



 POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Planning and Managing Software Projects 2011-12
Session 14

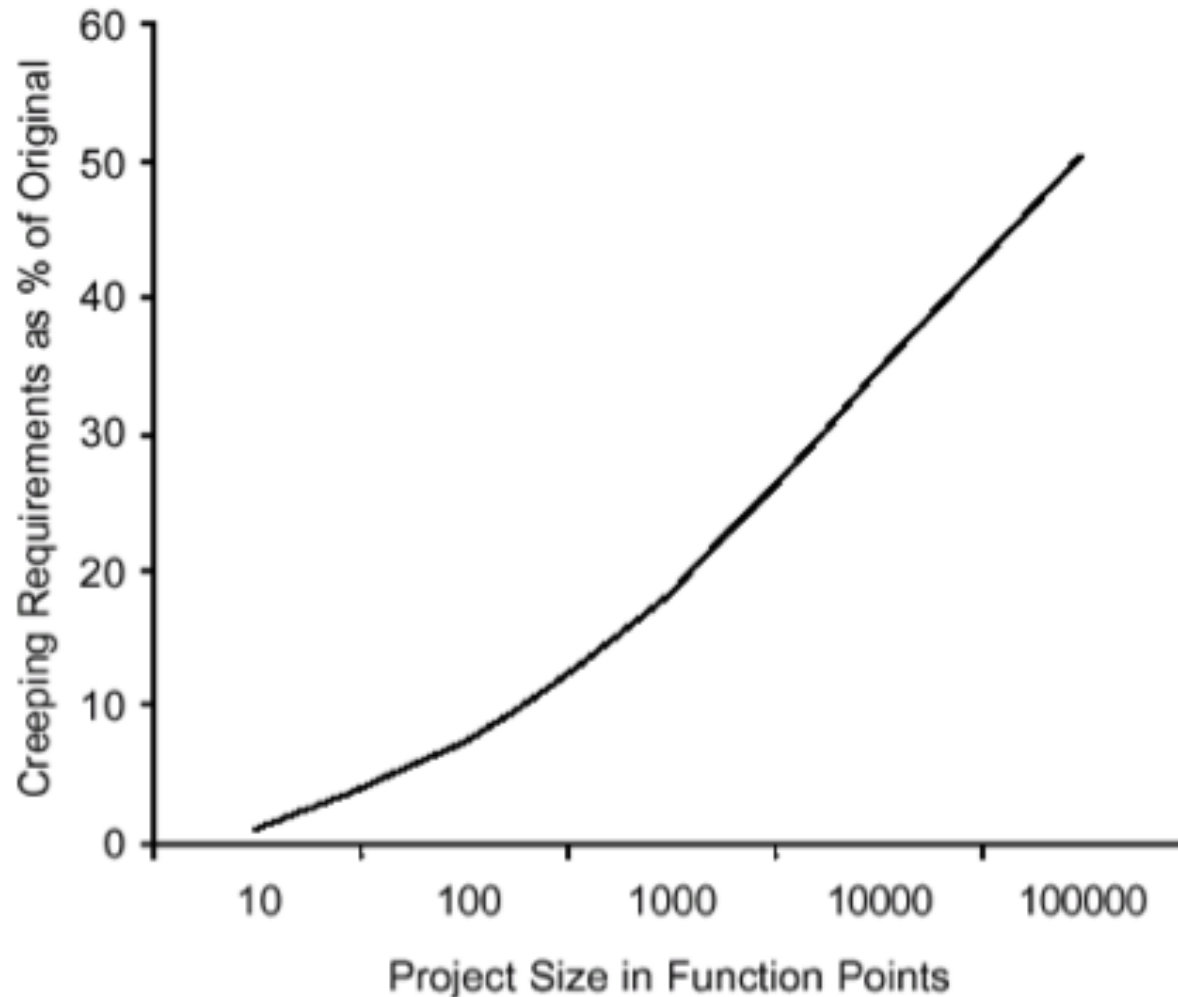
Change Management

Emanuele Della Valle
<http://emanueledellavalle.org>

- 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

- The Feature-Creep Phenomenon
- Feature Set Control

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



- 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 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

- 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

- 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

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

- 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.)

