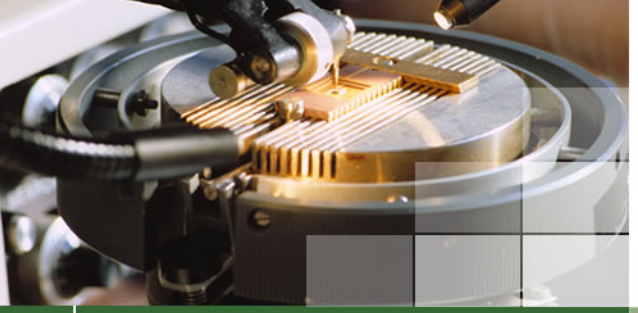


POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Planning and Managing Software Projects 2013-14

Session 13

Change Management

Emanuele Della Valle

http://emanueledellavalle.org

Credits

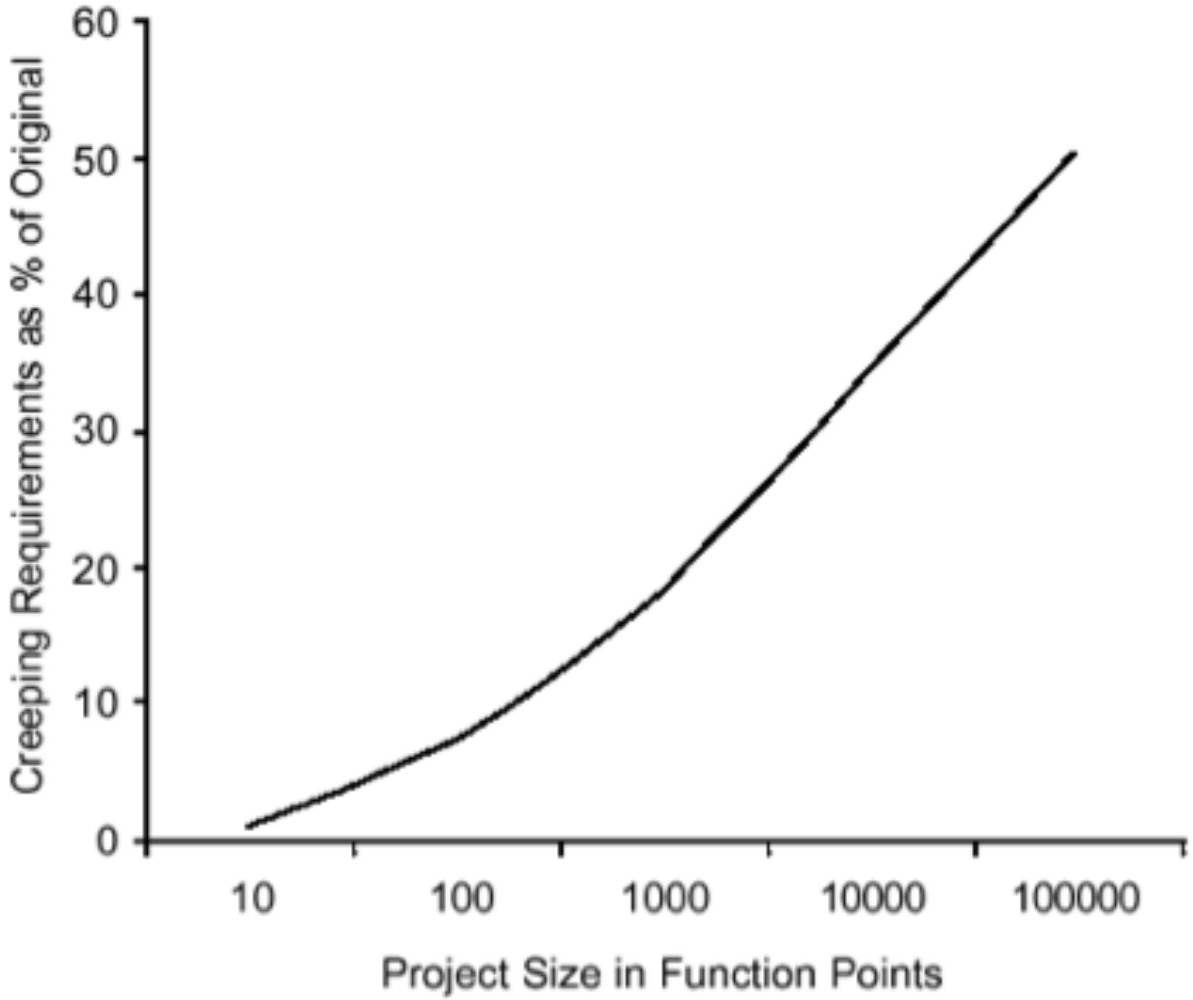
- These slides reuse concepts presented in Prof. John Musser class notes on “Principles of Software Project Management”
  - Original slides are available at http://www.projectreference.com/
- Reuse and republish permission was granted

