

▶ POLITECNICO DI MILANO

Dipartimento di  
Elettronica e Informazione

Session 7

# Risk and Change Management

Emanuele Della Valle

<http://home.dei.polimi.it/dellavalle>

- This slides are largely based on 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

3

- Risk Management
- Feature Set Control
- Change Control
- Configuration Management

- Problems that haven't happened yet
- Why is it hard?
- Some are wary of bearing bad news
  - No one wants to be the messenger
  - Or seen as "a worrier"
- You need to define a strategy early in your project

- Identification, Analysis, Control
- Goal: avoid a crisis
- Thayer: Risk Mgmt. vs. Project Mgt.
  - For a specific vs. all projects
  - Proactive vs. reactive

- Characterized by:
  - Uncertainty ( $0 < \text{probability} < 1$ )
  - An associated loss (money, life, reputation, etc)
  - Manageable – some action can control it
  
- For an example see Service-Finder's "Risk Management and contingency plan"
  - [http://www.emanueledellavalle.org/slides/P&MSP2009\\_12\\_Service-Finder\\_Risk-Management.pdf](http://www.emanueledellavalle.org/slides/P&MSP2009_12_Service-Finder_Risk-Management.pdf)

## Project Risk (2)

7

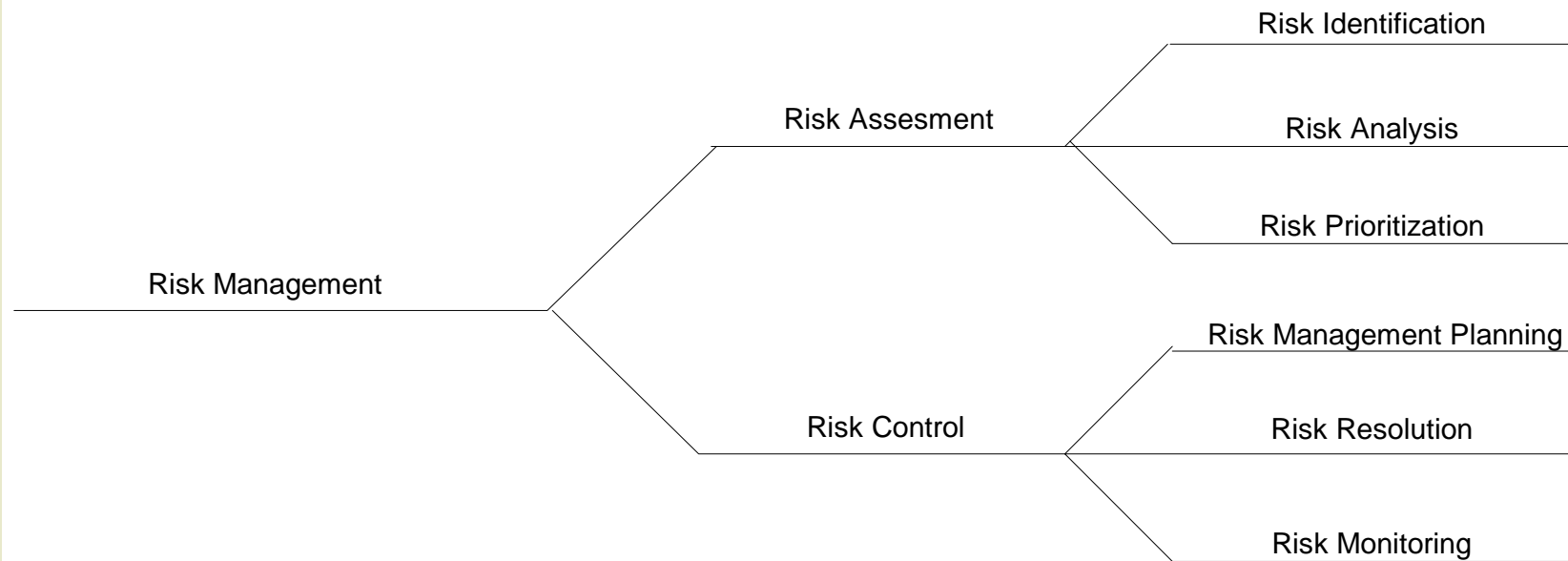
- Risk Exposure
  - Product of probability and potential loss
- Problem
  - A risk that has materialized

