# Software Configuration Management

LECTURE 9

## Change is inevitable: But why?

- **New Market Conditions** can cause changes product or business requirements
- **New Customer Needs**
  - ✓ Input data may change
  - ✓ The output data may change
  - ✓ They may require different functionality or services to be offered
- **Reorganization** or business growth or downsizing
  - ✓ This can cause changes to the product
  - ✓ This can cause changes in the team structure and resources available
- **Budgeting and Scheduling Constraints**
- **Newly available supporting technologies**
- **Bugs** are discovered and have to be corrected

## What can change?

- Documents
  - ✓ Requirements Document, Design Documents
- Data
  - Internal Data (eg. interest rate for a banking system)
  - External Data (eg. names of persons with special characters)
- Programs
  - Source Code
  - Executable Code

- *These will make up your Software Configuration Items **(SCI)***

## Why manage changes?

- Missed Deadlines
  - ✓ Changes can lead to significant time slippage due to re-work, new features which require more effort/time
- Budget Overruns
  - ✓ Changes incorporated without new provision of resources will possibly exhaust budget
- Bugs
  - ✓ Errors may be discovered and may need to be addressed to prevent the negative effects
- Poor Software Support
  - ✓ Changes not carefully managed can lead to inconsistent documentation of both code and documents which may be inconsistent when there is need to make adjustments or support users of the system
- Software Project Failure
  - ✓ Ultimately, the above issues can result in failure of a software project altogether.

How do we properly deal with changes to software?

Answer:
Software Configuration Management

## Software Configuration Management (SCM)

- Is the management of an **evolving** Software System
- It ensures that **changes** are incorporated into the system in a controlled manner
- Ensures records of the details of these **changes** are maintained
- Is concerned with the policies, processes and tools to manage the above **changes**

## There are established Standards to SCM

- IEEE 828-1998 - *IEEE* Standard for Software. Configuration Management Plans
- ISO 9000 - Quality management
- Capability Maturity Model (CMM)

## SCM has 4 major processes

- Change Management
  - ✓ is about **assessing change proposals** and **deciding** if or when they should be implemented
- Version Management
  - ✓ is about **keeping track** of different software components as changes are being made to them
- System Building
  - ✓ is the process of **assembling** system components to form an executable program to run on a target computer
- Release Management
  - ✓ involves making decisions about system release dates and **organizing all aspects of the system** including the code, data, configuration, system documentation and marketing documentation relating to that release

# Change Management

## Change Management

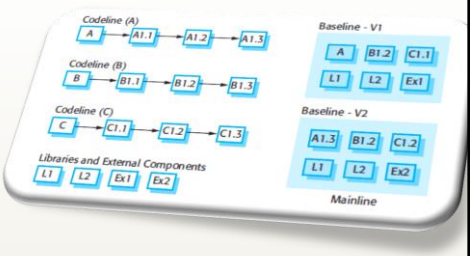
- "checking, costing, and approving" - Sommerville

- This involves:
  - ✓ **keeping track of requests for changes** to the software from customers and developers and **analyzing** them
  - ✓ working out the **costs and impact** of making these changes
  - ✓ **deciding** if and when the changes should be implemented

# Configuration

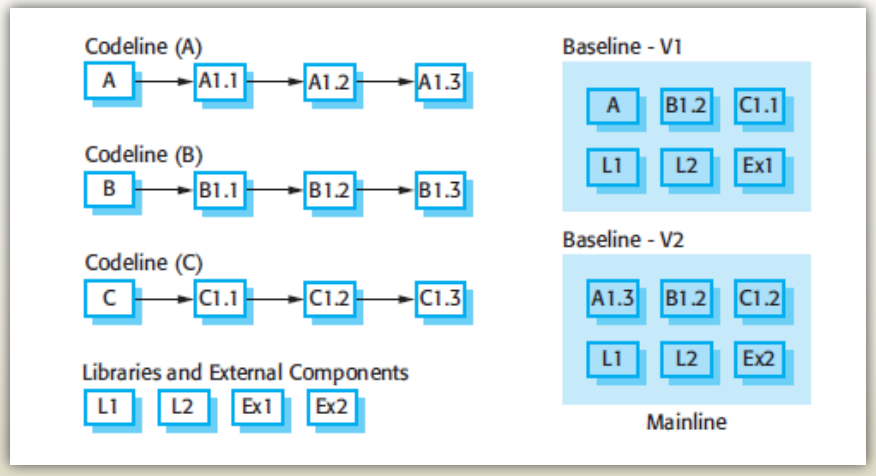
- **Configuration Control**
  - ✓ The process of ensuring that versions of systems and components are recorded and maintained
  - ✓ Ensures that changes are managed and all versions of components are identified and stored for the lifetime of the system
- **Configuration Items** or **Software Configuration Items** (**SCI**s)
  - ✓ Anything associated with a software project (design, code, test data, document, etc.) that has been placed under configuration control.
  - ✓ There are often different versions of a configuration item
  - ✓ Configuration items have a unique name or identifier

