Introduction, Project Plan: Textbook Chapter 2

CSE 4322, Fall 2013
Christoph Csallner
University of Texas at Arlington (UTA)

These slides contain textbook material by:

- Hans van Vliet [VV]: http://www.cs.vu.nl/~hans/SEbook.html
- Steve McConnell [MC]: Software Project Survival Guide

Overview

- A software development project is often complex
- Number of people involved
  - May be hundreds of people
- Project duration
  - May span multiple years

- Requires careful planning and controlling
- What are the key aspects?
  - Time, information, organization, quality, money

Software is not developed in isolation

- **Project has to satisfy boundary conditions**
- Boundary conditions may change during the project
  - E.g., management adjusts business model, technology, ...
- Overriding goals of the organization
  - E.g., management wants your project to use technology A

Software is not developed in isolation

- Other (software development) projects going on in the organization at the same time
  - Need to tune your project to these projects / interests
  - Other projects may develop common interfaces
  - E.g., the program you are developing will have to communicate with program B. B is currently developed by group C. C may be more important to management

Software is not developed in isolation

- Software is typically a part of a larger system
  - Software is an important part, but not the only part
- Software needs hardware to do any work
  - Do we have to use certain existing hardware?
  - Do we have to share this hardware with other software?
  - E.g., Google is developing their own server hardware
- Software has to satisfy laws and regulations
  - Laws not the same in every country..
  - Privacy laws, etc.
- Software has to be used and maintained by people
  - People are very different
  - Need people that understand the programming language

Example: Library Automation System

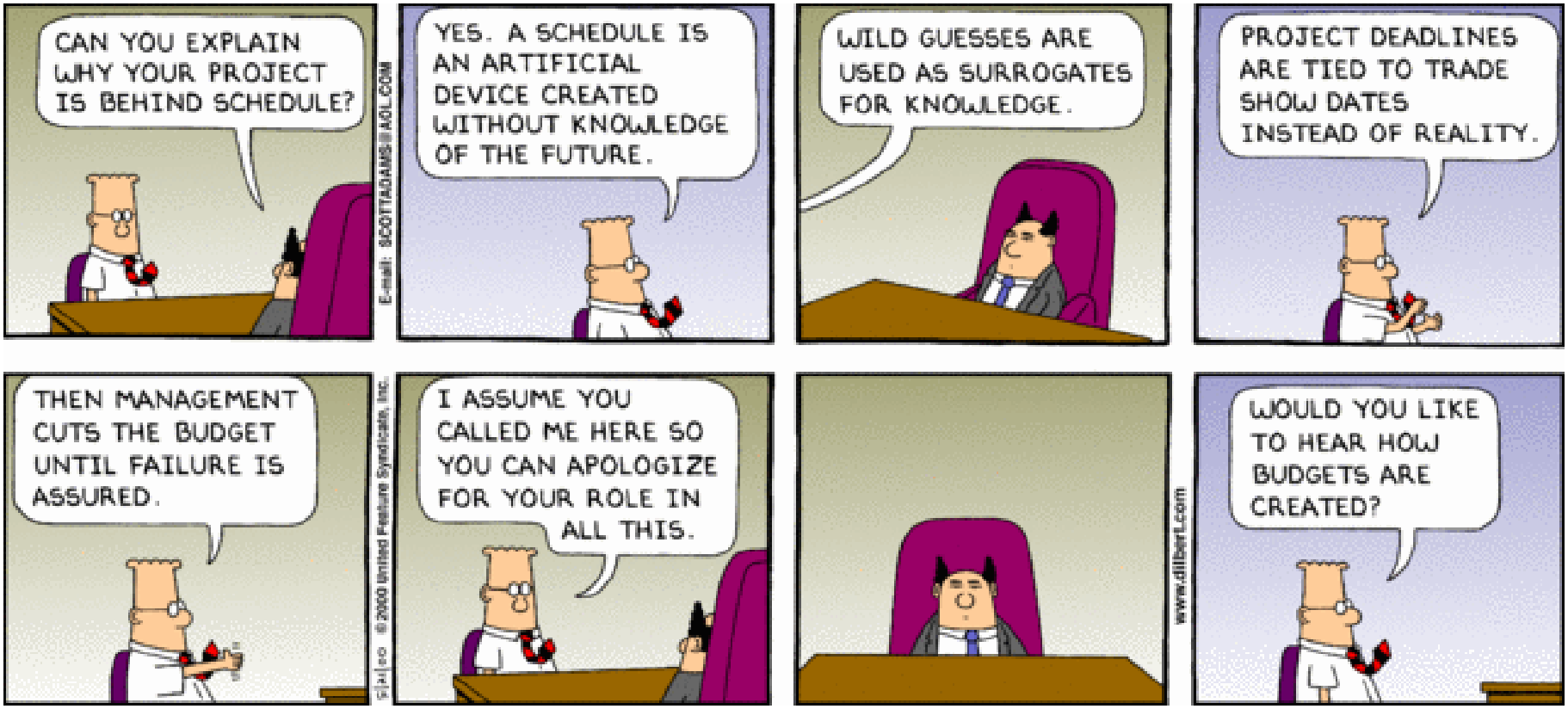
- System := Software + Database + Hardware + ...
- Book classification/ordering scheme
  - Organize books by ISBN, author name, etc.?
- Identify each book, distinguish copies of same book
  - Naming scheme? Bar code? RFID hardware?
  - Hardware to scan books quickly
  - Prevent theft: Need additional hardware in books?
- Add or discard a book
  - Need new label/RFID/anti-theft device?
- Requirements on staff/training
- Access over the web, from smart phones, ...

