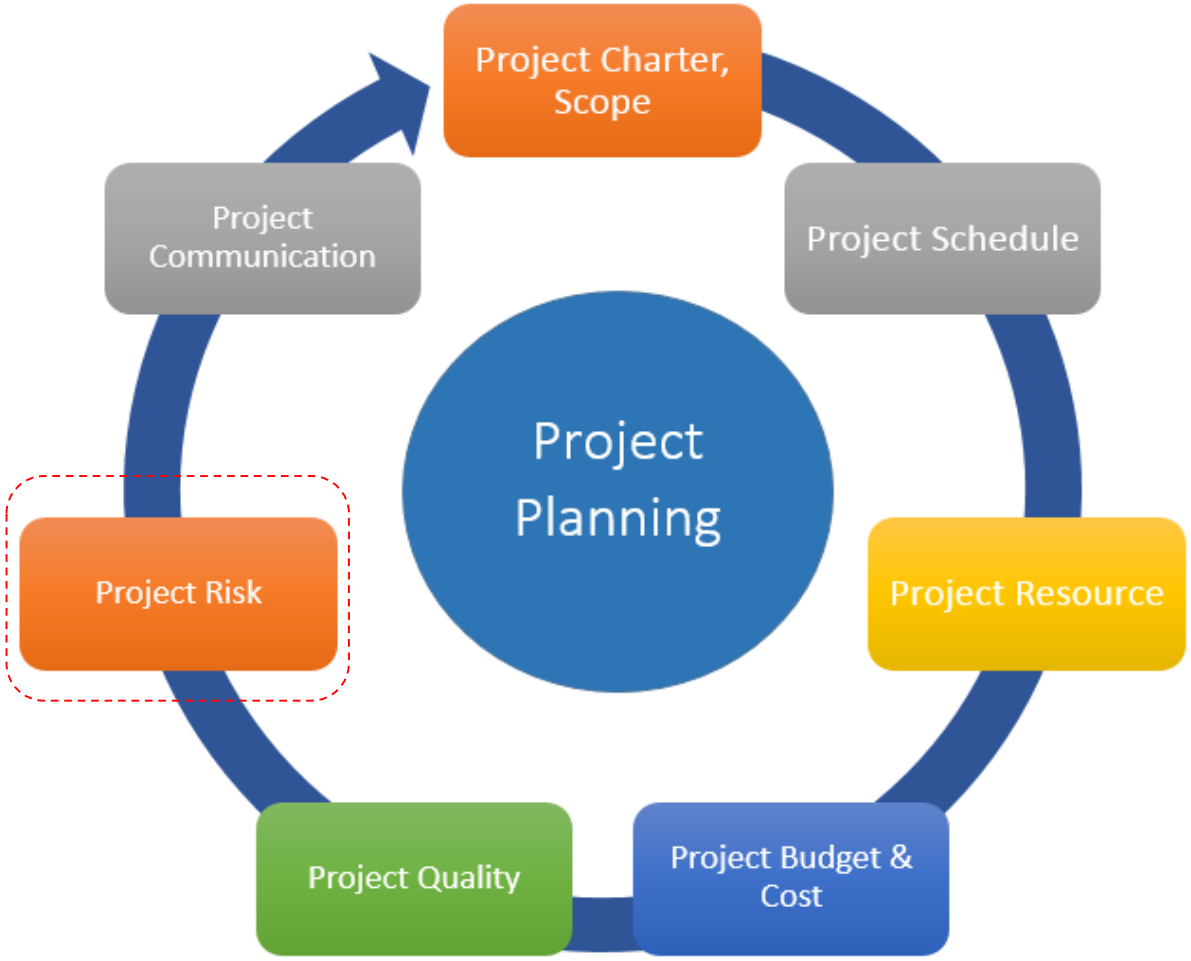
SWEN90016
Software Processes & Project Management

Quality Management
Configuration Management

Shanika Karunasekera
Department of Computing and Information Systems
The University of Melbourne
karus@unimelb.edu.au

2019 – Semester 2
Lecture 8

RECAP – Risk Management



RECAP – Risk Management

Learning Outcomes:

1. Understand the fundamentals of risk management
2. Understand the Risk Management Process
3. Understand how to:
   - plan risk management activities
   - identify risks
   - analyze and assess risks
   - respond to risks (risk strategies)
   - monitor and control risks

RECAP – Risk Management

Risk Planning → **Risk Management Plan**

Risk Analysis and Assessment:

Risk ID	Risk	Probability (0 – 100%)	Impact	Exposure	Rank
1	XXX	40%	4	1.6	4

Risk Impact Analysis Table

Risk Identification:

Kinds of Risks:

Project, Product, Business

Identification Techniques:

Pondering	Interviewing
Brainstorming	Checklists
Delphi	SWOT Analysis

Risk Response

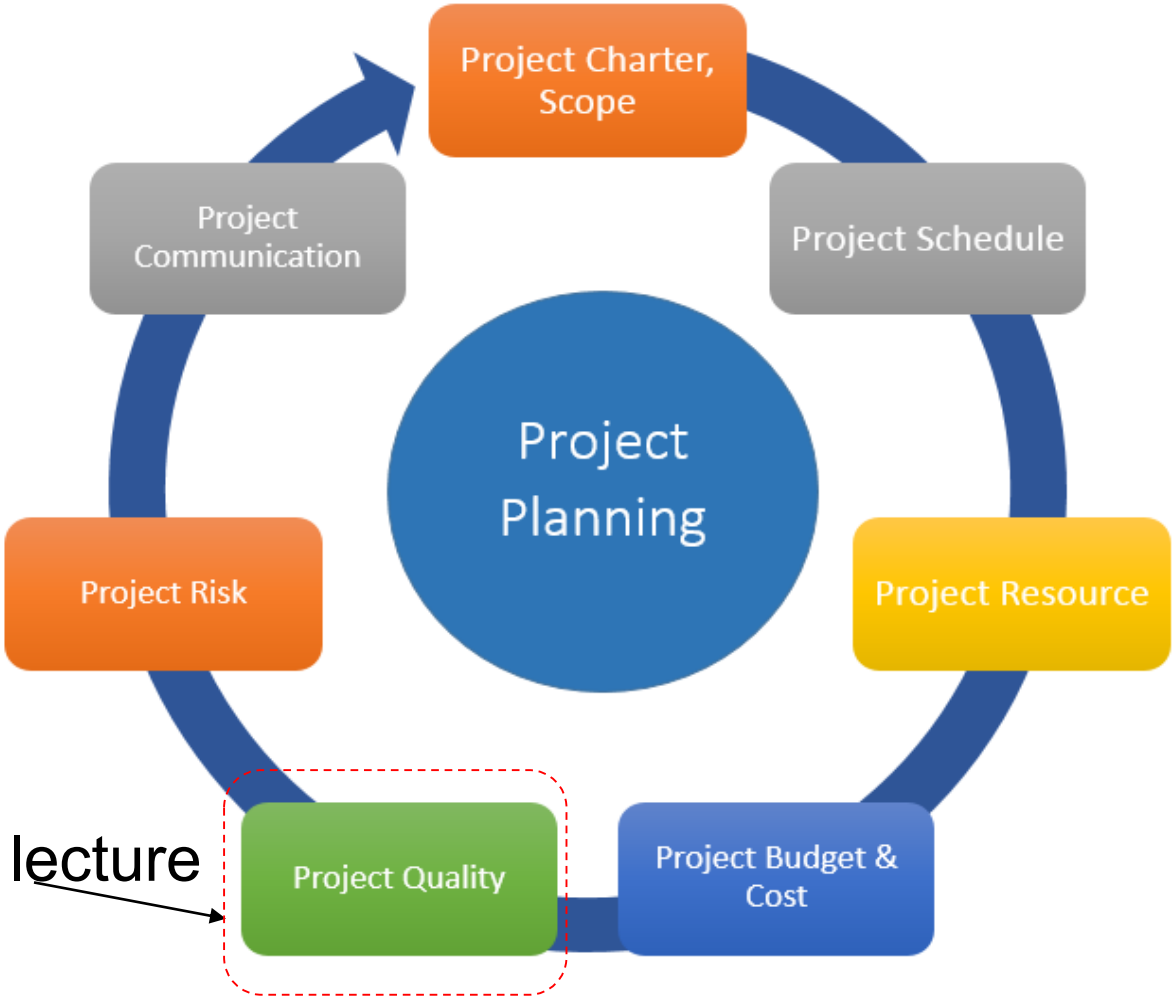
Risk ID	Trigger	Owner	Response	Resources Required

Risk Register

Which of the following risks is the best example of a business risk?