# Baseline




- A baseline is a **collection of component versions** that make up a system.
- Baselines are **controlled,** which means that the versions of the components making up the system cannot be changed
- This means that it should always be possible to re-create a baseline from its constituent components
- Baselines include **code** as well as **documentation**

## Baseline - Example




## Change Requests

- Sources
  - Client
  - Users (Indirectly)
  - Members of the Development or Management Team

- The Development Team
  - ✓ Analyzes the request and determines the changes that are necessary
  - ✓ Identifies other components that may be affected
  - ✓ Provides estimate of the time/cost required to implement the change
  - ✓ May approve and execute minor change requests
- The Change Control Board
  - ✓ Group of persons who will asses the change request and determine if it is cost effective and if it is to be implemented

## Change Analysis

- Significant Factors to consider when deciding on a change
  - ✓ **Consequence** of not making the change
  - ✓ **Benefits** of the change
  - ✓ Number of **persons affected** by the change
  - ✓ The **costs** of making the change
  - ✓ The **Product Release Cycle**
    - If a new version was recently released then it may be better to implement the change in the next planned release

## Tracking Implemented Changes

- Implemented changes can be captured on the CRF
- Implemented changes can be captured in source code

```
// SICSA project (XEP 6087)
//
// APP-SYSTEM/AUTH/RBAC/USER_ROLE
//
// Object: currentRole
// Author: R. Looek
// Creation date: 13/11/2009
//
// © St Andrews University 2009
//
// Modification history
// Version  Modifier  Date        Change      Reason
// 1.0      J. Jones  11/11/2009  Add header  Submitted to CM
// 1.1      R. Looek  13/11/2009  New field   Change req. R07/02
```

## Tracking Changes

- Change Request Form (CRF)
  - Essential Elements:
    - details about the issue
    - analysis of the issue
    - decision made regarding the change
  - Complex or critical systems may require more details to be captured than smaller systems

# Change Request - Sample

Example:
Simple
CRF

DJ-Banking System
Change Request Form (v1.0.0)
Change Request ID:

CHANGE DETAILS

**Change Requester (and Role)**

Mr. Eyton Ferguson

**Change Request Date**

03/11/2013

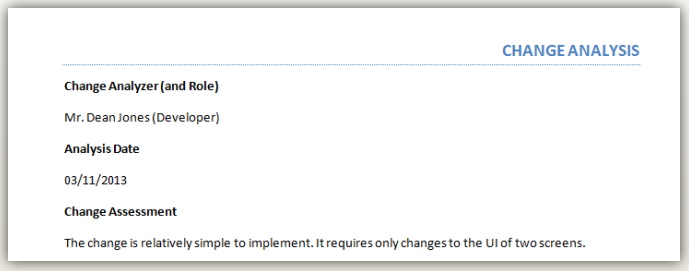
**Details of Change Requested**

The field names are out of place on the Home Screen (URL: http://www.djbanking.com/home)

The application was being accessed from the university campus with a user with the role of "Administrator"

# Assessment of Change Request

Example:
Simple
CRF



CHANGE ANALYSIS

**Change Analyzer (and Role)**

Mr. Dean Jones (Developer)

**Analysis Date**

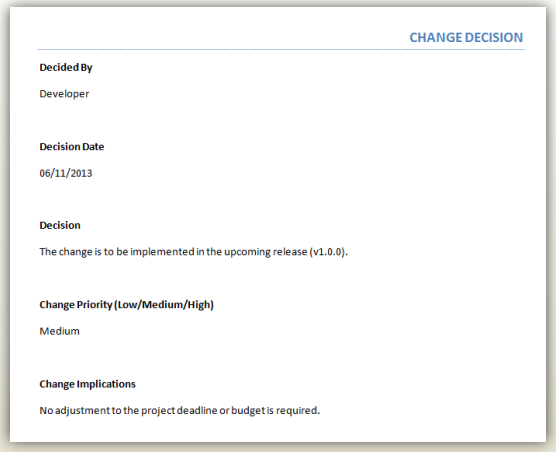
03/11/2013

**Change Assessment**

The change is relatively simple to implement. It requires only changes to the UI of two screens.

