SWEN90016
Software Processes & Project Management

Configuration Management

Marion Zalk
Department of Computing and Information Systems
The University of Melbourne
mzalk@unimelb.edu.au

2020 – Semester 1
Lecture 8

Learning Outcomes

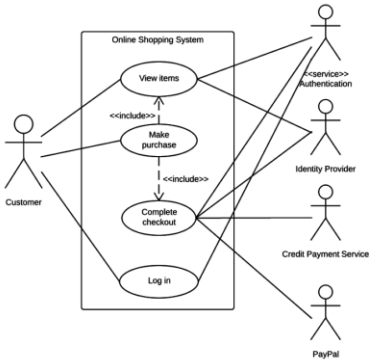
1. Understand the role of configuration management

2. Understand the configuration management process

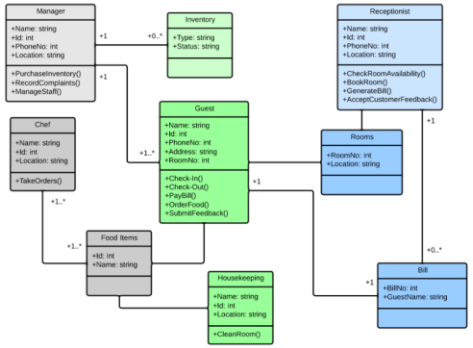
3. Understand the tasks associated with configuration management

What is a Software Configuration?