Keep planning throughout the project

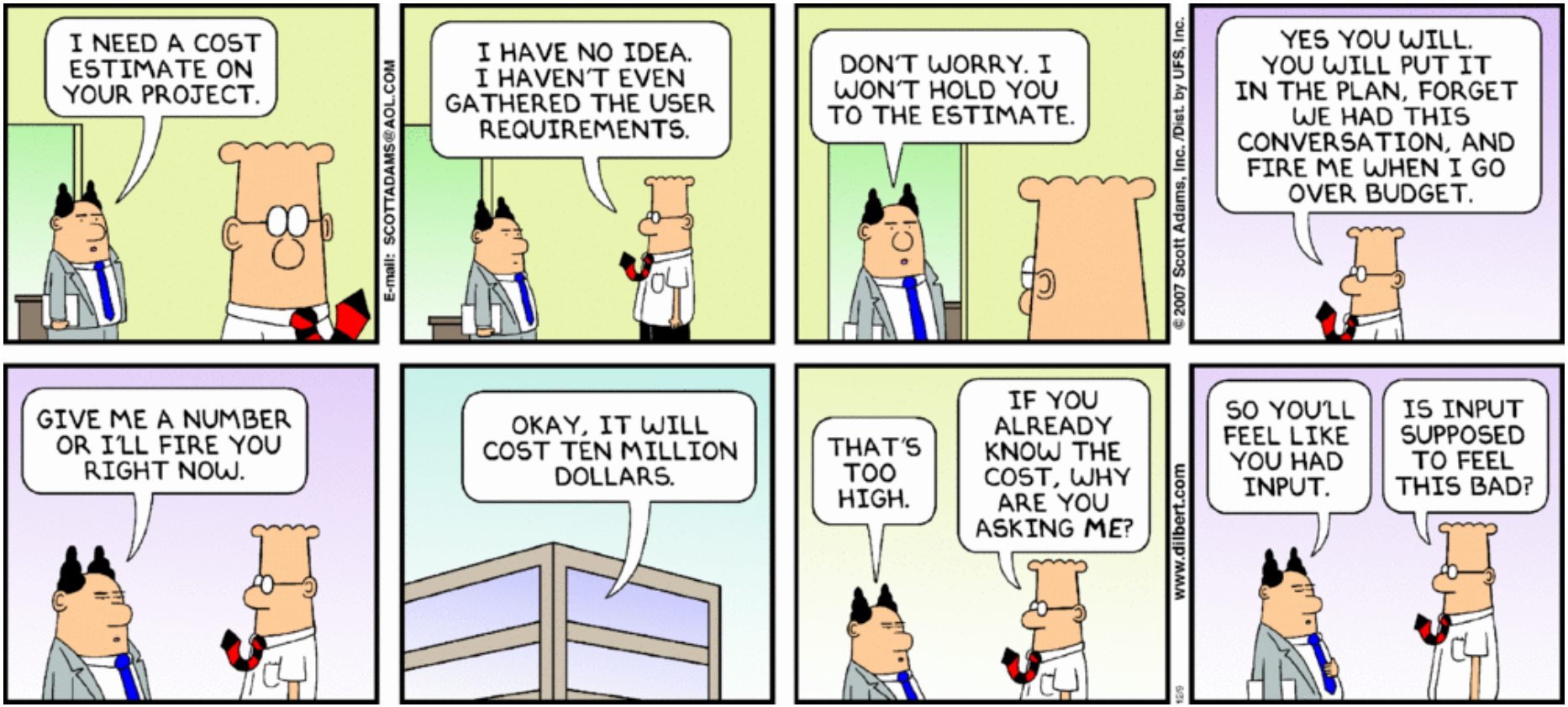
- One of the first activities, even before project starts
- How long will it take to complete the project?
  - For accurate planning, we need to know what project should deliver
  - But we likely do not (yet) know what project should deliver
  - Elaborating requirements is part of the project
- Planning early on in the project is imprecise
  - We cannot plan precisely what we know little about
- Planning is not a one-time activity

