Faculty of ECMS / School of Computer Science

Software Engineering & Project Configuration Management


· Aug 14
my sister, who doesn't work in tech, thought it was pretty cool that our team sets aside time to tell each other jokes and thats when i realized she had no idea what i was saying when i told her i had "standup" everyday
102
1.7K
13.4K

Outline

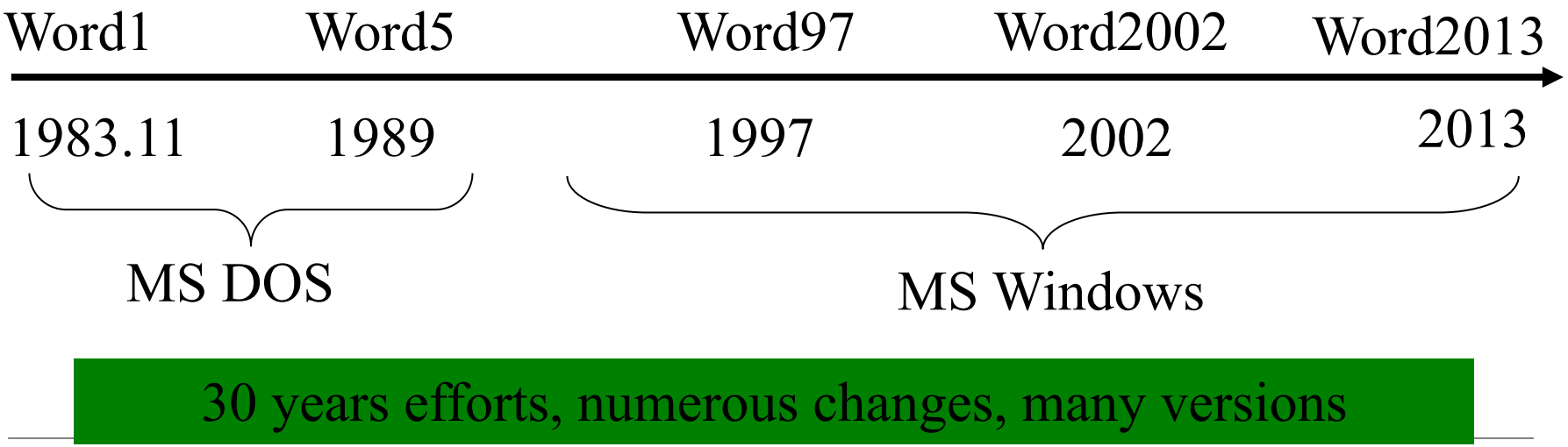
- Configuration Management: What and Why

- Fundamental CM Activities
  - Change management
  - Version and release management
  - System building
  - Configuration management planning

Configuration Management: What and Why

- **Tracking** and **controlling changes** in software systems
  - Releases
  - Security patches
  - Bug fixes
  - etc.

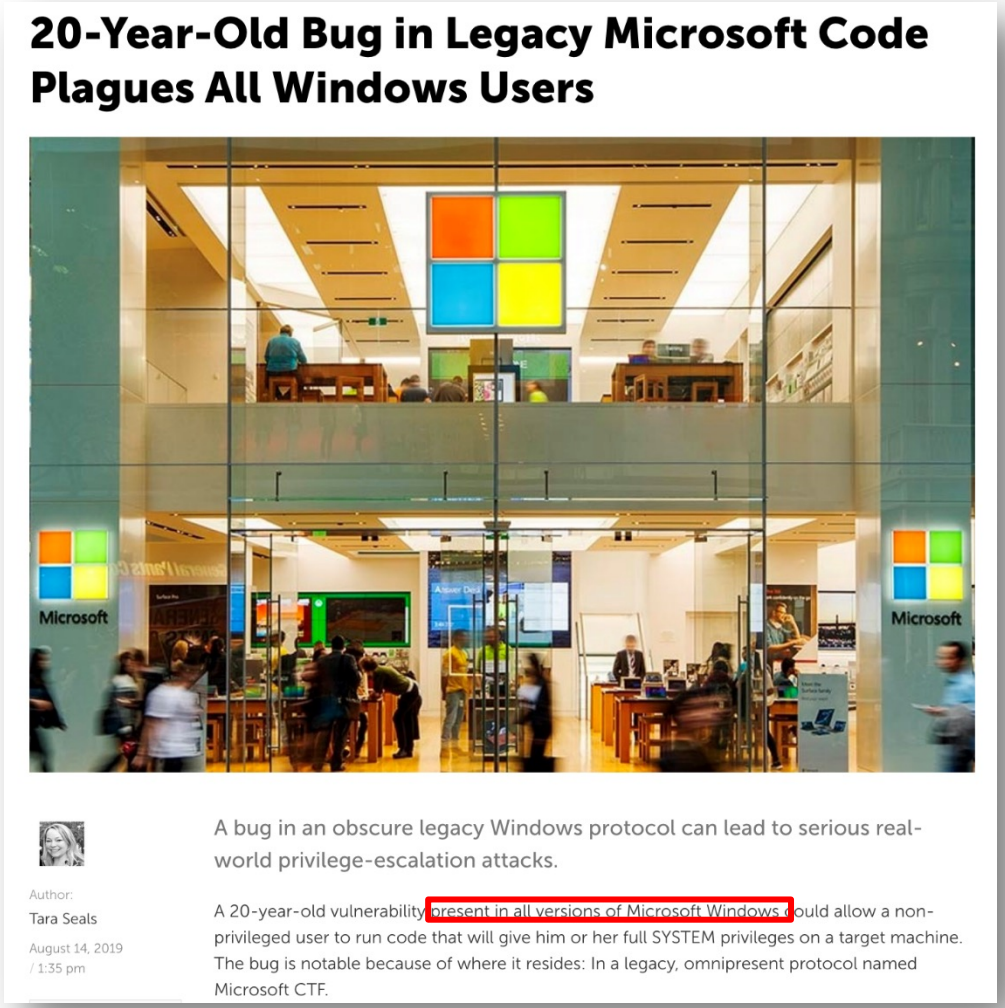
History of MS Word



30 years efforts, numerous changes, many versions

Well, but is the old code really relevant?

