

Project Management

- Who needs managers?
- What management functions exist in your team?

Project Management

- Project
 - unique task
 - limited in time
 - example
 - build a house
 - deadline: June 1, 2007
- Role: Project Manager
 - overall planning
 - single point of decision
- Issues
 - resources
 - time
 - staff
 - interface to upper management or customer

Project Management

- “You control a project to the extent that you manage to minimize surprises along the way” -- Tom DeMarco
- Tensions
 - between stakeholders
 - customers vs. developers
 - developers vs. testers
 - experienced staff vs. unexperienced staff
- Feedback
 - training
 - telling people how they are doing
 - without emotion...
- Resolve problems
- Create a good work environment for staff members!

Cost

- What is the biggest cost factor in software development?

Personnel Management

- Personnel
 - biggest cost factor in software development!
 - example:
 - a PC costs \$3000, lasts 2 years
 - with software, support: \$3000 / year
 - a developer earns \geq \$50,000 / year
 - plus 50% benefits, taxes; overhead for office space etc.
- Productivity
 - factor of 10 between good and bad developers
- Essential to get good staff
 - ability hard to see on a résumé
 - experience, experience, experience...
 - reduces training cost
 - sometimes unrealistic demands

Organizational Structure

- Centralized vs. distributed
- Centralized
 - hierarchical
 - quick acting
 - business goals
 - bad news are difficult to communicate
- Distributed
 - cooperative
 - slow to change
 - no single point of failure
 - chaotic
- Management styles
 - authoritarian versus democratic
 - deciding versus consulting

Motivation

- What motivates you to do a good job?
- What motivates your team members to do a good job?

Personnel Motivation

- Different people are motivated by different factors
 - money
 - power
 - technical challenges
 - honor
 - having fun, independence
- Motivation requires
 - feedback
 - realistic goals
 - market awareness
 - sense of community
- Self-fulfilling prophecy
 - unrealistic plan => demotivated team => failure

Process of Project Planning

1. establish constraints
 2. define milestones and deliverables
 3. while not completed
 4. make / review project schedule
 5. do work
 6. review progress
- Software projects are unique
 - development, not production
 - planning is hard!
 - experience
 - revise the plan as needed
 - better a bad plan than no plan!

Project Plan

- Sample structure
 - introduction
 - goals
 - budget
 - organisation
 - staff
 - risk analysis
 - resource requirements
 - hardware, software
 - office space
 - work breakdown
 - project schedule
 - time plan
 - monitoring and reports

Project Schedule

- What is a project schedule?
- Why do we need a project schedule?

Project Schedule

- Goals versus time
 - deadline for milestones
 - consider resource availability
 - staff
 - consider task dependencies
 - A has to be done before B
- Tools
 - spreadsheet
 - charts
- Rules of thumb
 - task/activity: not more than 16 personhours
 - smaller tasks are better!
 - estimate optimal outcome and add 50% buffer
- Depends on process model
 - what is an activity?

The “Mythical Man Month”

- A person month
 - amount of work a person can do in 1 month
 - example:
 - 12 people * 3 months = 36 person months
 - 1 person * 12 months = 12 person months
- “Mythical man month”
 - the idea that people and time are exchangeable
 - truth: there are limits!
 - training overhead
 - communication overhead
 - “adding more people to a late project will make it later” -- Fred Brooks (Brooks' Law)

Milestones

- An endpoint of an activity
 - measurable!
- A report or deliverable is due
- Goal:
 - check if project is progressing as planned
 - review project schedule
 - solve issues if necessary
 - feedback
- Deliverable
 - something the customer wants
 - running program, documentation,...
- Report
 - for internal management