Change of project objectives	**A**
Scope creep	**B**
No demand for the product	**C**
Design not well thought through	**D**
Lead architect leaving the project	**E**

Project Quality



Today's lecture

Semester Structure

Week #	Lecture Week Start	Old Arts Public Lecture Theatre Friday 3.15pm to 5.15pm	Assignment
1	29/07/19	Subject Introduction, Introduction to Projects and Project Management	
2	05/08/19	Project Management Plan & SDLC's	Assignment 1 Spec available on LMS 05/08
3	12/08/19	SDLC - Agile Scrum – continued Individuals, Motivation and Teams	
4	19/08/19	Stakeholder Management Communication Management	Assignment 2 available & Groups created during the workshops / tutorials – attendance mandatory
5	26/08/19	Project Planning and Scheduling	Assignment 1 (Individual) due Fri 31/08 @ 11.59 pm
6	02/09/19	Cost Estimation	
7	09/09/19	Risk Management	
8	**16/09/19**	**Quality Management/Configuration Management**	**Assignment 2 (Part 1) due Wed 18/09 @ 11.59 pm**
9	***23/09/19***	***University Holiday***	
	30/09/19	***Non Teaching Week – Mid semester break***	Assignment 2 (Part 2) due Sat 28/09 @ 11.59 pm
10	07/10/19	Ethics, Outsourcing & Procurement	Assignment 2 (Part 3) due Sat 12/10 @ 11.59 pm
11	14/10/19	Guest Lecture	Assignment 2 (Final) due Sat 19/10 @ 11.59 pm
12	21/10/19	Subject Revision and Exam Prep	Assignment 2 Project Demonstration during tutorials

Learning Outcomes

Quality Management

1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
   - Quality Assurance
   - Quality Planning
   - Quality Control and Monitoring

Learning Outcomes

1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
   - Quality Assurance
   - Quality Planning
   - Quality Control and Monitoring

What is quality?

Quality is not an act, it is a habit — Aristotle

- Evidence shows that we cannot simply fix up our software *post-hoc* and add in quality attributes after building the system
- Quality must be *built into the software from the beginning*
- In this topic you will learn how to built quality into the software through a range of *Quality Management* activities

Which of the following would you regard as a high quality product?

A product that meets client requirements	**A**
A product that has passed 100% of the test cases	**B**
A product that is reliable and efficient	**C**
A product that is easy to use	**D**
A product that is easy to maintain and extend	**E**

What is software quality?

- We define quality from two broad perspectives:
  - **End-user's Perspective:**

    Typically, end-users judge the quality of a product by their interaction with it. For users, a system has quality if it is fit for purpose, is reliable, has reasonable performance, is easy to learn and use, and helps the users in achieving their goals. Sometimes, if the functionality is hard to learn but is extremely important and worth the trouble of learning, then users will still judge the system to have high quality. These are termed ***external quality characteristics***, because they are typically associated with the external behaviour of the system.
  - **Developer's Perspective:**

    The developer's perspective typically also includes the number of faults that the system has, ease of modifying the system, ease of testing the system, the ease of understanding the system design, the re-usability of components, conformance to requirements, resource usage, and performance. These are mainly ***internal quality characteristics***, because they are concerned with the quality of the internal structure of the system.

Cost of quality

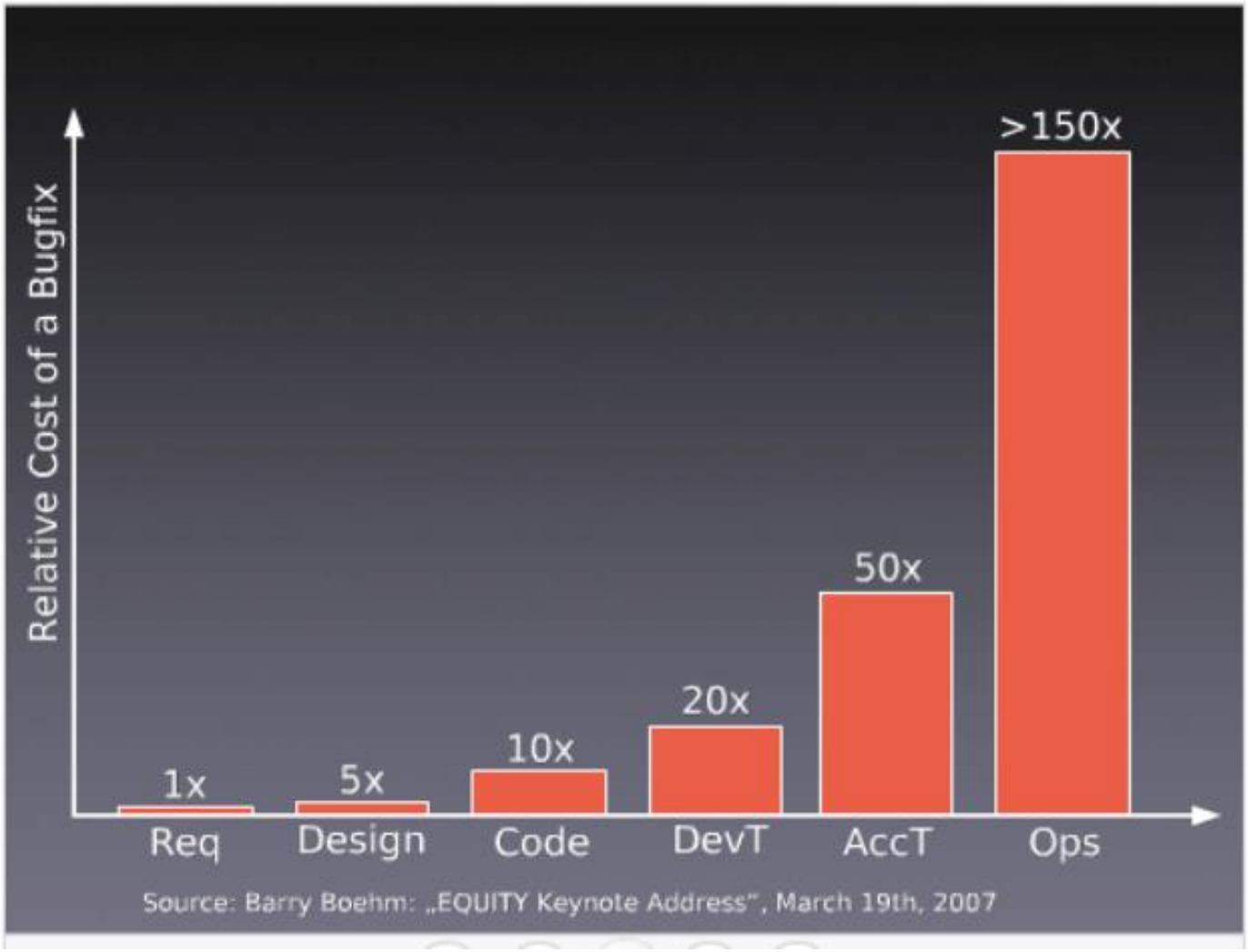
- Some claim:

  Most quality assurance activities are too costly - savings made from not using resources is greater than the cost incurred in fixing the faults

  - For example, instead of performing formal reviews of requirements specification documents, it is far better to build the system, ask the client/user for feedback, and to correct any faults from there.
  - Alternatively, one can simply release the system and correct faults as users report them.

