SOFTHOUSE

CONFIGURATION MANAGEMENT

på fem minuter

INTRODUCTION TO CONFIGURATION MANAGEMENT

Fundamentals useful for EEE

CONFIGURATION MANAGEMENT - DEFINITION

A controlled process for managing development and change of documents, source code and product releases, during the complete life cycle.

BACKGROUND

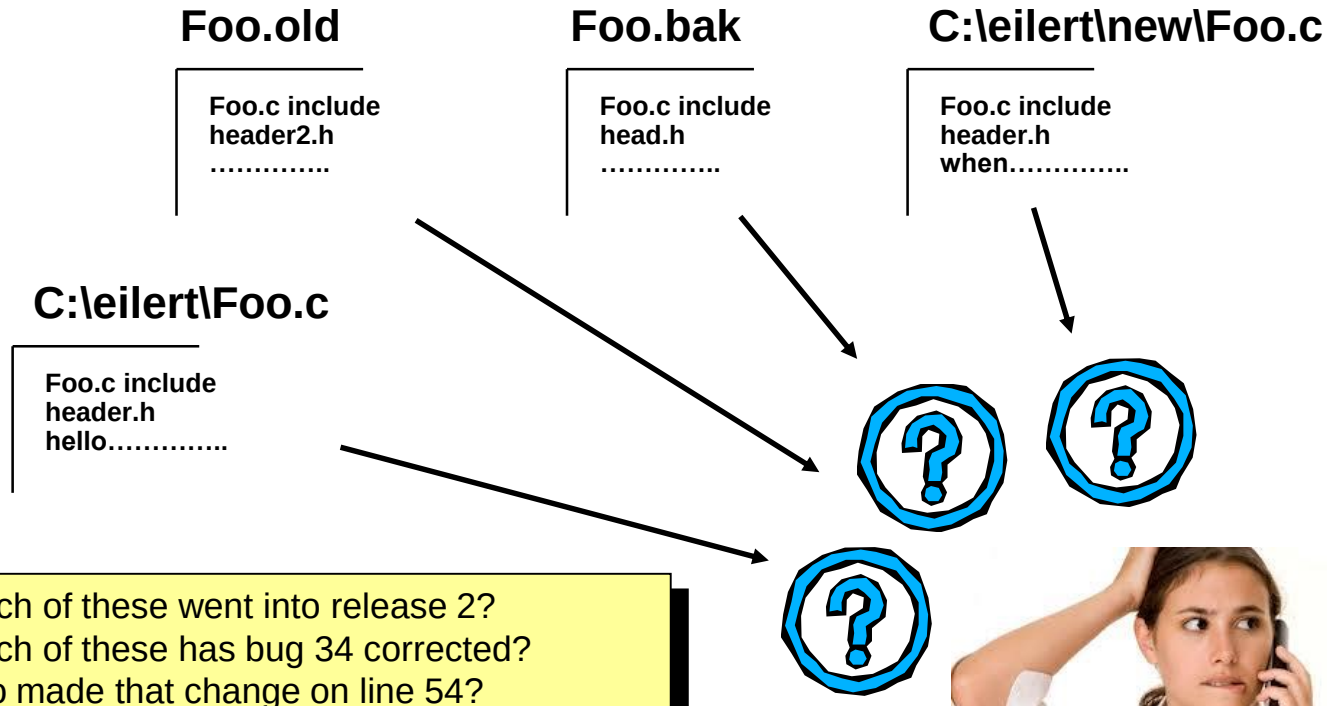
- Started in US military in the 1950's
  - Only hardware oriented since software did not exist yet
  - Process developed to ensure that a unique product could be manufactured in exactly the same way multiple times
- Best practice
  - Processes developed at many major companies during many years
  - CM principles are the same regardless of type of business
- Standards
  - ITIL (Information Technology Infrastructure Library)
  - CMMI (Capability Maturity Model Integration)
  - SPICE (Software Process Improvement and Capability Determination)
  - ISO (International Organization for Standardization)
  - IEEE (Institute of Electrical and Electronics Engineers)
- Today
  - Today CM is an integrated part of the development process in major companies with substantial product development
  - CM is a part of quality assurance in development and maintenance projects

CM ENABLES CONTROL

Examples on issues addressed:

- Can you show me the current versions of the software files for RCIOM, with a list of all changes made since November 1, 2012?
- Which of these 14 CR's were implemented in the last System Release?
- Can we rebuild the solution that worked last month?
- Which versions of this software are released in which branch?
- How do I see which System Specification Baselines that version 18 of this specification is included in?