- 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

- 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

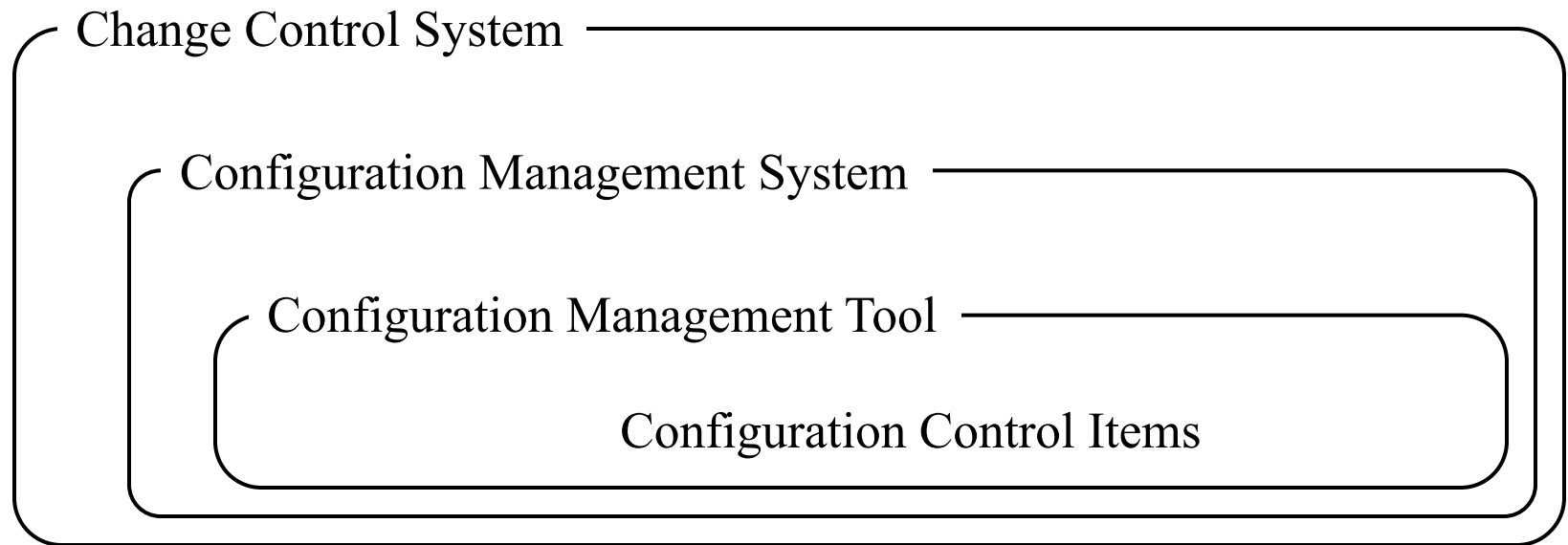
Requirements Scrubbing

- 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

- 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 ;-)