Yes, it is!



20-Year-Old Bug in Legacy Microsoft Code Plagues All Windows Users

Author:
Tara Seals

August 14, 2019
/ 1:35 pm

A bug in an obscure legacy Windows protocol can lead to serious real-world privilege-escalation attacks.

A 20-year-old vulnerability present in all versions of Microsoft Windows could allow a non-privileged user to run code that will give him or her full SYSTEM privileges on a target machine. The bug is notable because of where it resides: In a legacy, omnipresent protocol named Microsoft CTF.

https://threatpost.com/20-year-old-bug-legacy-microsoft-windows-users/147336/

Configuration Management: What and Why

- **New versions** of software systems are **created over time**:
  - For different machines/OS/platforms
  - Offering different functionality
  - Tailored for particular user requirements

Configuration Management: What and Why

- Configuration management is concerned with **managing evolving software** systems:
  - Managing change is a **team activity**
  - CM aims to **control the costs and effort** involved in making changes to a system

Configuration Management: What and Why

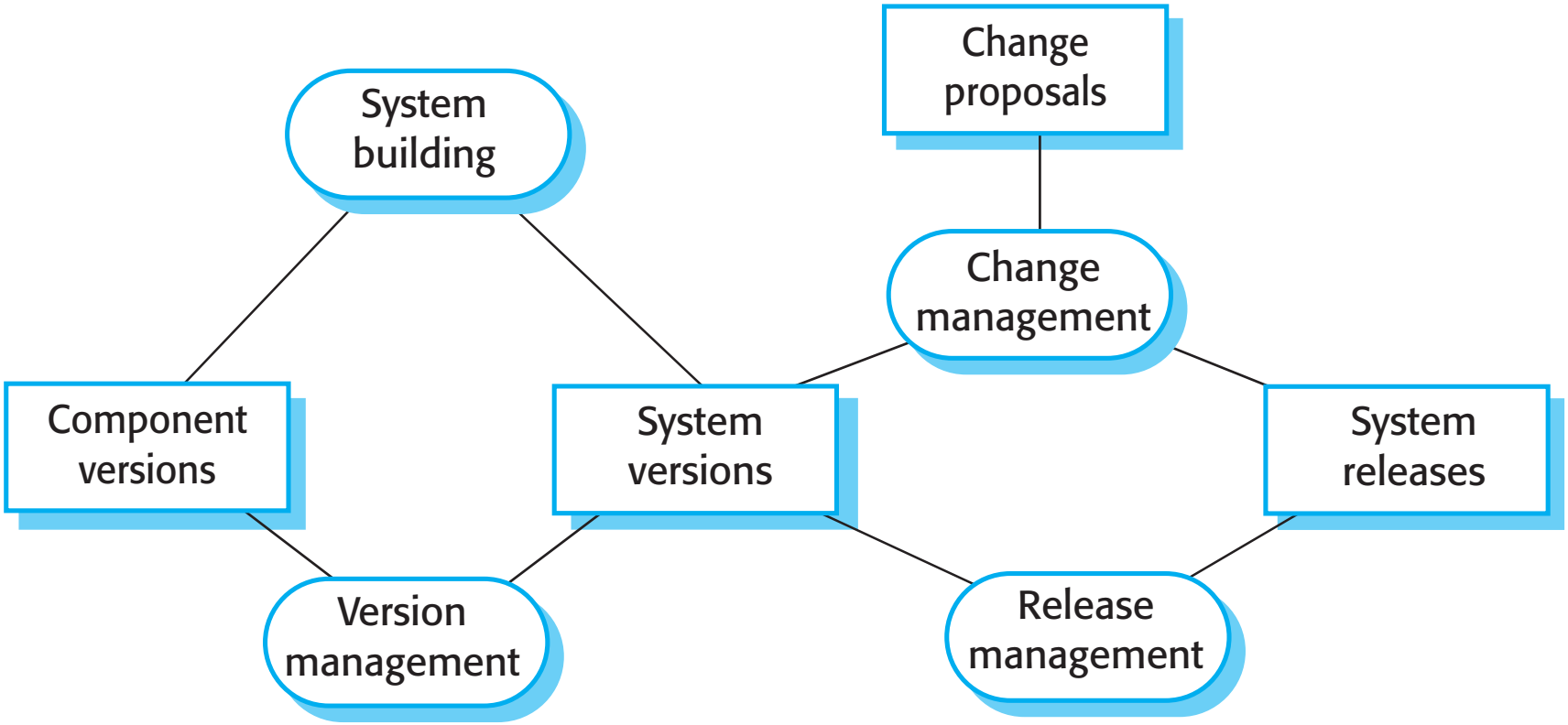
- Involves the development and application of **policies**, **procedures**, standards, and **tools** to manage an evolving software product
- May be seen as part of a more general **quality management** process (see previous lecture)
- If no there is no configuration management
  - May **waste time** modifying the wrong version
  - May **lost track** of where the source code is stored
  - May deliver the **wrong version** to customers
  - May be fixing a bug that is **irrelevant** now
  - And so on

Configuration Management Planning

- **All products of the software process** must be **managed:**
  - Requirement specifications
  - Design documents
  - Source Code
  - Executable programs
  - Test data
  - User manuals