Today

- The Feature-Creep Phenomenon
- Feature Set Control
- Project Recovery

The Feature-Creep Phenomenon 1/3

- Avg. project has 25% change in requirements during development



The Feature-Creep Phenomenon 2/3

- Sources of change
  - Marketing: want to meet customer's check-list
  - Developers: want to perfect r1 deficiencies
  - Users: want more functionality or now 'know' what they want
- They will all try to 'insert' these during dev.

The Feature-Creep Phenomenon 3/3

- The devil is in the details
- McConnell's example: "trivial" feature can have +/- weeks of impact
- Developers can insert things when you're not looking
- No specification can cover all details. You must specify!
- Programmer ideal: flip switch
- Up to 10-1 differences in program size with the same specifications

Feature-Set Control

- It is a classic mistake avoidance technique
- Early Stages
  1. Minimal Specification
  2. Requirements Scrubbing
  3. Versioned Development
- Mid-Project
  - Effective change control
- Late-Project
  - Feature cuts (and if needed project recovery)

Feature-Set Control

- It is a classic mistake avoidance technique
- **Early Stages**
  1. **Minimal Specification**
  2. **Requirements Scrubbing**
  3. **Versioned Development**
- Mid-Project
  - Effective change control
- Late-Project
  - Feature cuts (and if needed project recovery)

Traditional Specs

- Drive for “traditional” specs
  - Necessity
  - Downstream cost avoidance
  - Full control over all aspects
- As McConnell notes:
  - “But the goal is not to build exactly what you said you would at the beginning. It is to build the best possible software within the available time.”
  - Idealistic but worth remembering

Minimal Specification 1/2

- Tradition spec. issues
  - Wasted effort
    - Too much detail
  - Obsolescence
  - Lack of efficacy -- details do not guarantee success
  - Overly constrained design
  - User assumption that costs are equal (UI ex.)

Minimal Specification 2/2

- Benefits
  - Improved morale and motivation
  - Opportunistic efficiency
  - Shorter requirements phase
- Costs and Risks
  - Omission of key requirements
  - Unclear or impossible goals
  - Gold plating
  - Used for the wrong reasons
    - Lazy substitute for doing good requirements
- Success Factors
  - Used only when requirements are flexible
  - Capture most important items
  - Involve key users

Minimal Specification vs. Extreme Programming

- This is not XP (extreme programming)
- In XP Requirements are
  - expressed as automated acceptance tests rather than specification documents
  - defined incrementally, rather than trying to get them all in advance
- XP has broader goals
  - An attempt to reconcile humanity and productivity
  - A mechanism for social change
  - A path to improvement
  - A style of development
  - A software development discipline

Requirements Scrubbing

- Removing a feature from the product
  - Eliminates all effort: spec., design, dev., test, doc.
  - The earlier the better
  - Typically done during or right after Requirements
- Less risky than minimal specification
- Aims
  - Eliminate all but absolutely necessary requirements
  - Simplify all complicated requirements
  - Substitute cheaper items