- 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

- A set of fully fledged methods and tools is available

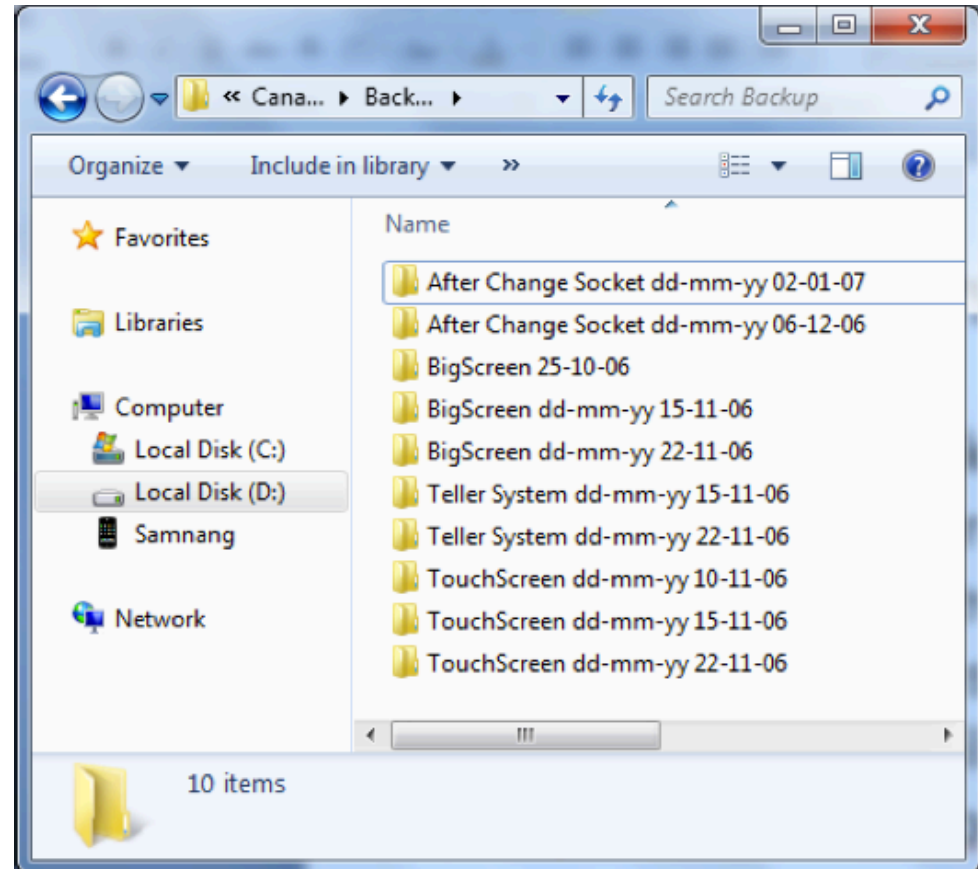


- Configuration Control Item (CCI)
 - Anything suitable for configuration control
 - Source code, documents, diagrams, etc.
- Configuration Control Tool (CCT)
 - Any support for managing CCIs
- Configuration Control
 - process of evaluating, approving and disapproving, and managing changes to CCIs using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

The Configuration Control Problem 1/2

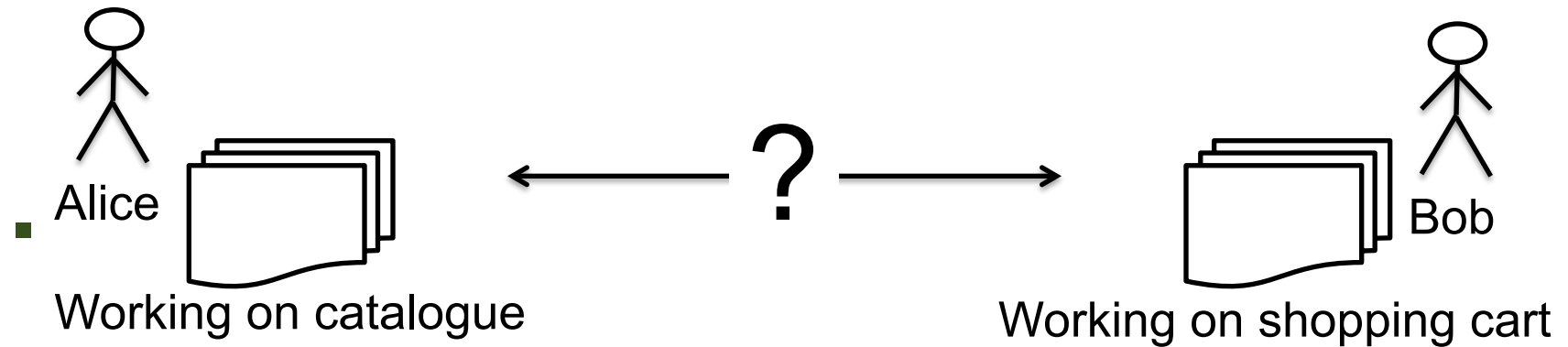
Working alone without Configuration Control

