An Introduction to Software Engineering

IGATE is now a part of Capgemini

People matter, results count.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Objective

- To Understand the following :
  - What is Software Engineering (SE)
  - Common life cycle models
  - Phases in SE
  - Familiarizing Requirements Phase
  - Familiarizing Design Phase
  - Familiarizing Construction Phase
  - Familiarizing Testing and acceptance Phase
  - Review and Configuration Management Process

Introduction to Software Engineering

Overview of the Session

- What is Software Engineering?
- Software Development Life Cycle
- Software development Models
- Life cycle selection

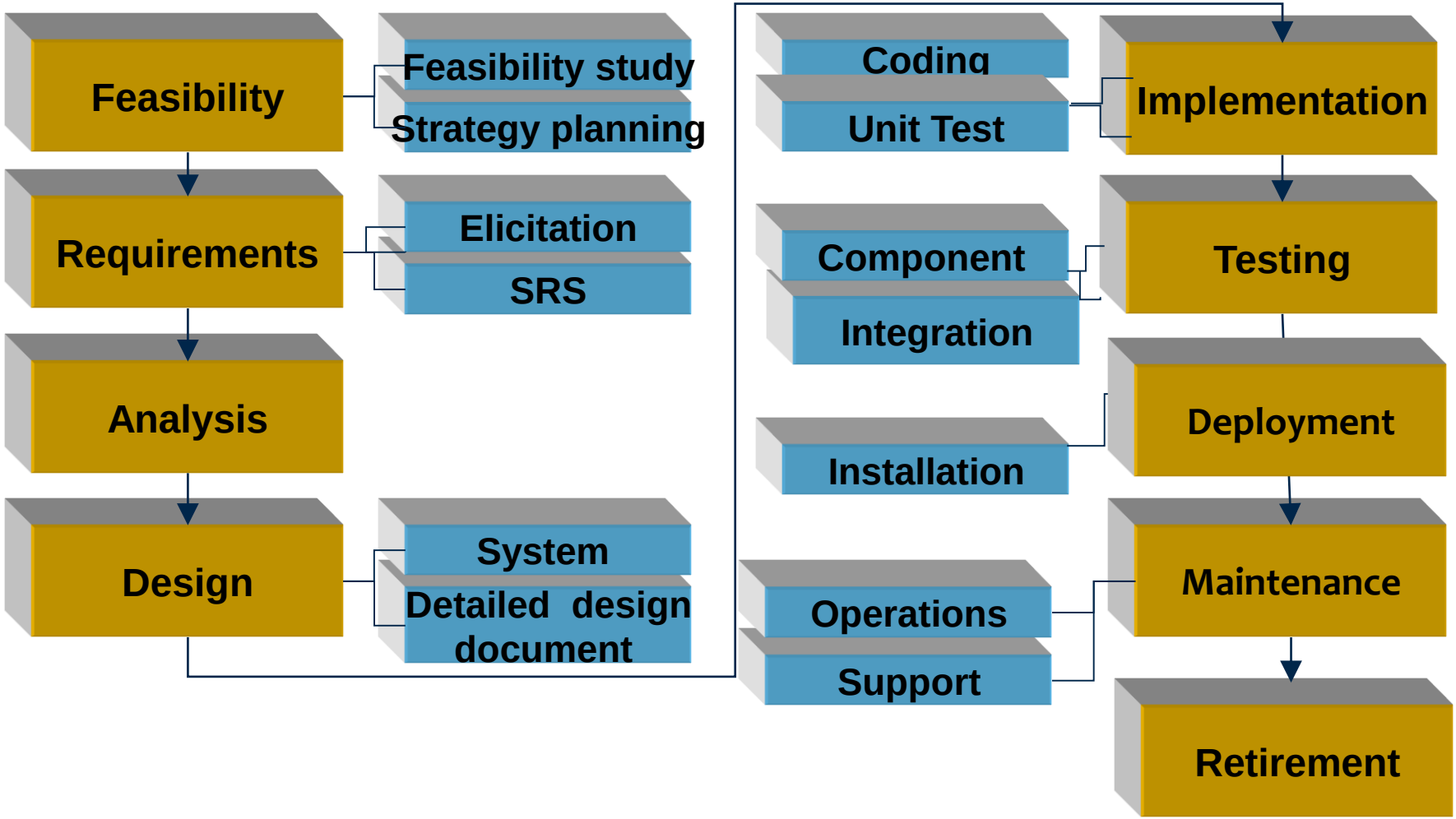
Software Engineering – What

- A systematic, disciplined and measurable approach towards development, operation and maintenance of a software
- Concerned with creating and maintaining software applications by applying technologies and practices from
  - Computer science,
  - Project management,
  - Engineering,
  - Application domains etc..
- It is broad term covering not only the technical aspects of building software , but also other factors like team management, schedule, budget and resource management etc..

Software Development Life Cycle (SDLC)

- Also known as software development process or Systems development life cycle
- A set of processes, standards and tools used to develop, alter software in a optimal manner
- Starts when a product is conceived and ends when the product is no longer available or is effective to use
- Composed of phases , where each phase is dependent on the previous phase's result
- Each phase is a limited period of time starting with a definite set of data and having a definite set of results