- Hundreds of separate documents are generated for a large software system
- The **CM plan** describes the **standards and procedures** that should be used for configuration management

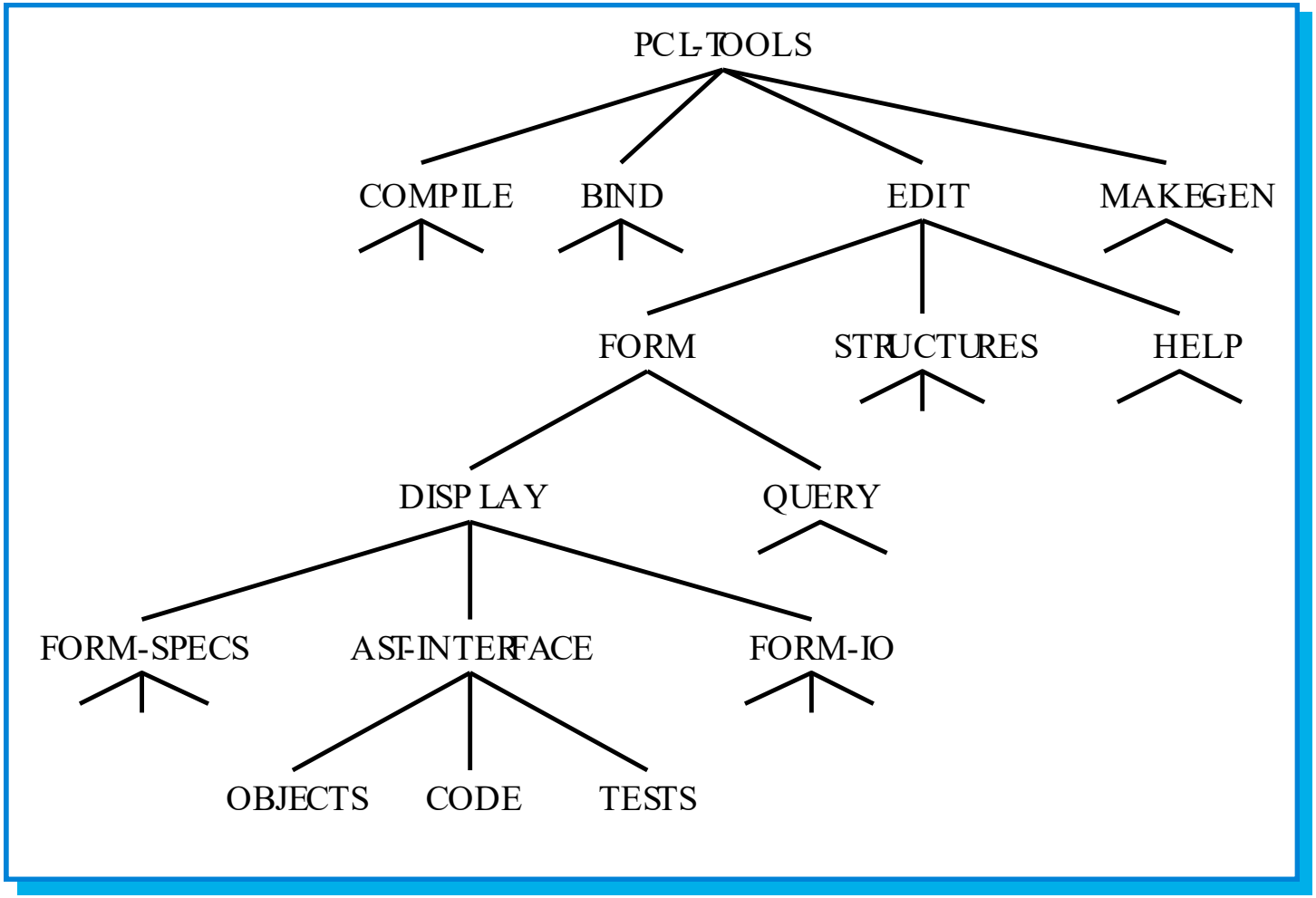
Configuration Management Activities



Configuration Item Identification

- Large projects typically produce hundreds of **documents** which must be **uniquely identified**
- Some of these documents must be maintained for the **lifetime** of the software
- A **document naming scheme** should be defined so that related documents have related names
- Example: **Hierarchical scheme** with multi-level names

Configuration Item Identification Example



Change Management

- Software systems are subject to **continual change requests**
  - From users
  - From developers
  - From market forces

- Change management is concerned with the **managing of these changes** and ensuring that they are implemented in the most (cost-)effective way

Handling change request in Waterfall

The Change Management Procedure

Request change by completing a *change request form*
Analyze change request
if change is valid **then**
    Assess how change might be implemented
    Assess change cost
    Submit request to *change control board*
    if change is accepted **then**
        repeat
            Make changes to the software
            Submit software changes for quality approval
        until software quality is adequate
        Create new system version
    else
        Reject change request
else
    Reject change request

Change Request Form

- The **definition of a change request form** is part of the CM planning process

- This form records the **change proposed**, **requestor** of change, the **reason** why change was suggested and the **urgency** of change (from requestor of the change)

- It also records **change evaluation**, **impact analysis**, change **cost** and **recommendations**

Change Request Form Example

Change Request Form

Project: HEMP **Number:** 007/13
Change requester: Jarad Anderson **Date:** 20/05/13
Requested change: When a new computer is selected from the catalog, display the name of the school who owns the device.

Change analyser: Simon/Anthony **Analysis date:** 22/05/13
Components affected: Display-Icon.Select, Display-Icon.Display

Associated components: FileT able

Change assessment: Relatively simple to implement as a file name table is available. Requires the design and implementation of a display field. No changes to associated components are required.

