselines:

- ✓External product releases
- ✓Internal testing releases

For example:

- ✓The development organization may release a configuration baseline to the test organization
- ✓A project may release a configuration baseline to the user community for Beta testing

Software Configuration Management

Three Levels of Baseline Releases

- ✓Major : represents new generation of product
- ✓Minor : Same basic product with enhanced features
- ✓Interim or Patch : transient, "Bug fixes"

Each Level corresponds to a numbered identifier

- ✓N.M.X
- ✓Where N : major, M: minor, X: interim

Release Focus:

- ✓Major and Minor for external product release
- ✓Interim for development configuration

Software Configuration Management

Choosing a Method of Version Numbering:

- ✓ The basic rules for any system of version numbering:
  - There shall be ONE system
  - The numbers shall always go up
    - ☞ It doesn't *really* matter if 2.1 is a big jump and 2.0.005 is a small jump but it does matter that 2.1 is more recent than 2.0.005.

Software Configuration Management

The most common technique for version numbering is the "major level," "minor level," "patch level" version numbering scheme.

- ✓The first number is the major number and it signifies major changes or rewrites.
- ✓The second number is the minor number and it represents added or tweaked functionality on top of a largely coherent structure.
- ✓The third number is the patch number and it usually refers to releases fixing bugs.

Software Configuration Management

Other version numbering systems worth looking at:

- ✓Linux kernel version numbering
- ✓Mozilla milestones

Software Configuration Management

Linux kernel version numbering:

- ✓The Linux kernel uses a versioning system where any odd minor version number refers to a development or testing release and any even minor version number refers to a stable version.
  - ☞Under this system, 2.1 and 2.3 kernels were and always will be development or testing kernels and 2.0, 2.2. and 2.4 kernels are all production code with a higher degree of stability and more testing.

Software Configuration Management

Mozilla milestones

- ✓Mozilla's version numbering structure has historically been made up of milestones.
  - ☞From the beginning of the mozilla project, the goals of the project in the order and degree to which they were to be achieved were charted out on a series of road maps.
  - ☞Major points and achievements along these road-maps were marked as milestones.
  - ☞Although Mozilla was built and distributed nightly as "nightly builds," on a day when the goals of a milestone on the road-map had been reached, that particular build was marked as a "milestone release."

Software Configuration Management

Documentation, Why we need it?

- ✓A piece of software without documentation is not worth using.

Software Configuration Management

What people expect from documentation?

- ✓ **Man pages (The fat manuals)**
- ✓ **Command line accessible documentation**
  - ☞ Users want to be able to type "man yourprojectname" end up with a nicely formatted man page highlighting the basic use of your application..
- ✓ **Users expect the following files**:
  1. *README or Readme: basic installation, compilation, and use*
  2. *INSTALL or Install: how to build and install*
  3. *CHANGELOG: describes what has been changed*
  4. *FAQ*

Software Configuration Management

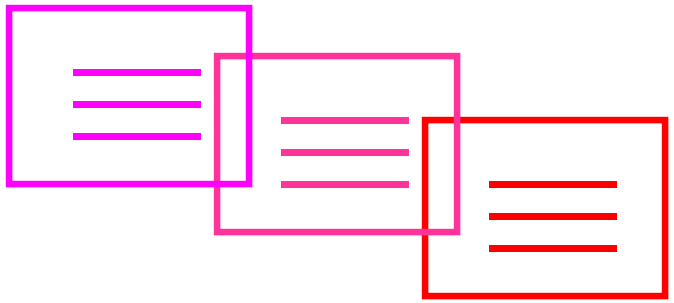
Configuration Items:

- ✓Requirement document
- ✓Design Document
- ✓Test Plan
- ✓Development tools
- ✓Emulators/Simulators
- ✓Management/Monitoring tools
- ✓operating system
- ✓Data/hardware used
- ✓Libraries
- ✓Test data