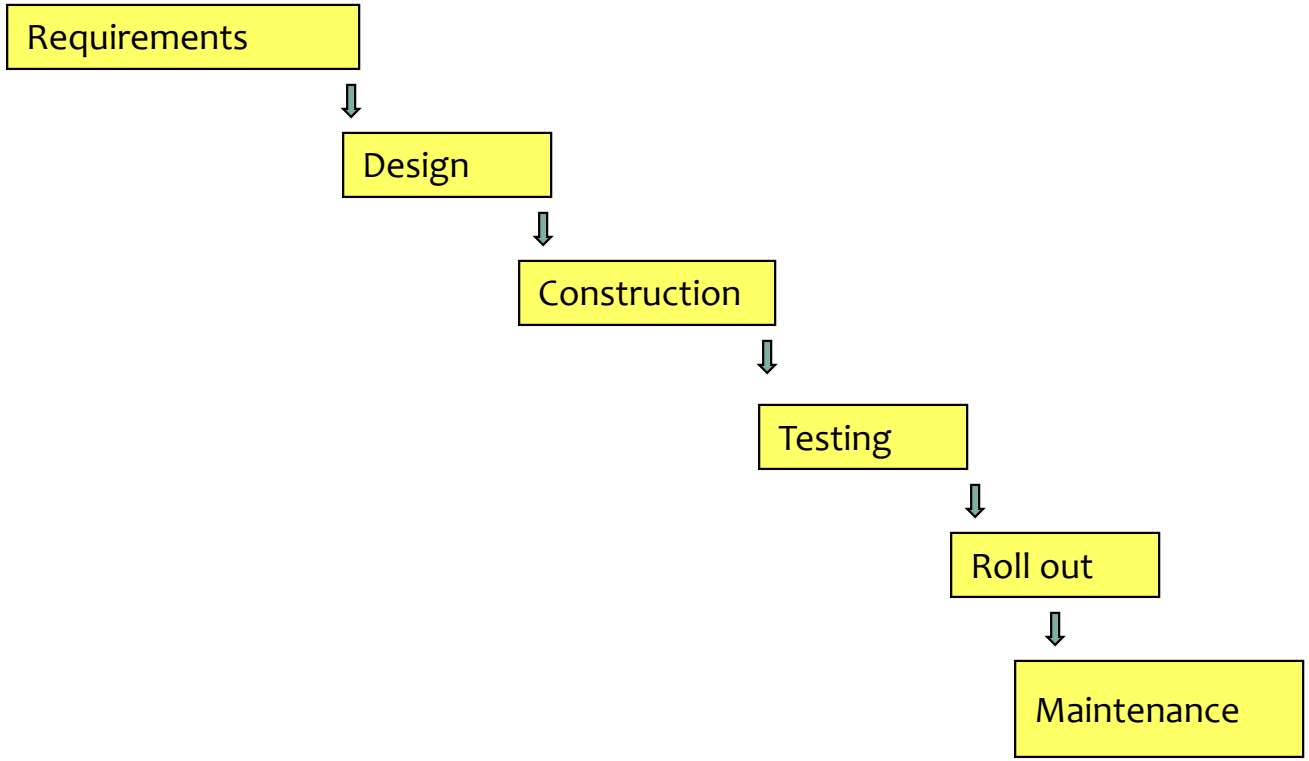
Typical Phases in Software Development



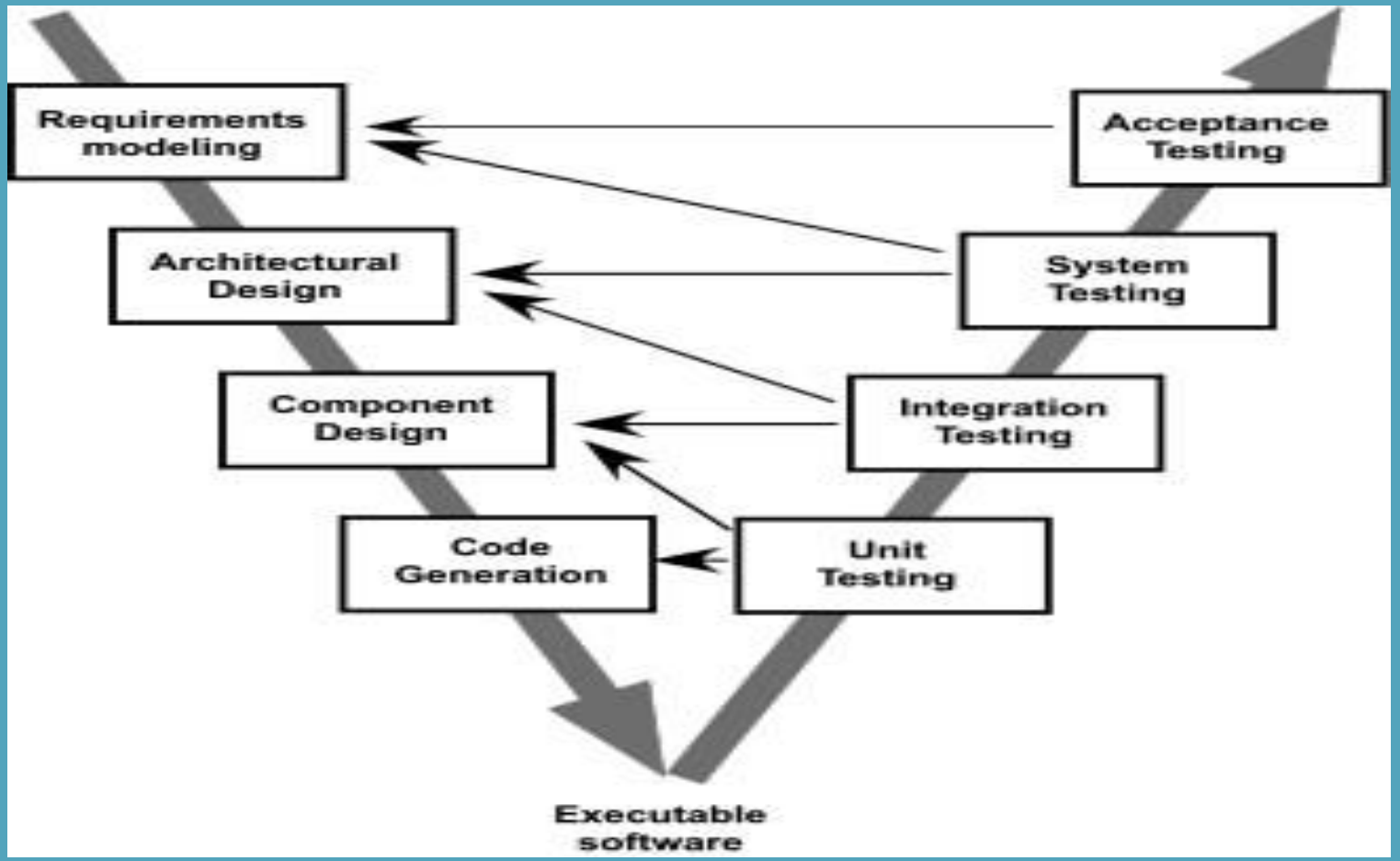
SDLC Models

- A life cycle model covers the entire lifetime of a software – from birth of an idea to phase out
- More than one possible life cycle models can be adopted
- The type of SDLC model is defined by the way it links the phases.
- Every life cycle focusses its phase towards a goal and has a definite milestone
- Some of the common developmental models defined are
  - Waterfall /Enhanced Waterfall
  - V – model
  - Evolutionary Prototyping (aka Incremental)
  - Throw-away Prototyping (aka Rapid))
  - Incremental
- Following models are typically used in the organisations Iterative , V-model , Agile and semi waterfall