→ **Start with rough project plan draft and keep adapting it throughout the project as needed**

Schedule: An artificial device created *with limited knowledge* of the future



Cost Estimate: Created with limited knowledge of the future



Project Plan (part 1)

- Summary
- Background and history
- Aims / Vision
- Deliverables
- Names of people responsible
- Process model
- Activities
- Milestones + how we know milestones are reached
- Critical paths

Project Plan (part 2)

- Relationship of the project with other entities
  - Management
  - Related projects: Shared interfaces
  - Users + information, services, resources, facilities provided by users
- Roles in project team: What do they mean?
  - PM, e.g.: Traditional vs. Microsoft
  - Analyst
  - Architect, technical lead
  - Developer (“dev”, “engineer”)
  - Tester

Project Plan (part 3)

- Standards, guidelines, procedures and the tools to enforce them
  - Coding standards: IDE settings, code analysis/review tools
  - Documentation: Review procedures
  - Shared source code: Source control system
  - Bug reporting: Bug tracking system
  - Task planning: Task management system
- Management activities
  - Regular status/progress reports
  - Prioritize requirements, schedule, cost

Project Plan (part 4)

- Risk management
  - Every project faces risks, especially software projects
  - What are the risks? Few examples:
    - Competitor provides feature A earlier than us
    - Users do not like how we provide feature B
    - Required human/software/hardware/etc. resources not available
    - Some used technology not as reliable as anticipated
  - For each risk, how likely will it become reality?
  - What will we do if a risk becomes a reality?
- Staffing
  - For each role, how many people do we need when?
  - Skills required

Project Plan (part 5)

- Methods and techniques: How we do
  - Requirements engineering, design, implementation, testing, etc.
  - Testing order and equipment
  - Acceptance testing (testing under supervision)
  - Maybe merge with standards, guidelines, procedures and the tools to enforce them
- Quality Assurance
  - Organizations and procedures used
  - Maybe merge with standards, guidelines, procedures and the tools to enforce them

Project Plan (part 6)

- Work packages
  - Break project into separate tasks
  - Do that recursively → task hierarchy
- Resources needed
  - Hardware, CPU cycles, storage
  - Personnel, maybe merge with staffing
- Budget and schedule
  - Map budget to required resources and task hierarchy
  - Schedule items of task hierarchy
  - How we track resource consumption / task completion

Project Plan (part 7)

- Changes
  - The world changes rapidly, will affect project
  - Customer may change his mind, etc.
  - A single change may affect all project deliverables: requirements, design, code, documentation, tests, etc.
  - How will we register/review/approve/include changes?
  - Example: Agile processes have built-in mechanisms to deal with change
- Delivery
  - How system will be handed over to the customer

