CS 307: Software Engineering

Lecture 11: Project Management

Prof. Jeff Turkstra
Announcements

- Remember your weekly reports
  - Now due Mondays
  - Team report should be sent to project coordinator AND jeff@purdue.edu
Lecture 11

- Project management
- Risk analysis and management
- Classic mistakes
- Lowering risk
Project management

- Encompasses all activities needed to plan and execute a project
  - Deciding what needs to be done
  - Estimating costs
  - Ensuring that there are suitable people to undertake the project
  - Defining responsibilities
  - Scheduling
  - Making arrangements for the work
- Directing
- Being a technical leader
- Reviewing and approving decisions made by others
- Building morale and supporting staff
- Monitoring and controlling
- Coordinating the work with managers of other projects
- Reporting
- Continually striving to improve process
Software Process Models

- General approaches for organizing a project into activities
  - Help the project manager and team decide:
    - What work should be done
    - The sequence
  - Aids to thinking, not rigid prescriptions
  - Each project has its own unique plan
Models

- Code and Fix
- Stagewise and Waterfall
- Prototyping
- Evolutionary
- Spiral
- Agile/Scrum

See Lecture 3
Re-engineering

- Periodically project managers should set aside some time to re-engineer part or all of the system
  - Clean up code to make it more readable
  - Completely rewrite a layer
  - **Refactor** part of the design
- General goal is to increase maintainability
Refactoring

- Process of restructuring existing code without changing external behavior
- Improves nonfunctional attributes of the software
  - Software quality
  - Improve readability
  - Reduce complexity
  - Improve extensibility
  - etc
Cost estimation

- Estimating how much software engineering time will be required to do some work
  - Elapsed time: start date to end date of the task or project
  - Development effort: amount of labor used in person-months or person-days
    - Development time estimate → money
Effective cost estimation principles

1. Divide and conquer
   - Divide the project into subsystems
   - Divide each subsystem into development activities
   - Estimate time for each activity
   - Sum results
2. Include all activities when making estimates

- Prototyping
- Design
- Inspecting
- Testing
- Debugging
- Writing documentation
- Deployment
3. Leverage past experience and knowledge of current project

- Look for similarities with past work
  - Similar amount of time and effort
- Base estimates on personal judgment of experts
- Use algorithmic models developed by software industry
  - Formulas to estimate cost
Algorithmic models

- Systematically estimate development effort
  - Number of use cases
  - Number of distinct requirements
  - Number of classes in domain model
  - Number of widgets in UI
  - Estimated number of lines of code (LoC)

- Generally less reliable and accurate than experience
Effective cost estimation

4. Account for **differences** when extrapolating from other projects

- Different software developers
- Different development processes and maturity levels
- Different customers and users
- Different schedules
- Different technology
- Different requirements complexity
- Different domains
- Different requirements stability
5. Anticipate worst case and plan for contingencies

- Develop and identify critical use cases first
- Estimate
  - Optimistic – everything goes perfectly
  - Likely – allow for “typical” things going wrong
  - Pessimistic – everything goes wrong
6. Combine independent estimates

- Use several techniques and compare results
- Delphi technique
  - Several individuals make cost estimates in private
  - Share their estimates, analyze discrepancies
  - Adjust estimates until consensus reached
Scrum poker

- Consensus-based technique for estimating effort
- Variation of Delphi method
- Group members make estimates by playing numbered cards face-down on the table
  - Don’t speak
- Cards are revealed and estimates are discussed
Avoids “anchoring” - first number sets a precedent for subsequent estimates
- Similar to “priming” in psychology

Some organizations use standard playing cards (Ace, 2, 3, ..., King)
- King: “too big or complicated to estimate”

Apps
Fibonacci

- Recall the Fibonacci sequence: 0 1 2 3 5 8 13 ....
- Often times used instead of linear progression
Effective cost estimation

7. Revise and refine estimates as work progresses
   - As you add detail
   - As the requirements change
   - As the risk management process uncovers problems
Teams

- **Software engineering is a human process**
  - Selecting appropriate people and assigning roles and responsibilities is an important project management skill
  - Team organization can vary

![Diagram showing different team organizations]

- a) Egoless
- b) Chief programmer
- c) Strict hierarchy
Egoless

- “Self-organizing team”
- Everybody is equal
  - Work together toward a common goal
- Decisions made by consensus
- “Most suited to difficult projects with many technical challenges”
  - Open to debate
- Typical Agile team organization
Strict hierarchy