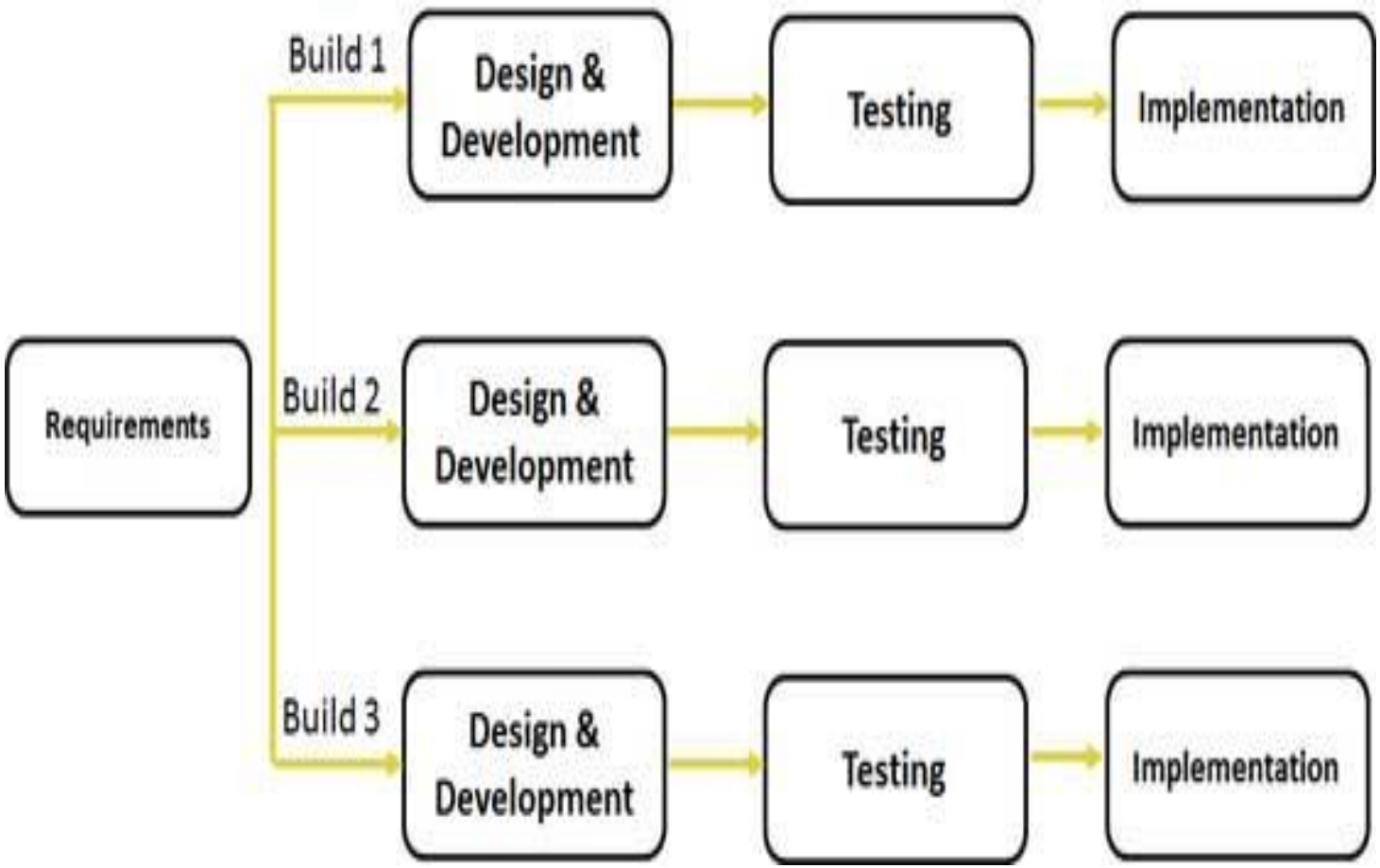
Software Development Models- Waterfall



Software Development Models – V Model



Software Development Models – Iterative and Incremental



Agile Modeling

- It is a variant of the Incremental model
- It enables developing customized software with a process that helps in meeting current requirement as well as future through suitable adjustment
- The following principles that enable this methodology to be effective and light weight
  - **Communication**
    - Open communication between stakeholder and development team at every stage
  - **Simplicity**
    - The model emphasize the need to keep concepts and ideas in simple manner like simple tools , simple design , content etc..
  - **Feedback**
    - Model allows quick feedback from shareholders to ensure that things are on track
- Typically used when requirements are volatile and applications are time critical and the team is aware of the agile practices

Software Operations and Maintenance

- Software maintenance in Software Engineering is the process of modifying a software product after delivery

- The modification of s/w could be for various reasons
  - Fixing a defect - Corrective
  - Address incremental and performance Improvements Perfective
  - Perfecting and adapting the code to the changes in operating environment - Adaptive
- Maintenance activity over the years has evolved to become a crucial source of input and a key driver for new product requirements
- Software maintenance and support projects also follow a life cycle model
  - Quick Fix Model
  - Code reuse Model
  - Iterative enhancement Model, etc..

Software Maintenance Life Cycle – Typical phases

- Application Assessment
  - Understand the application and client expectation
  - Prepare Project Plan
- Knowledge Transition/Responsibility Transition
  - Ramp up the team
  - Sign off service level agreement
- Steady State – Maintenance Release (MR) execution
  - Provide maintenance support
  - Provide production support
  - Monitor Performance

Selection of life cycle and support

- Based on nature of project
  - Clarity of requirements
  - Priority of implementation
  - Need to address change management
  - Need for prototypes
- Organization Support
  - QMS Procedure /guidelines for performing various activities
  - Templates/ forms/checklist/metrics for tracking , measuring and analysing various activities and taking corrective action
  - Tools for automating and improvement
    - SVN /TFS (for CM) (Recommended)

Note : In some cases we may use tools/templates suggested by clients

Introduction to Requirements Phase

What is a Requirement?

- Simply put , it is the needs of the stakeholder which needs to be met/satisfied (by a s/w)
- A Software capability needed by the User to solve a problem to achieve an objective.
- Can be a high level abstract statement indicating needs to a details of the system which the client can validate
- Requirements needs to be
  - Elicited
  - Analyzed
  - Specified
  - Managed
- The engineering process covering all activities leading to discovery , document and manage requirement is known as Requirement Engineering