Change priority: Low

Change implementation:
Estimated effort: 0.5 days

Date to CCB: 22/05/13 **CCB decision date:** 23/05/13
CCB decision: Accept change. Change to be implemented in Release 1.1.
Change implementor: Scott/James/Luke **Date of change:** 28/05/13
Date submitted to QA: **QA decision:**
Date submitted to CM:
Comments

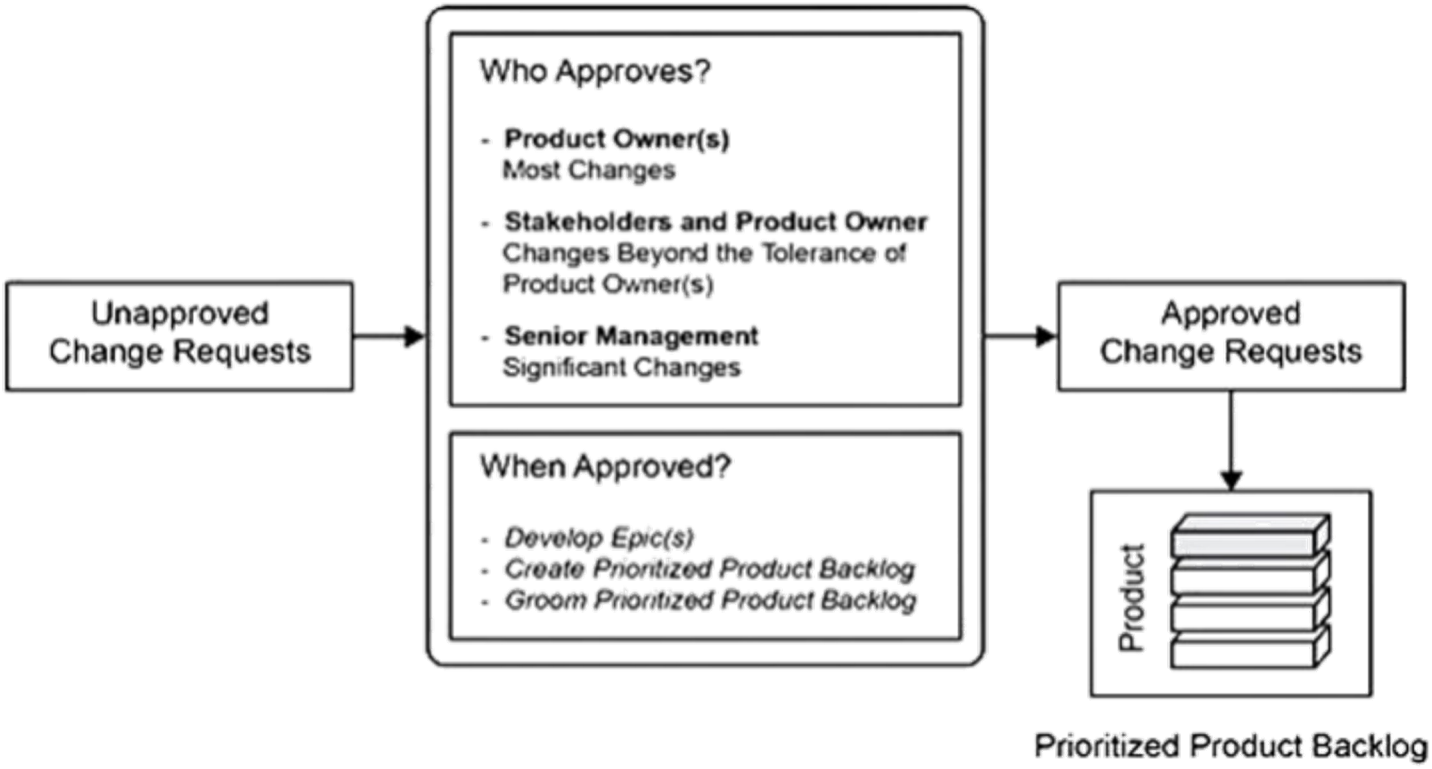
Change Control Board (CCB)

- Changes should be **reviewed by an external group** who decide whether or not they are cost-effective from **a strategic and organizational viewpoint** rather than a technical viewpoint.
  - The consequences of not making the change
  - The benefits of the change
  - The number of users affected by the change
  - The costs of making the change
  - The product release cycle

Change Control Board (CCB)