THE CM NIGHTMARE



SUB-SECTIONS OF CONFIGURATION MANAGEMENT - 1 ACCORDING TO CM STANDARDS

- **Configuration Identification**
  - Identify configuration items, components, and related work products to be placed under configuration management and apply an identification system for all items
- **Configuration Control**
  - The process of controlling modifications to the system's design, hardware, software, and documentation.
- **Configuration Status Accounting**
  - The process of recording and reporting configuration item descriptions (e.g., hardware, software, firmware, etc.) and all departures from the baseline during design and production
- **Configuration Audits**
  - Independent review of hardware and software for the purpose of assessing compliance with established performance requirements, commercial and appropriate standards, functionality and product baselines.

SUB-SECTIONS OF CONFIGURATION MANAGEMENT – 2
COMMONLY USED TERMS

- **Version Control**
  - Having a structured way of storing produced work products
  - Increasing the quality of the development process by enabling tracking of all changes made
  - Facilitating parallel development. i.e. bug fixing at same time as new functions are developed for next release of the system
- **Change Management**
  - Change Requests
  - Defects
- **Release Management**
  - What is included in a release
  - How is a release distributed
  - How is a release documented
- **Build Management**
  - How different parts of a software are built together into a binary module

VERSION CONTROL - FUNDAMENTAL TERMS

- **Configuration Item (CI)**
  - A file or document, or a collection of files/documents that are considered to be an Item
  - Can also be called ”artifact”, but the term CI is used in this material
- **Version**
  - Sequentially stored changes of a CI
- **Baseline**
  - A specific version of each CI at a certain occasion, that can be used as a logical basis for test, further development or release
- **Branch**
  - Changes in a CI that are isolated from other changes in the same CI
- **Merge**
  - Include changes from different versions into the same version of a CI

VERSION CONTROL

- Version is a basic unit for distinguishing different content between two consecutive “issues” of a CI, with a relationship that is ordered in time