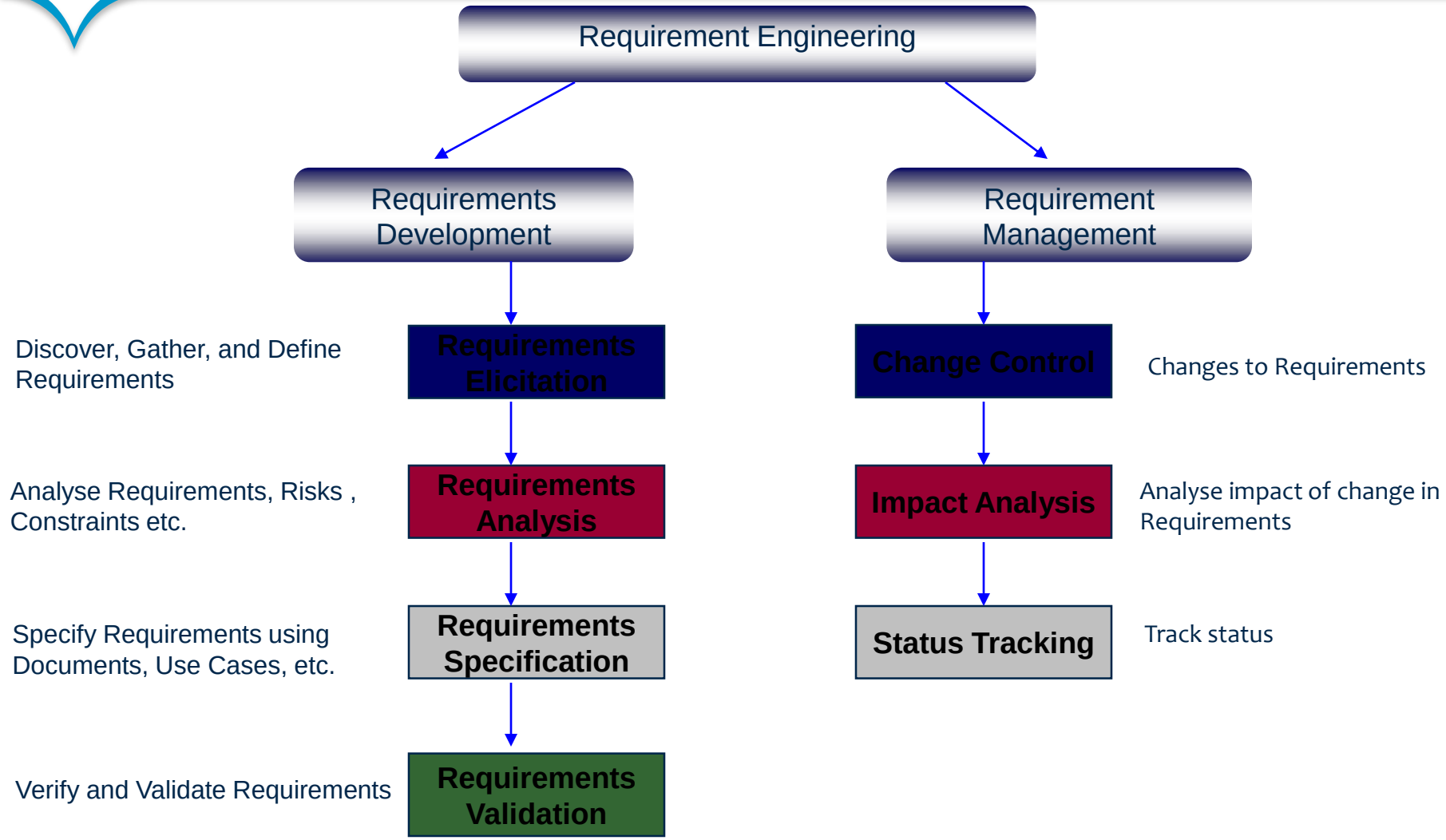
Requirement phase

- This is the initial phase of the development process
- The development team works closely with the customer to determine the customer's requirements for the product – functional, non functional and other characteristics which the product must mandatorily have .
- The requirements identified in this phase serve as a foundation for the remaining phases of the development process, and the customer acceptance criteria.
- The main participants involved in the requirement phase are
  - Stake holders
  - Requirement Engineer

Need for good requirements

- Requirement Problems are the single No.1 reason for projects failing over
  - Schedule
  - Budget
  - Scope
  - Quality
  - And even getting Cancelled!!
- Reworking requirements cost 40-50% of project effort
- Many problems found during design, testing, or operation of a system are the result of incorrect, incomplete, or missing requirements

Requirement Engineering activities



Requirement Engineering activities

- Requirement Elicitation
  - This phase focuses on examining and gathering desired requirements and objectives for the system from different stakeholders
  - Various techniques are followed to gather requirements viz interviews, document examining , brainstorming , prototyping etc
- Requirement Analysis
  - This phase focusses on analyzing rigorously ,classifying, prioritizing , documenting the gathered requirements within business context
- Requirement Specification and Validation
  - A formal document is prepared after collating all requirements which contains a complete description of the external behavior of the software system.
  - Requirements are specified in
    - URS User Requirement Specification
    - SRS System Requirement Specification
    - Use Case Documentation
  - **T**he requirement documented in the SRS is verified , validated and agreed upon by all parties .

Requirement Validation and Management

- Requirement Management
  - Requirements Management (RM) involves recognizing and planning changes occurring in requirements due to various factors during the life of the project
  - RM is a continuous activity that can occur post development during maintenance as requirements may continuously change
  - When a sizeable set of changes are received, the project may decide to go thru a change request process , to get approval for time and budget

Requirement phase key points

Pre-requisites

- Contract/Statement of Work
- Finalization of Engagement boundaries

Activities

- Capture requirements
  - Functional
  - Technical
  - Performance
- Gap Analysis where applicable
- Define interfacing requirements
- Documentation of complete requirements
- Develop Requirements Traceability Matrix
- Present requirements document to client team
- UI prototyping where necessary
- Finalize Acceptance Criteria
- Identify data migration requirements

Completion Criteria

- Client signoff on technical, functional and performance requirements
- Validation of UI prototypes
- Signoff on Acceptance criteria

Deliverables

- SRS /Use Case document
- UI design
- Acceptance criteria
- Interface requirements

Introduction to Design Phase

Architecture and Design

- Architecture
  - It is the high level organizing structure of the system
  - It defines the components, interfaces, and behaviors of the system.
  - The process of architecting a software involves defining a structured solution that meets all of the technical and operational requirements, along with attributes such as performance, security, and manageability.
  - This phase usually involves the technical/solution architect

- Design
  - It is a process of creating a detailed specification for a software module .
  - It involves algorithmic design and other implementation specific approaches for a s/w component such as modularity , control hierarchy, data structures etc
  - Designers /Technical leads ,senior developers , architects are involved in this phase

Architecture deals with Non functional requirements whereas design deals with functional

Key activities in Design phase

- The design phase includes following activities
  - Identify solution which will meet the customers non functional requirements like performance , security etc..
  - Identify technology stack
  - Identify framework and design pattern
  - Create software architectural overview document
  - Identify major modules and its interfacing with each other as well as external systems if any
  - Defining the logical and physical database model
  - Create test design
  - Plan of the unit and integration test cases
  - Detailing the overall logic of the module in pseudo code or flow charts
  - Detailed database design including constraints data types etc.. (Physical)
  - Detailed interfacing reference (with API and parameters
  - Prepare design documents