- Schedule Risks
  - Schedule compression (customer, marketing, etc.)
- Cost Risks
  - Unreasonable budgets
- Requirements Risks
  - Incorrect
  - Incomplete
  - Unclear or inconsistent
  - Volatile

## Types of Risks (2)

9

- Quality Risks
- Operational Risks
- Most of the “Classic Mistakes”
  - Classic mistakes are made more often



“Software Risk Management”, Boehm, 1989

- Get your team involved in this process
  - Don't go it alone
- Produces a list of risks with potential to disrupt your project's schedule
- Use a checklist or similar source to brainstorm possible risks
  - <http://www.construx.com/Page.aspx?hid=1134>

- Determine impact of each risk
- Risk Exposure (RE)
  - a.k.a. "Risk Impact"
  - $RE = \text{Probability of loss} * \text{size of loss}$
  - Ex: risk is "Facilities not ready on time"
    - Probability is 25%, size is 4 weeks, RE is 1 week
  - Ex: risk is "Inadequate design – redesign required"
    - Probability is 15%, size is 10 weeks, RE is 1.5 weeks
  - Statistically are "expected values"
  - Sum all RE's to get expected overrun
    - Which is pre risk management

- Estimating size of loss
  - Loss is easier to see than probability
    - You can break this down into “chunks” (like WBS)
- Estimating probability of loss
  - Use team member estimates and have a risk-estimate review
  - Use Delphi or group-consensus techniques
  - Use gambling analogy” “how much would you bet”
  - Use “adjective calibration”:
    - highly likely
    - probably
    - improbable
    - unlikely
    - highly unlikely

- Remember the 80-20 rule
- Often want larger-loss risks higher
  - Or higher probability items
- Possibly group 'related risks'
- Helps identify which risks to ignore
  - Those at the bottom

## Types of Unknowns

15

- Known Unknowns
  - Information you know someone else has
- Unknown Unknowns
  - Information that does not yet exist

- Risk Management Plan
  - Can be 1 paragraph per risk
  - McConnell's example
    - <http://www.construx.com/Page.aspx?hid=1286>

- Risk Avoidance
  - Don't do it
  - Scrub from system
  - Off-load to another party
  
- Risk Assumption
  - Don't do anything about it
  - Accept that it might occur
  - But still watch for it

- Problem control
  - Develop contingency plans
  - Allocate extra test resources
- Risk Transfer
  - To another part of the project (or team)
  - Move off the critical path at least
- Knowledge Acquisition
  - Investigate
    - Ex: do a prototype
  - Buy information or expertise about it
  - Do research

- Top 10 Risk List
  - Rank
  - Previous Rank
  - Weeks on List
  - Risk Name
  - Risk Resolution Status
- A low-overhead best practice
- Interim project post-mortems
  - After various major milestones
- McConnell's example
  - <http://www.construx.com/Page.aspx?hid=1293>

- Don't be afraid to convey the risks
- Use your judgment to balance
  - Sky-is-falling whiner vs. information distribution

- Miniature Milestones
- Feature-Set Control
- Traditional Specs
- Minimal Specification
- Requirements Scrubbing
- Versioned Development
- Mid-Project Feature-Creep
- Change Control Board (CCB)
- Configuration Control

- A risk-reduction technique
- Use of small goals within project schedule
  - One of McConnell's Best Practices (Ch. 27)
- Fine-grained approach to plan & track
- Reduces risk of undetected project slippage
- Pros
  - Enhances status visibility
  - Good for project recovery
- Cons
  - Increase project tracking effort

- Can be used throughout the development cycle
- Works with will hard-to-manage project activities or methods
  - Such as with evolutionary prototyping
- Reduces unpleasant surprises
- Success factors
  - Overcoming resistance from those managed
  - Staying true to 'miniature' nature
- Can improve motivation through achievements