A project plan is just a tool

- What customers really want is a high-quality software system
  - Do not care how you get there
- If you can achieve this without planning, do not plan
  - But experience shows that some planning is needed and will save you a lot of money and problems down the road
- A project plan is not an end in itself
  - You are typically not getting paid for producing a highly formatted Word document

Executive Sponsorship

- Project in trouble if it does not have an executive sponsor
- Final authority over project-wide decisions
  - Commit to feature set
  - Approve UI design
  - Decide if ready to release
- One person or a group
  - If group, then one person per interest: Marketing, management, etc.
  - Get group to agree
- Project plan should identify executive sponsor

Publicize Plans

- Get input, feedback, corrections from stakeholders if stakeholders are aware of plans
- Otherwise stakeholders are not aware of plan and plan becomes obsolete
  - Project runs out of control
- Review and approve by people carrying out the work
  - Developers, testers review & approve plan created by manager
- Managers chooses to get disapproval early (of the plan) or late (when team fails to deliver software as planned)

Managers Need Team Buy-In

- Need good relation between team and management
- Team wants to do good work
- Do not try to trick your developers or testers
  - Will not magically give you more or better software
- Role of management is to coordinate activities so that efforts do not go to waste
- All planning documents should be readily available for all developers and testers

Publicize Progress Indicators

- All progress indicators should be readily available to all project stakeholders
  - Task list & completion status
  - Defect statistics
  - Top 10 risks
  - Percent of schedule used
  - Percent of resources used
  - Status reports to upper management
- Create web site that contains all indicators
  - Also link to project deliverables and background information
  - Easier to access than source control

Controlling a project

- Time, information, organization, quality, money
  - Will revisit these in following lectures
- Adding people to a late project may delay it further
  - Project team has to spend time to train new people
  - Fred Brooks law
- Quality vs. time/money
  - When to stop testing / fixing bugs?
  - Will never find all bugs
  - If we do not ship we will never get paid
  - Microsoft ships new version of Windows operating system with thousands of known bugs (plus additional unknown ones)

Shared, elevating, exclusive, and achievable

PROJECT VISION

Team needs to buy in to shared vision

- High-performance team work requires a shared vision
  - Study of 75 projects showed that each effective team had a shared vision
- Helps streamline decisions on smaller issues
- Helps avoid time-wasting side-activities
- Builds trust among team members
  - Everyone working on same goal
  - Enables good cooperation among team members
- Do not have to revisit big questions over and over again throughout project

Vision should be elevating

- Effective vision motivates team
- Team needs a challenge / mission
- Team does not form around mundane goal
- Will a team form around the following vision?
  - “We’d like to create the third-best web site designer on the market and deliver it 25% later than the industry average.”

Vision should be elevating

- Team response to challenge is emotional
- Team response is influenced by how it is presented
- Re-present challenge from last slide:
  - “We are going to create a web site designer that will take us from zero market share to 25% market share in the first 6 months. We are cash-poor so we’re going to make it our goal to define a feature set and ship date that will allow us to use a small, sharp, highly efficient team to establish a foothold in the market before we run out of cash.”

Vision must be achievable

- Some people can be motivated with impossible goals
  - Sales and marketing people
- Software developers tend to see impossible goals as illogical
  - Impossible goals often de-motivate software developers
- Same applies if each sub-goal is achievable but the collection of goal is not achievable
  - Example: short schedule + low cost + many features

Vision must be achievable

- On some projects it is not clear initially if goals are achievable or impossible
- Developers may find out that goals are impossible before management finds out
  - Creates bad dynamics
- Management may insist on impossible goals
  - De-motivates developers

Define what to leave out

- Vision statement should be written such that it is clear what is part of project and what is not
- A good vision statement is exclusive
  - Word the vision statement such that it excludes things that should not be part of the project
- Exclusive vision helps team to produce the minimal amount of software
  - Key to software project management