- Empirical studies refute the above claim:
  - There are many studies in the area

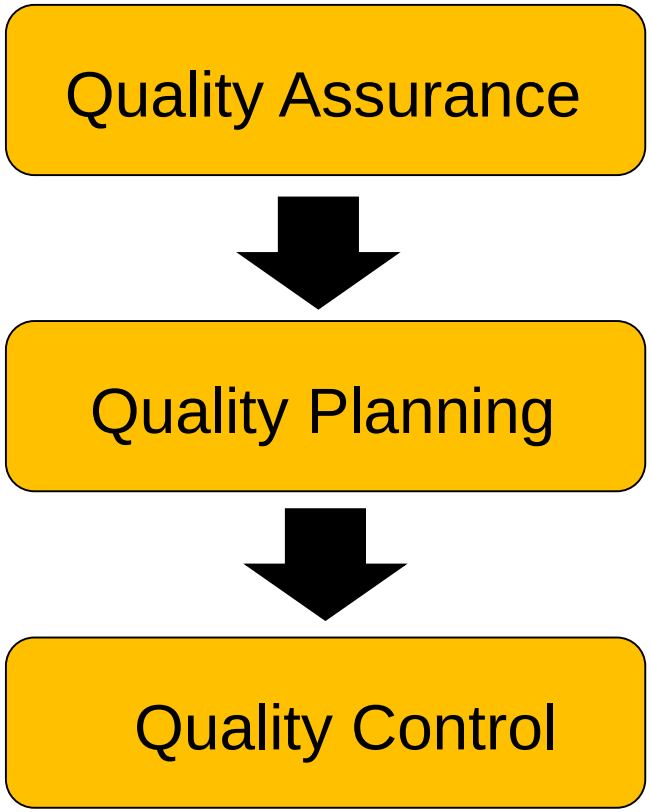
Cost of quality



Learning Outcomes

1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
   - Quality Assurance
   - Quality Planning
   - Quality Control and Monitoring

Quality Management Process



Quality Management Process

1. ***Quality assurance:***

   The establishment of a framework of organizational procedures and standards that lead to high-quality software

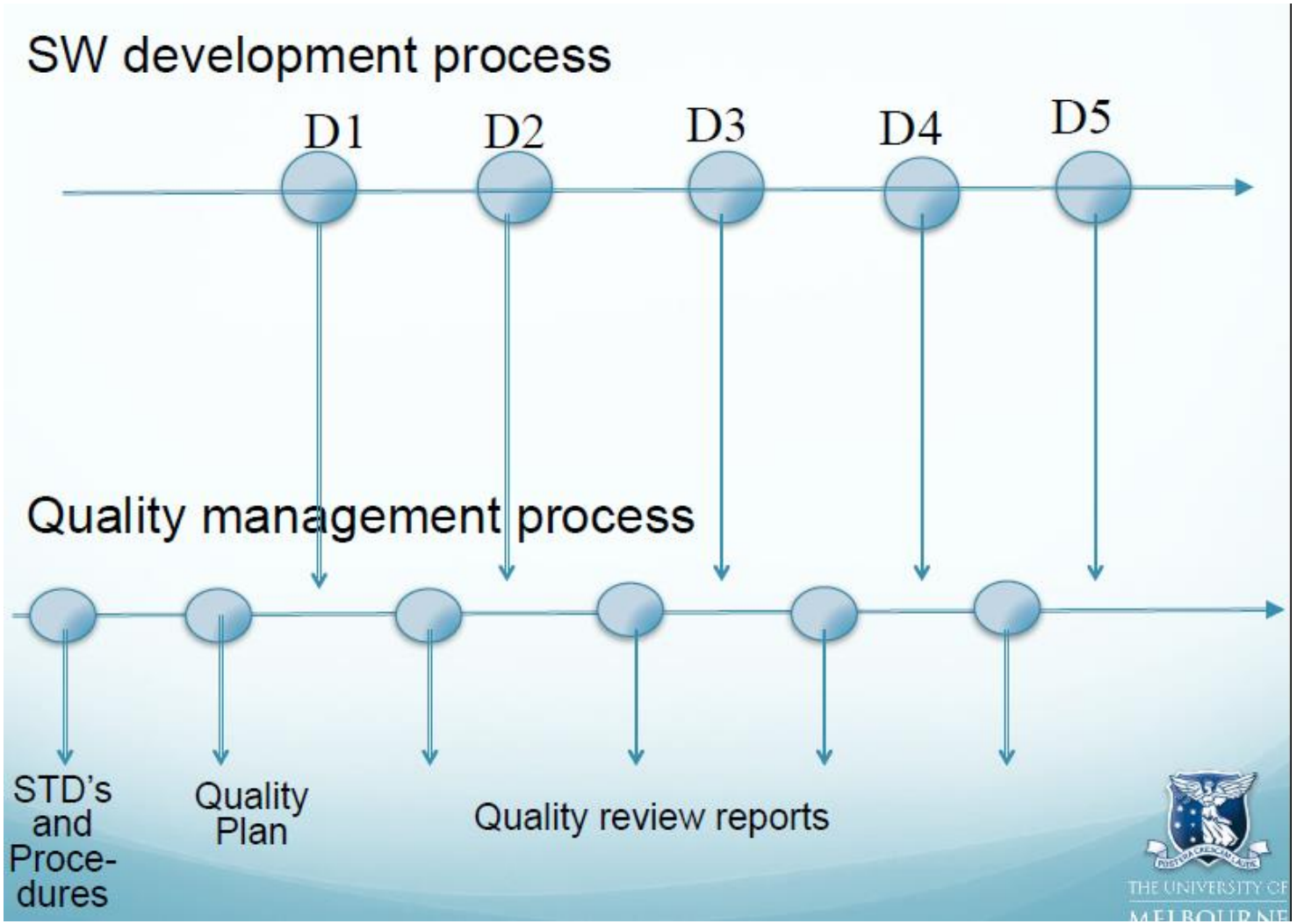
2. ***Quality planning:***

   The selection of appropriate procedures and standards from the framework, adopted for the specific project

3. ***Quality control:***

   Ensuring that the software development team has followed the project quality procedures and standards

Quality Management Process



Learning Outcomes

1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
   - Quality Assurance
   - Quality Planning
   - Quality Control and Monitoring

Quality Assurance

- Quality assurance process is primarily concerned with defining or selecting the *quality standards*
  - A standard might simply be defined as a *set of rules for ensuring quality*
  - Standards play an important role in the quality management process

- There are two types of standards:
  - Product standards:
    - These apply to the product being developed
  - Process standards:
    - These standards define the processes that should be followed during software development

Product vs Process Standards

Product Standards	Process Standards
Design review form template	Design review conduct
Requirements document structure	Design validation process
Documentation standards	Version release process
Coding standards to follow	Project plan approval process
Project plan format	Change control process
Change request from template	Test recording process

Product vs process standards

Documentation standards

- Why are documentation standards important?
  - documents are the tangible manifestation of the software
- Documentation process standards
  - How documents should be developed, validated and maintained
- Document standards
  - Concerned with document identification, structure, presentation, changes highlighting, etc.
- Document interchange standards
  - How documents are stored and interchanged between different documentation systems
  - XML is an emerging standard for document interchange which will be widely supported in future

Software Standards and Systems

- Advantages of standards
  - Provide a framework around which the quality assurance process may be implemented
  - Provide encapsulation of best, or at least most appropriate, practice
  - Customers sometimes require a particular quality standard/level when choosing a software vendor
- Problems with standards
  - Not seen as relevant and up-to-date by software engineers
  - Involve too much bureaucratic form filling
  - Unsupported by software tools so tedious manual work is involved to maintain standards

Standards should not be avoided, but should be tailored as needed!

