Software Project Planning

CS470

What is Planning?

• Phases of a project can be mostly predicted
• Planning is the process of estimating the time and resources required to complete each phase
• Helpful in overall management of a project, adherence to deadlines, etc.
• Plan depends on software development methodology
Common Development Methodologies

• Classic: “Waterfall Model”
• Prototyping
• Agile Development Techniques
  – Extreme Programming
  – SCRUM
• Incremental Development

• Pick the right methodology for your project

Sequential or Iterative approach?

• You might choose a sequential approach:
  – Requirements are stable
  – Design is well understood
  – Development team familiar with the application
  – Low risk
  – Long term predictability
  – Cost of changing requirements, design and code downstream is high
Sequential or Iterative?

- You might choose an iterative approach:
  - The requirements are not well understood or you expect them to be unstable
  - The design is complex or unknown
  - The development team is unfamiliar with the application area
  - The project contains a lot of risk
  - Long term predictability is not important
  - The cost of changing requirements, design, and code downstream is low

Documentation

- This class forces you to do some documentation along the way
  - Initial document, presentations, reviews
- Benefits of Documentation:
  - Forces you to think the problem all the way through
    - If you can’t write it down, you probably don’t understand it
    - Defines why you are doing it, what you will be doing, and how it will be done
  - Vehicle for communications
  - Defines commitments and reduces misunderstandings and “feature creep”
How much time?

- A well-run project devotes 10-20% of its effort and 20-30% of its schedule to requirements, architecture, and up-front planning.
- If requirements are unstable you’ll probably need to resolve requirements issues yourself:
  - Allow time for defining requirements well enough that their volatility will have a minimal impact on construction.
  - Treat requirements or architecture work as its own project if unknown; estimate remaining project.

Planning – Estimating Effort

- Planning is estimating the effort for each activity.
- Example for Waterfall Model (would be different for other methodologies):

![Pie chart showing distribution of effort across phases: Requirements (15%), Design (20%), Implementation (35%), Testing (15%), Writeup (10%), Presentation (5%)](chart.png)
Plug in Actual Hours

- E.g. if have 150 hours over the entire semester to work on the project:
  - $150 \times 0.05 = 7.5$ hours for presentation
  - $150 \times 0.35 = 52.5$ hours for development
- Etc.
- Can begin to see how realistic a schedule may be and what work is actually involved
- This is just your best guess at this point

Project Scheduling Guidelines

- Use previous experience in estimate
- Build in contingency time
- Software engineers are notoriously optimistic
- Use bottom-up scheduling for better accuracy
- Take deadlines and other commitments into account and limit project activities accordingly or make tradeoffs
- Set clear milestones with specific timelines
Gantt Chart

- Can create on paper, in Excel, Project, Web applets
- Graphical view of schedule with dependencies
Project Skills

• Success of a project is not just good project management, but effective documentation, communications, publishing, writing, and presentation skills
• Simplest Method – Project Notebook
  – Can be hardback or electronic
  – Keep references, new ideas, software designs, reports, timelines, etc.
  – Where project management software is useful: these things are easily categorized, dates, etc.

Project Communications

• Regular meetings with customer
  – Keep a weekly status report of what issues arose, what was accomplished, etc.
  – Makes it much easier to write the final report and maintain a history of the project
  – Useful tool for communications w/supervisor
  – Develop agenda for meetings
    • Report of prior material
    • Current progress
    • Plans
    • Issues
    • Other business
As Your Project Progresses

• Report problems early
  – Flag problems or potential slips early, don’t try to hide them and hope it will work out later

• Articulate unknowns
  – Margin of error and major unknowns should be declared

• Ready, Fire, Aim
  – Everyone is always behind, don’t try to make up by coding before defining the work, deliverables, etc.

This Class

• In this class we’ll focus on coding and debugging techniques that are not normally covered in other classes

• Practice with technical writing, giving presentations, evaluating code
  – Exercises built around your project so there is not too much “extra” work

• Additional material on society, ethics, IP
Which language?

• Many students often use a new programming language for their project
• Many languages are better suited to particular projects
  – E.g. PHP for Web; better than C++

• Use an appropriate language but it can be a big risk to learn a new programming language during the short span of the semester
  – Usually better to stick with the language you know best, if it applies. If it doesn’t consider changing requirements if possible

What should be in my design document?

• The document is both a requirements and design document
  – As much detail as possible to nail down what your project will be and how you will know when you’re done
  – But not a giant comprehensive document covering all the little details like what you may have produced in some classes or jobs
• Major Sections
  – Overview / Hypothesis / Background
  – Requirements
    • English or formal, mock-ups
  – Design
    • English or more formal, architecture, decomposition
  – Planning
    • Schedule with milestones and deliverables
  – References
Proposal Guidelines

• How long?
  – Depends; probably 5-10 pages, but be succinct

• Writing style
  – Formal document, okay to use “I”
    • Instead of: “You’ll probably do something like clicking a button or pressing enter, to trigger the login screen”
    • More formal: “Click the submit button to begin the login process”
  – Number each section, e.g. 2. Requirements, 2.1 Functional specifications, 2.2 Non-Functional specifications, etc.

• Spell check and proofread!