Committing to the vision

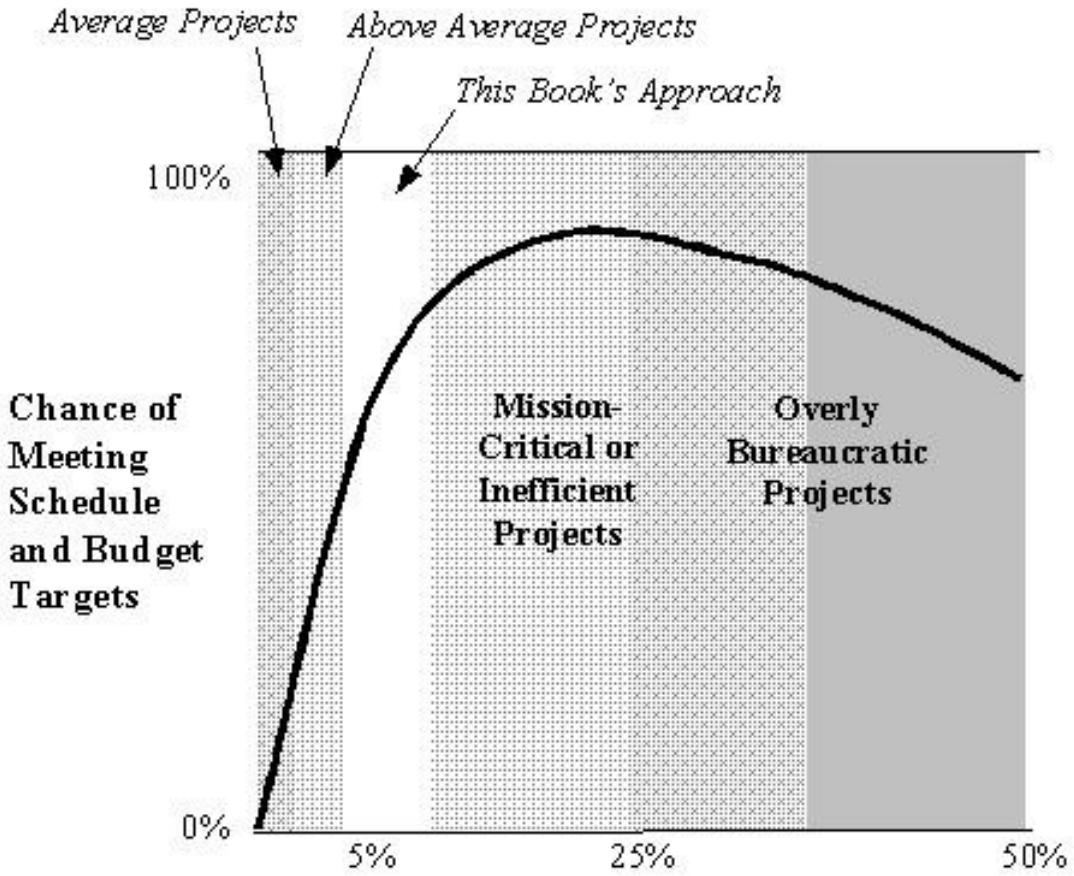
- Vision statement should be written down
- Vision statement is the first item put under version control
  - Check into Subversion repository
- Should not be changed frequently
- Should be changed when understanding of project changes

RISK MANAGEMENT

Risk Management

- Risk = Possibility of loss (or injury)
  - Lose money
- Risk management is an overhead
  - You are paid for delivering software
  - You are not paid for risk management
- It is possible to do too much risk management
  - Risk management can become a risk
  - But most projects do too little risk management
- More risk management improves most projects
- Goal: Use risk management to trade a small investment (risk management) for a large reduction of risk

Risk Management Payoff



Risk Management

- Commit to risk management
  - Describe risk management approach in project plan
  - Include in budget: Funds for risk resolution
  - Include impact of assessed risks in project plan
- Enable risk management
  - Easy part is assessing risks
  - Hard part is setting up a risk reporting channel from developers to higher managers

Risk Officer

- Team member
- Ideally not the project manager
- Responsible for detecting emerging risks
  - Play devil's advocate
  - Frequently look for potential problems
- Need respect from management
  - Otherwise becomes the "pessimistic guy"
- Typically a senior technical person
- But all team members should look out for risks