Progress Monitoring

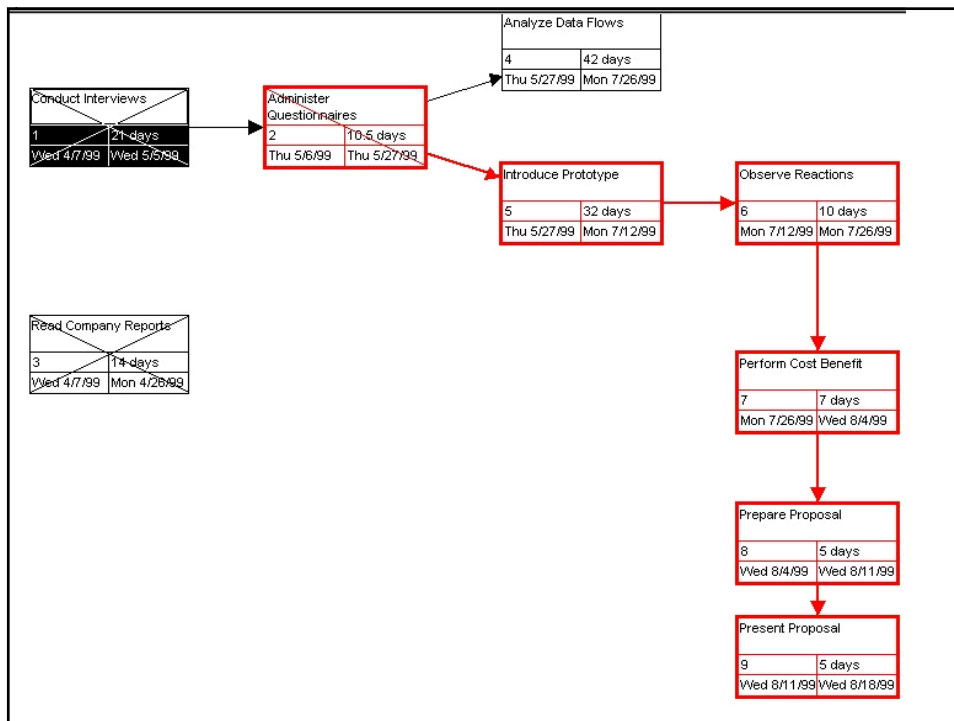
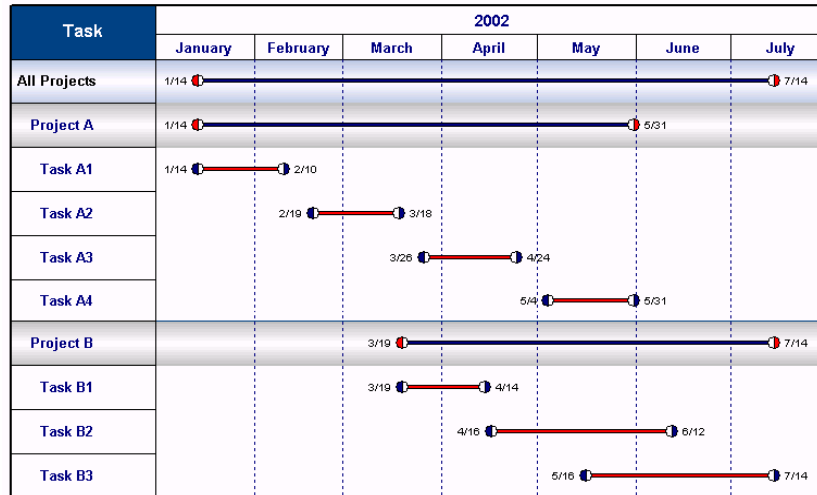
- Goals
- Checking the goals
 - checking without goals: pointless annoyance
 - goals without checking: lack of leadership
- Ensuring that...
 - milestones are met
 - deliverable are delivered
 - goals are met 100% (not 90%...)
 - project schedule does not slip
- Meetings
 - should have an agenda
 - do only if everybody is needed
 - should have minutes that describe the results

Priorities

- Have numbered priorities
 - at least three categories
- How to identify priority of a task?
- Why does one have to prioritize?
- Bad ideas:
 - almost all tasks are highest priority
 - pointless!
 - basing priority on urgency
 - often, urgent tasks do not have high-priority!

Gantt Chart

Widgets-R-Us



Pert Chart

- Program Evaluation and Review Technique
- Shows dependencies between tasks
 - can be combined with Gantt chart
- Critical path
 - those activities that cannot be delayed
- Shows
 - parallelism
 - possible subprojects

Risk Management

- Risks in your project?
 - recovery cost?
 - probability?
 - how to prevent?
 - how to recover?

Risk Management

- Risk = Recovery cost * Probability
- Causes of risks
 - bad planning
 - example: underestimated complexity of task
 - external events
 - example: price for tool goes up
 - bad information
 - example: customer changes mind
- Strategical risk
 - involves the business plan
 - changing markets, competitors, funding
- Operational risk
 - involves daily tasks
 - behind schedule, too complex, wrong architecture

Risk Management

- Develop risk plan
 - lists potential risks
 - impact
 - probability
 - what to do if it happens
 - monitor risks: notice problems early
- Tools
 - checklists
 - project schedule
 - alternative plans
 - measurements

Common Risks

- Bad personnel
 - untrained, unexperienced
 - to do: hire good people, match people to tasks, train people
- Bad schedule
 - unrealistic
 - to do: incremental process, reuse, monitor progress
- Bad requirements
 - wrong features
 - to do: Gui prototype, customer feedback, validation
- Bad tools and external components
 - buggy, undesired behavior
 - to do: read reviews, testing
- Bad architecture
 - too slow, too complex
 - to do: reviews, incremental process, exploratory prototype