Software Configuration Management

Software Products:

- ✓Are produced in different releases
- ✓Each release is associated with certain release/version of a configuration item
- ✓New features are added to the software product and existing features could be replaced or removed
- ✓Bottom line, software changes … for all kind of reasons and we need to track it



Software Configuration Management

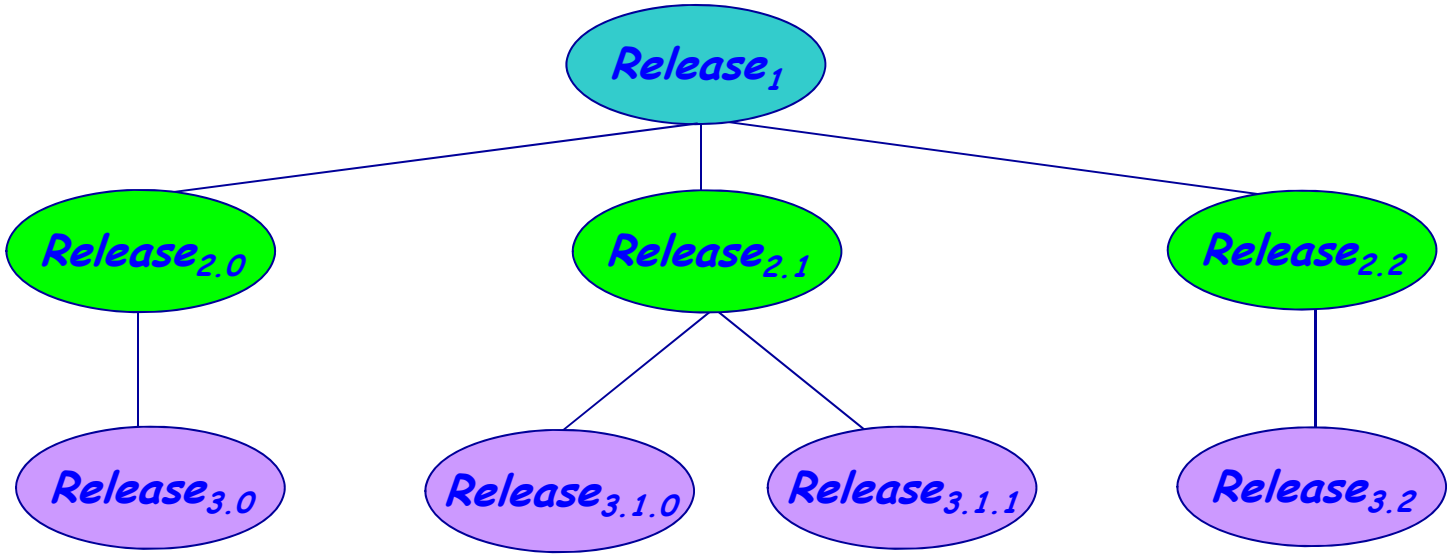
Rational behind software changes:

- ✓Different versions of the software product for different Customers
- ✓But over time customers request different enhancement for their versions of the software product.
- ✓So overtime different versions of the software product will less and less in common.
- ✓Finally, we end up maintaining different versions of the same product.

Software Configuration Management

Rational behind software changes:

✓Over time, we end up with a monster of "different versions for the software system" that is expensive to maintain



Software Configuration Management

Why we need the CM?

- ✓To preserve the **integrity** of the product, that is to say, we need to deliver a product that has all the required configuration items in the right release/version or otherwise the product may NOT work as intended

Software Configuration Management

Does software age?

- ✓Yes : technological advances in the computer hardware will make our software system suffer from the aging symptoms.
- ✓No : because software unlike hardware doesn't wear out

Software Configuration Management

How to cope with software aging?