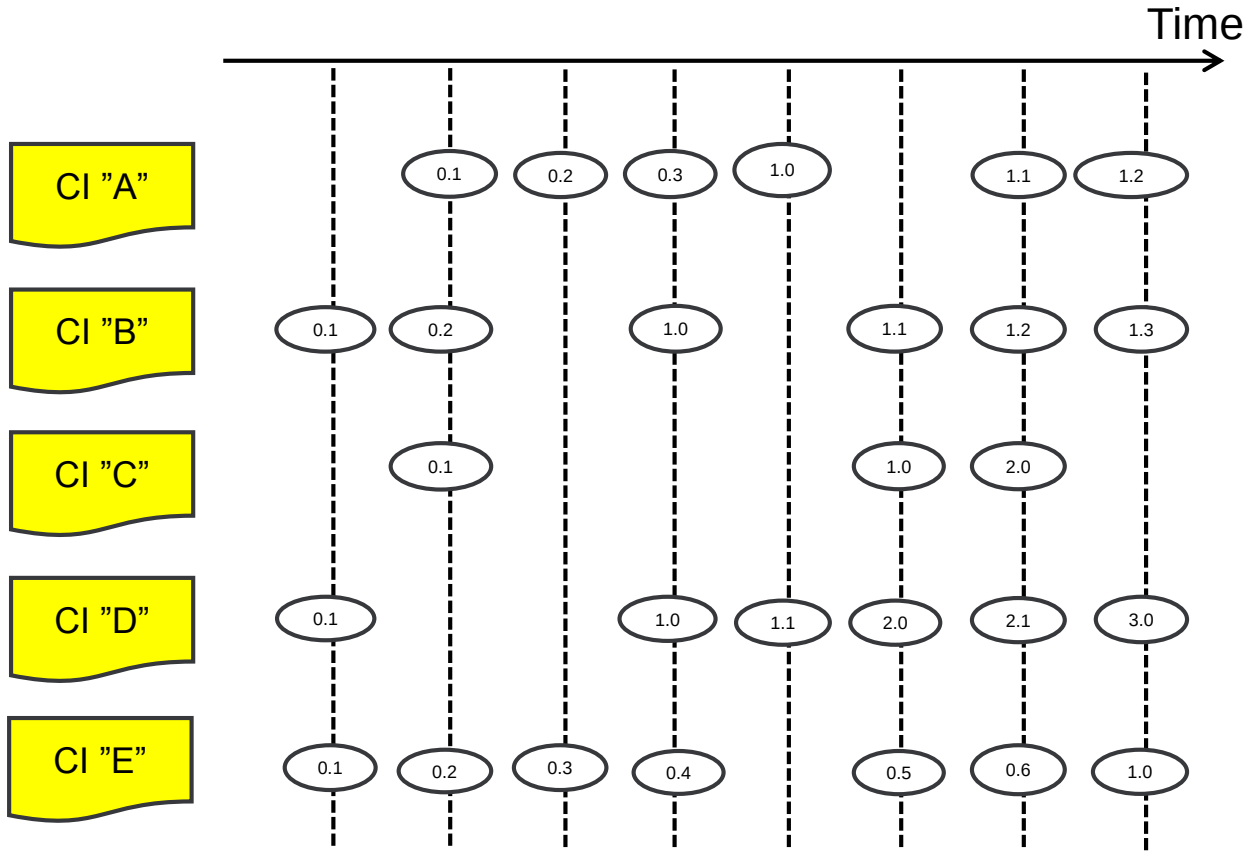
- CI’s that are under version control normally don’t change name. It would generate a new CI. Often a file name is important for identifying a CI, which makes it improper to change

- CI names don’t include version numbers, dates, or any other information that is valid for a specific version of the CI’s

VERSION NUMBERS

- CI's have automatically generated version numbers provided by the tool where the CI's are stored, which is a simple serial number
- Many of those CI's (e.g. documents) also have a more extensive version number with more digits, e.g. 5.2, specified in the document
- The version number of each CI is individual and independent of all other CI version numbers, still using a version number standard
- A version number can also be used for a collection of CI's that together compile a product
- Version numbers follow a consecutive number series. No number is skipped in such a consecutive number series
- Version numbers should not be mixed up with release names

THE PRINCIPLES FOR CONSECUTIVE VERSIONS OF CONFIGURATION ITEMS



BASELINE - 1

- Baselines are used for traceability reasons, to be able to go back to a certain situation that was important to preserve, and may be used as recovery point
- A baseline contains a specific version of all CI's in a specified collection, i.e. CI's that somehow belong together
- A baseline contain all CI's in the specified collection, not only the ones that are changed since last baseline
- The specific baselines have a certain purpose related to e.g. product evolution or specific project event
- Baselined CI's can only be changed via a formal change management process
- A baseline is not version handled or changed. Instead a new baseline is created if changes are made after the previous baseline was established

BASELINE - 2

- Example of baselines at Electrical department
  - System Specification Baseline E0
  - Component Delivery Baseline ABS_2.2.1.0_2
  - System Integration Baseline E0_VT06_1.0
  - System Release Baseline E0_VT_REL03_1.0

- A baseline contains a specific version of all CI's in a specified collection, i.e. CI's that somehow belong together