Software Standards and Systems

- Many standards and systems related to software quality exists today

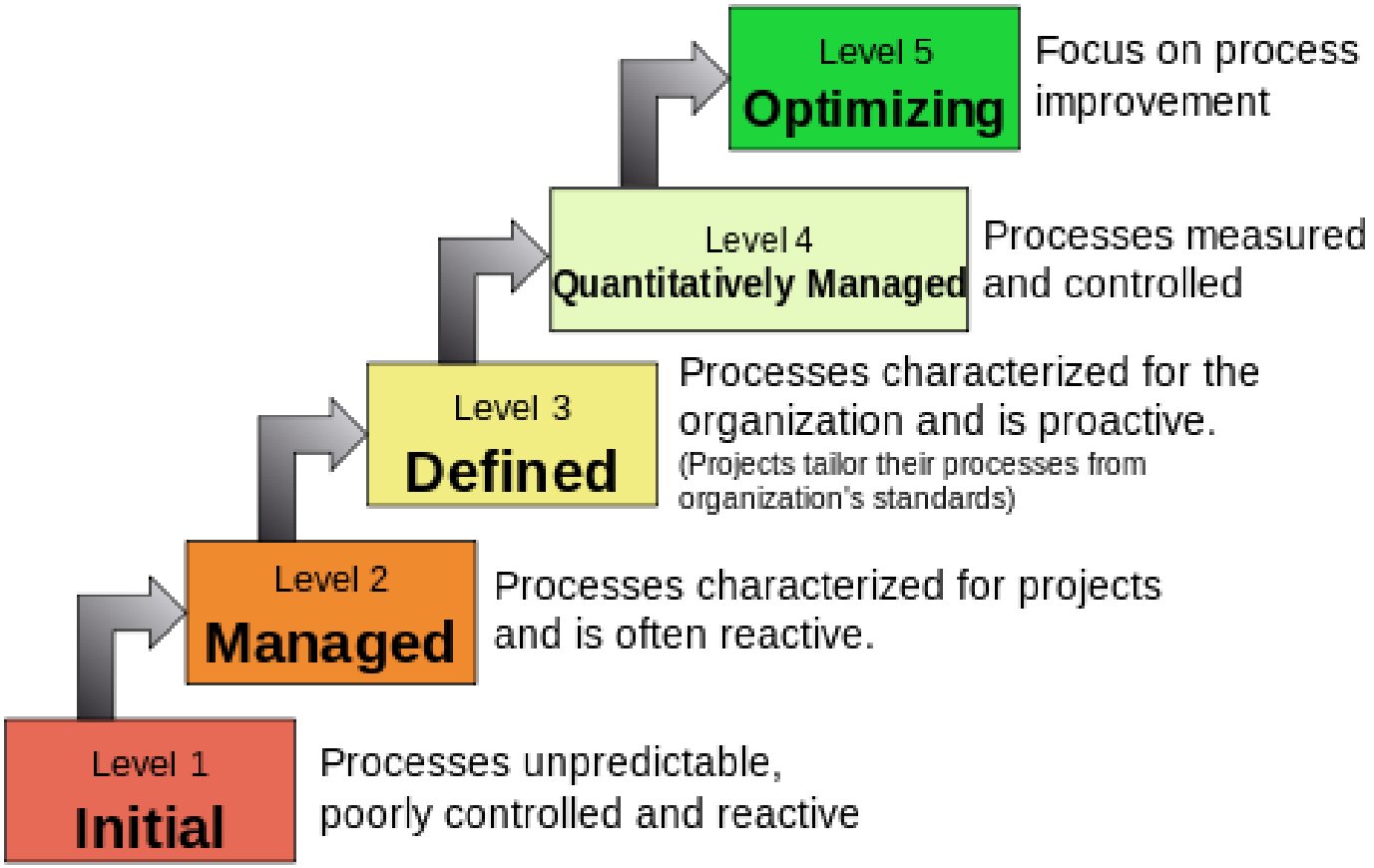
- Some examples of software standards and systems
  - ISO 9000
  - Capability Maturity Model

Capability Maturity Model

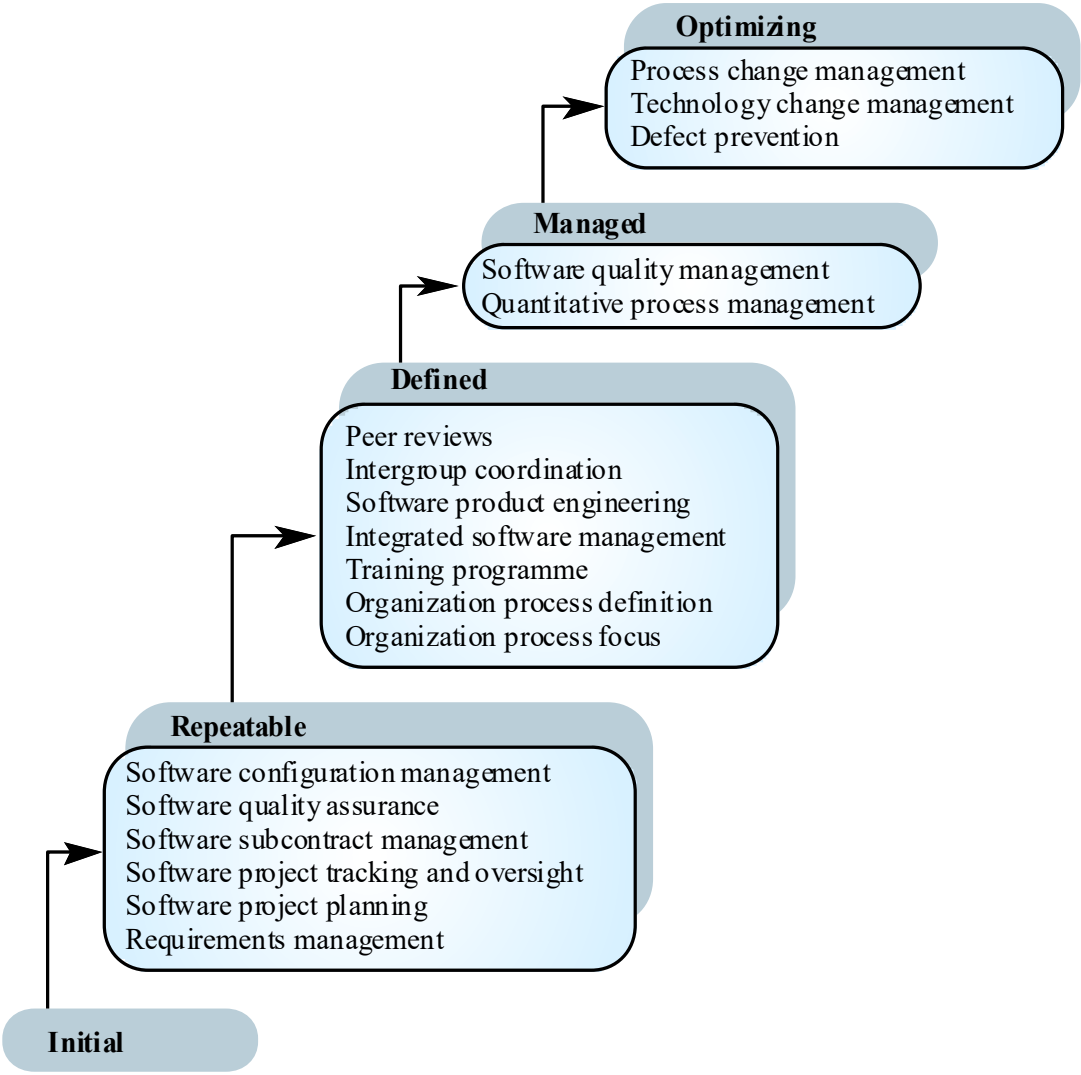
- Developed by the Software Engineering Institute (SEI) at Carnegie Mellon University
- Describes the key elements of an effective software development process
- Describes an approach for software companies to move from an ad-hoc, immature process to a mature developed process
- Organizations are characterised being at a Level from 1-5 based on the processes they follow

Capability Maturity Model

Characteristics of the Maturity levels



Capability Maturity Model



Learning Outcomes

1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
   - Quality Assurance
   - Quality Planning
   - Quality Control and Monitoring

Quality Planning

- The process of selecting those standards and systems that are appropriate to a particular organization and project

- The outcome of the planning process is a:
  - Software Quality Plan (SQP), sometimes called a Software Quality Assurance Plan (SQAP)

SQP - Template

- Software Quality Assurance Plan
  - Product Overview

    A description of the product, intended market, and quality expectations
  - Product Plan

    The critical release dates and responsibilities – could point to the schedule
  - Quality Goals

    The quality goals and plans for the product, including identification and justification of critical product quality attributes
  - Process Description

    The quality assurance processes that should be used for product development and management (reviews, audits etc)
  - Document and Coding Standards

    Standards for the documents and coding standards
  - Risks and Risk Management

    The key risks that might affect product quality and the actions to address these risks (could provide a link to appropriate risks in the Risk Management Plan)

Software Quality Attributes

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Software quality attributes

- Some of the quality attributes matter only matter to developers while others matter to end-users
- It is not possible for any system to be optimised for all attributes – trade-off is necessary to select the most important ones

Verification vs Validation

- Verification and Validation (V &V) are important aspects of quality assurance
- **Verification:**
  - Verification is an attempt to ensure that the product is built correctly, in the sense that the output products of an activity meet the specifications imposed on them in previous activities.
  - Verification normally involves two (sets of) artifacts: req. spec. vs design, design vs code; this is an internal developer activity.
  - Verification is ensuring you are *building the system right* (the right way).
- **Validation:**
  - Validation is an attempt to ensure that the right product is built—that is, the product fulfills it specific intended purpose.
  - Validation involves going back to the stakeholders to check if the product meets their requirements; this normally involves something/someone external.
  - Validation is ensuring that you are *building the right system* (to meet stakeholder needs).