- ✓As we get older, our visits to doctors' office becomes more frequent to ***maintain*** our health and well being.
  - ✓Benefits: less pain + few more years to live
  - ✓Cost: insurance bill becomes bigger
- ✓The same for software:
  - ✓Benefits: few more releases to live
  - ✓Cost: increased effort=time+money

Software Configuration Management

CM bottom line:

Maintain the integrity of the product by controlling the changes to the product

Software Configuration Management

Causes for Loss of Integrity:

- ✓ *Concurrent:*
  - ➢ *Uses : read/write*
  - ➢ *Users*
- ✓ *Multiple Software Releases for different customers*
- ✓ *Different versions for Tools/Compilers*
- ✓ Software artifacts, design, user guide, are not in sync with the code changes
- ✓ Not knowing what version of the software to use when executing modification requests

Software Configuration Management

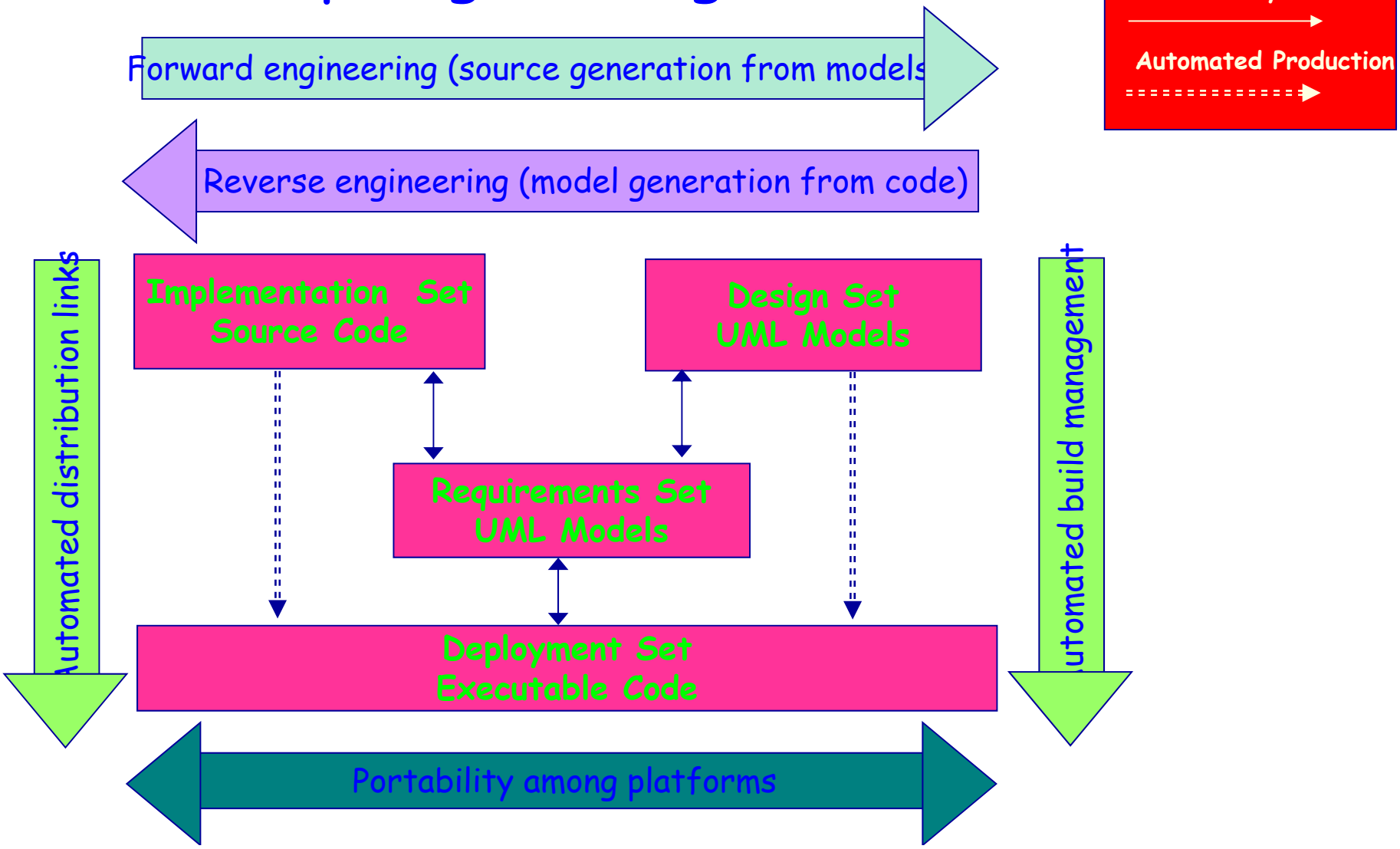
The software project environment:

- ***Round-Trip Engineering***: *Tools must be integrated to maintain consistency and traceability.*

- ***Change management*** *must be automated and enforced to manage multiple iterations and to enable change freedom*

Software Configuration Management

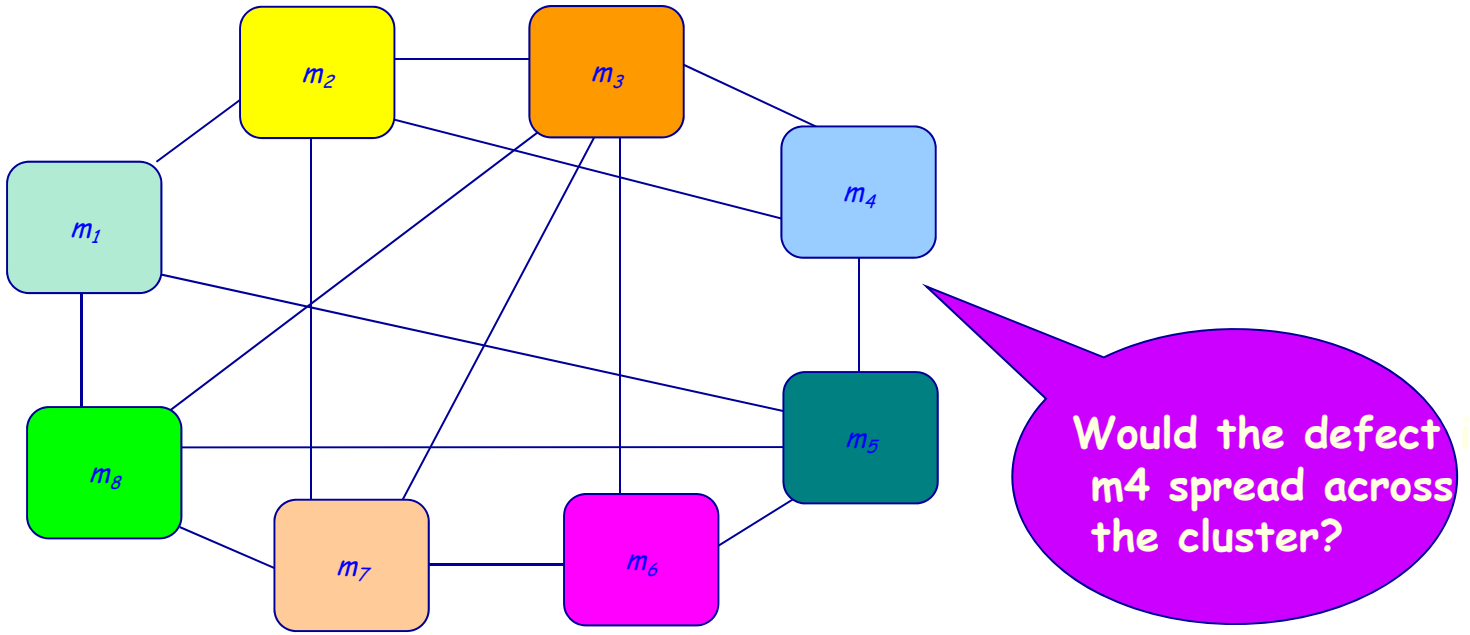
Round-Trip Engineering:



Software Configuration Management

Root cause for CM problems:

✓If a software system is composed of the following modules: m_1, m_2, ..., m_8, then there will be a probability that a change in module m_2 will result in the introduction of defect in m_4



Software Configuration Management

Configuration Management - Scenarios

✓ ***More than one change request on module m***

The manager assigns the change requests to different people to modify module m. X saves it inadvertently overwriting the changes Y made a day earlier

Software Configuration Management

Configuration Management - Scenarios

✓ ***Key person leaving***

The key person leaves for emergency shortly before the deadline.The team members try to find out the file he was working on and come up with umpteen versions. Finally starting with some Version, one of the team-mate develops and tests the unit redoing the work that the key person has almost finished before leaving.

Software Configuration Management

Configuration Management - Scenarios

- ***Releasing an older version***

The team finishes development on time and the final testing with no bugs. The product was released to the customer for implementation.

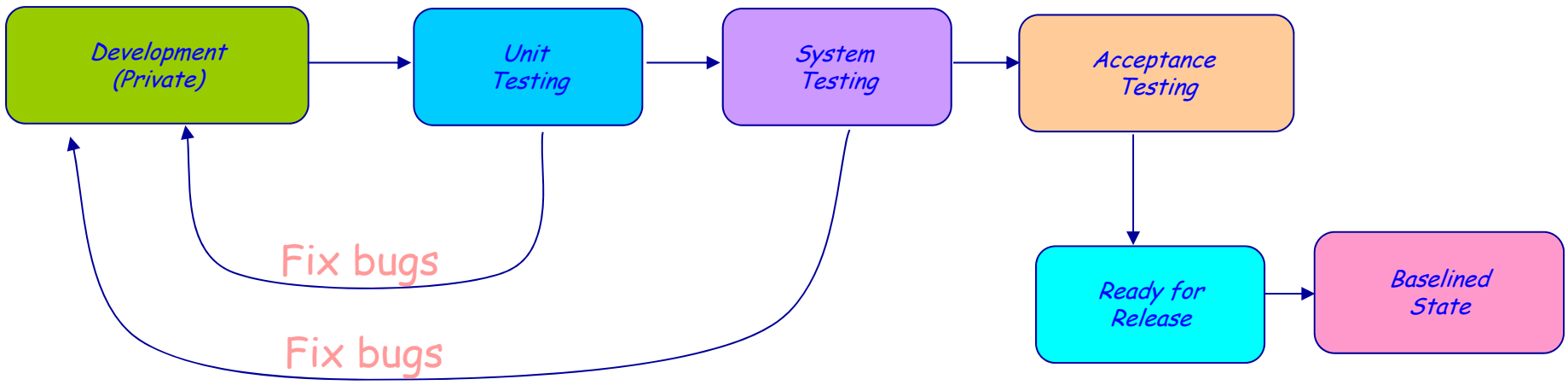
The next day they receive several angry emails and complaints. After frantic effort on the teams part, the cause was found that the release version of the software contained an older version of a key component.

Software Configuration Management

CM Concepts

✓ Primary objective of CM is to manage the evolving configuration of the software system.

Stages of Program evolution



Software Configuration Management

CM Functionality Requirements

✓ ***Give the states of the programs***. Need this to decide when to start testing or when to release the software.

✓ ***Give the latest version of the program.*** So that modification can be carried out in the latest version of the program. Otherwise earlier changes will be lost.

Software Configuration Management

CM Functionality Requirements

✓ ***Handle concurrent update requests.*** One of the changes could potentially overwrite the other change.

- Access control should be used so that only one person can make changes to a program at a time.
- If multiple parallel changes are allowed, reconciliation procedures should be implemented to ensure that all changes are reflected in the final version.