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



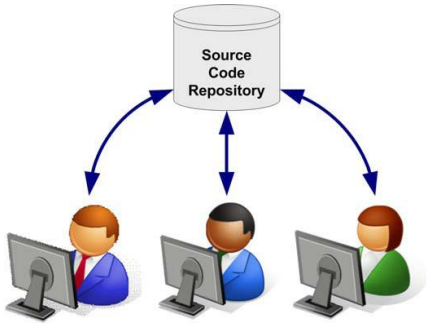
Use-case diagrams



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

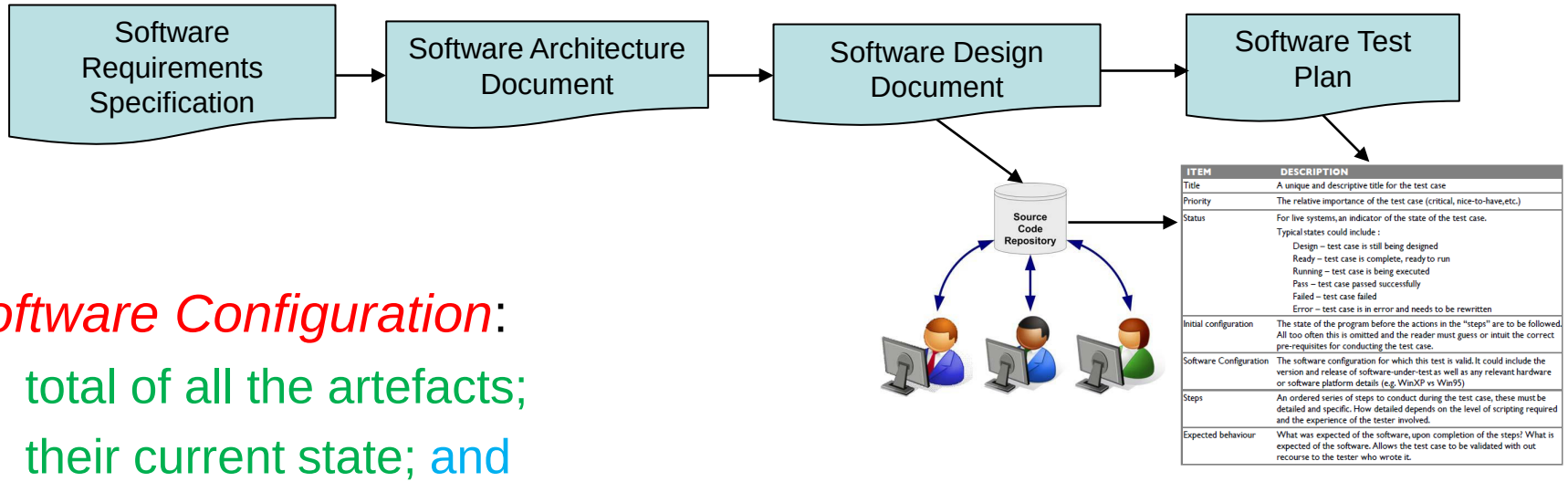
Software Architecture Document

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

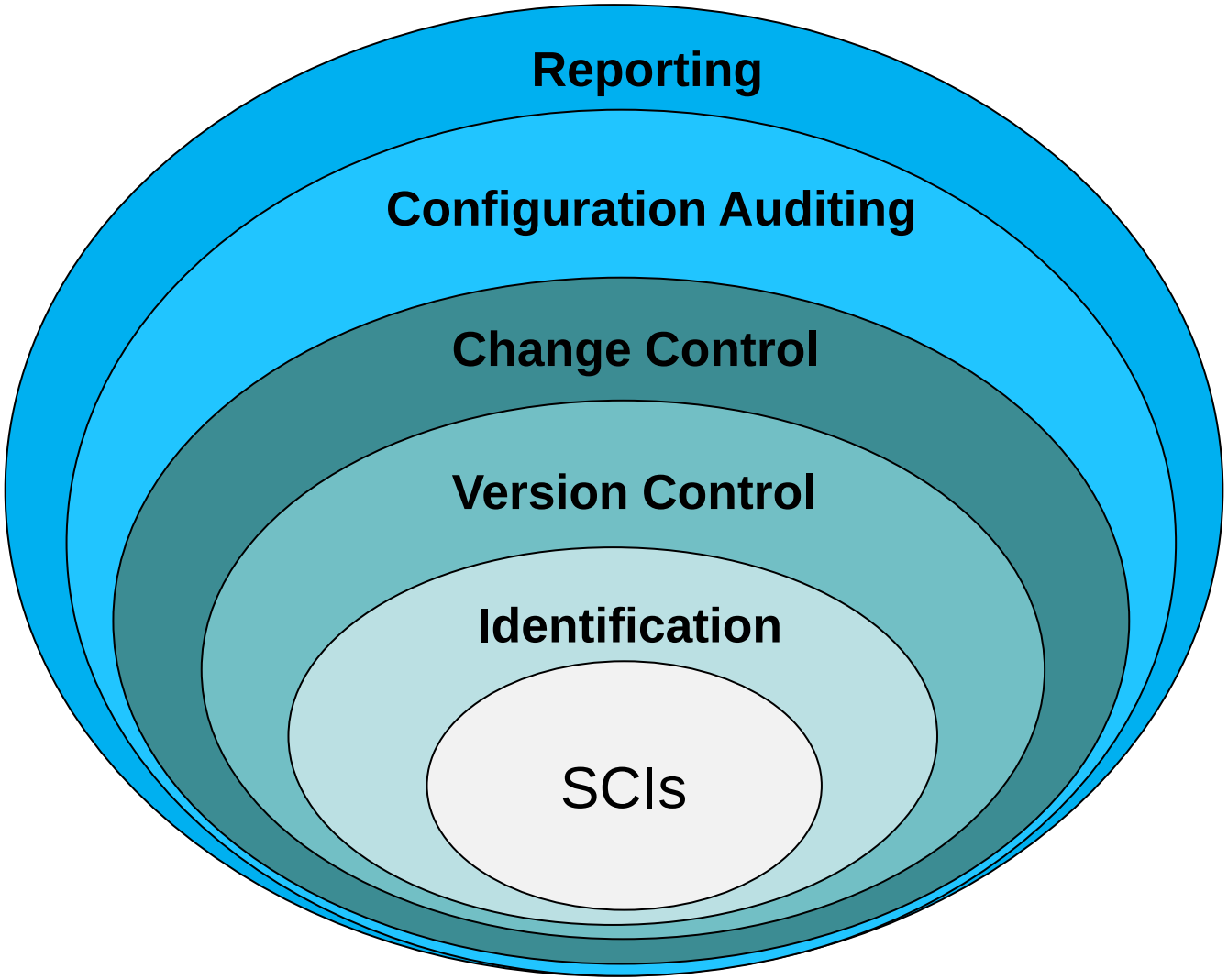
1. Understand the role of configuration management
2. Understand the configuration management process
3. Understand the tasks associated with configuration management