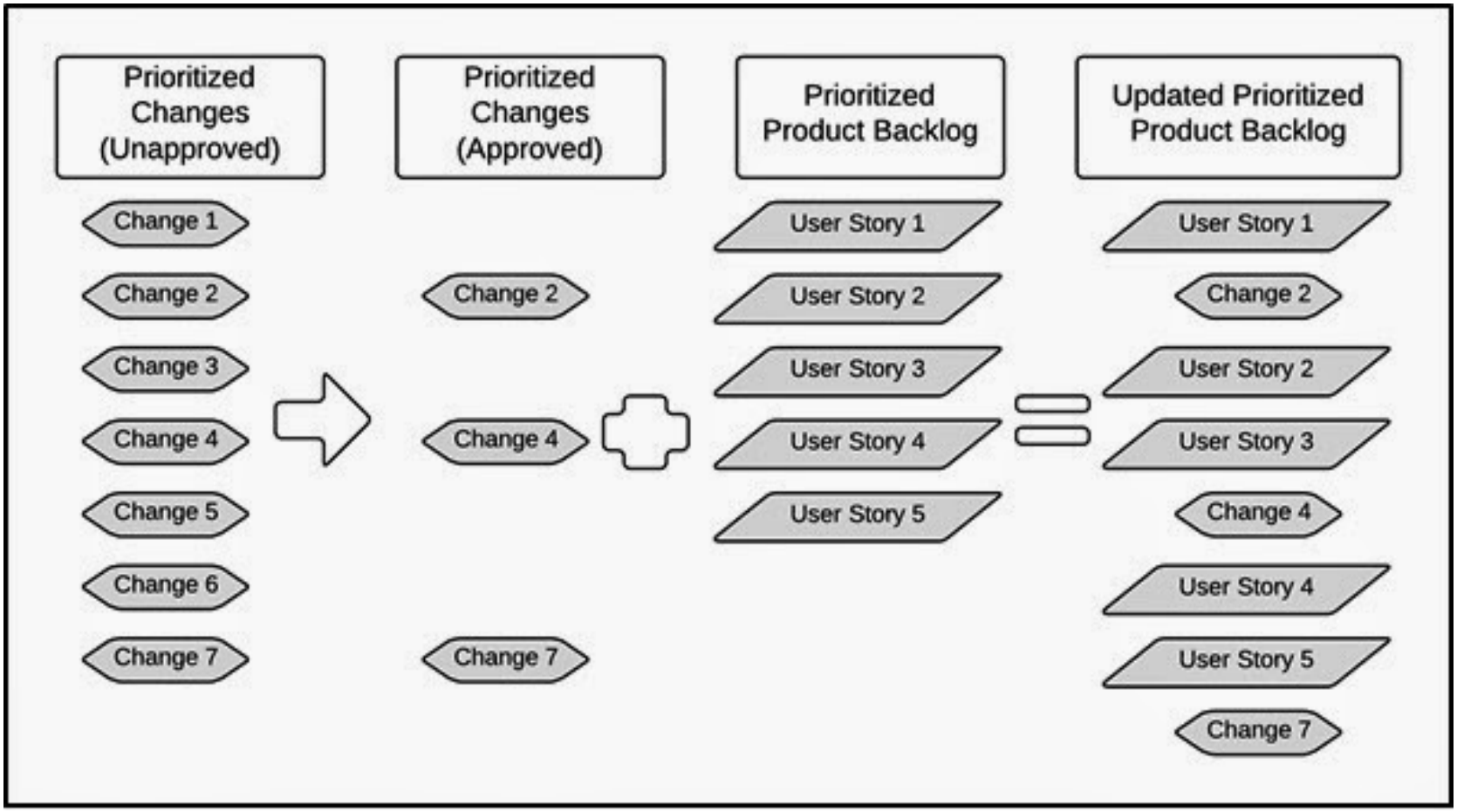
- Should be **independent** of project responsible for system. The group is sometimes called a *change control board* (CCB).
- CCB may include **representatives from client** and contractor staff (usually a formal requirement for military projects)

Handling change request in SCRUM



http://blog.scrumstudy.com/how-to-handle-request-for-change-or-change-requests-in-a-scrum-project-2/

Handling change request in SCRUM



http://blog.scrumstudy.com/how-to-handle-request-for-change-or-change-requests-in-a-scrum-project-2/

But…

"Change control is a traditional project management process for managing change. In a traditional project change control typically consists of filling out a detailed change request form which includes attributes like detail of change, impact to the project, risks, mitigations etc. It also needs approval of several people. **Traditional change control is at odds with Agile because it conflicts with the principle of "Responding to change over following a plan"**. Responding to change becomes difficult when there are huge forms to fill and list of approvals required. In an interesting discussion on the Lean Agile Scrum group, members of the group discuss the need to track changes in Scrum and the possible way in which changes could be tracked."

https://www.infoq.com/news/2008/12/change-requests-in-scrum/

Recommendations

- Log change to the **backlog** or **change tracker**
- **Eliminate** as many **approvals** as possible
- Have a **light change control form**, if necessary
- Keep the stakeholders and operations **involved**

https://www.infoq.com/news/2008/12/change-requests-in-scrum/

Version and Release Management

- Decide on **identification scheme** for system versions
- Plan when new system version is to be **produced**
- Ensure that **version management** procedures and tools are properly applied
- Plan and **distribute** new system **releases**

Some Terminology

- **Version:** An instance of a system which is **functionally distinct** in some way from other system instances
  - Example: MS Word97, Word2007

- **Variant:** An instance of a system which is **functionally identical but non-functionally distinct** from other instances of a system
  - MS Word for Windows or for Mac OS