Versioned Development

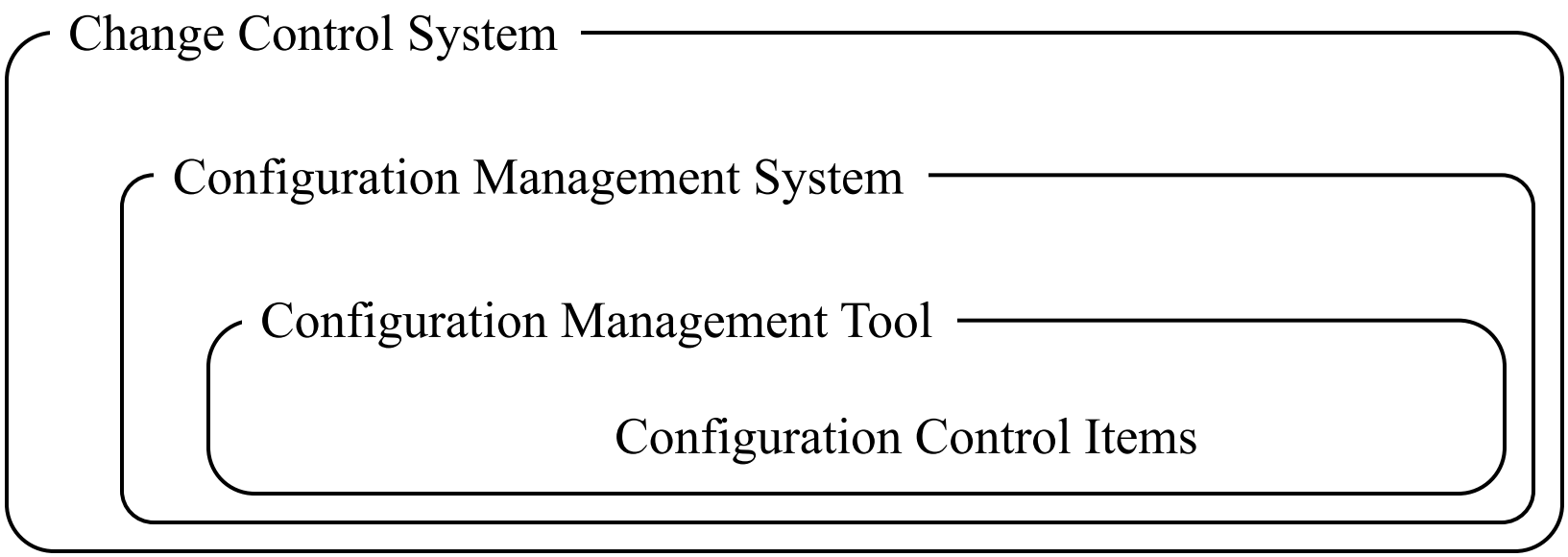
- Eliminate them from the current version
- “Let’s put it in release 1.1”
  - You’re still saying “Yes”, not “No”
- By next rev. the list has changed anyway
- My favorite ;-)

Feature-Set Control

- It is a classic mistake avoidance technique
- Early Stages
  1. Minimal Specification
  2. Requirements Scrubbing
  3. Versioned Development
- **Mid-Project**
  - **Effective change control**
- Late-Project
  - Feature cuts (and if needed project recovery)