Design phase key points

Pre-requisites

- Signed off requirements
- Signed off prototype
- Finalized acceptance criteria
- Finalized interface requirements

Activities

- Detailed System Design
- Prepare Object Models (Class diagrams, Sequence Diagrams)
- Prepare Database Models (Conceptual Data Model, Physical Data Model)
- Design review
- Develop QA plan
- Develop data migration plan
- Develop Integration Test plans and test cases

Completion Criteria

- Approved System Design documents
- Approved Models – Db , Application
- Approved QA plan

Deliverables

- SAD
- HLD
- LLD
- ITP

Introduction to Construction Phase

Construction phase

- Also known as implementation phase
- Main objective of this phase is to translate the software design into code , each component identified in design is implemented as a program module following coding guidelines
- Each module in this phase is reviewed and unit tested to determine correct working (White Box testing)
- Unit tested code are then integrated in a planned and a phased manner .
- In each integration step the partially integrated system is tested

Construction phase

- In addition to the major activities the following activities are also carried out as well
  - Prepare unit test plan and test case
  - Prepare unit test data
  - Setup coding guidelines
  - Setup the environment for Configuration Management as per CM guidelines
  - Provide suitable environment for base lining code and continuous integration
  - Defect reporting and fixing

- The main role players in this phase are
  - Developers
  - Team Leads

Construction phase – key activities

Pre-requisites

- Approved design documents
- Approved QA plan
- Standard Coding guidelines
- Review checklists

Activities

- Development environment setup
- Prepare Unit Test plan and data
- Build Code
- Code review
-
- Perform Unit Test
- Rework and re-test
- Baseline source code

Completion Criteria

- Code ready for System testing

Deliverables

- Test reports
- Baseline source code

Introduction to Testing Phase

System Testing

- System testing involves testing of all subsystems together
- Also known as Black Box testing It is ideally done by the QA team
- The following types of testing are done as part of system testing
  - Functional testing to validate functional requirements
  - Performance testing to validate non functional requirements

System Testing Key Activities

Pre-requisites

- **System Test plans**
- **Integrated Application**

Activities

- **Test environment setup**
- **Validate test plan with Requirements**
- **Perform system testing**
- **Write additional test cases if needed**
- **Log defects**
- **Rework based on test results**

Completion Criteria

- **Satisfactory completion of System tests**

Deliverables

- **System tested code ready for UAT**

Acceptance Testing

- Usually done at the client location by the client , after the findings of System testing is fixed
- Focus of Acceptance test is to evaluate the system's compliance with the business requirements and assess readiness for delivery.
- Acceptance Testing is done in two ways
  - Alpha Testing or Internal Acceptance Testing
    - **done by s/w vendors**
  - Beta Testing or User Acceptance testing
    - **Done by end users of customers or customer's customer**
- Outcome of the acceptance testing will enable the user, customers or other authorized entity to determine whether or not to accept the system.

Acceptance Testing - Key activities

Pre-requisites

- **System and Integration tested code**

Activities

- **Assist client in setting up testing environment**
- **Support users / client team, in acceptance testing**
- **Fix defects / bugs**
- **Acceptance and signoff from the client**
- **Assist client team in preparing implementation plan**

Completion Criteria

- **User accepted application**

Post Acceptance phase

- After successful acceptance testing plans are made to move the application to the “live environment”
- Activities like knowledge transfer , end user training , project signoff are also done .
- Once when the customers starts using the developed system the maintenance team supports and monitors the system to resolve errors and performance .

Software Reviews

Reviews – What

- An assessment of a work product created during the software engineering process
- Ensure completeness and consistency of the work product
- Identify needed improvements
- It is a Quality Assurance mechanism to identify discrepancy /deviation from the accepted standards
- Goal of Review
  - To detect and eliminate defects early, effectively and before delivering the product to the customer

Reviews – Why

- Intermediate software products which are not testable as standalone units
- Can find errors not possible through testing
  - E.g., Maintainability: Comments, Consistency, Standards
- Are proactive measure to find out 60-80 % of defects
- **Reduce Rework Effort and Improve Schedule adherence**
- Enables Quantitative Quality Assessment of any work product

Software Reviews – When , where

- Can happen in all phases of SDLC
- All artifacts can go for reviews
  - Proposals, contracts, statement of work
  - All project work products - Plans, Configuration Mgmt, Test Plans
  - Deliverable and non deliverable work products
  - Software(e.g.: source code) and non software work products (e.g.: documents, test data, etc.)
- Process descriptions
- Policies, brochures, reports, guidelines, standards, training material where required

Types of Review

- Self Review
  - Done by the author himself with the aid of tools like checklists , review guidelines , rules etc..
- Peer Review
  - Done by “peer” or colleague formally or informally using various approaches
    - Inspection
    - Walk through
    - Pair Programming

Review Process

- Input
  - Work Product , Specifications, Checklists, Guidelines, Historical Data
- Process
  - Prepare for Review
  - Conduct Reviews
  - Analyze Deviations
  - Correct Defects
- Output
  - Review Form, reviewed work product,