Reviews

- ***Review*** is a common technique used for verification and validation

- Artefacts produced during the development process are reviewed as a way of identifying problems seeking ways to improve them early

- Three common types of reviews:
  - Technical Reviews
  - Business Reviews
  - Management Reviews

Technical Reviews

- Reviews of artefacts is performed by **peers** in the development team but the author/s are involved
- The aim is uncovering problems in an artefact and seeking ways to improve the artefact
- Is considered a “soft” method for quality assurance - that is, nothing is executed
  - Some developers greet reviews with scepticism - however, empirical evidence suggests that such scepticism is unjustified

Technical Reviews

- Advantages of technical reviews:
  - **Can be performed on any software artefact**, whereas many “hard” methods of quality assurance, such as testing and measurement, can be performed only on executable artefacts.
  - **Earlier detection of problems** in software artefacts leads to lower costs of resolution.
  - Studies show that roughly 30-70% of all programming faults found in a project were located using **source code reviews**, and up to 80% according to studies performed by IBM. Some studies demonstrated that **review techniques found several types of faults that testing failed to find**, and vice-versa**.**
  - **Reviews find the actual faults in source code**, in contrast to testing, which merely indicates that there is a fault somewhere in the program. After a fault is detected with testing, it must then be located.
  - Due to internal pressure of getting software releases out the door, programmers make more mistakes when correcting faults that were found during testing than they do **correcting faults during the review phase**

Technical Reviews

- Disadvantages of technical reviews:
  - Could be time and resource consuming
  - Should be carefully planned and executed to get the desired outcomes

- Types of technical reviews
  - Informal Reviews
  - Formal Reviews
  - Walk throughs
  - Code inspections
  - Audits

Technical Reviews

- **Informal Reviews:**
  - A simple desk check or casual meeting with a colleague which aims to improve the quality of a document
  - No formal guidelines or procedures that are followed
  - The effectiveness of informal reviews is considerably less than formal reviews, because of the lack of diversity found in a group
  - Checklists are tools that can help to improve the effectiveness of a review.
  - A checklist is a list of questions that the reviewer must answer about an artefact, however, the questions are generic questions about that type of artefact
  - Less time and resource consuming than a formal review

Technical Reviews

Example checklist for a Requirements Specification

Checklist for software requirements specification artifact

Organisation and Completeness

- ☐ Are all internal cross-references to other requirements correct?
- ☐ Are all requirements written at a consistent and appropriate level of detail?
- ☐ Do the requirements provide an adequate basis for design?
- ☐ Is the implementation priority of each requirement included?
- ☐ Are all external hardware, software, and communication interfaces defined?
- ☐ Have algorithms intrinsic to the functional requirements been defined?
- ☐ Does the specification include all of the known customer or system needs?
- ☐ Is the expected behaviour documented for all anticipated error conditions?

Correctness

- ☐ Do any requirements conflict with or duplicate other requirements?
- ☐ Is each requirement written in clear, concise, unambiguous language?
- ☐ Is each requirement verifiable by testing, demonstration, review, or analysis?
- ☐ Is each requirement in scope for the project?
- ☐ Is each requirement free from content and grammatical errors?
- ☐ Is any necessary information missing from a requirement? If so, is it identified as "to be decided"?
- ☐ Can all of the requirements be implemented within known constraints?
- ☐ Are any specified error messages unique and meaningful?

Quality Attributes

- ☐ Are all performance objectives properly specified?
- ☐ Are all security and safety considerations properly specified?
- ☐ Are other pertinent quality attribute goals explicitly documented and quantified, with the acceptable tradeoffs specified?

Traceability

- ☐ Is each requirement uniquely and correctly identified?
- ☐ Is each software functional requirement traceable to a higher-level requirement (e.g., system requirement, use case)?

Special Issues

- ☐ Are all requirements actually requirements, not design or implementation solutions?
- ☐ Are all time-critical functions identified, and timing criteria specified for them?
- ☐ Have internationalisation issues been adequately addressed?

Technical Reviews

- **Formal Reviews:**
  - A meeting with multiple stakeholders such as developers, testers, client
    - The group approach has benefits of bringing out different perspectives
  - Meeting should adhere to the following constraints
    - The review team should be 3-5 members carefully chosen
    - The meeting should last no longer than 90 minutes
    - Following are the critical roles
      - Review Leader: responsible for organizing the review
      - Author: at least one author should be present
      - Reviewers: at least two or three non-author stakeholders
      - Recorder: responsible for recording all important review comments
  - The review meeting could recommend one of the following:
    - Accept without further changes
    - Accept with proposed changes
    - Reject the artefact – this requires a re-review after modifications

Technical Reviews

- **Walkthroughs**
  - Walkthrough could be for code or a document
  - This is a review process where the author (the programmer or designer) leads a group of reviewers
  - Following are the main differences from a formal review:
    - Moderator, that leads the review is the author of the artefact being reviewed
    - Reviewers do not need preparation
    - When defects or inconsistencies are found, possible solutions are discussed

- **Code Inspections**
  - These are very similar to formal reviews, expect that the focus is on the code

Technical Reviews

- **Audits**
  - Reviews of processes and products to determine if a particular product or process conforms to standards
  - It is a type of technical review where the authors of the artefact being audited are not involved in the audit process at all – all the other roles are similar to a formal review
  - Audits are typically performed by a team that is completely external to an organisation
  - Two types of audits:
    - Product audits: to confirm that the product meets the standards
    - Process audits: to ensure that the team follows processes

Business Reviews

- The goal of a business review is to ensure that the IT solution provides the functionality specified in the project scope and requirements document
- A business review can include all project deliverables to ensure that:
  - It is complete
  - Provides the information needed to move to the next phase or process
  - Meets the standards

Management Reviews

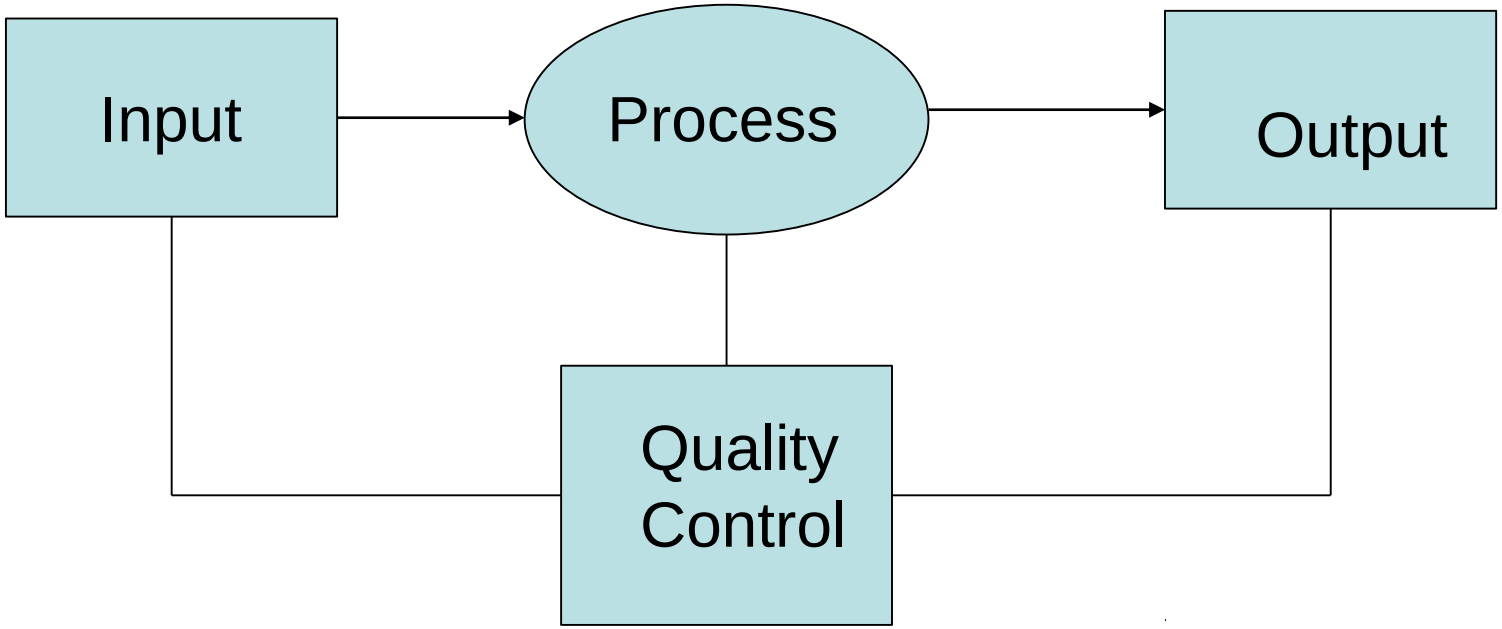
- Compares the project's actual progress against a baseline project plan
- Project Manager is responsible for presenting the project progress and providing a clear picture of the current status
- Issues need to be resolved – e.g. resources reallocated as needed, change to the project course if needed
- May involve reviewing if the project meets the scope, schedule, budget and quality objectives