- Requires a detailed schedule
- Have early milestones
- McConnell says 1-2 days
  - Longer is still good (1-2 weeks)
- Encourages iterative development
- Use binary milestones
  - Done or not done (100%)

- Class mistake avoidance
- Early Stages
  - 1. Minimal Specification
  - 2. Requirements Scrubbing
  - 3. Versioned Development
- Mid-Project
  - Effective change control
- Late-Project
  - Feature cuts

- 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

- This is not XP (extreme programming)
- 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

- 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

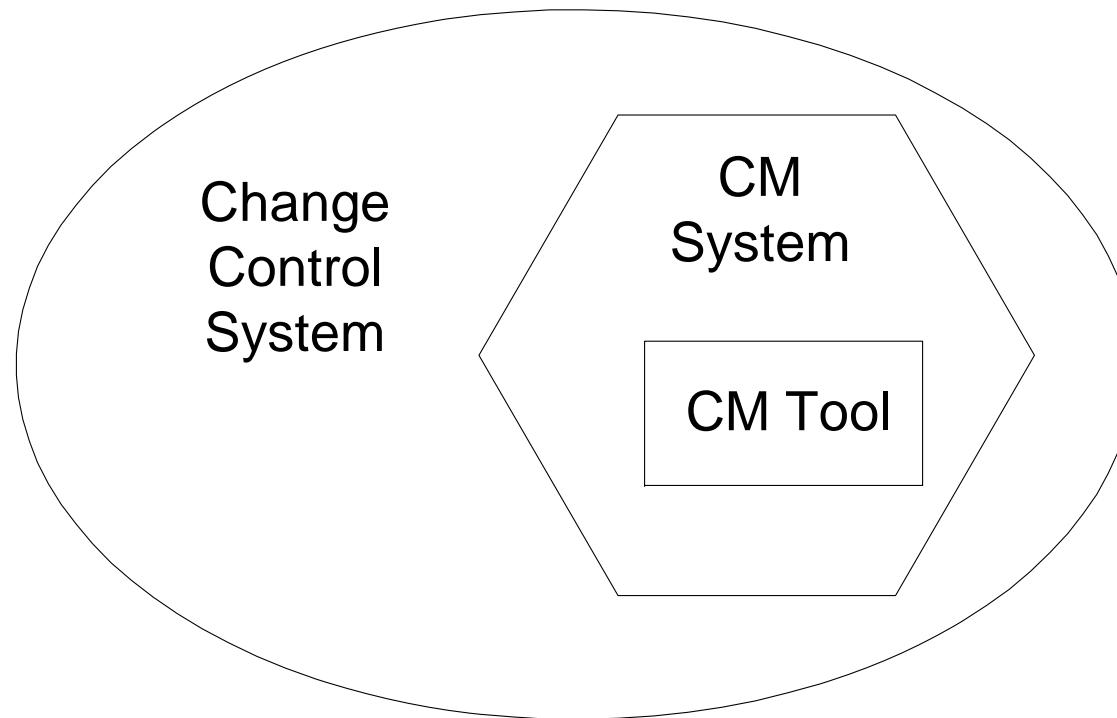
- Eliminate them from the current version
- "Let's put it in release 1.1"
  - You're still saying "Yes", not "No"
- By next rev. the list has changed anyway
- My favorite ;-)

- 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 spec. can cover all details. You must.
- Programmer ideal: flip switch
- Up to 10-1 differences in prog. size with the same specs.