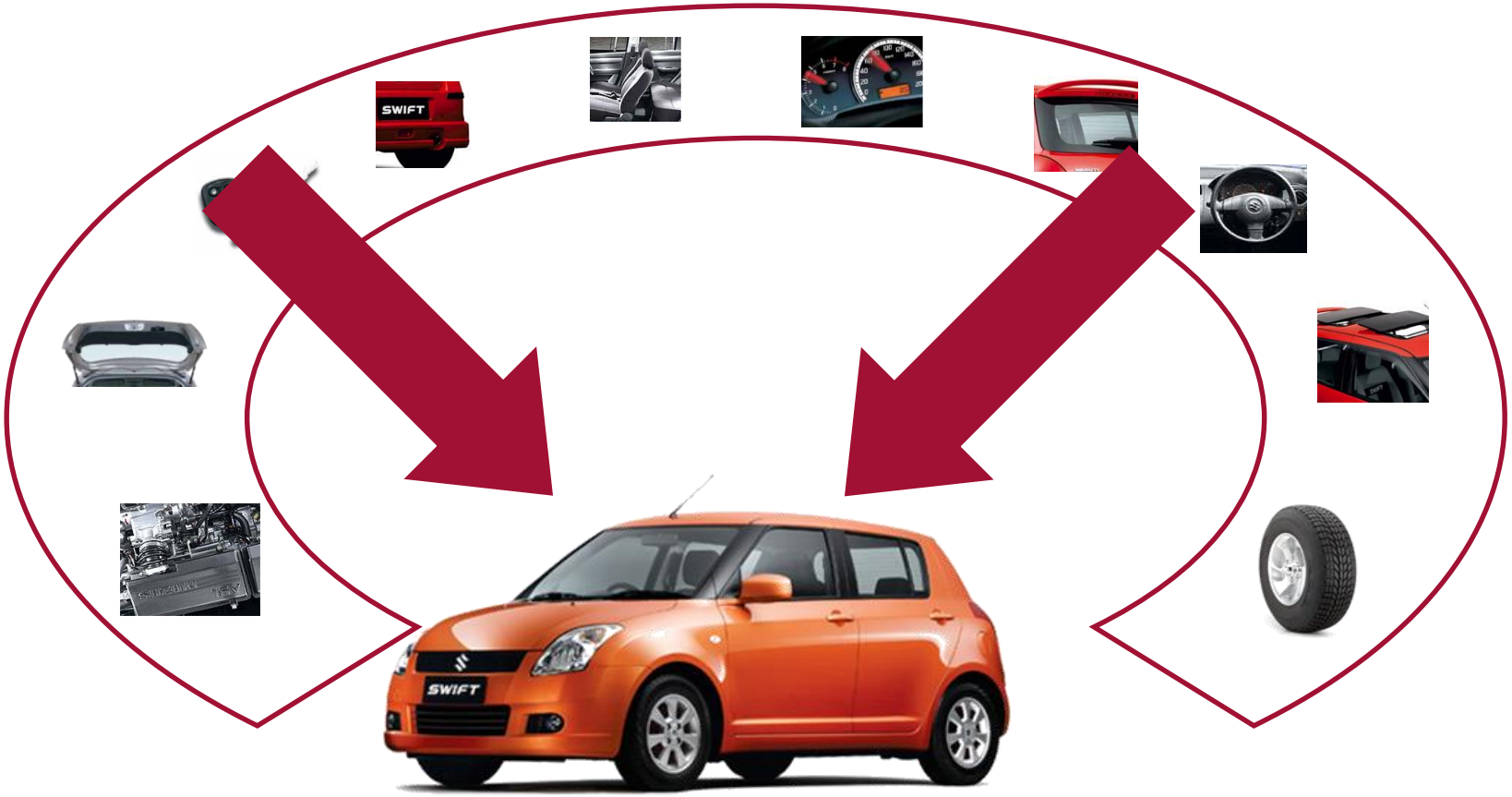
Introduction to Configuration Management Process

Agenda

- What is SCM?
- Why SCM?
- Elements of SCM
- Change Management
- Information on SCM tools

What is a “Configuration”?

- Arrangement of functional unit of a system in a particular order



What is Software Configuration Management?

- SCM is the overall management of a software project as it evolves into a software system.
- This includes managing , tracking, organizing, communicating, controlling modifications made in project including release plan
- Also includes the ability to control and manage change in a software project
- Configuration Managers Are responsible for planning the CM activities of their project
- The configuration details of the project are documented in the CMP (Configuration management Plan)

Why do we need SCM?

- Some of the frustrating problems we face are
  - The latest version of the source code not found
  - A developed and tested feature is mysteriously missing
  - A fully tested program suddenly does not work
  - A wrong version of code was tested

- SCM answers who, what, when and why
  - Who makes the changes?
  - What changes were made to the system?
  - When were the changes made?
  - Why were the changes made?

Elements of SCM

- Configuration identification
  - CI
  - NCI
- Configuration control (Elements)
  - Library Control
  - Access Control
  - Version Control
  - Establish Naming conventions
  - Establish Baselines
  - Branching, Merging and Labeling
- Change Management
- Auditing (Verification)

Elements of SCM

- Configurable item (CI)
  - CI is a collection of items, treated as a unit which are likely to undergo change during the project life cycle and a change to them is likely to affect other CIs.
  - Items that needs to be accessed, controlled, secured and archived is a configurable item

    (E.g.) Design document, project plan etc..

- Non Configurable item (NCI)
  - Any item / file for which changes need NOT be tracked) i.e. no need to roll back to earlier versions is called a Non-Configured Item.

  (E.g.) Minutes of Meeting(MOM)

Library Structure

- Controlled collection of software and related documentation designed to aid in
  - software development
  - use
  - Maintenance

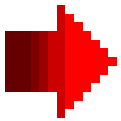
Organized structure

- The folder structure is indicated in the CMP

Example of Libraries Structure

- Input Library
- Development Library
- Testing / Review Library
- Release / Delivery Library
- Template Library
- Project Management Library

Tip: The library (folder) can be created on need basis for the project. No thumb rule to create the same.

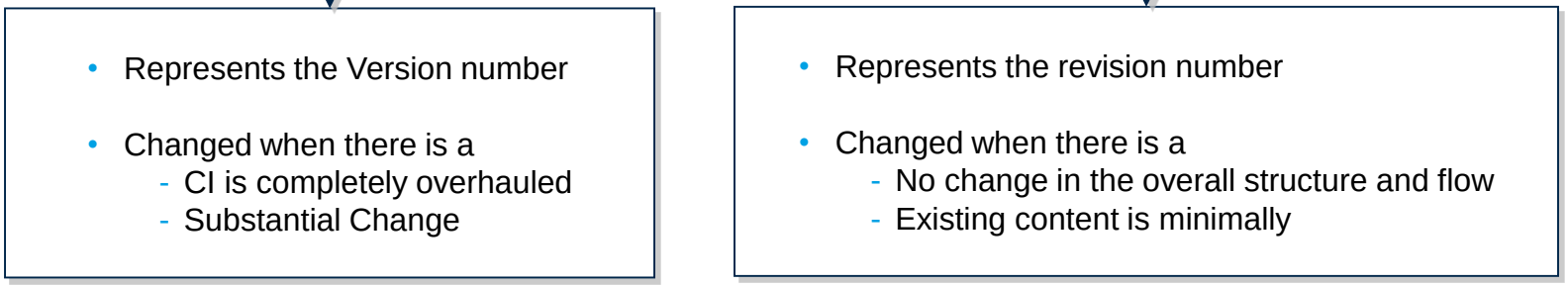
SDLC  **Folders**

Usage of library - example

- Coding and Testing scenario
  - Development done in Development library by development team who have access to development folder
  - QA team (testing team) would be doing the testing
  - As per CM policy QA team wont have permission on Development folder ,
  - The code is **moved** from development folder to testing folder
  - The code **is moved back** to development folder for rework
  - The Re-testing happens in Testing library following the above steps
  - Once all the bugs are fixed , the code is moved to release folder .