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

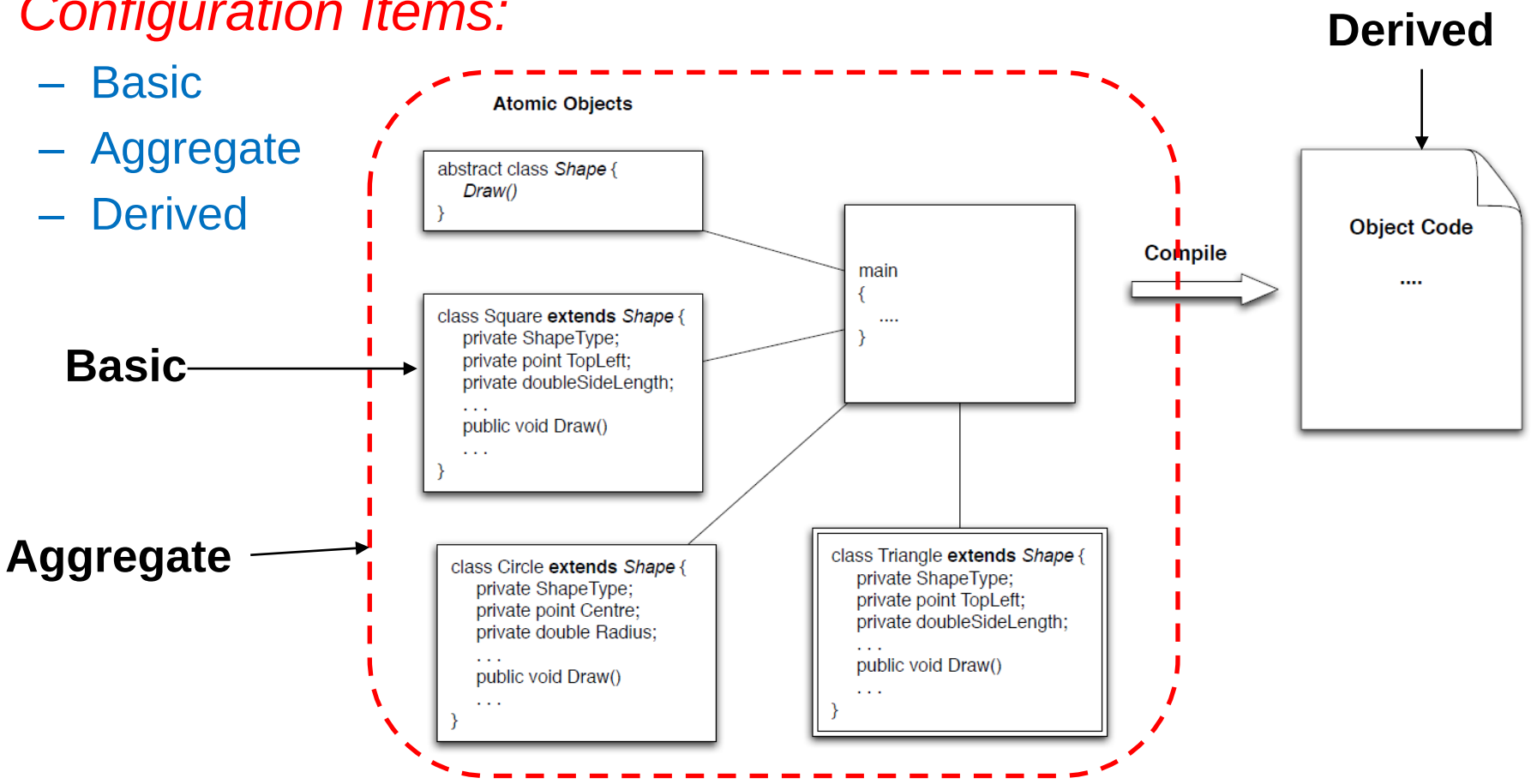
1. Understand the role of configuration management

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

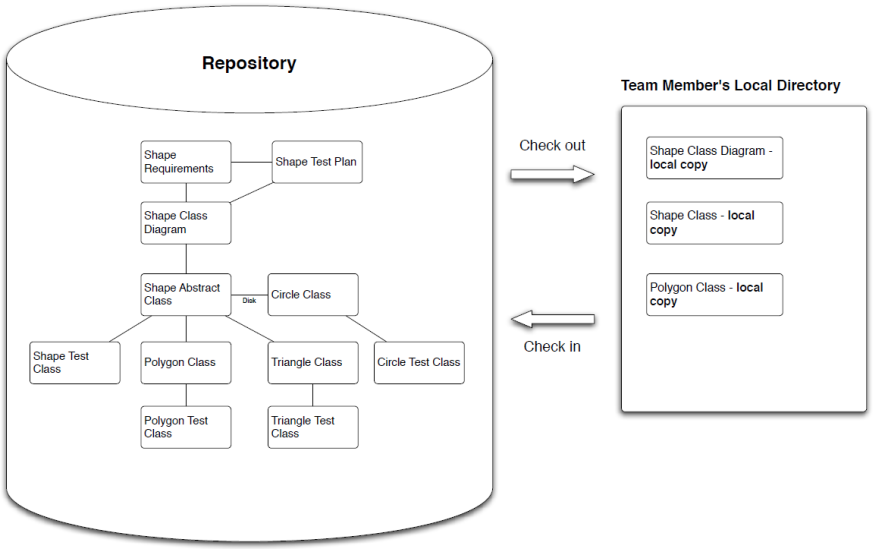
- A typical list of configuration items
  - requirements specifications, requirements models, sections of the requirements specification, and individual requirements
  - use-cases, user stories
  - design models, design documents, design elements, and class designs
  - source code modules
  - object 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.