Top 10 Risk List

- Maintain list of current top-10 risks to project
  - Keep list current throughout project
- Small project such as class project may work with Top-3 or Top-5 list
  - Big project may need Top-15, etc.
- Each project team already has a preliminary list
- Risk officer, project manager, and upper management review list every two weeks
- Publicize to all stakeholders
- Management should reward team members for reporting the risks team members find in their domain

Top-10 Risk List Template

- Five columns
- For each risk:
  - This week's rank
  - Last week's rank
  - Weeks on list
  - Short risk description
  - Risk resolution progress

Top-10 Risk List Example

This wk	Last wk	Wks on list	Risk	Risk resolution progress
1	1	5	Creeping requirements	•UI prototype used to gather high quality requirements •Requirements spec placed under explicit change control •Staged delivery to enable changing features if needed
2	5	5	Requirement gold-plating	•Vision statement specifies what is not included •Design emphasis on minimalism •Reviews check for extra design or implementation

[Excerpt from MC Table 7-2]

Risk Management tool support

- Can use defect-tracking system
  - But separate risks from regular defects
- Supports marking risk as open or closed
- Can assign each risk to an owner

Detailed risk management plans

- Each risk in Top-10 Risk List should have a plan
  - Why is a plan needed for this risk? Describe risk probability, consequences, and severity
  - How will the risk be resolved in general? List the options that were considered
  - What specific steps will be taken to resolve the risk? List steps and deliverables
  - Who is responsible for completing each step?
  - When will each step be completed?
  - How much budget is allocated to resolve the risk, broken down to each step?

Anonymous risk reporting

- Reporting bad news is hard
- Need channel from team to upper management
  - Can be simple box in break room
  - Can be simple intranet-based app (example next slide)
- Examples:
  - Tester can report that developers hand over their code later than planned
  - Developer can report that project manager misreports project progress
- Top management wants to hear warning signals as early as possible

www.construx.com/Thought_Leadership/Books/Survival_Guide/Resources_By_Chapter/Anonymous_Feedback_Pag

Both good and bad news should be able to flow up and down the project hierarchy. If you believe an issue is not being addressed properly through normal channels, please use this form to give feedback about the project anonymously. Using this form will not reveal your identity in any way.

Feedback

Severity of the issue: Sev 1 - Risk to Project Completion

Please describe the issue that needs to be addressed:

Who needs to know about this issue:

Management	Customer	Project Team
☐ Executive Sponsor	☐ Marketing Rep	☐ Engineering Lead
☐ Program Manager	☐ End-User Rep	☐ QA Lead
☐ Project Manager	☐ Other	☐ Documentation Lead
☐ Engineering Manager	☐ Other	☐ Editor
☐ QA Manager	☐ Other	☐ Other
☐ Documentation Manager		☐ Other
☐ Other		☐ Other
☐ Other		

Submit the Feedback Form

Submit Reset

TIME ACCOUNTING

Time Accounting

- Begin detailed time accounting now
- Critical for project visibility and control
- Lays foundation for more accurate estimates and planning in subsequent projects
- Enables comparing estimated with actual times
- Review time spent on rework
  - Was enough time spent on fixing problems when they were introduced?
- Use categories of following slides
  - [MC, Table 7-4]: http://www.construx.com/Sample_Time_Accounting_Categories/

Time Accounting Categories (1)

- Management
  - Plan
  - Manage customer/end-user relations
  - Manage change
- Administration
  - Downtime
  - Lab setup
- Process development
  - Create development process
  - Review development process
  - Rework development process
  - Educate customer or team members about dev. process

Time Accounting Categories (2)

- Requirements Development
  - Create specification
  - Review specification
  - Rework specification
  - Report defects detected during specification
- User Interface Prototyping
  - Create prototype
  - Review prototype
  - Rework prototype
  - Report defects detected during prototyping

Time Accounting Categories (3)

- Architecture
  - Create architecture
  - Review architecture
  - Rework architecture
  - Report defects detected during architecture
- Detailed Design
  - Create design
  - Review design
  - Rework design
  - Report defects detected during design