Version Numbering

- A version number is a unique number or set of numbers assigned to a specific release of a software/hardware/firmware
- As updates and new editions of product are released, the version number will increase
- Version numbers are usually divided into sets of numbers, separated by decimal points
- Draft version has X . Y



Naming Conventions

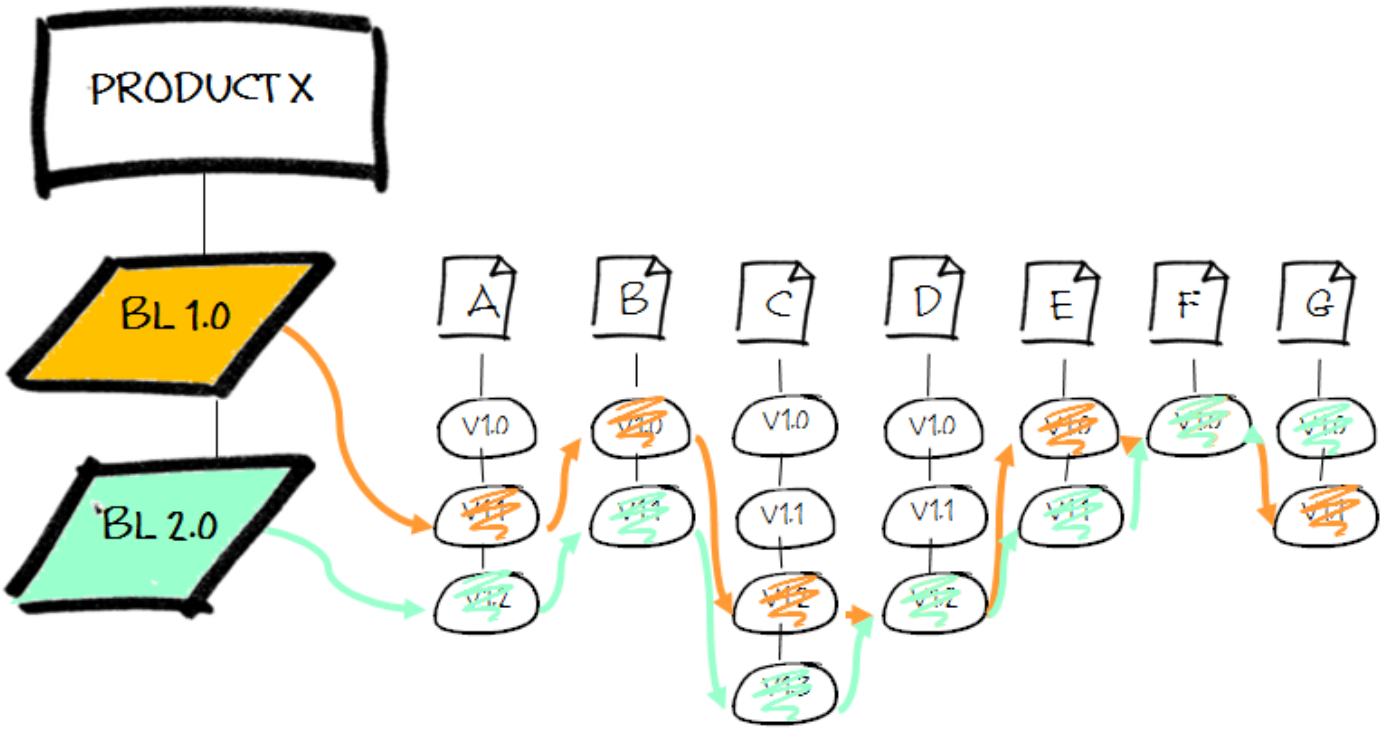
- Helps in easy identification
- The name may include
  - Project name/ Application name/ Request ID
  - Document/ Work product name
  - Version number/ Date (If manual configuration)
  - Status – Draft, review, approval etc.,

Sample Document	Sample Naming conventions
Management documents	Est_<Project name>_dd_mm_yy_<ver_ x.y>.doc, PP_<Project name>_<ver_x.y>.doc, SCH_<Project Name>_<ver_x.y>.xls/mpp
Source code	<Component name>_<ver_ x.y>.java, <Module name>_<component name>_<ver_ x.y>.java
Unit Test Plan	UTP_<Component name>_<ver_ x.y>.doc
Quality Records	C-rev_<component name>_<ver_ x.y>**.xls, C-rev_<Module name>_<component name>_<ver_ x.y>**.xls

Baselines

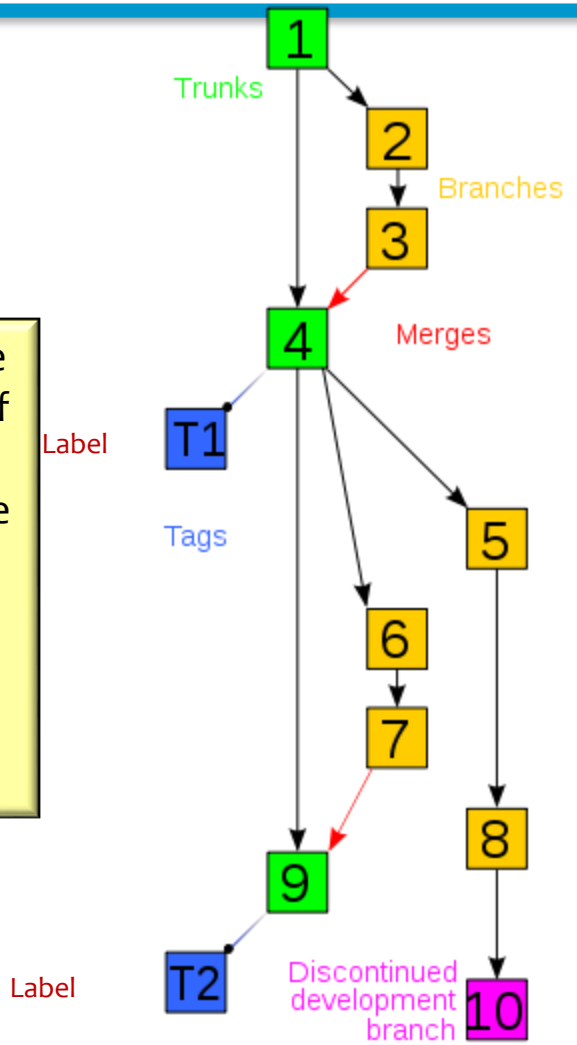
- Baseline
  - A 'baseline' is a CI that has been reviewed and agreed upon and is a basis for further development.
  - After base-lining all changes to CI are controlled through a formal change process (Such as Change Management and Configuration Control).
  - For example a reviewed and approved Project plan is used as a basis for execution of the project.
  - One baseline may have several work product , each having different version number
  - Baselining can be of many types – Input baseline , design baseline , code baseline etc