- Each individual reports to a manager and is responsible for performing the tasks delegated by that manager
- Suitable for large projects with a strict schedule
  - Everybody is well-trained and has a well-defined role
- Often couples with waterfall model
Chief programmer

- Middle road between egoless and hierarchical
- Chief programmer leads and guides project
- Consults with and relies on individual specialists
  - Linux kernel
Team size

- Doubling the size of the team never halves the development time
  - The Mythical Man-Month: Essays on Software Engineering
- Subsystems and teams should be sized such that total amount of required knowledge and exchange of information is reduced
- Number of people on a team may not be constant
Behind schedule? Adding more people will make it worse.
Skills

- Architect
- Project Manager
- Configuration management and build specialist
- UI specialist
- Hardware and third-party software specialist
- Documentation specialist
- Tester
Scheduling and tracking

- **Scheduling** is the process of deciding:
  - Sequence of activities to be performed
  - When they should start and complete

- **Tracking** is the process of determining how well you are sticking to the cost estimate and schedule
PERT chart

- **Shows the sequence in which tasks must be completed**
  - Each node of a PERT chart contains elapsed time and effort estimates
  - **Critical path** indicates minimum time to project completion
Gantt chart

- Graphically illustrates start and end dates for each software engineering task
  - One axis shows time
  - Other shows activities to be performed
  - Black bars are top-level tasks
  - White bars are subtasks
  - Diamonds are milestones
    - Important deadline dates
Risk

- What is risk analysis and management?
- Classic mistakes
- Lowering risk
Dictionary definition

“to expose to the chance of injury or loss”
- The American College Dictionary
Types of risk

1. Speculative – implies the possibility of either a profit or a loss
   - Playing the stock market

2. Static – only have losses
   - Not wearing a seat belt
What is risk all about

- Risk is only concerned with future events. Past and present events are already known and can no longer be considered as risks.
- Risk involves any change which can have a negative effect on the project outcome.
- Risk involves choice and all the uncertainty that goes with making choices.
Uncertainty

- **Structure uncertainty** – missing information needed to fully describe the system
- **Measurement uncertainty** – missing information on how to measure the system variables
- **Event outcome uncertainty** – missing information on the possible outcomes of actions taken which could affect the system
Choice

- There must be a choice for there to be a risk
- Risk involves making decisions which can affect the size of the loss or gain
How do risks come about?

1. Acts of God
2. We place ourselves at risk
3. We place others at risk
4. Other people place us at risk
For an event to be considered a risk...

- There must be a possible loss associated with it
- There must be uncertainty or chance involved
- There must be choices
Why don’t we do risk analysis?

- We don’t want to admit that real risks exist
- We like to, and tend to work, on the easy parts first
- It costs money, time, and requires trained people up front to do good risk analysis
Rowe’s definition of risk

- Risk is the potential for the realization of unwanted, negative consequences of an event

“Anatomy of Risk”, 1988
Questions

1. What are the risks?
2. What is the probability of the loss they might cause?
3. What will the losses cost?
4. What is the worst case scenario?
5. What alternatives do we have?
6. How can the losses be reduced or avoided?
7. What have we forgotten?
Risk analysis

1. Identification
2. Estimation
3. Risk evaluation
Risk identification

- Attempts to reduce the structural uncertainty – missing information
- Tries to identify the system variables
- Tries to determine the potential threats to the system’s success

- What is the risk?
- Can we categorize it?
Risk estimation

- Attempts to reduce the measurement uncertainty
- Tries to assign meaningful values to the system variables
- Tries to determine the effects of the identified threats on the system
- Tries to assign meaningful values to the losses associated with the identified threats
Estimation

- How big is the risk?
- What will it affect?
- What is the probability of it occurring?
Risk evaluation

- Attempts to reduce the event outcome uncertainty
- Tries to assign priorities to the risks
- Tries to generate ways to reduce, avoid, or eliminate the risks
Evaluation

- What are the limits of acceptable risk?
- Which risks are acceptable?
- How badly can the unacceptable ones affect us?
- What can be done to reduce or avoid the unacceptable risks?
Risk management

1. Planning
2. Control
3. Monitoring
Risk planning

- Chooses a course of action acknowledging all of the known risks involved
- Develops a plan for carrying out the selected course of action
- Develops a risk aversion plan for the selected course of action
Risk control

- Implements the risk aversion plan
- Updates and supports the plan as needed with:
  - Needed resource changes
  - Needed schedule changes
  - Needed budget changes
Risk monitoring

- Watches for risk changes
- Refines the plans as needed to account for new or changing risks
Comments

- Risks should not limit or dictate the way we manage projects
- Risk assessment should help set up reasonable boundary conditions for the management process
When should it be done?