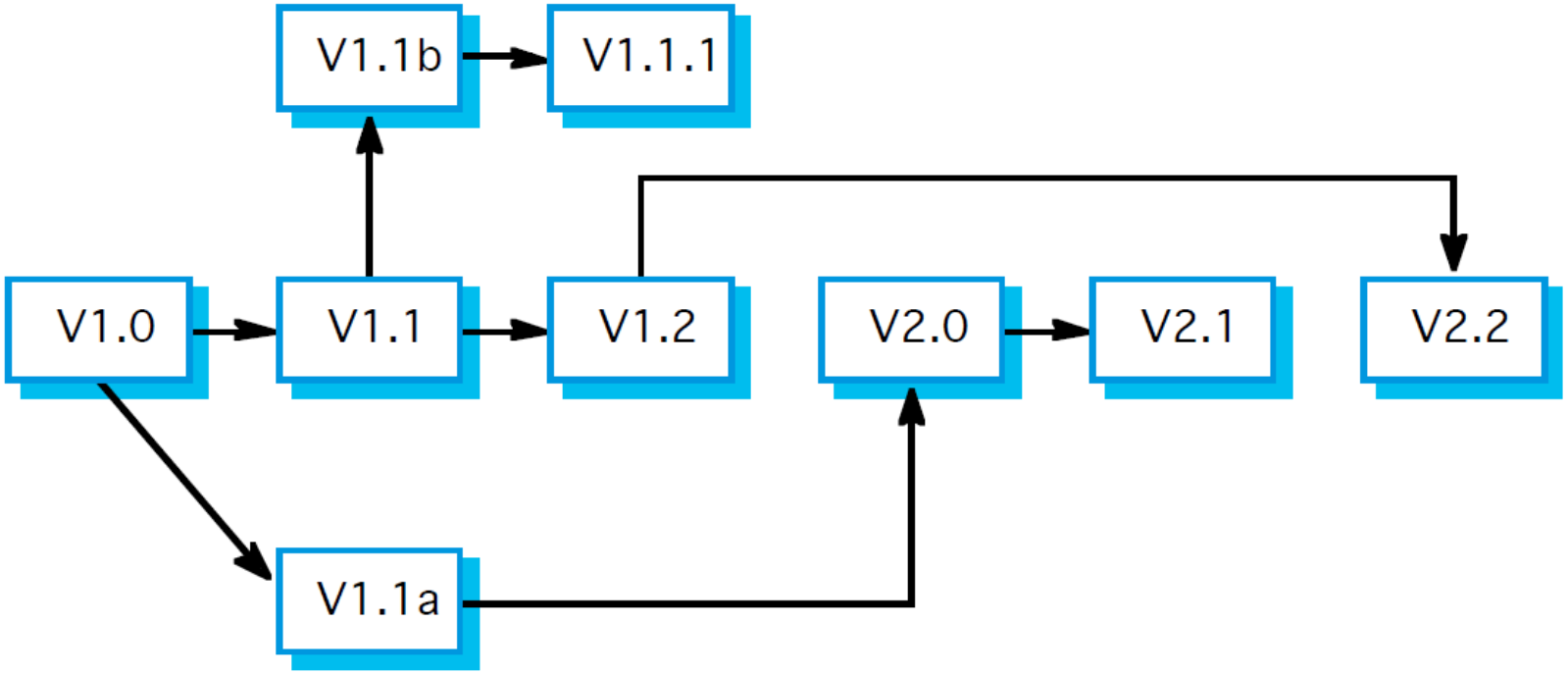
Version Control

- *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.