- 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
- Triage
  - Allocating scarce resources
  - Some will not receive treatment
  - Life-critical to the project
- Will say “No” more than “Yes”
- Watch for bureaucracy

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



- A management support function
- Includes
  - Program code changes
  - Requirements and design changes
  - Version release changes
- Essential for developed items
  - Code, documentation, etc.
- Example
  - The case of the code that used to work
    - But didn't in time for the demo

- Software Configuration Control Item (SCCI)
  - a.k.a. Source Item (SI)
  - Anything suitable for configuration control
  - Source code, documents, diagrams, etc.
- Change Control: process of controlling changes
  - Proposal, evaluation, approval, scheduling, implementation, tracking
- Version Control: controlling software version releases
  - Recording and saving releases
  - Documenting release differences
- Configuration Control: process of evaluating, approving and disapproving, and managing changes to SCCIs.

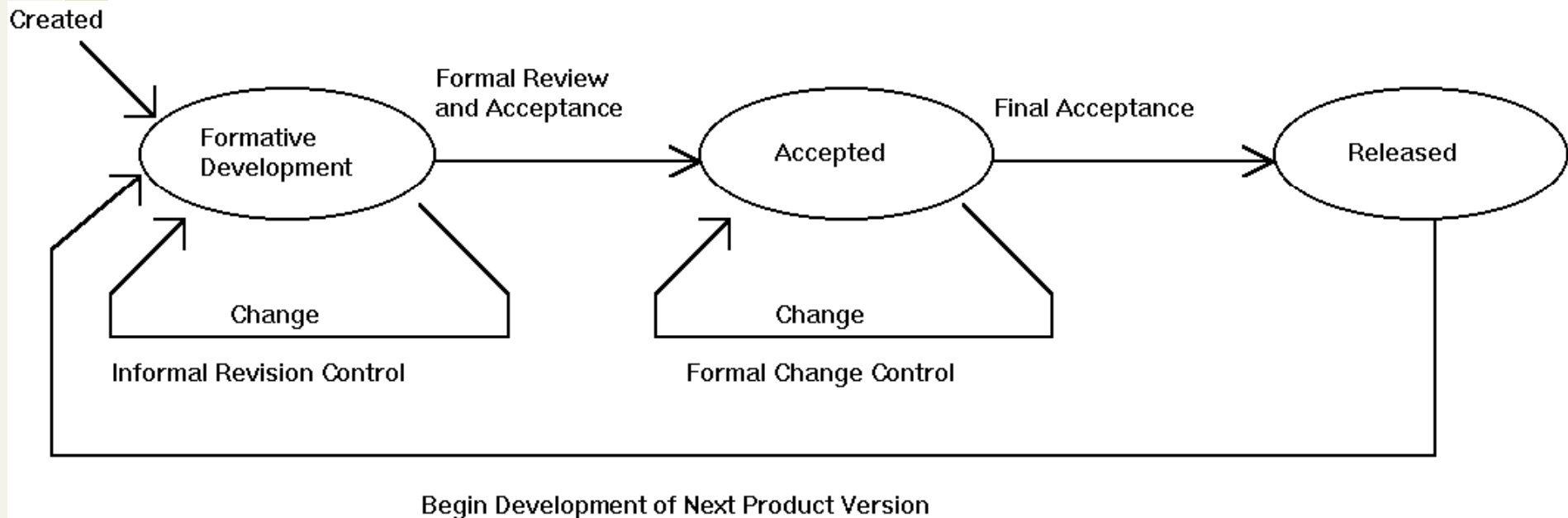
- Software Configuration Management
- Formal engineering discipline
- Methods and tools to identify & manage software throughout its use

- See also  
[http://en.wikipedia.org/wiki/Software\\_configuration\\_management](http://en.wikipedia.org/wiki/Software_configuration_management)

- Establish clearly defined mgmt. 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
  - See <http://www.construx.com/Page.aspx?hid=1424>
  - Often part of Software Development Plan

# Example of Change Control Procedure

39



[Source <http://www.construx.com/Page.aspx?hid=1117>]

- SCM is very important during all phases starting with Requirements
- Continues to be important during Maintenance

## Optional Readings

41

- McConnell: 11 “Motivation”, 13 “Team Structure”
- Schwalbe: 8 “Project Human Resource Management”

- Top 10 Risk List for your project
- Get inspired by <http://www.construx.com/Page.aspx?hid=1293>

# Questions?

43