Time Accounting Categories (4)

- Implementation
  - Create implementation
  - Review implementation
  - Rework implementation
  - Report defects detected during implementation
- Component acquisition
  - Investigate/acquire components
  - Manage component acquisition
  - Test/review acquired components
  - Maintain acquired components
  - Report defects in acquired components

Time Accounting Categories (5)

- Integration
  - Automate build
  - Maintain build
  - Test build
  - Distribute build
- System testing
  - Plan system testing
  - Create manual for system testing
  - Create automated system test
  - Run manual system test
  - Run automated system test
  - Report defects detected during system test

Time Accounting Categories (6)

- Software release
  - Prepare and support alpha, beta, or staged release
  - Prepare and support final release
- Metrics
  - Collect measurement data
  - Analyze measurement data

THE POWER OF PROCESS

Software Process

- Commit all requirements to writing
- Systematic procedure for changing requirements
- Systematic technical reviews of all requirements, design, code
- Systematic Quality Assurance Plan includes Test Plan, Review Plan, Defect Tracking Plan
- Implementation Plan defines order for developing and integrating software components
- Use automated source control (Subversion, etc.)
- Revise cost & schedule estimates when a milestone is reached

Anti-Process View *(Wrong!)*

- Process = Unnecessary overhead, waste of time
  - Rigid, restrictive, inefficient
- Just hire the best people and give them the resources they ask for
  - Let them run wild
- Throughout project, each developer is productive for 90% of each work day
  - Remaining 10% spent on making & fixing mistakes
- Adding process will take away big chunk of these 90%
  - Overall productivity will therefore decrease

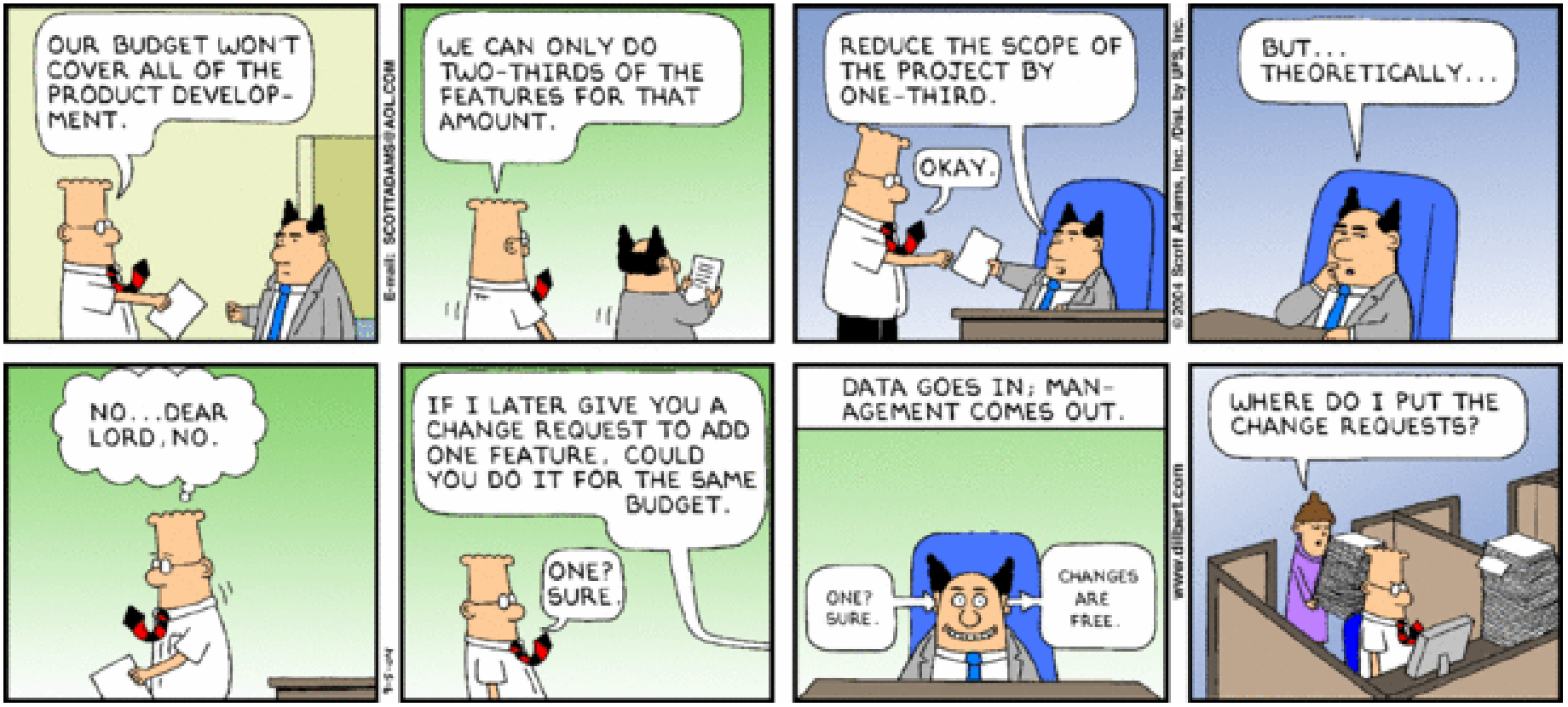
Why Is Negative View Wrong?