✓ ***Undo a program change.*** A change is made to a program but later a need arises to reverse this change request.

Software Configuration Management

CM Functionality Requirements

✓ ***Prevent unauthorized changes or deletions.***
Access control mechanisms are required to disallow unapproved changes.

✓ ***Provide traceability between requirement change requests and program changes***. A mechanism to track change requests that can specify all programs to be changed as well as the state of each program should be established.

Software Configuration Management

CM Functionality Requirements

- ✓ ***Undo requirement change.***

- ✓ ***Show associated changes.***

Suppose a bug is found in a program, and it is suspected that this bug came from the implementation of a change request. It is desirable to review all changes made as a result of that change request.

Software Configuration Management

CM Functionality Requirements

✓ ***Gather all sources, documents and other information for the current system.***

File corruption or system crash or a change to an existing system might make it essential to obtain all the information.

Software Configuration Management

CM Mechanisms

- ✓ Conventions for naming and organization of files
- ✓ Version control
- ✓ Change request traceability
- ✓ Access control
- ✓ Reconciliation procedures
- ✓ Modification login program

Software Configuration Management

Naming Program files

- ✓ Quick access – Naming according to convention and keeping the files in specific directories
- ✓ Nature of file contents without looking at the files
- ✓ Segregating by states and keeping them in specific directories help developers to identify the program state easily

Software Configuration Management

Version Control

- ✓ Many tools are available
- ✓ Helps in preserving the older versions whenever they are changed

Software Configuration Management

Change request traceability

✓ A change request traceability mechanism provides mapping from a requirement change request to subsequent changes in the programs.

✓ Modification log is useful to trace a change back to the change request

Software Configuration Management

Access Control and Reconciliation Procedures

✓ Ensures that only authorized people can modify certain files and that only one person can modify a file at any given time.

✓ Reconciliation procedures specify how two changes made independently to a program can be merged to create a new version that reflects both.

Software Configuration Management

CM Process

- ✓ The CM process defines the sequence of activities that must be performed in support of the CM Mechanisms.

- ✓ CM process main activities:
  1. Planning and setting up Configuration Management
  2. Perform configuration Control
  3. Status Monitoring and Audits

Software Configuration Management

Planning and setting up CM

✓ Planning for CM involves identifying the configuration items and specifying the procedures to be used for controlling and implementing changes to them.

✓ Types of items are identified but not listed in detail. This omission reflects the fact that some items may not be known during CM planning

✓Establish Naming conventions and Version numbering

✓Set the Directory structure for managing the states of the program using either tool or explicit handling

Software Configuration Management

Planning and setting up CM

✓ The Configuration Controller or Project Manger does the CM planning.

✓ Begins only when the project has been initiated and the operating environment and requirements specifications are clearly documented

Software Configuration Management

Planning and setting up CM – Activities:

1. Identify configuration items, including customer-supplied and purchased items.
2. Define a naming and numbering scheme for the Configuration items.
3. Define access restrictions
4. Define change control procedures
5. Identify and define the responsibility and authority of the CC or Configuration Control Board (CCB)

Software Configuration Management

Planning and setting up CM - Activities

6. Define the directory structure needed for CM
7. Define a method for tracking the status of configuration items
8. Define a backup procedure
9. Define a reconciliation procedure, if needed
10. Define a release procedure
11. Define an archival procedure
12. Identify points at which the configuration items will be moved to the baseline

Software Configuration Management

Planning and setting up CM – when to have CCB?

