Project proposal

- **due Friday March 10**
- **1-2 pages**
  - email to bwk, Subject: Project proposal
  - content: URL or Word doc or text file
- **name / title**
- **people**
  - names, email addresses, primary role(s)
  - list one person as project manager, acts as contact
- **project vision / goal**
  - 1-2 sentences (or a short paragraph) on what it is
- **feature list**
  - what language(s) are we doing, major pieces (in order if possible), how they fit together
- **major design choices**
  - web vs. standalone, languages, tools, environment

- **these are not binding commitments but should be your best guess based on thought and discussion among team members**
- **I'm looking for evidence that you have spent some time thinking about it**
  - don't just throw together a page at the last minute because it's due

Process: organizing what to do

- **use an orderly process or it won't work**
- **this is NOT a process:**
  - talk about the software at dinner
  - hack some code together
  - test it a bit
  - do some debugging
  - fix the obvious bugs
  - repeat from the top until semester ends

- **classic "waterfall" model**

specification
requirements
architectural design
detailed design
coding
integration
testing
delivery

- **this is too much overkill for 333**
- **however, some process is essential …**

"Staged delivery" model

- **conceptual design**
  - roughly, what are we doing?

- **requirements definition ("what")**
  - gather ideas about what it should do
  - potential users, competitive analysis, prototyping
  - specify with written docs, scenarios, prototypes
  - this should generally not change once you're started
    it's too hard to hit a moving target
- **architecture / design ("how")**
  - map out structure with design diagrams, prototypes
  - explore options & alternatives on paper
  - partition into major subsystems
  - specify interactions between subsystems
    interfaces, information flow, control flow
  - decide pervasive design issues
    language, environment, storage, error handling
  - make versus buy decisions taken here
    [aside on what you can use from elsewhere]
- **implementation ("what by when")**
  - deliver in stages, each of which is complete, working
    what will be in each release?
  - test as you go

Deciding what to do

- **formal processes are nice, but you still have to do a lot of thinking and exploring informally**
- **do this early, so you have time to let ideas gell**
- **make big decisions first, to narrow the range of uncertainty later**
  - Web based or standalone, Unix or Windows, what target language?
    build the GUI in Java or VB or Tcl/Tk?
    what kinds of windows will be visible?
    what do individual screens and menus look like?

- **McConnell: "large grain" decisions before "small grain"**
- **think through decisions at each stage so you know enough to make decisions at next stage**

- **this is more iterative than this might imply**
  - don't make binding decisions until you are all fairly comfortable with them

Other ways to think about it

- **"elevator pitch"**
  - what would you say if you were alone in an elevator with Bill Gates for 60 seconds?
  - attention-grabbing description
  - a paragraph without big words but good buzzwords
- **5-7 slides for a 5-10 minute talk**
  - what would be the titles and 2-3 points on each slide?
- **1 page advertisement**
  - what would be the main selling points?
- **talk outline**
  - how would you organize a talk to give at the end of the semester?
- **business plan**
  - how would you pitch it to an angel or venture capitalist?
    - what does it do for who?
    - who would want it?
    - what's the competition?
    - what are the stages of evolution or major releases?

Things to do from the beginning

- **think about schedule**
- **plan for a sequence of stages**
  - do not build something that requires a "big bang" where nothing works until everything works
  - always be able to declare success and walk away
- **simplify**
  - do not take on too big a job
  - do not try to do it all at the beginning
- **use source code control for everything**
- **leave room for "overhead" activities**
  - testing: you have to have a Quality Assurance plan
    - build quality in from the beginning
  - documentation: you have to deliver written material
  - deliverables: you have to package your system for delivery
  - changing your mind: some decisions will be reversed and some work will have to be redone
  - disaster: lost files, broken hardware, overloaded systems are all inevitable
  - sickness: you will lose time for all kinds of unavoidable reasons
  - health: there is more to life than this project!
- **keep records, report where the time goes**

Roles

- **not all of these need be explicit, but projects have to do these tasks**
- **project manager**
  - orchestrates code, testing, documentation, etc.
  - in charge, but not necessarily the technical lead
- **architect**
  - how do the pieces fit together
  - makes it look like the product of a single mind
- **user interface designer**
  - makes it look like the product of a single mind
- **developer**
  - you all have to do some significant part of this
- **quality assurance / testing**
  - responsible for making sure it always works
- **toolsmith**
  - support, builds, export packaging
- **documentor**
  - manual, internals doc, web page, blurbs, presentation
- **risk officer (McConnell)**
  - what are the risks? what could go wrong?
  - not the project manager!!