- Every project will eventually adopt some process
  - Early on and well-planned
  - Or later, forced by cost overruns, bugs, etc.

Without good process from early on (1)

- Without Change Control:
  - Manager tells some developers about new requirements
  - Developers agree informally
- No systematic review of requirements or their impact
  - Developers implement such new requirements
  - Size of project keeps growing
- Recipe for cost & schedule overruns
  - Have to consider consequences of requirement changes
  - Impact on cost and schedule
  - Impact on other requirements

Change Management



Without good process from early on (2)

- Without Quality Assurance:
  - Do not immediately fix bugs
  - Let bugs linger in documents and code base
  - Build software on top of buggy documents and code
- Recipe for cost & schedule overruns
  - Lot of bugs
  - Hard to implement and test new code as old code buggy
  - Spend a lot of time finding and fixing old bugs
  - Development speed decreases, lot of debugging

Without good process from early on (3)

- Without Defect Tracking:
  - No central bug database or tracking system
  - Bugs not recorded & tracked
  - Many bugs forgotten and not fixed
- Some bugs are easy to fix
  - Not fixed but included in shipped software

Without good process from early on (4)

- Without Source Control:
  - Some developer accidentally overwrite master version with old version
  - Work of many hours, days, or weeks can get lost
- Developers email archives to each other
  - Time wasted on manual file copying
  - Hard to keep track of different versions, branches
  - Hard to roll back to old versions

Without good process from early on (5)

- Saved some time early in project
  - Did not spend the time to set up process
  - Initially higher productivity than with process
  - Initial high productivity is deceiving
- Soon after productivity decreases
  - Developers spend a lot more time on bug hunting, debugging, reworking earlier work
  - Developers work longer hours to compensate
  - Developers get frustrated, de-motivated

Without good process from early on (6)

- Eventually productivity approaches zero
  - Project is in serious trouble
- Around this time stakeholders turn to process
  - Try to use process elements to rescue project
  - But much damage already done, very hard to recover
- After a few weeks or months management or customers discover that project makes little or no progress
  - Project is typically cancelled, failed
- **Failed project = 100% overhead**
  - Use process to make project a success
  - Successful project is better than failed project

Good process

- Should be neither rigid nor inefficient
- Time spent on process issues should decrease during project
- Time spent on making and fixing mistakes should also decrease during project
- Results in high overall productivity
- Leads to higher return on investment
  - Higher productivity
  - Shorter time to market
  - Fewer costs and defects
- A lot of success stories across industries

Process vs. Creativity & Morale

- Process can reduce creativity and morale
- But surveys show that companies with more attention to process have a higher morale
  - Highest morale in most process sophisticated companies
- Programmers feel best when they are most productive
  - Good management uses process that makes programmers most productive
  - Use process to avoid mistakes, rework, wasted efforts
- Programmers do not like weak leadership
  - Resulting poor process leads to mistakes, rework, low productivity