## Decision on Change Request

Example:
Simple
CRF



CHANGE DECISION

**Decided By**

Developer

**Decision Date**

06/11/2013

**Decision**

The change is to be implemented in the upcoming release (v1.0.0).

**Change Priority (Low/Medium/High)**

Medium

**Change Implications**

No adjustment to the project deadline or budget is required.

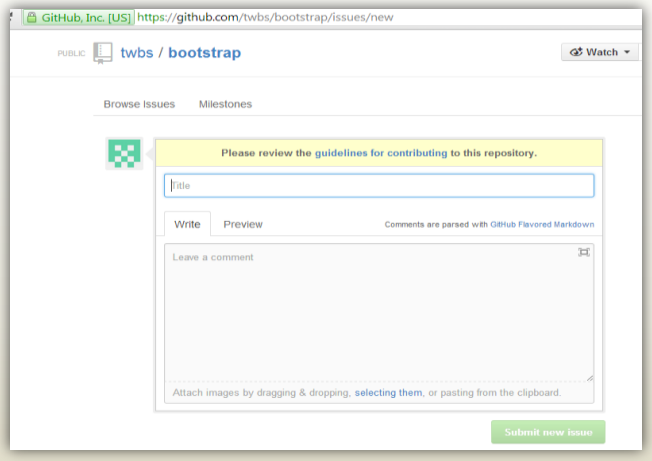
## Change Management Tools

- Tools
  - Issue Tracking Systems
  - Support Ticket System

Examples

- ✓ Trac
- ✓ Redmine
- ✓ Github's Issue Tracking System

Example:
Electronic CRF




As changes are made to the system what happens to the different versions?

We keep track of them using

**Version Management**

# Version Management

## Version Management

- This involves:
  - ✓ keeping track of the multiple **versions of system components** and
  - ✓ ensuring that **changes made to components by different developers do not interfere** with each other
- Version Management is the process of managing **Codelines** and **Baselines**

## Version Management

- Version
  - ✓ An instance of a configuration item that differs, in some way, from other instances of that item
  - ✓ Versions always have a unique identifier, which is often composed of the configuration item name plus a version number

- Example of a three-sequence version number:

**"ABC Banking System SRS - v1.2.8"**

## Version Management

- Example of a three-sequence version number:

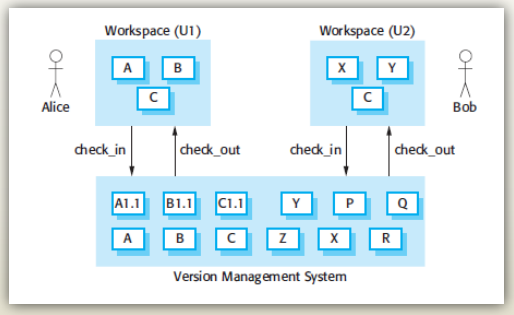
**"ABC Banking System SRS - v1.2.8"**

| Major Version | Minor Version | Change/Revision |
|---|---|---|
| 1 | 2 | 8 |

# Workspace

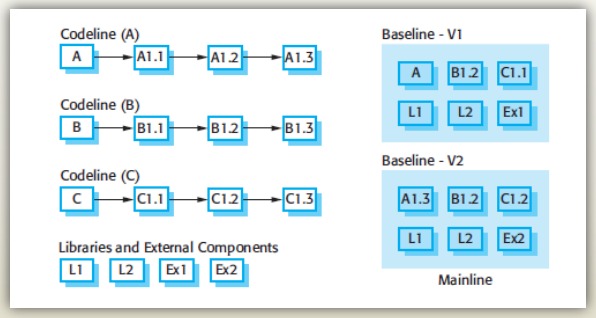
- ✓ A private work area where software can be modified without affecting other developers who may be using or modifying that software.