Effective change control

- A set of fully fledge methods and tools is available



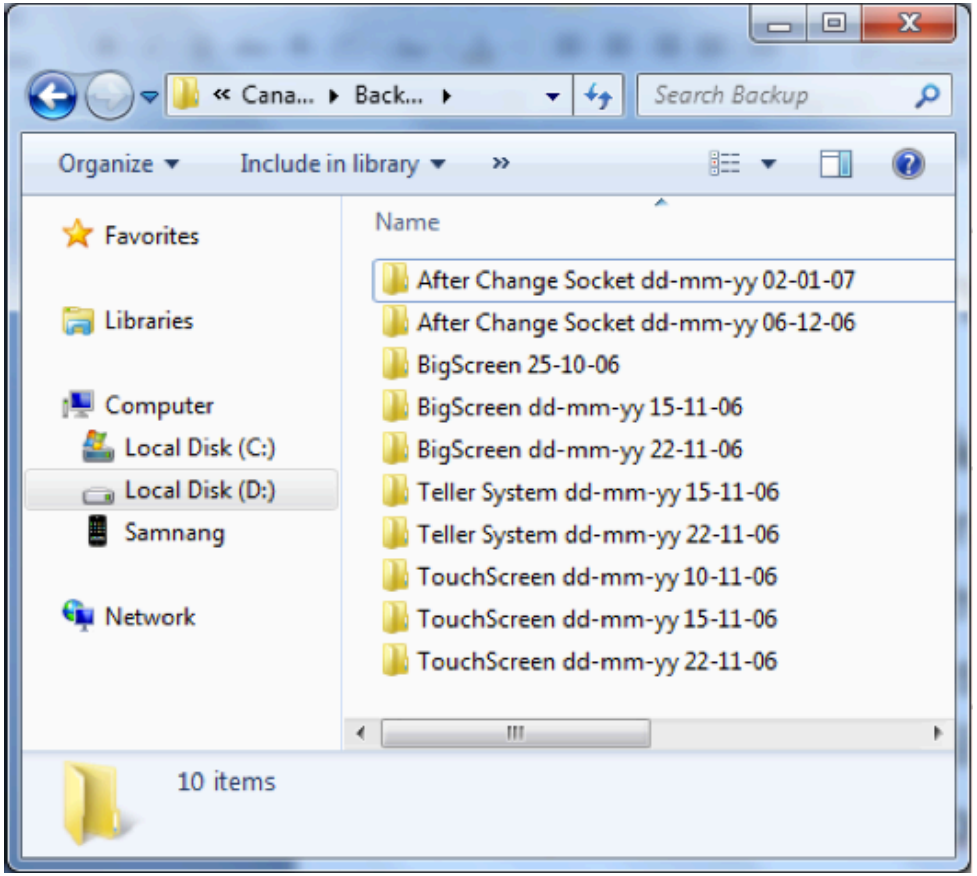
Change and Configuration Mng. Terminology

- Configuration Control Item (CCI)
  - Anything suitable for configuration control
  - Source code, documents, diagrams, etc.
- Configuration Control Tool (CCT)
  - Any support for managing CCIs
- Configuration Control
  - process of evaluating, approving and disapproving, and managing changes to CCIs using CCTs
- Change Control
  - process of controlling changes
  - typical steps
  - Proposal, evaluation, approval, scheduling, implementation, tracking

The Configuration Control Problem 1/2

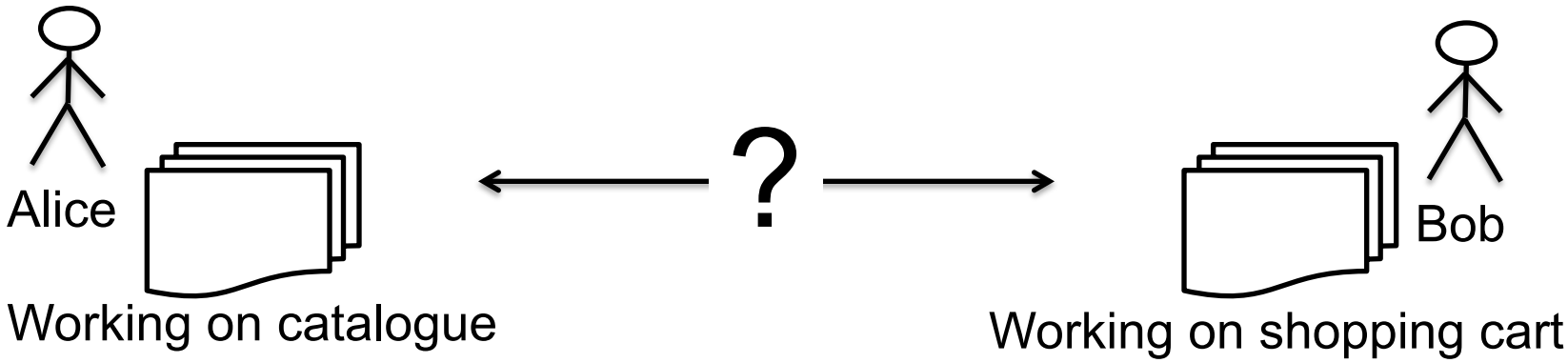
Working alone without Configuration Control

- Which version works?
- Which versions have bug X?
- Which versions have feature X?
- What's the different between certain versions?