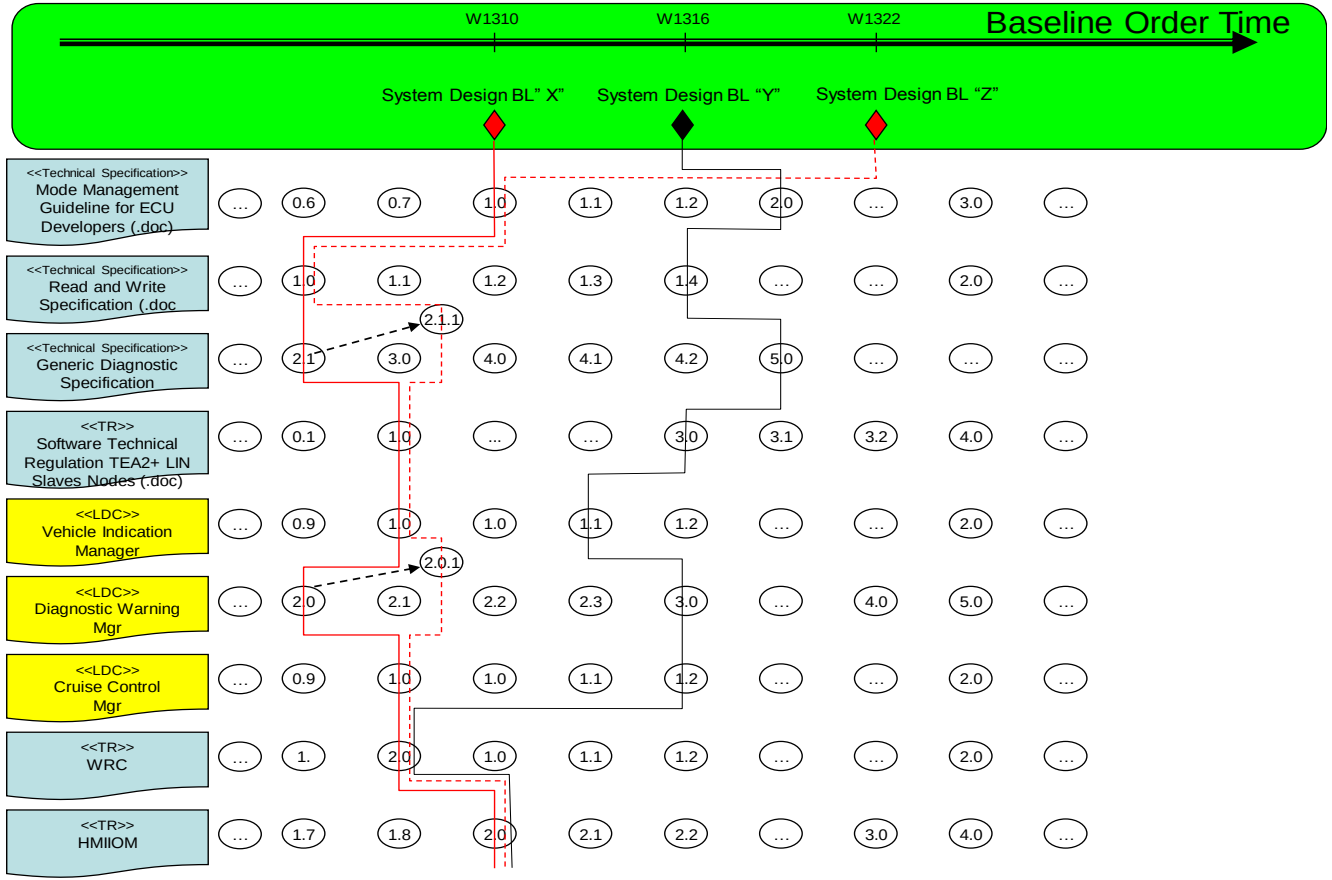
BASELINE - 3
VERSION OF A PLATFORM CONFIGURATION

A specific version of a platform configuration, e.g:

- E0_VT01_**1.0** E0_VT_REL01_**1.0** Charlie **1.0**
- E0_VT02_**1.0** E0_VT_REL02_**1.0** Delta **1.0**
- E0_VT03_**1.0** E0_VT_REL03_**1.0** Echo **1.0**

N.B. 1 Platform configuration version numbers are not connected. I.e. E0_VT01_**1.0** does not necessarily correspond to E0_VT_REL01_**1.0** nor to Charlie **1.0**.