Example :
C:\Users\Alice\Project1
C:\Users\Bob\Project1




# Codeline

- ❖ A set of versions of a software component and other Configuration Items on which that component depends



## Version Management Systems

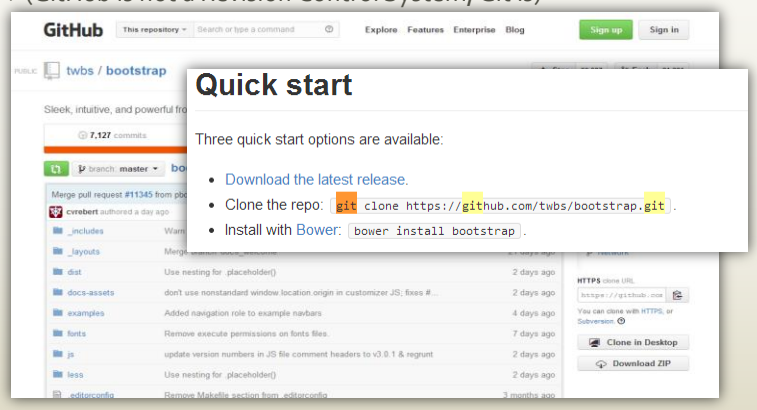
- Features of Version Management Systems/Tools
  - Version and release identification
  - Storage management
  - Change history recording
  - Independent development
  - Project support

❖ Video:
- https://www.youtube.com/watch?v=zbKdDsNNOhg

## Version Management Systems

❖ GitHub.com is a service that is based on Git Revision Control System.

❖ (GitHub is not a Revision Control System, Git is)

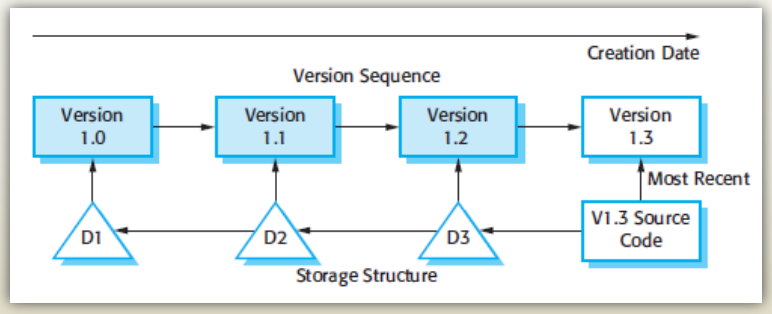


## Version Management systems

- Version Management Systems/Tools
  - ✓ Also known as Version Control Systems
  - ✓ Include
    - Git
    - CVS
    - Subversion

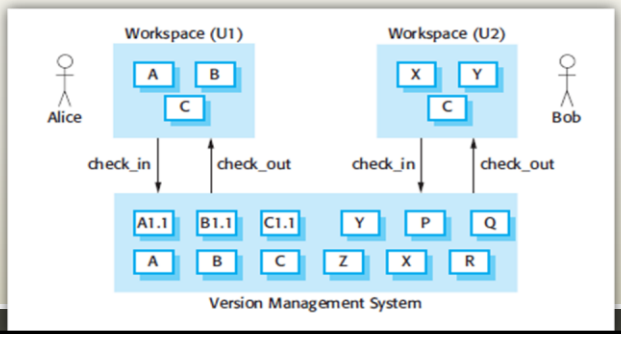
## Version Management

- Features of Version Management Systems/Tools
  - ✓ Storage Management
    - Typically, the most recent version of a component is stored
    - *Deltas* are created that specify what changes to make to the most recent file in order to get back to a specific version



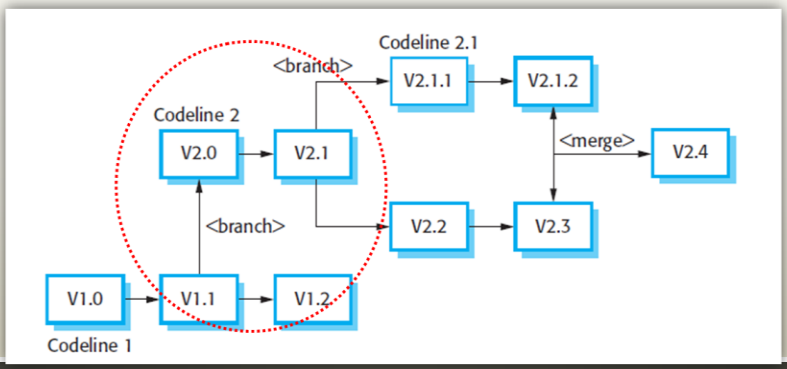
# Features of Version Management Systems/Tools