Code Review

- Systematic review of source code by others
 - also known as code inspection
 - review team, facilitator, review report
- Checklist
 - project-specific
 - lists common problems to watch out for
- Goals
 - finding bugs
 - finding potential bugs
 - finding violations of other guidelines
 - coding conventions
 - design documents
- Reviewers are not...
 - expected to understand everything
 - if something is unclear: mark as questionable
 - expected to test the code

Code Review Checklist (Short Example)

- Design rules
 - see handout on design smells
 - examples: operation too long, class too long, badly named attribute, too many parameters
- Coding conventions
 - no proper indentation
 - no comments
- Possible bugs
 - parameter may be null
 - infinite loop / infinite recursion
 - possible data type overflow
 - including: array index out of bounds
 - wrong operation called
 - wrong parameter given
 - security problem
 - race condition / deadlock
 - ...

Cost Estimation

- How long is it going to take?
 - Time = cost
- Methods
 - algorithmic
 - experts
 - analogy
 - Parkinson's Law
 - "work fills available time"
 - size determined by cost
- Goal
 - beforehand: planning
 - afterwards: productivity measurement
- Units of productivity
 - Lines of Code
 - Function Points

Productivity Factors

- Complexity of product
 - does it have to be very reliable?
 - e.g., airplane control
 - is it a poorly known domain?
 - experimental product vs. standard product
 - high performance?
 - real-time systems
- Experience and skills of team
 - have they done something similar before?
 - learning curve
- Process model
 - fast feedback?
 - reviews, inspections, testing...

Measuring Productivity

- How would you measure productivity of software developers?

Lines of Code

- Metric for the size of a program
- Standardized LOC:
 - no comments
 - no empty lines
 - no more than one statement per line
- Advantage
 - easy
 - independent of domain
- Disadvantage
 - more code is not better code
 - reuse
 - reusing a library reduces LOC but improves quality
 - easy to manipulate

Function Points

- Requirements-based metric
- Weights:
 - number of inputs 4
 - data items entered by user, example: amount deposited
 - number of outputs 5
 - data items presented: account balance
 - number of inquiries 4
 - user commands: calculate interest
 - number of files 10
 - nontemporary files read or written
 - number of interfaces 7
 - interfaces to other software programs, networks

Function Points

- Advantage
 - independent of implementation
- Disadvantage
 - magic numbers
 - mainly for file processing software
- LOC per FP
 - assembly: 320
 - C: 150
 - C++: 53
 - Java: 53
 - spreadsheets: 6

Cocomo II

- Constructive Cost Model
 - algorithmic
 - uses statistics and empirical data
 - tool-based
 - more complex than LOC and FP
- Steps
 - estimate delivered LOC
 - determine project type
 - determine cost drivers
 - let the tool calculate...
- Output: time estimate
 - in person-months
- Formula: $c * LOC^k$
 - c, k depend on project

Cocomo Drivers

- Scaling Drivers (selection)
 - Precedentedness PREC
 - Team Cohesion TEAM
 - Process Maturity PMAT
- Cost Drivers (selection)
 - Required Reuse RUSE
 - Personnel Capability PERS
 - Platform Experience PEXP
 - Execution Time Constraints TIME
- Early Design Model versus Post-Architecture Model

Metrics

- Measuring software
 - putting a number to it!
 - tool-based
- Examples
 - LOC
 - Function Point
 - Cyclomatic complexity
 - control-flow graph
 - $CC = \text{edges} - \text{nodes} + 2$
 - related to number of branches
 - Fog metric
 - length of words in natural language
 - Fan-in/fan-out
 - example of a design metric
- Nail plant...

Delphi Technique

- Expert estimation
- Consensus-based
- Steps
 - coordinator elected
 - coordinator develops questions
 - each expert answers questions anonymously
 - coordinator collects answers, sorts them, and makes new questionnaire
 - experts comment on issues again
 - repeat until there is desired consensus
- For example
 - for agreeing on a design
 - coordinator lists choices in questionnaire, asks for pros and cons
 - for cost estimation

Measurement Example

- Red Hat Linux 7.1
 - 30 Million LOC
 - cost: 1 billion dollars
 - 8,000 person years
 - according to Cocomo II
 - Linux Kernel: 2.3 million LOC in C
 - Mozilla: 1.3 million LOC in C++, 0.8 million in C
 - Xfree86: 1.8 million in C
- MS Windows XP: 20 million
- MS Windows 3.1: 3 million
- Space Shuttle On-Board: .4 million