CS 307: Software Engineering

Lecture 14: Project Management

Prof. Jeff Turkstra
Announcements

- **Sprint 1 planning document due Monday, February 13**
  - Also when Sprint 1 begins
  - Might be worth reviewing Lecture 3, slides 31 onward

- **Remember your weekly reports**
  - Team report should be sent to project coordinator AND jeff@cs.purdue.edu
Lecture 14

- Project management
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
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

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
Questions?