- Independent Development
  - ✓ Independent Development is facilitated by the concept of **public repository** and a **private workspace**
  - ✓ The workspaces of Alice and Bob (in the following slide) are Private
  - ✓ The Version Management System depicted is the public repository



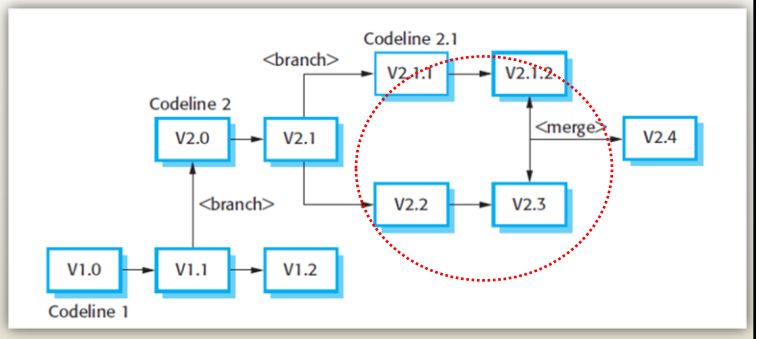

# Branching

- The creation of a new codeline from a version in an existing codeline.
- The new codeline and the existing codeline may then develop independently



# Merging

- The creation of a new version of a software component by merging separate versions in different codelines
- These codelines may have been created by a previous branch of one of the codelines involved




VIDEO: **https://www.youtube.com/watch?v=QQhD5VveZhw**

# System Building

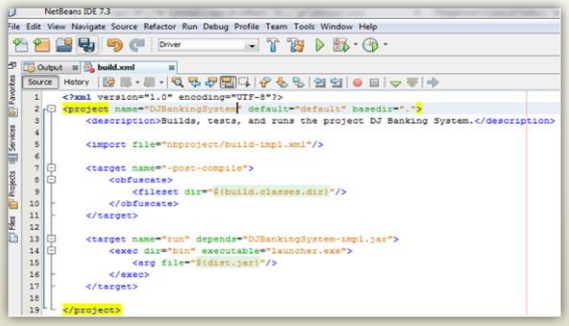
As new versions of the code are being created they are usually continuously compiled and tested in some way. The next section describes this build process – **System Building**

## System Building

- System Building is the process of:
  - **assembling** program components, data, and libraries
  - compiling and linking these to create a **complete**, executable system
- Release
  - ✓ A version of a system that has been built and released to customers (or other users in an organization) for use.

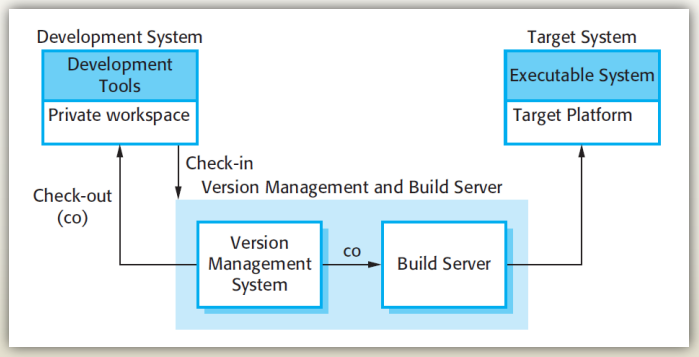
## System Building

- Build Script
  - ✓ Also called the build configuration file
  - ✓ Is the definition of the system to be built
  - ✓ Contains information about components and their dependencies, and the versions of tools used to compile and link the system



# Build Tool and Version Control

- The Build Tool must communicate with the Version Control System in order to retrieve specific versions of components




- The Build Tool must communicate with the Version Control System in order to retrieve specific versions of components
- The Versions of components selected for a build are noted in the build configuration file (or build script). This includes:
  - ✓ Specific Versions of components that are included
  - ✓ Dependent Components
  - ✓ Details of External Libraries that are used
    - Inclusion of configuration files that define the target installation
- When the system needs to be rebuilt the build configuration file is reused

## Build Tools Features

- Build script generation
- Minimal recompilation
  - ✓ Based on signatures of already compiled components. Checksums can be based on:
    - Modification Timestamps
    - Source Code Checksums
- Executable system creation
- Test automation
- Reporting (about success or failure of tests or other build activities)
- Documentation generation (eg. release notes)

## Build Tools

- ✓Apache Ant
  - (comes with Netbeans/Eclipse IDEs)
- ✓Make

## Release Management

There will be many builds of a system. A few of which will be stable. Even fewer of these will be organized and packaged for the end user – these will make it into Releases.