The Configuration Control Problem 2/2

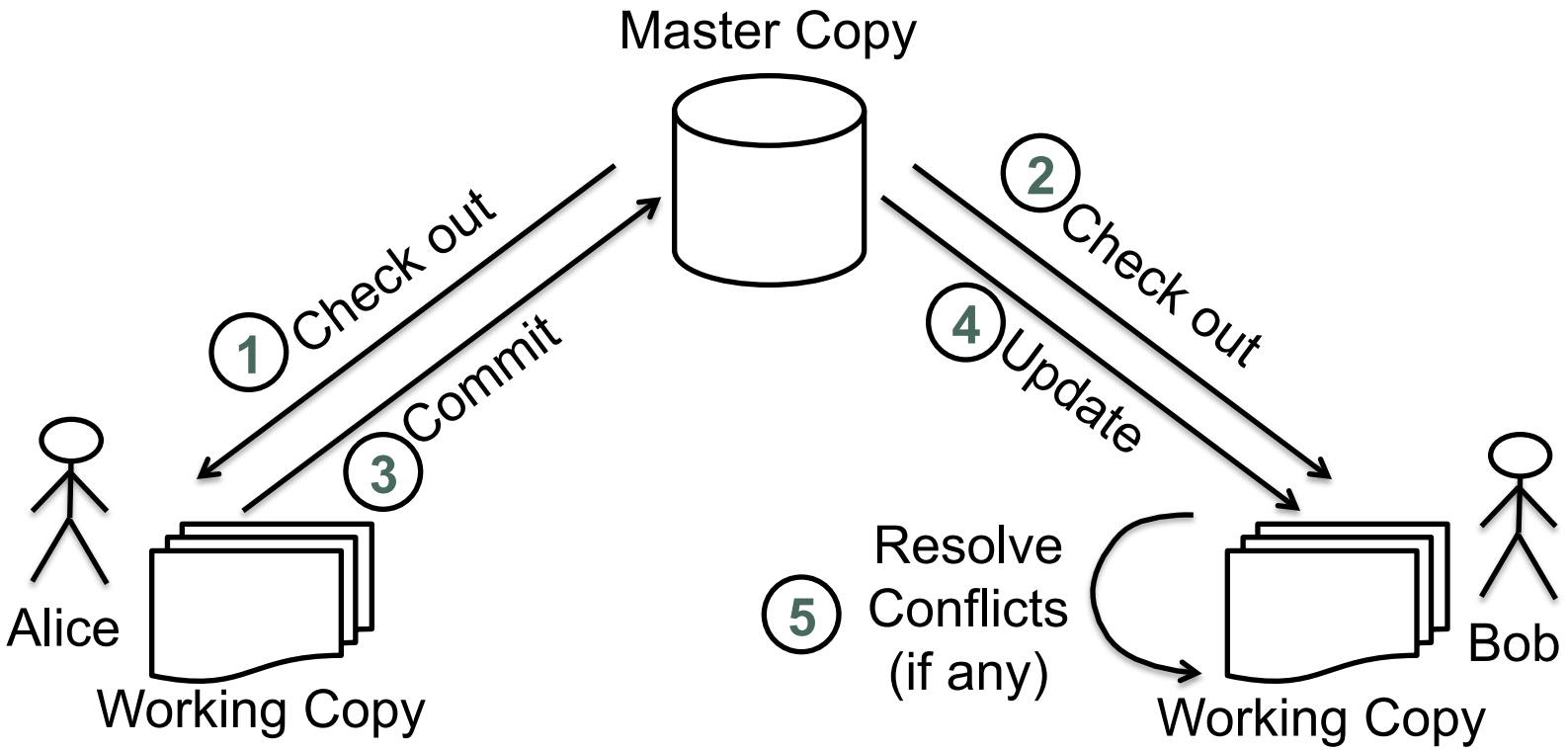
Working with others without Configuration Control



- Who is responsible on keeping the latest version?
- How to combine independently developed versions into one working program?

Basic Configuration Control Solution

- The Working Cycle



- Tools that support this working cycle
  - CSV, SVN, Git, etc.
- Hands on in Class 13

Configuration Control – Why?