Illustration of a Baseline



Branching, Merging and Labeling

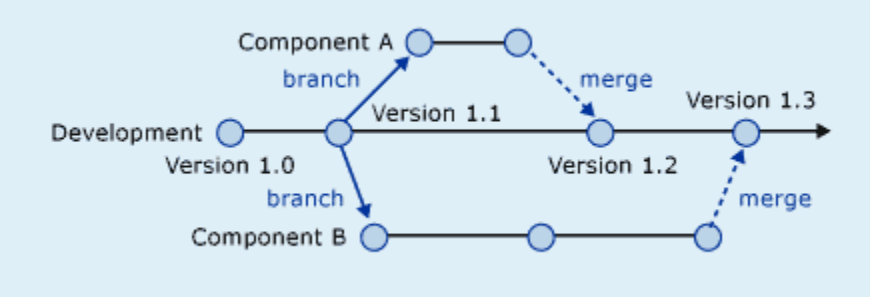
Branching and Merging is the process of duplicating part of a software development project baseline so that some parallel development can take place. Then once completed it's merged back into the baseline.



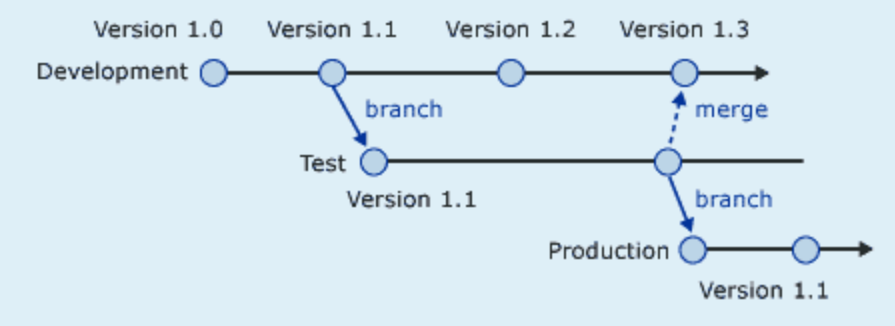
Label is kind of bookmark on the elements. All current versions of the elements are marked with label which makes it easy to retrieve elements later based on the label.

Different Ways of Branching

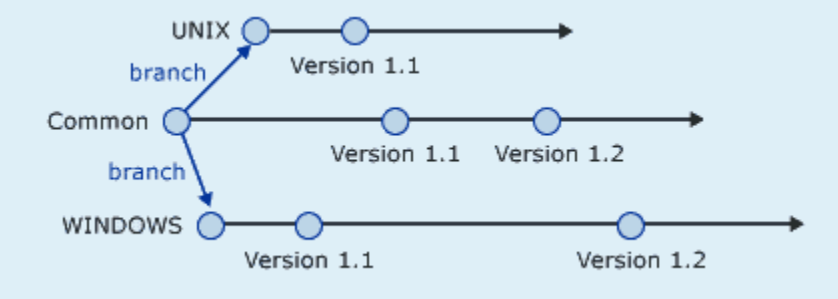
Branch per Release



Branch per Component



Branch per Promotion



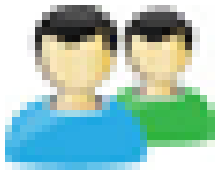
Branch per Technology

SCM Tools

- Tools help in managing projects better as it reduces a lot of manual effort

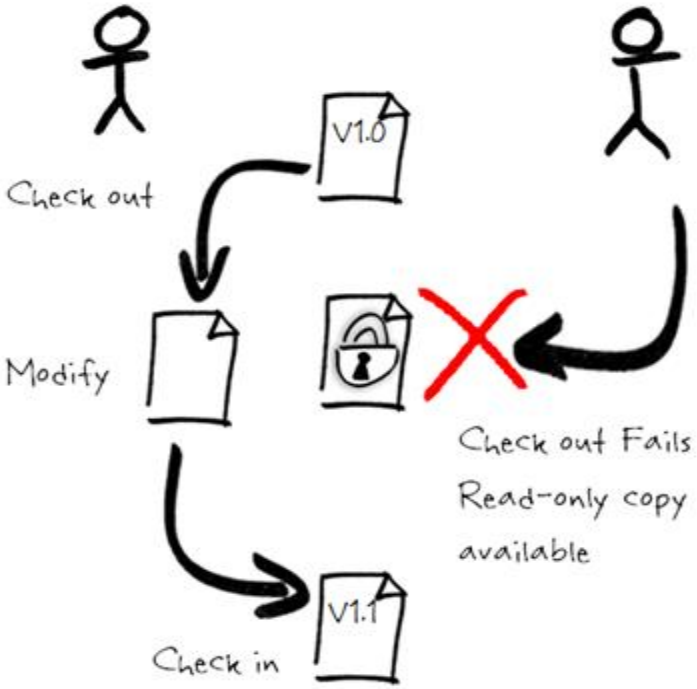
- Early SCM tools had facilities for controlling and managing CI , but now it comes with a lot of features like
  - Build and Release management
  - Defect Management and tracking
  - Packaging control etc ..
- Major advantages of SCM tools are
  - Sharing of project information across teams with accurate and reliable information,
  - Flexible in supporting parallel development
  - Provide better decision-making capability by managing and tracking

Different Roles and Accesses in SCM Tool



- Administrator
  - All access to all project folders
  - Can add users, create and manage projects and modify their access levels
- Configuration Manager
  - Greater access than all team-members.
  - Creates the basic environment for his projects configuration management
  - Responsible for moving files across projects, establishing baselines, adding requirements files, preparing guidelines, etc
- Team-member
  - Varying access depending on their responsibilities. For e.g. PM gets add/modify access to Management project

Check –in and Check -Out



A check-out is the act of creating a local working copy from the repository. A user may specify a specific revision or obtain the latest.

A check - in is the action of writing or merging the changes made in the working copy back to the repository.

Test your understanding

- SRS (System Requirement Specification) are prepared in which phase ?
- A software can be built using more than 1 life cycle model . (T/F)
- Name some non functional requirements
- Architecture focuses on functional requirement T/F ?
- Alpha testing and beta testing happens in which phase ?

Test Your Understanding!

- In which phase UNIT Test Plan is written ?

- White box testing includes
  - Unit Testing
  - System Testing
  - Operations Acceptance Testing
  - Integration testing

- A single baseline may contain many files.(T/F)
- A Tester can test in the development library.(T/F)