- When there are large teams or when two or more teams/groups are involved in the development of the same or different portions of the software, then the CCB(Change control Board) is necessary

  - ★ the CCB includes representatives from each of the teams
  - ★ The CM plan must clearly define the roles and responsibilities of the CCB

Software Configuration Management

Planning and setting up CM – Procedures and policies

- Procedures and policies to control:
  - ✓ ***Normal Changes*** – Managed through library Mechanism and directory structure
  - ✓ ***Requirement Change Requests*** – Tracked through spreadsheets that lists all items that must be changed as well as the directory for each item(thereby giving its state)

Software Configuration Management

Planning and setting up CM – Reconciliation

- Reconciliation procedure for concurrent updates:
  - ✓ ***Concurrent updates*** – One possible procedure is to state the differences between the original version and the new versions will be examined, and the changes in the version having fewer changes will be merged in the other version.
  - ✓ If changes affect different part of the program merging is straight-forward. Otherwise programmer must review the overlap and then accommodate both changes.

Software Configuration Management

Perform Configuration Control

- ✓ Done during the Execution phase of the project. Two configuration control activities:

1. First one deals with managing the state transitions of programs(and documents); this state transition involves moving items from one directory to another; official node and personal development node.

2. Second one deals with managing the change requests that must be implemented

Software Configuration Management

Perform Configuration Control

- ***State transition management*** - Involves moving the items from one directory to another when the state changes and then creating versions when changes are made
- ***CM Tools*** - Employ check-in/check-out procedure for controlling access and handling version control
- ***Controlled Environment*** - When a program is in CE it cannot be modified, even by the original author without authorization.

Software Configuration Management

Perform Configuration Control

- ✓ An item is modified after it has been checked out.
- ✓ Checks are done to ensure that the changed item is suitable before it is checked back in with regard to other members using it.
- ✓ When an item is checked back into the controlled environment, the older copy is not destroyed, instead a new version is created
- ✓ ***Modification Log*** – Identifies the start and end of a change and includes a reference to the change request that prompted it

Software Configuration Management

Perform Configuration Control & the CM tools

- ✓ Majority of CM tools provide support for:
  1. Checking in
  2. Checking out
  3. Version maintenance
  4. Creation of modification log

Software Configuration Management

Perform Configuration Control

✓ A change request is first analyzed by performing an impact analysis.

✓ Determines the programs and documents that need to be changed and the cost and schedule implications of making the change

✓ Once the change is approved by project leader and CC, all programs and documents identified in the impact analysis must be changed appropriately.

Software Configuration Management

Perform Configuration Control

Activities of implementing a change request

1. Accept the change request(with impact analysis)
2. Set up a tracking mechanism
3. Check out configuration items that need to be changed
4. Perform the changes
5. Check in the configuration items
6. Take the item through its life cycle

Software Configuration Management

Status Monitoring and Audits

✓ A configuration item can exist in one of several states

✓ Set of possible states vary depending on whether the item is a program or document

✓ Important to represent the state accurately as they can lead to state-related mistakes

Software Configuration Management

Status Monitoring and Audits

State Related mistakes

- A program which has not been unit tested but moved to 'ready for release' state

- System failing to reflect the fact that a program has been checked out from 'baseline' to implement change request could lead to delivery without change

Software Configuration Management

Status Monitoring and Audits

✓ Regular status checking of the Configuration Items and also the change requests

✓ Report about discrepancies should be produced and all discrepancies should be resolved

✓ Checks to ensure that all modified items go through their full life-cycle before they are incorporated I the baseline

Software Configuration Management

Status Monitoring and Audits

Configuration Audit

✓ Ensures that the CM process of the project is indeed being followed

✓ The baseline for the system is also audited to ensure that its integrity is not being violated and that items are moved to and from the baseline in a manner consistent with the CM plans

Change Management

✓ Tracking changes in the technical artifacts is crucial to understanding the true technical progress trends and quality trends

✓Baseline configuration management techniques for technical artifacts - A late life-cycle activity

✓ In a modern process in which requirements, design, and implementation set artifacts are captured early – change management has become fundamental

Change Management

Software Change Orders

✓ Atomic unit of software work that is authorized to create, modify, or obsolesce components within a configuration baseline is called a software change order(SCO)