Learning Outcomes

1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
   - Quality Assurance
   - Quality Planning
   - Quality Control and Monitoring

Quality Control and Monitoring

- Involves monitoring the software development process to ensure that the quality assurance procedures and standards specified in the SQP are being followed



Learning Outcomes

1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
   - Quality Assurance
   - Quality Planning
   - Quality Control and Monitoring

Lecture Break

BREAK

Please return promptly as the

Lecture will re-start in *5 mins*

Learning Outcomes

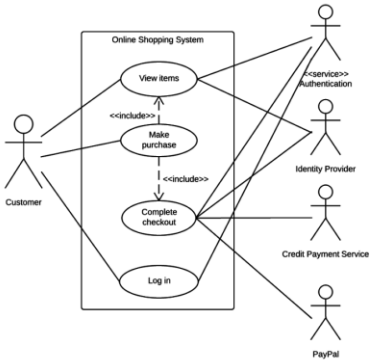
1. Understand the role of configuration management

2. Understand the configuration management process

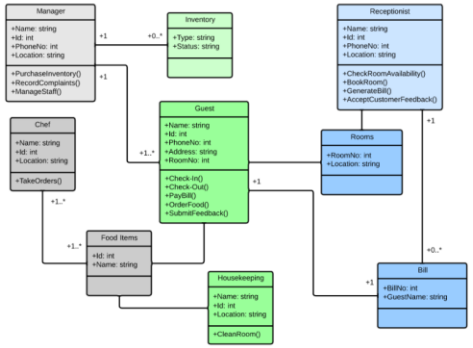
3. Understand the tasks associated with configuration management

What is a Software Configuration?

- Software projects generate a large number of different types of artefacts – e.g.:



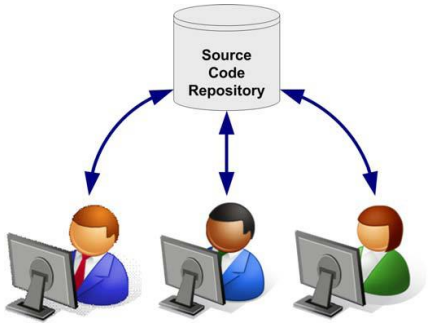
Use-case diagrams



Class diagrams

ITEM	DESCRIPTION
Title	A unique and descriptive title for the test case
Priority	The relative importance of the test case (critical, nice-to-have,etc.)
Status	For live systems, an indicator of the state of the test case. Typical states could include : Design – test case is still being designed Ready – test case is complete, ready to run Running – test case is being executed Pass – test case passed successfully Failed – test case failed Error – test case is in error and needs to be rewritten
Initial configuration	The state of the program before the actions in the "steps" are to be followed. All too often this is omitted and the reader must guess or intuit the correct pre-requisites for conducting the test case.
Software Configuration	The software configuration for which this test is valid. It could include the version and release of software-under-test as well as any relevant hardware or software platform details (e.g. WinXP vs Win95)
Steps	An ordered series of steps to conduct during the test case, these must be detailed and specific. How detailed depends on the level of scripting required and the experience of the tester involved.
Expected behaviour	What was expected of the software, upon completion of the steps? What is expected of the software. Allows the test case to be validated with out recourse to the tester who wrote it.

Test Cases



Source Code

Software Requirements Specification

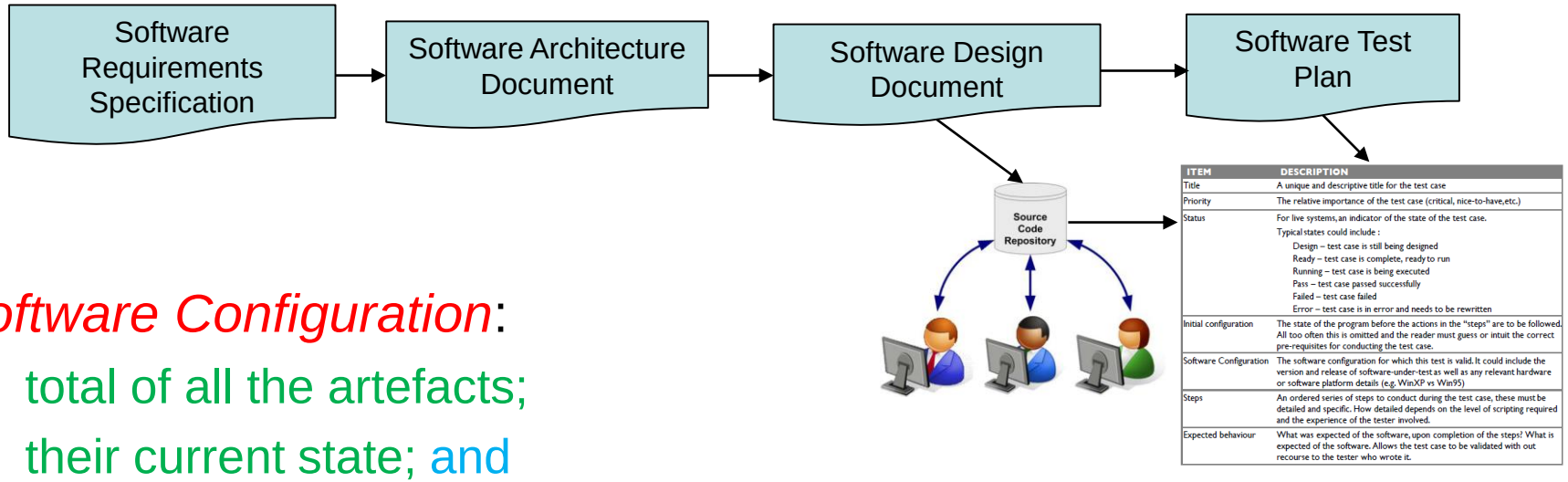
Software Architecture Document

Software Design Document

Software Test Plan

What is a Software Configuration?

- There are *dependencies* between all of these artefacts.
  - For example, a code module may depend on a design element such as a class diagram or state chart, as well as on a design element such as a design class diagram. In turn these may depend on a combination of textual requirements, use-cases and analysis classes.



- *Software Configuration*:
  - total of all the artefacts;
  - their current state; and
  - the dependencies between them.

Role of Configuration Management

The problem is change!

- If we make a change to an artefact, it may impact all of that artefact's dependencies

- If we are not careful then changes to artefacts may leave the configuration in an inconsistent state
  - For example, a change to the requirement will have an impact on the system design and all of the code modules that depend on the design. Also, the test plan, test cases and testing scripts for the code will also be impacted. *The danger is that we may change one module without changing one of its dependent modules leaving the configuration inconsistent.*

Role of Configuration Management

- The aim of configuration management is to manage change properly without losing overall consistency through:
  - establishing processes;
  - setting up repositories; and
  - using other appropriate tools and techniques

- Configuration Management (CM) addresses the following:
  - How do we manage requests for change?
  - What and where are the software components?
  - What is the status of each software component?
  - How does a change to one component affect others?
  - How do we resolve conflicting changes?
  - How do we maintain multiple versions?
  - How do we keep the system up to date?