BASELINE - 4


W1310
W1316
W1322
Baseline Order Time
System Design BL" X"
System Design BL "Y"
System Design BL "Z"
<<Technical Specification>>
Mode Management Guideline for ECU Developers (.doc)
… 0.6 0.7 1.0 1.1 1.2 2.0 … 3.0 …
<<Technical Specification>>
Read and Write Specification (.doc
… 1.0 1.1 1.2 1.3 1.4 … … 2.0 …
2.1.1
<<Technical Specification>>
Generic Diagnostic Specification
… 2.1 3.0 4.0 4.1 4.2 5.0 … … …
<<TR>>
Software Technical Regulation TEA2+ LIN Slaves Nodes (.doc)
… 0.1 1.0 … … 3.0 3.1 3.2 4.0 …
<<LDC>>
Vehicle Indication Manager
… 0.9 1.0 1.0 1.1 1.2 … … 2.0 …
2.0.1
<<LDC>>
Diagnostic Warning Mgr
… 2.0 2.1 2.2 2.3 3.0 … 4.0 5.0 …
<<LDC>>
Cruise Control Mgr
… 0.9 1.0 1.0 1.1 1.2 … … 2.0 …
<<TR>>
WRC
… 1. 2.0 1.0 1.1 1.2 … … 2.0 …
<<TR>>
HMIIOM
… 1.7 1.8 2.0 2.1 2.2 … 3.0 4.0 …