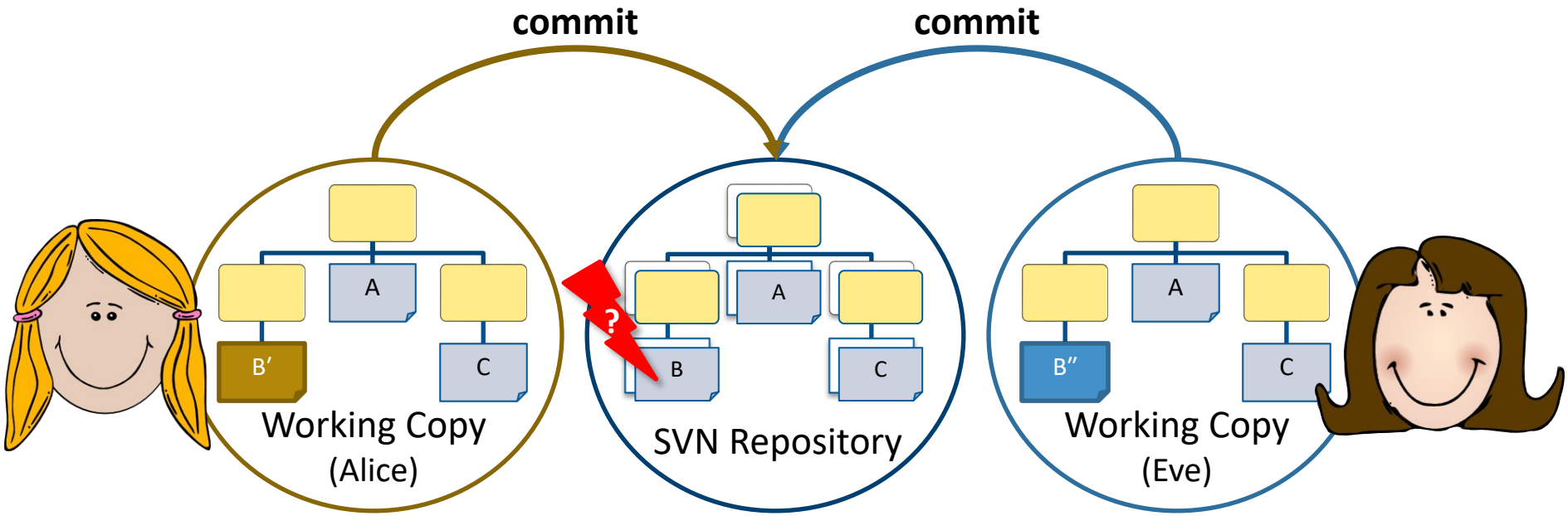
- **Release:** An instance of a system which is **distributed to users outside of the development team**
  - Word 2007 for Windows Vista, released 30/01/2007

Version Management

- Version management (VM) is the process of **keeping track of different versions** of software components or configuration items and the systems in which these components are used

- It also involves ensuring that **changes** made by different developers to these versions do **not interfere with each other**

Example: SVN Conflicts



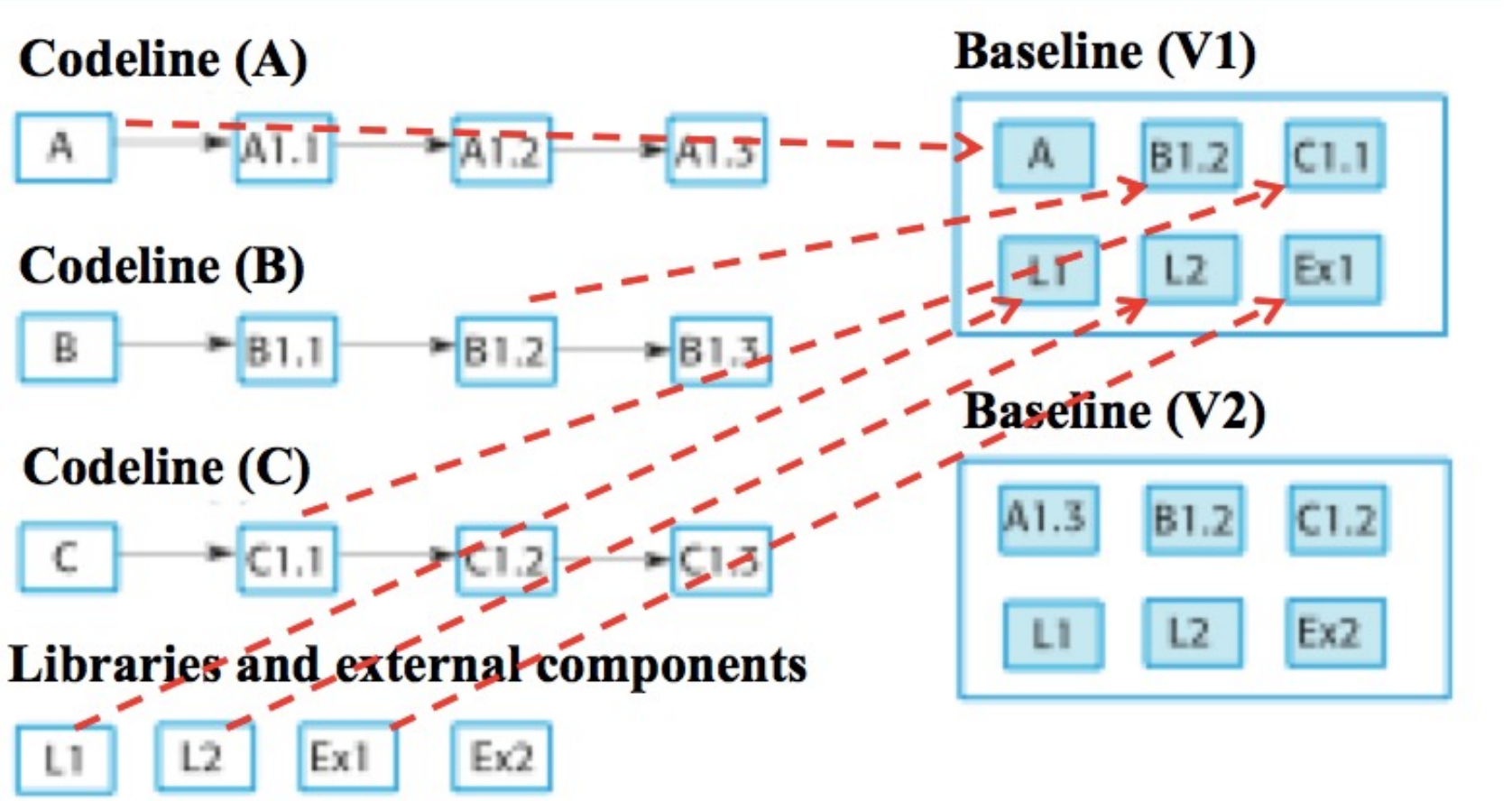
Problem: Multiple developers work on the same file during the same time (actually, they work on copies in their working directories)
Whose changes should be applied to the code in the repo?