Change Management

Software Change Orders

✓ SCO are a key mechanism for partitioning, allocating, and scheduling software work against an established software baseline and for assessing progress and quality

Change Management

Software Change Orders

- ✓ SCO should be written against a single component so that it is easily allocated to a single individual

- ✓ ***SCO Fields***
  - ❖ Title
  - ❖ Description
  - ❖ Metrics
  - ❖ Resolution
  - ❖ Assessment
  - ❖ Disposition

The primitive components of SCO

Title:

Description

Name: _____ Date: _____

Project: _____

Metrics

Category: _____ (0/1 error, 2 enhancement, 3 new feature, 4 other)

Initial Estimate **Actual Rework Expended**

Breakage: _____ Analysis: _____ Test: _____

Rework: _____ Implement: _____ Document: _____

Resolution

Analyst: _____

Software Component: _____

Assessment

Method: _____(inspection, analysis, demonstration, test)

Tester: _____ Platforms: _____ Date: _____

Disposition

State: _____ Release: _____ Priority: _____

Acceptance: _____ Date: _____

Closure: _____ Date: _____

Change Management

Software Change Orders

- ***Title*** - Suggested by originator and finalized by CCB

- ***Description*** – Includes name of the originator, date of origination, CCB-assigned SCO identifier, and relevant version identifiers of related support software.

Should provide as much detail as possible along with attached code experts, display snapshots, error messages

Change Management

Software Change Orders

✓ ***Metrics*** – Important for planning, scheduling, assessing quality improvement

❖ ***Change categories***

- Type 0 – critical bug, must fix before product release
- Type 1 – bug, no major impact on system
- Type 2 – Enhancement, improve "ilities"
- Type 3 – New feature or requirement change
- Type 4 – other, like documentation or upgrade for commercial component.

Change Management

Software Change Orders

- ***Breakage*** – Quantifies volume of change. Defined in units of SLOC, function points, files, components or classes

- ***Rework*** – Quantifies complexity of change

- ***Analysis*** – Identifies number of staff hours expended in understanding the required change

Change Management

Software Change Orders

- ***Implement*** – Identifies staff hours necessary to design and implement the resolution

- ***Test*** – identifies hours expended in testing the resolution

- ***Document*** – identifies all effort expended in updating other artifacts like user manual or release description

Change Management

Software Change Orders

✓ ***Resolution***

❖ Includes the name of the person responsible for implementing the change, the components changed, the actual metrics, and a description of the change

❖ The lowest level of component references should be kept the level of allocation to an individual

Change Management

Software Change Orders

✓ ***Assessment***

❖Describes assessment technique as either inspection, analysis, demonstration, or test

❖Reference all existing test cases, new test cases, identify all different test configurations, like platform, topologies and compilers

Change Management

Software Change Orders

✓ ***Disposition :*** SCO is assigned one of the following states by the CCB

Proposed : written, pending CCB review
Accepted : CCB – approved for resolution
Rejected : closed, with rationale, such as not a problem, duplicate, obsolete change, resolved by another SCO
Archived : accepted but postponed until a later release
In Progress : assigned and actively being resolved by the development organization
In assessment : resolved by dev. Org. and assessed by testing org.
Closed : Completely resolved

Change Management

Software Change Orders

Proposed – A proposed change is drafted and submitted to the CCB. Includes a technical description of the problem and an estimate of the resolution effort

Accepted,archived or rejected – CCB assigns unique identifier. Acceptance includes the change for resolution in the next release; archiving accepts the change but postpones; rejection judges the change to be without merit

Change Management

Software Change Orders

In Progress – The responsible person analyzes, implements, and tests a solution to satisfy the SCO. Upon completion submits for assessment team

In assessment – The test team assess whether SCO is completely resolved. When satisfactory SCO is submitted for final disposition and closure

Closed – When development organization, independent test team, and CCB concur that the SCO is resolved, it is transitioned to a closed status