- Backup & Restore
- Synchronization
- Short-Term Undo
- Long-Term Undo
- Track Changes
- Track Owner
- Branching & Merging

Configuration Control

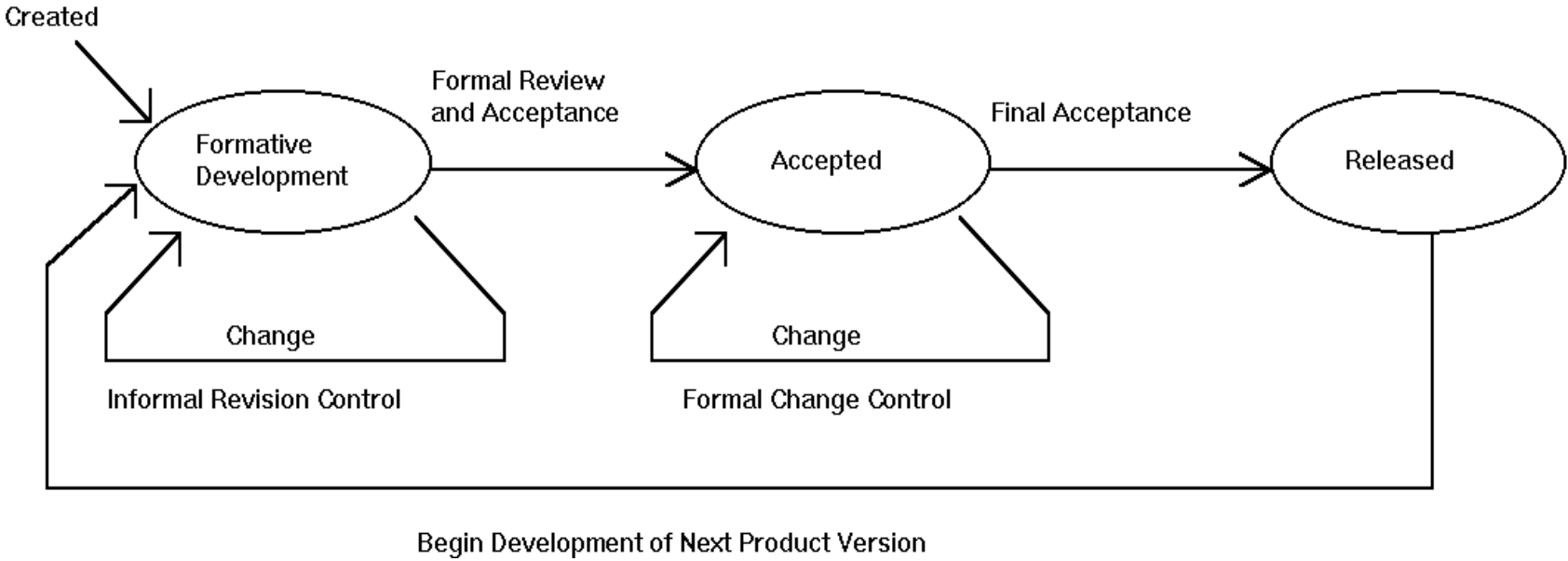
- A management support function
- Includes
  - Program code changes
  - Requirements and design changes
  - Version release changes
- Essential for developed items
  - Code, documentation, etc.

Configuration Control Needs

- Establish clearly defined management authority
- Setup control standards, procedures and guidelines
  - All team members must be aware of these
- Requires appropriate tools and infrastructure

Configuration Management Plan

- Must be produced during planning phase
- Often part of Software Development Plan
- Example of Change Control Procedure