## Release Management

- This involves:
  - ✓ preparing software for external release and
  - ✓ keeping track of the system versions that have been released for customer use

- Major Releases
  - ✓ Significant new functionality
  - ✓ Are usually sold separately
- Minor Releases
  - ✓ Bug fixes
  - ✓ Other minor changes
  - ✓ Usually distributed free of charge
  - ✓ Automatic upgrade is favourable

## Release Management

### Releases in everyday life

- Microsoft Office

# Release Management

## Releases in everyday life

- Microsoft Exchange Server Releases

| Version | Build number | Release date |
|---|---|---|
| Microsoft Exchange Server 4.0 | 4.0.837 | April 1996 |
| Microsoft Exchange Server 4.0 (a) | 4.0.993 | August 1996 |
| Microsoft Exchange Server 4.0 SP1 | 4.0.838 | May 1996 |
| Microsoft Exchange Server 4.0 SP2 | 4.0.993 | August 1996 |
| Microsoft Exchange Server 4.0 SP3 | 4.0.994 | November 1996 |
| Microsoft Exchange Server 4.0 SP4 | 4.0.995 | April 1997 |
| Microsoft Exchange Server 4.0 SP5 | 4.0.996 | May 1998 |

# Release Package Includes

- Components' source code for specific versions
- Corresponding executables
- Data and Databases required by the system to function
- Configuration files that define target environments
- Build scripts
- Details of the build system including the versions of:
  - ✓ Operating System, Libraries, Compilers, Other tools
- Electronic and printed documentation
- Packaging and associated marketing material that have been designed for the release

## Issues in Release Management

- Managing releases is a complex and expensive process
- Special releases may have to be produced for different customers or groups of customers
  - ✓ Eg.:
    - separate releases for iPhone and Android users
    - separate releases for iPhone 3G (with no front camera) and iPhone 6 users
- Different users may be running different releases at the same time
- Previous releases may need to be rebuilt exactly as they were in order to facilitate maintenance (including security patches)

## The Software Configuration Management Plan

## The Software Configuration Management Plan

- The IEEE 828-1998 Standard recommends inclusion of the following sections within the SCM Plan
  - ✓ Introduction
  - ✓ SCM Management
  - ✓ SCM Activities
  - ✓ SCM Schedules
  - ✓ SCM Resources
  - ✓ SCM Plan Maintenance

## The Software Configuration Management Plan

- Essentially the SCM Plan documents
  - **what SCM activities** are to be done
  - **how** they are to be done
  - **who** is responsible for doing specific activities,
  - **when** they are to happen
  - and **what resources** are required
- It can address SCM activities over any portion of software product's life cycle

*For more details see the complete standard*

## The Software Configuration Management Plan

- Good SCM Planning includes
  - ✓ Specifying what elements within the system are to be controlled (i.e. Defining SCIs)
    - Eg. :
      - SRS
      - SDD
      - RFP
      - Test Plans
      - Source Code
  - ✓ Defining the rules of access Project Database which should keep the controlled documents
  - ✓ Defining the procedures for handling changes in the system

## Activities

A brief introduction to Git

- URL: http://youtu.be/mYjZtU1-u9Y

A brief introduction to Redmine (for Project Management)

- URL: http://youtu.be/sAobJ1kmyqA

- Answer the *simple* questions at the end of Sommerville's chapter on SCM

# References

This lesson was adapted primarily from:

- Sommerville, Ian. *Software engineering*. 9th ed. Boston: Pearson, 2011. Print.

Other sources:

- IEEE 828-1998 - IEEE Standard for Software Configuration Management Plans

Suresh, P. V. (Director) (2008, October 15). Software Configuration Management. Lecture conducted from School of Computer and Information Science, New Delhi.

How to : Determine the version number, build number & the service pack of Exchange Server?. (n.d.). *MSDN Blogs*. Retrieved November 04, 2013, from http://blogs.msdn.com/b/deva/archive/2009/07/29/how-to-determine-the-version-number-build-number-the-service-pack-of-exchange-server.aspx

Originally Compiled by Dean J. Jones, Updated by R.A. Anderson