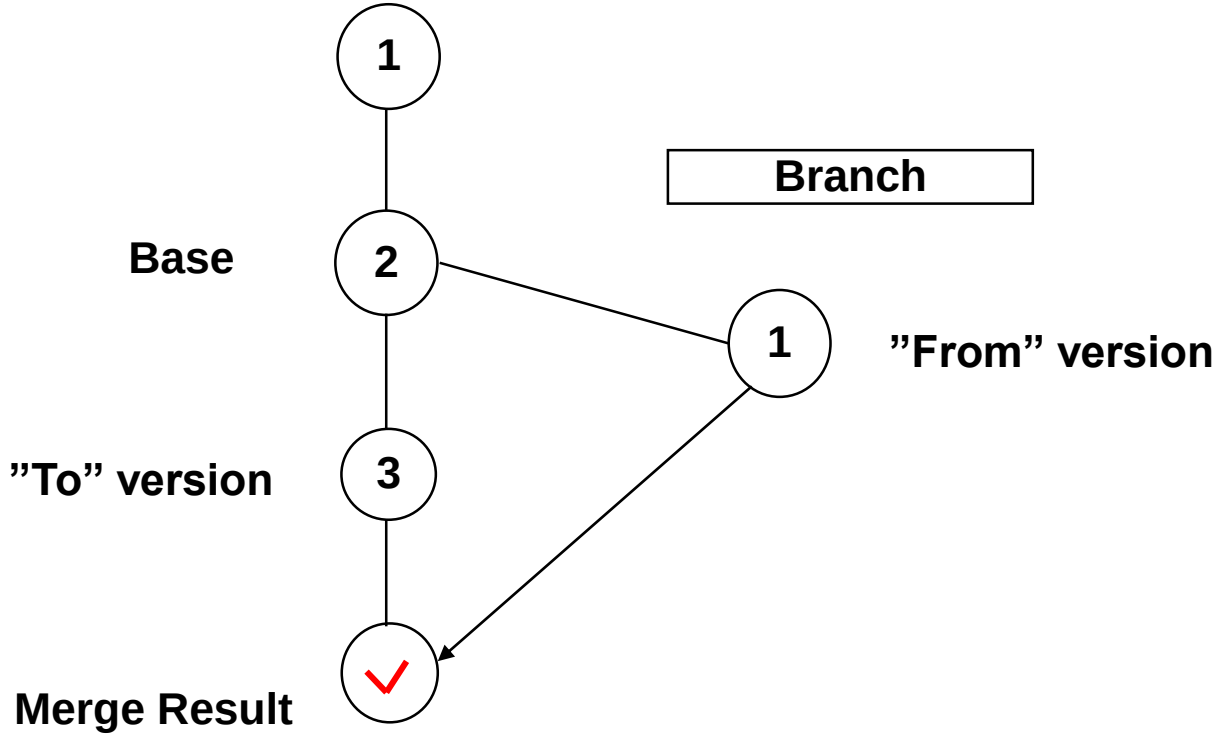
BRANCH

- A branch is a set of changes that are isolated from other changes in the same CI
- A branch can be created for different reasons, for example
  - Parallell development
  - Testing new functionality without disturbing other development
  - To support corrections/updates after a CI is formally released – at a baseline
- Branching can be applied on different levels, for example
  - On system level, e.g. Volvo/Renault
  - On code level, e.g. bug correction
  - For production branches, e.g. P2540
- Branching is applied in different ways depending on tool and storage used for the CI's
  - The simpliest way is to make a copy of a CI, or mark in the CI what is branched
  - Version handling tools have a more advanced functionality for branching

MERGE - 1

- Merge is mostly used when changes in different branches of a specific CI are included in the same (new) version of the CI
- Merge can also be done to gather changes from different CI's
- Version handling tools usually have functionality for automatic merge
- Merge must sometimes be done manually, if the CI type is binary or if a version handling tool is not used

MERGE - 2



CHANGE MANAGEMENT

- The purpose of Change Management is to have control of changes in relation to a predefined baseline
- Issues involved in Change Management can be divided into Change Requests, Defects, and sometimes Deviations
- Changes are controlled by using a Change Management tool, e. g. Jira or ClearQuest
- Changes should be prioritized and approved in CCB (Change Control Board)

RELEASE MANAGEMENT - 1

- The purpose of Release Management is to record and manage content, deviations and issues related to the new and/or changed deliverables, and take necessary corrective actions
- A release is baselined in order to preserve information about the release content
- A release contains new or changed software and/or hardware required to implement approved changes
- An important part of Release Management is to have a clear communication between sender and receiver of releases
- Releases can be of two types
  - Delta release, only part of products that are changed are included in the release
  - Full release, everything of concerned produts is included in the release
- A release should always include a Release Note

RELEASE MANAGEMENT - 2
SOME EXAMPLES OF PREVIOUS RELEASES

- ACM.0.6.20 / EEP5 / EEP5.0.y
- APM FP3.11 / W942:1
- BL1.3.0 / 08w38 / LDS0.7.0 / CSwC R4
- Rel FP2.1 08w45 / 08w50
- SW EEP5
- AECU R7 BTL02.08
- Audio_Head_Unit w1006_FP5.0_1
- OBSOLETE - CCIOM_SW.2.0.0
- AHU_W1117_EEP5.1.Y-e
- SW_EEP5.0_D4c
- 1.3.6.0

INFORMATION CATEGORIES

- **Product related information**
  - Describing the product throughout its life cycle, including requirement, implementation and verification specifications
  - Information that can not be removed after project closure
  - Must be traceable, under version control, and included in baselines
  - High CM needs
- **Project related information**
  - Information that can be removed after project closure
  - Might be desirable to baseline at some occasions
  - Limited CM needs, some kind of version control is desirable
- **Line/organizational related information**
  - Operations and process information
  - Limited CM needs, some kind of version control is desirable

AUTHENTIC MAIL FROM 2012

Hej Hannes,

Jag behöver få tag i filer som levererades när jag arbetade som komponentägare 2009, var hittar jag sådan dokumentation?

Tack på förhand, det är lite bråttom i detta ärendet sitter i tråkiga diskussioner med leverantör om FDC1.

Hello Hannes,

I need to find files that were delivered when I was working as component owner 2009. Where do I find such documentation?

Thanks in advance, it’s a bit short of time with this issue because of tedious discussions with the supplier about FDC1.

TRACEABILITY
VISIBILITY
CONTROL