Some Terminologies: Codelines and baselines

- A codeline is a sequence of versions of source code with later versions in the sequence derived from earlier versions.
  - Codelines normally apply to components of systems so that there are different versions of each component.

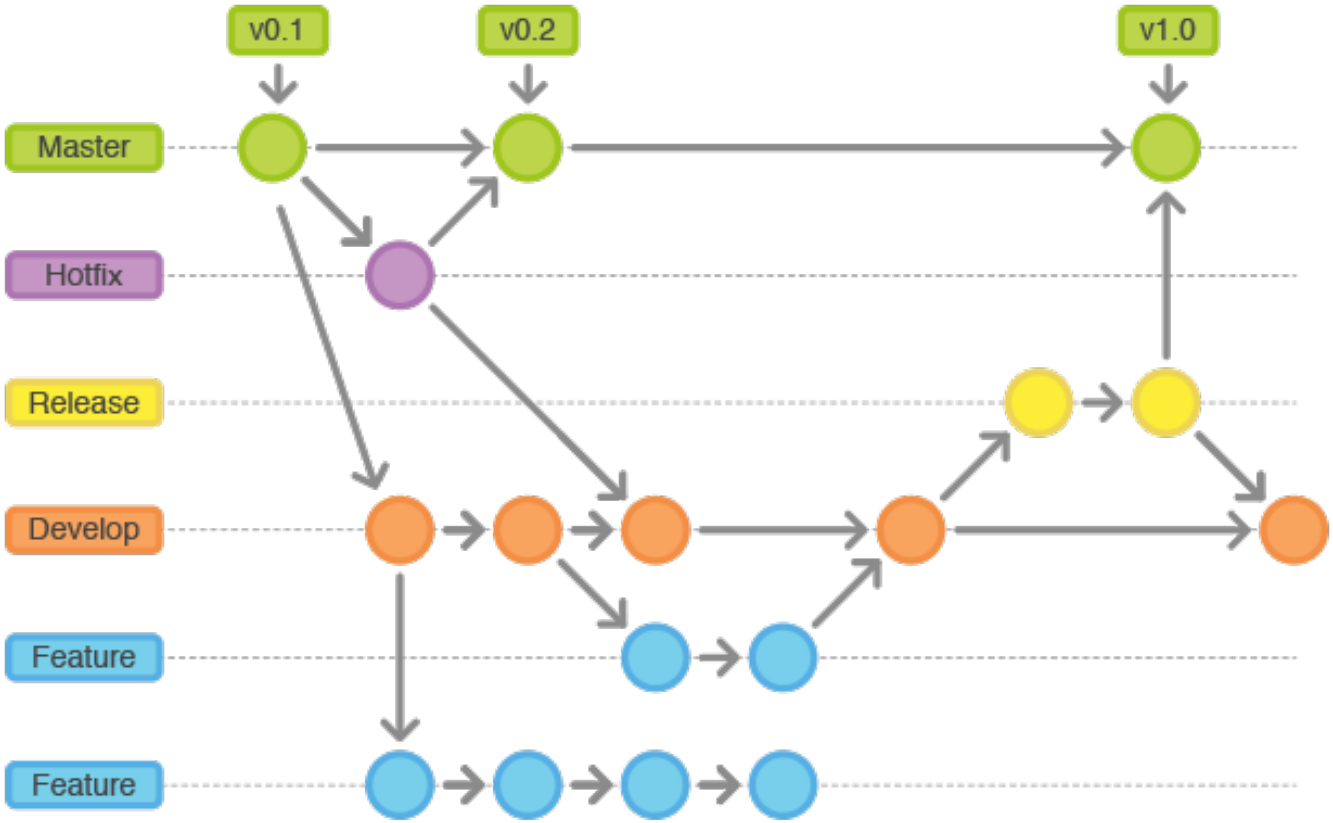
- A baseline is a definition of a specific system.
  - The baseline therefore specifies the component versions that are included in the system plus a specification of the libraries used, configuration files, etc.

Codelines and Baselines



Version Management Example in Git

- Branching practices



https://nvie.com/posts/a-successful-git-branching-model/

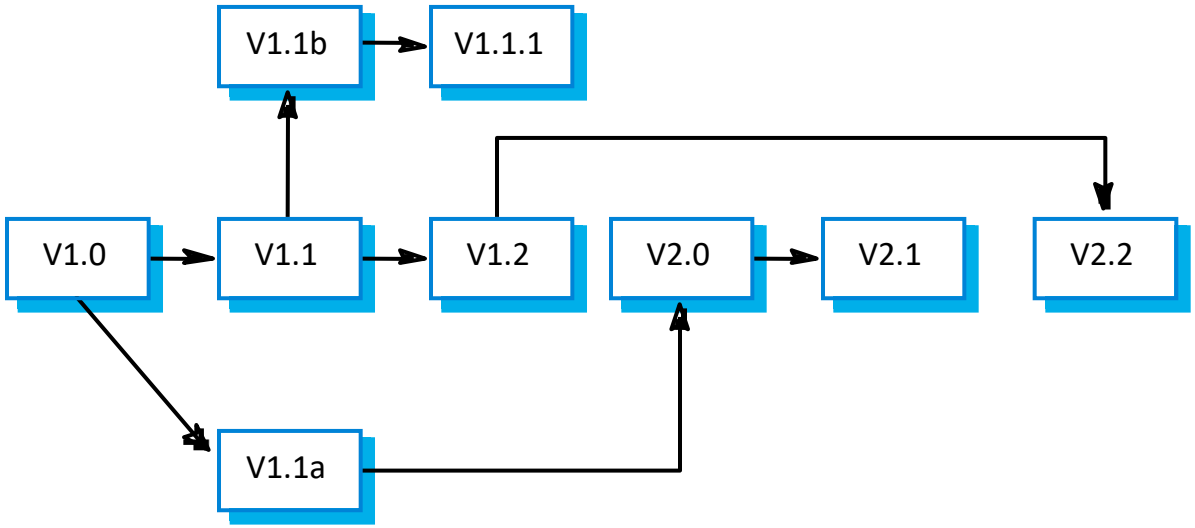
Version Identification

- Procedures for **version identification** should define an unambiguous way of identifying component versions