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



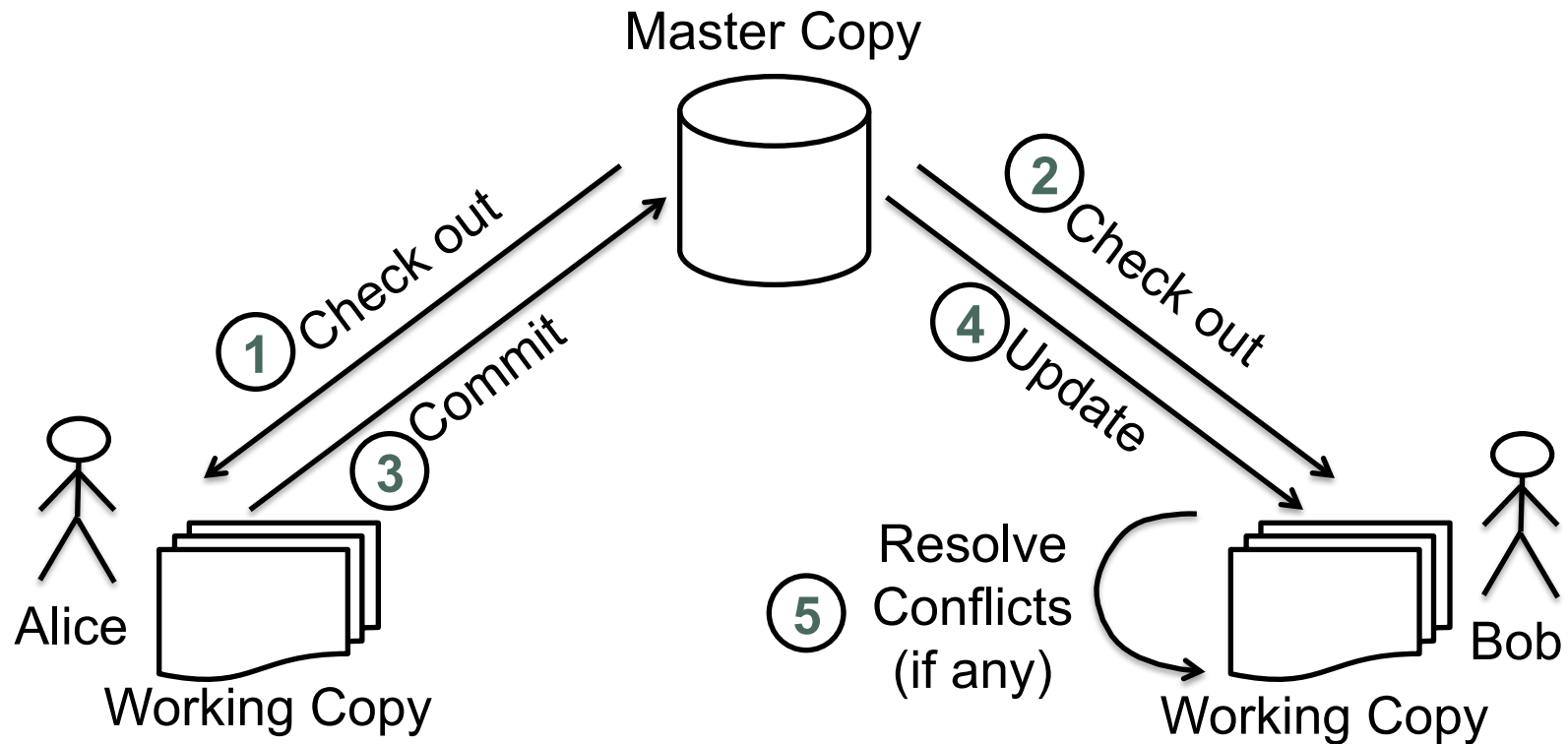
The Configuration Control Problem 2/2

Working with others without Configuration Control



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

- The Working Cycle



- Tools that support this working cycle

- CSV, SVN, Git, etc.