- Only when the risks are expected to exceed the cost of doing the assessment
- In parallel with other planning activities

“The worst thing that risk analysis and risk management can do is to plan and contain a failure that doesn’t occur.” – Charette
Risk is based on probabilities. It is therefore possible that the harmful event might not ever happen.

Risk management is supposed to help reduce the project’s total life cycle costs. If it doesn’t help, don’t use it.
Build a risk table

- For each risk $i$, add an entry to each of the three columns
  1. Scenario $S_i$ – what can go wrong
  2. Likelihood $L_i$ – probability of it occurring
  3. Damage $D_i$ – how will it affect us
Environments where risks exist

- Technical
- Development
- Operational
- Maintenance
- Political
- Social
- Legal
- Economic
- Others
Sources of risk information

- Traditional or folk lore
  - Perceptions
  - Software mythology
  - Stereotypical information

- Analogies to well-known cases
  - Historical data
  - Risk checklists

- Common sense assessments
- Results of experiments or tests
  - Prototypes
  - Buying information
- Inadvertent exposure
  - Product used in ways not intended
  - Product used in ways not imagined
What can lead to a risk?

1. Lack of information
2. Lack of control
3. Lack of time
4. Lack of resources
Develop your own scale for risk

<table>
<thead>
<tr>
<th>Classification</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not likely</td>
<td>0.01 – 0.09</td>
</tr>
<tr>
<td>Low</td>
<td>0.10 – 0.24</td>
</tr>
<tr>
<td>Moderate</td>
<td>0.25 – 0.49</td>
</tr>
<tr>
<td>High</td>
<td>0.50 – 0.64</td>
</tr>
<tr>
<td>Very High</td>
<td>0.65 – 0.89</td>
</tr>
<tr>
<td>Almost certain</td>
<td>0.90 – 0.99</td>
</tr>
</tbody>
</table>
Cautions

- Putting things in classes does not improve the accuracy of the estimate
- Risks are generally not independent events
  - Change requests can affect schedule, budget, resources, and even product quality
Describing risk damage

- Need to describe its character: what is likely to be damaged?
  - Physical – loss of life, equipment, data, etc
  - Economic – loss of market share, return on investment, etc
  - Something else – schedule, resources, etc
- Likely extent of the damage
  - Severity – amount measured in pain, time cost
  - Distribution of damage – who or what is affected and by how much?
Timing of the damage

- Immediate – personnel leave
- Spread over time – lack of trained software engineers slowly causes the schedule to flip
- Future impact – not doing risk analysis, no test plans
Precision vs Accuracy

- **Precision** deals with the exactness by which we describe a variable.
- **Accuracy** deals with freedom from error.

The temperature is precisely 74.456789 degrees with an accuracy of ±3°F. “... the last digit is correct about 10% of the time.”
Estimation biases

- People tend to believe their own estimates are more accurate than they really are.
  - Bias can be too high or too low

- Lack of understanding of probability and statistics can lead to really bad estimates
  - Picking wrong probability distribution
Some people have a tendency to not revise estimates. Watch out for the...

- “I’ve made up my mind, don’t confuse me with the facts” syndrome

Tend to be insensitive to sample sizes

- We have tested it successfully three times with three different users, must be ready to ship!
Bias of association: if risk is low then the damage must also be low!