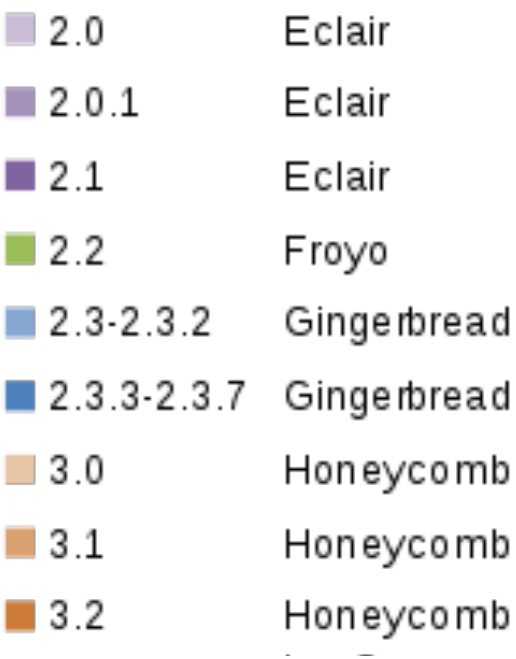
- Basic technique for component identification: Version **numbering**

Version Numbering

- Simple naming scheme uses a **linear derivation**
  - V1, V1.1, V1.2, V2.1, V2.2 etc.
- The actual derivation structure is a **tree** or a network rather than a sequence
- Names should be **meaningful**



Version Numbering



- A **consistent scheme** of version identification should be established
  - When should increase the first digit?
  - When should increase the second/third digit?
- Example:
  - Android Éclair 2.0 vs 2.0.1:
    - Minor API changes, bugfixes and framework behavioral changes
- Each version should have **change log**

https://en.wikipedia.org/wiki/Android_version_history

Semantic Versioning

- Given a version number **MAJOR.MINOR.PATCH** increment the:
  - **MAJOR** version when you make incompatible API changes,
  - **MINOR** version when you add functionality in a backwards compatible manner, and
  - **PATCH** version when you make backwards compatible bug fixes.
- Additional labels for pre-release and build metadata are available as extensions to the default format, e.g. 1.0.0-alpha

https://semver.org/

Changelogs

- What is a changelog?
  - A changelog is a **file which contains a curated, chronologically ordered list of notable changes** for each version of a project.

- Why keep a changelog?
  - To make **it easier for users and contributors** to see precisely what **notable changes** have been made between each release (or version) of the project.

- Who needs a changelog?
  - People do. Whether consumers or developers, the end users of software are **human beings who care about what's in the software**. When the software changes, people want to know why and how.

https://keepachangelog.com/en/1.0.0/

Release Management

- A system **release** is a version of the system that is **distributed to customers**

- Release management includes:
  - deciding **when** to issue a system version as a release
  - managing the **process** of creating the release
  - **documenting** the release so that re-creation is possible

System Releases

- Not just a set of executable programs
- May also include
  - **Configuration files** defining how the release is configured for a particular installation
  - **Data files** needed for system operation
  - An **installation program** or shell script to install the system on target hardware
  - Electronic and paper **documentation**
  - **Packaging** and associated publicity
  - **Changelog** (description of changes/known issues)
- Systems are now normally **released online**, either as downloadable installation files or in app stores

Release Problems

- Customer **may not want** a new release of the system
  - They may be happy with their current system as the new version may provide unwanted functionality

- Release management **should not assume** that all previous releases have been accepted
  - **All files required** for a release should be re-created when a new release is installed
  - **Tradeoff** between efficient release deployment and less complicated release migration process

Release Problems: Example

NEWS

Microsoft plans to sell post-2020 support for Windows 7

The developer agreed to offer 'Windows 7 Extended Security Updates' for three years beyond the operating system's January 2020 retirement date.

By Gregg Keizer

Senior Reporter, Computerworld | 09 SEPTEMBER 2018 11:47 PT

https://www.computerworld.com/article/3304309/microsoft-plans-to-sell-post-2020-support-for-windows-7.html

Release Decision Making

- Preparing and distributing a system release is an **expensive** process

- **Factors** influencing the decision when to issue a new system release:
  - technical quality of the system
  - competition
  - marketing requirements
  - customer change requests

- Too frequent or too infrequent releases are not good

System Building

- The process of **compiling and linking** software components into an executable system
- Different systems are built from **different combinations of components**
- This process is typically supported by **automated tools** that are driven by 'build scripts' (e.g., make, Ant, Maven, Gradle)

Key Points Revisit

- Configuration management is the **management of change** in software products
- A formal **document naming scheme** should be established and documents should be managed in a central repository
- The configuration repository should **record** information about **changes** and **change requests**

Key Points Revisit

- A consistent scheme of **version identification** should be established (e.g. semantic versioning)
- **System releases** include executable code, data, configuration files and documentation
- **System building** involves assembling components into a system
- Software **tools** are available to support all CM activities