Learning Outcomes

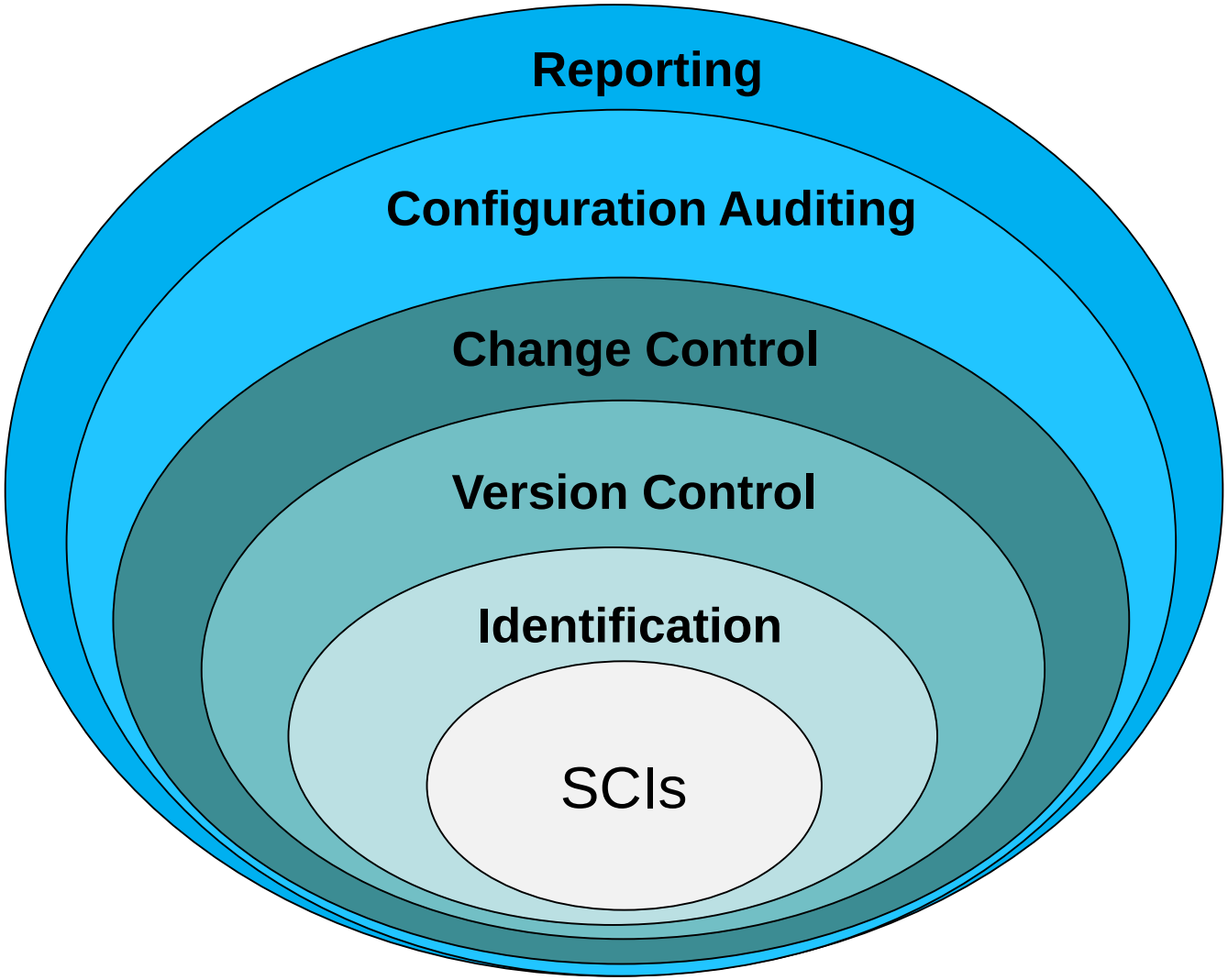
1. Understand the role of configuration management
2. Understand the configuration management process
3. Understand the tasks associated with configuration management

CM Processes

CM Aims:

1. To identify all items that collectively will make up the configuration
2. To manage changes to one or more of these items so that the collection remains consistent
3. To manage different versions of the product
4. To assure software quality as the configuration evolves over time

CM Tasks


Reporting
Configuration Auditing
Change Control
Version Control
Identification
SCIs

CM Tasks

- Identification
  - the configuration items necessary for the project are identified
- Version control
  - processes and tools are chosen to manage the different versions of configuration items as they are developed
- Change control
  - changes that affect more than just one configuration item are managed
- Configuration auditing
  - the consistency of the configuration is checked
- Configuration reporting
  - the status of configuration items is reported

Learning Outcomes

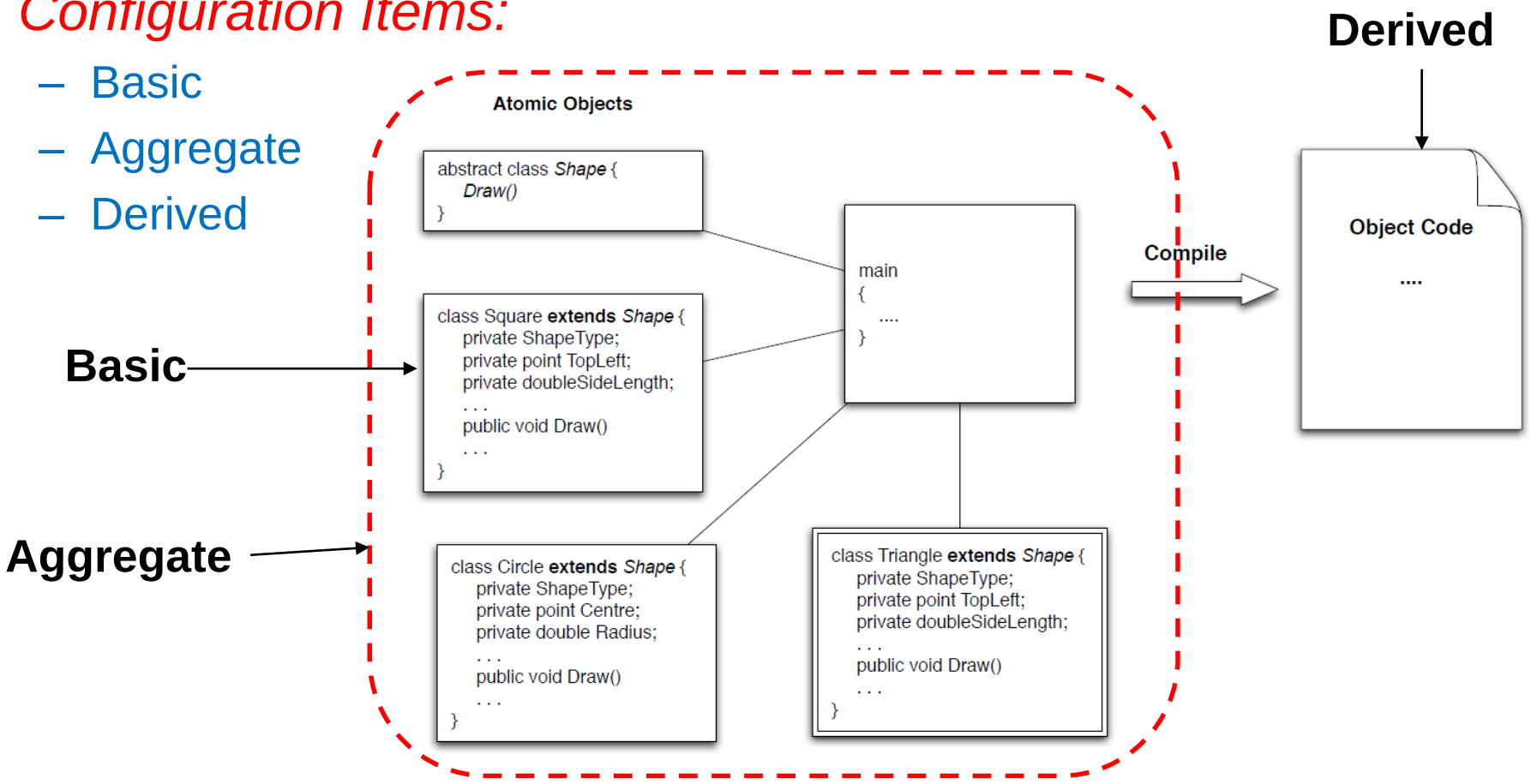
1. Understand the role of configuration management

2. Understand the configuration management process

3. Understand the tasks associated with configuration management

Identification

- The set of artefacts that require configuration management are called the *configuration items*
- *Configuration Items:*
  - Basic
  - Aggregate
  - Derived