- False coupling is an easy mistake to make
- Tend to make predictions based on our intuitive impression of the distribution
- Do you wear a seat belt when you drive?
Must not forget that even if the probability of component failures is small, the probability of system failure can be high given a large number of components.

\[ \text{P}(n, f) = 1 - (1 - f)^n \]
## Components Probability of Failure

<table>
<thead>
<tr>
<th>n</th>
<th>f</th>
<th>P(n, f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.100</td>
<td>0.65</td>
</tr>
<tr>
<td>100</td>
<td>0.010</td>
<td>0.63</td>
</tr>
<tr>
<td>1000</td>
<td>0.001</td>
<td>0.63</td>
</tr>
</tbody>
</table>
Classic mistakes

- Material from Chapter 3 of Rapid Development by Steve McConnell
  http://www.stevemcconnell.com/rdenum.htm

1. People-related
2. Process-related
3. Product-related
4. Technology-related
People-related

1. Undermined motivation
2. Weak personnel
3. Uncontrolled problem employees
4. Heroics
5. Adding people to a late project

Brooks Law

6. Noisy, crowded offices
7. Friction between developers and customers
People-related

8. Unrealistic expectations
9. Lack of effective project sponsorship
10. Lack of stakeholder buy-in
11. Lack of user input
12. Politics placed over substance
13. Wishful thinking
Process-related

14. Overly optimistic schedules
15. Insufficient risk management
16. Contractor failure
17. Insufficient planning
18. Abandonment of planning under pressure
19. Wasted time during the fuzzy front end
20. Shortchanged upstream activities
Process-related

21. Inadequate design
22. Shortchanged quality assurance
23. Insufficient management controls
24. Premature or overly frequent convergence
25. Omitting necessary tasks from estimates
26. Planning to catch up later
27. Code-like-hell programming
Product-related

28. Requirements gold-plating
29. Feature creep
30. Developer gold-plating
31. Push-me, pull-me negotiation
32. Research-oriented development
Technology-related

33. Silver-bullet syndrome
34. Overestimated savings from new tools or methods
35. Switching tools in the middle of a project
36. Lack of automated source-code control
Lowering the risks

- Let us examine Barry Boehm’s list of top ten risks and some possible ways for reducing these risks
Risk 1: Personnel shortfalls

- Staff with the best people you can afford
- Provide training
- Improve your work environment
- Spend time developing teams
- Plan ahead
  - Pre-schedule key personnel
  - Anticipate job turnover
Risk 2: Unrealistic schedules and budgets

- **Improve your current estimation process**
  - Be sure you know and can estimate all of your costs
  - Be sure that the people who approve the schedule, budget, and resources are the same people that have to do the work
Use an incremental development model (e.g., scrum), so that the needs of early and late users can be met

- Allows schedule and budget to reflect current needs

Buy components

- Fight NIH (Not Invented Here) syndrome

Develop a software reuse program

Run a requirements scrubbing review with all pertinent parties in attendance
Risk 3: Developing the wrong software functions

- Reevaluate your market forecasting
- Utilize user surveys
- Build and evaluate prototypes
- Develop and circulate early versions of your user manuals
- Get your users involved in the testing and reviewing process
Risk 4: Developing the wrong user interface

- About 40% of many current development efforts deal with the user interface
  - Build prototypes and actively involve real users in the evaluation process
  - Video tape and study how users really use your systems
  - Look for ways to reduce the user workload
Risk 5: Gold plating

- Design to cost. Customer pays with time and money
- Run a requirements scrubbing review with all pertinent parties in attendance
- Build and evaluate prototypes before adding unnecessary functionality
- Do cost-benefit analysis
- Prioritize the requirements and then follow an incremental development plan
- Set up a frequent build policy
- Inspect the product frequently
Risk 6: Continuing stream of requirements changes

- **Set a change threshold**
  - Set up a change control board to arbitrate requests
  - Make sure all needed changes can get through
  - Make it painful for the rest
    - Requester provides money, accepts schedule delays, or provides extra resources (people, equipment, etc)
Risk 7: Shortfalls in externally furnished components

- Benchmark components
- Do compatibility analysis
- Set up an inspection program
- Second source as much as possible
- Check the credentials of your suppliers
- Maintain a contingency plan
Risk 8: Shortfalls in externally performed tasks

- Check the credentials of your suppliers
- Fund competitive designs or prototypes
- Run pre-award audits
  - Are they at least at a process level equivalent to level 3 of the SEI software process model
- Develop a win-win strategy and relationship
Risk 9: Real time performance shortfalls

- Use simulators to check out the requirements and the design
- Develop system models
- Build and evaluate prototypes
- Extensively benchmark the system
- Instrument the system so that it can be tuned
Risk 10: Straining computer science capabilities

- Run technical analysis studies of the project and the project team
- Build prototypes to better understand the problems
- Check the credentials of your technical consultants
- Start and support a technical training program for your project personnel
- Support your technical supplier (Universities)
Questions?