- Hands on in Class 18 and 20 on Software Factory

Configuration Control – Why?

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

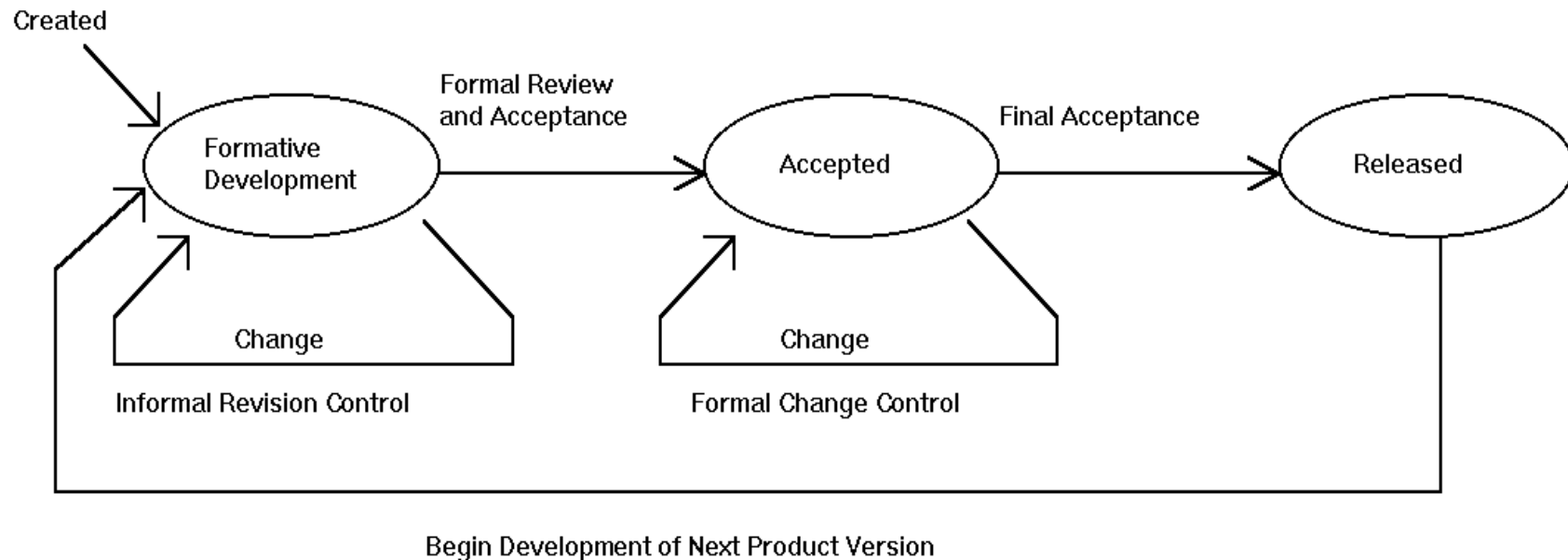
Configuration Control

- 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

- 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&MSP2012_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

Personal Property Receipt
Evidence Tag *413730*

Destination _____
Via _____ *413730*

TRIAGE TAG *413730*

S L U D G E
Solution Laceration Sprinkler Defectives G.I. Distress Emission

AUTO INJECTOR 1 2 3 4 5

Tri No. Gross Decon
Tri No. Secondary Decon

Solution

Blunt Trauma
Burn
C-Spine
Cardiac
Crushing
Fracture
Laceration
Penetrating Injury

Age _____
 Male Female

Other: _____

VITAL SIGNS

Time	S/P	Pulse	Respiration

Time	Drug Solution	Dose

EVIDENCE

MORGUE
Pulseless/Non-Breathing *413730*

IMMEDIATE
Life Threatening Injury *413730*

DELAYED
Serious, Non Life Threatening *413730*

MINOR
Walking Wounded *413730*

- 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**
 - **Not today, see “Project Recovery” section in class 19**

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

Questions?