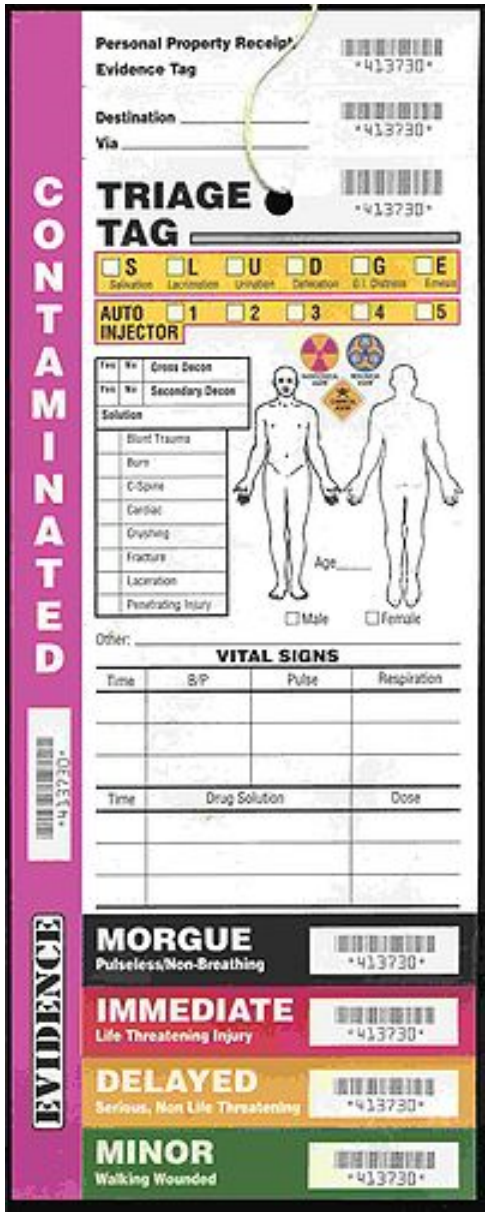
  - Cached version available
    - http://www.emanueledellavalle.org/slides/P&MSP2013_12b_Change-Procedure-by-McConnel.pdf

Change Control Board (CCB) 1/2

- McConnell “best practice” (see Ch. 17)
- Structure: representatives from each stakeholder party
  - Dev., QA, Marketing, Mgmt., Customer support
- Perform “change analysis”
  - Importance, priority, cost, benefit

Change Control Board (CCB) 2/2

- **A CCB is similar to a triage***
  - Allocating scarce resources
  - Some will not receive treatment
  - Life-critical to the project
- CCB will say “No” more than “Yes”
- Watch for bureaucracy



* a term from emergency medicine that refers to the activity of sorting injured people into groups so that the people who will most benefit from medical treatment receive it first

Feature-Set Control

- It is a classic mistake avoidance technique
- Early Stages
  1. Minimal Specification
  2. Requirements Scrubbing
  3. Versioned Development
- Mid-Project
  - Effective change control
- **Late-Project**
  - **Feature cuts (and if needed project recovery)**

Feature cut

- Stabilize the requirements
  - No more change requests will be accepted
- Trim the feature set
  - Determine priorities
  - cut the low ones

Project Recovery

- How to save a “drowning project”
- 3 Approaches
  1. Cut the size of the software
  2. Increase process productivity
  3. Slip the schedule, proceed with damage control
- Opportunity for decisive leadership action
- Not a time to ‘just cut corners’
  - Be realistic (not foolish)
- Timing: politically important
  - Not too early, not “too” late

Steps

- Assess situation
  - Is there a hard deadline, what's negotiable, etc.
- Don't do what's been done already
- Ask team what needs to be done

People Steps

- Restore morale
  - Sacrifice a sacred cow
    - Dress code, off-site, catered meals, etc
  - Cleanup personnel problems
- Focus people's time
  - Remove non-essential work

Process Steps

- Fix classic mistakes
  - Inadequate design, shortchanged activities, etc?
- Create “Miniature Milestones”
  - Small (in day(s)), binary, exhaustive
  - Boosts morale: getting things done!
- Track progress meticulously
- Recalibrate after a short time
- Manage risk painstakingly

Product Steps (partially seen in slide 28)

- Stabilize the requirements
- Raise the bar on change requests
- Trim the feature set
  - Determine priorities, cut the low ones
- “Take out the garbage”
  - Find error-prone modules; re-design
- Get to a known, stable state & build from there

Good Reference

- “Quality Software Project Management”, Futrell, Shafer, Shafer
  - Preview available on Google Books Search http://books.google.com/books?id=8GqC7xHTwGsC

Questions?