Identification

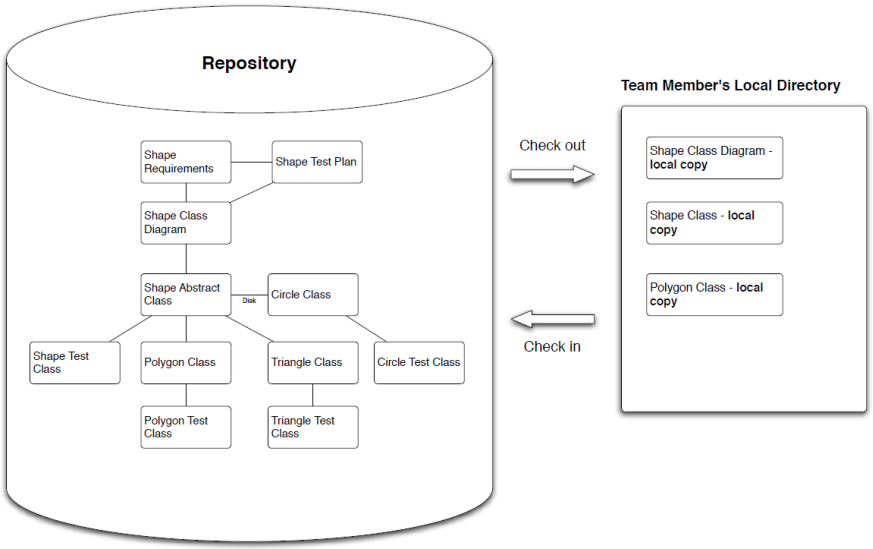
- A typical list of configuration items
  - requirements specifications, requirements models, sections of the requirements specification, and individual requirements
  - use-cases, user stories
  - design models, design documents, design elements, and class designs
  - source code modules
  - object code modules
  - release modules
  - software tools
  - test drivers and stubs, and test scripts
  - documents or sections of documents associated with the project

Version Control

- Requirements for a version control system:
  1. A *repository* for storing configuration items
  2. A *version management function* that allow software engineers to create and track versions, and roll the system back to previous versions if necessary – e.g. git, svn, cvs
  3. A *make-like facility* that allows engineers to collect all of the configuration objects for a particular target together and to build that target – *e.g. Apache Maven, Apache Ant, make (unix, linux)*

Version Control

- SCM information is maintained in a *repository* or *configuration database*



Version: An instance of a model, document, code, or other configuration item which is functionally distinct in some way from other system instances.
Variant: An instance of a system which is functionally identical but non-functionally distinct from other instances of a system.
Release: An instance of a system which is distributed to users outside of the development team.

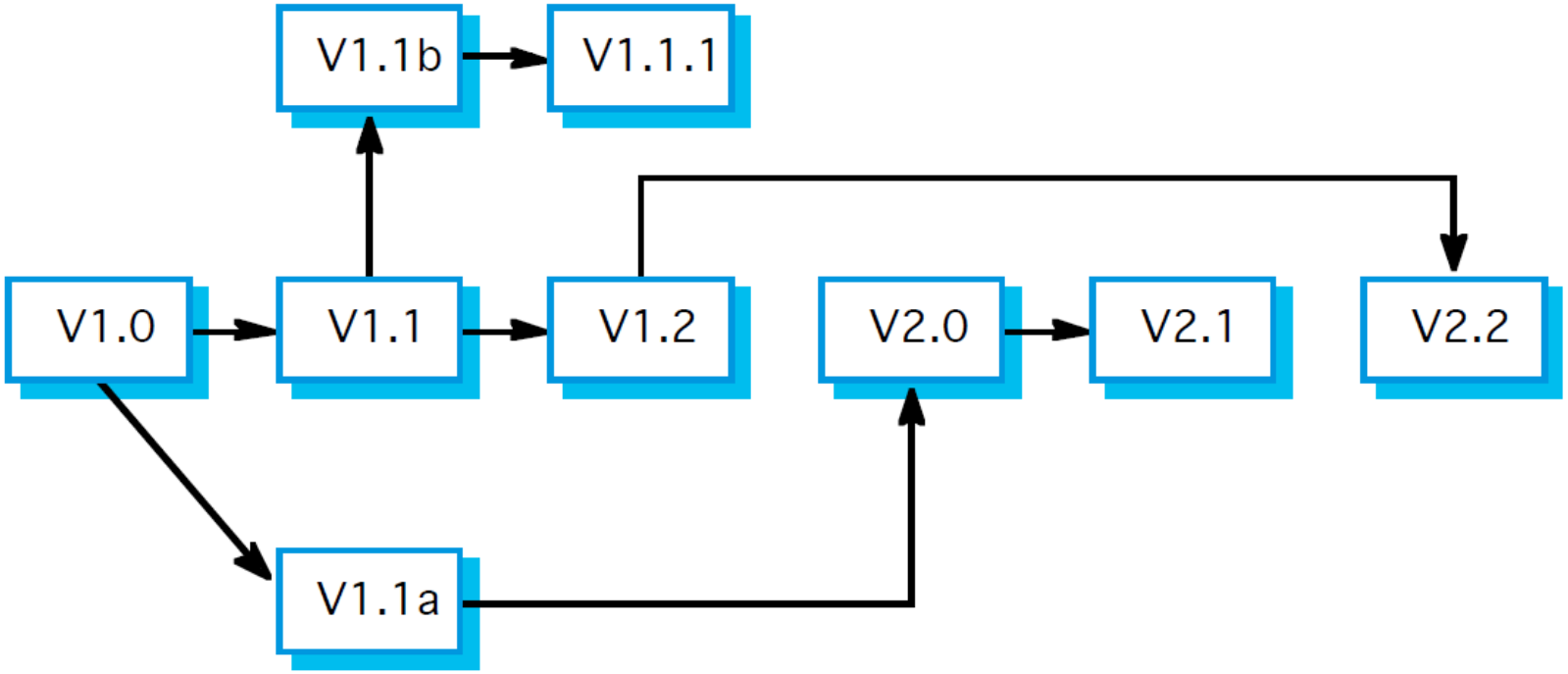
Version Control

- *Derivation History:*
  - This is a record of changes applied to a configuration object

- Each change should record:
  - the change made
  - the rationale for the change
  - who made the change
  - when it was implemented

- A common method of tracking versions in a repository is through *version numbering*
  - Version numbers could have meanings – for example a reviewed version of a document (major versions) vs un-reviewed changes

Version Control



A derivation structure for a project using version numbering to mark branches and merges

Change Control

- *Change Management Plan*
  - A part of an overall configuration management plan to specifically control these changes to the configuration
  - Changes must be made in a way that allows everyone on the project team to find out:
    - exactly what changes need to be made
    - what they need to do to affect the change
    - why the change is being made
    - how it will impact them

  More importantly, in distributed control structures, some changes may need to be carefully negotiated so that everyone understands the need for the change and supports it

Change Control

Element	Impact on the Process
Initiate the Change	Why is the change being made? What information will be needed to evaluate the change? How will the change be evaluated?
Evaluate the Change	How will the change affect the configuration? Which artefacts need to change and what are their dependencies? What are the benefits of the change? What are the costs of the change? Do the benefits of the change outweigh the costs of the change? Who will be impacted by the change?
Making the Change	Who will put the change into effect? How will the change be managed? How will other people working on the project understand the change? How will they be notified of the change? How will people working on the project know when the change is completed?

Change Control

- ***Baseline***
  - A baseline is an artefact that is *stable*
  - It has been formally reviewed and agreed upon, that is now ready for future development
  - It can only be changed through a *formal change management procedure*

Configuration Auditing

- *Configuration audits:*
  - complement the other configuration management activities by assuring that what is in the repository is actually consistent and that all of the changes have been made properly

Have the changes requested and approved been made?	Have any additional changes other than required by a request been made?
Did the configuration objects that were changed pass their quality assurance tests?	Do the objects in the configuration meet the required external standards?
Do the attributes of the configuration item match the change?	Does every configuration item have appropriate change logs?

Typical questions for a configuration audit

Status Reporting

- *Status Reporting*
  - Is a common way for large projects to keep track of the status of the repository
  - The idea is to review the configuration objects for consistency with other configuration objects, to find any omissions or to look for potential side effects
  - Status reporting can take many forms, but most commonly the aim is to report on the status of the configuration items of interest and the baselines that have been achieved
    - For example, we may have a design element that is in one of the states: not-initiated, initial-work, modified, approved, baselined – the status report can compare the state with what is in the project schedule

Learning Outcomes

1. Understand the role of configuration management

2. Understand the configuration management process

3. Understand the tasks associated with configuration management

References

1. R. S. Pressman. Software Engineering: A Practitioner's Approach. McGraw Hill, seventh edition, 2009.

2. I Somerville. Software Engineering, Addison-Wesley Publishing, ninth edition, 2010.

3. ISO. Information technology – software product evaluation – quality characteristics and guidelines for their use, international organization for standardization. International Standard ISO/IEC 9126, International Electrotechnical Commission, Geneva, 1991.

References

4. Marco Palomino, Abraham Dávil, Karin Melendez, Marcelo Pessoa. Agile Practices Adoption in CMMI Organizations: A Systematic Literature Review. International Conference on Software Process Improvement, 2016.