# 333 Project

- **a simulation of reality**
  - building a substantial system
  - in groups of 3 to 5 people
- **"three-tier" system for any application you like**
- **3 major pieces**
  - graphical user interface ("presentation layer")
  - processing in the middle ("business logic")
  - storage / data management
- **examples: many web-based services**
  - Amazon, Ebay, other web stores
  - news, information services, bots, mashups
  - email, chat, search, code tools, maps, ...
  - cellphone systems are often like this too
- **your project**
  - make something of roughly this structure
  - but smaller, simpler, defined by your interests

# Getting started

- **right now, if not sooner**
  - think about potential projects
    talk to TA's, bwk; look at previous ones; look around you;
    check out the external project ideas page
  - form a group
- **by Fri Mar 6 short meeting with bwk (earlier is fine)**
  - to be sure your project idea is generally ok
  - should have one pretty firm consensus idea
- **Fri Mar 13: design document draft (before break)**
  - ~3 pages of text, pictures, etc.
    (a template will be posted)
  - overview
    project name / title, short paragraph on what it is
    list one person as project manager, acts as contact
  - components & interfaces
    major pieces, how they fit together
    major design choices
    web vs. standalone, languages, tools, environment, …
  - milestones: clearly defined pieces either done or not
  - risks
- **not frozen, but should be your best guess based on significant thought and discussion**
  - we are happy to talk about your ideas
- **don't throw it together at the last minute**
  - all components of the project are graded

# 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 the semester ends

- **classic "waterfall" model: a real process**

  specification
  requirements
  architectural design
  detailed design
  coding
  integration
  testing
  delivery

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

# Informal process

- **conceptual design**
  - roughly, what are we doing?
  - blackboard sketches, scenarios, screens
- **requirements definition ("what")**
  - precise ideas about what it should do
  - explore options & alternatives on paper
  - specify more carefully with written docs
  - this should not change a lot once you're started
    it's hard to hit a moving target
- **architecture / design ("how")**
  - map out structure and appearance with diagrams, prototypes
  - partition into major subsystems or components
  - specify interactions and interfaces between components
  - decide pervasive design issues
    languages, environment, database, …
  - make versus buy decisions
    [aside on what you can use from elsewhere]
  - experiments to resolve connectivity, access, etc.
- **implementation ("what by when")**
  - make prototype
  - deliver in stages, each that does something and works
    what will be in each release?
  - test as you go: if (easy to break) lower grade

## Make versus buy

- **you can use components and code from elsewhere**
  - copy or adapt open source
- **design has to be your own**
- **so does selection and assembly of components**
- **so does the bulk of the work**
- **it's fine to build on what others have done**
  - identify what you have used, where it came from

## Interfaces

- **the boundary between two parts of a program**
- **a contract between the two parts**
- **what are the inputs?**
- **what are the outputs?**
- **what is the transformation?**
- **who manages resources?**
  - especially memory, shared state
- **hide design & implementation decisions behind interfaces, so they can be changed later without affecting the rest of the program**
  - data representations and formats
  - what database system is being used
  - specific algorithms
  - visual appearance
- **"I wish we had done interfaces better" is one of the most common comments**
  - less often: "We thought hard about the interfaces so it was easy to change things without breaking anything."

# 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 gel**
- **make big decisions first, to narrow the range of uncertainty later**
  - "large grain" decisions before "small grain" (McConnell)
  - web/standalone/phone? Unix/Windows/Mac/iPhone?
    framework (GWT, Django, Rails) or roll your own?
    GUI in Java or .NET or IB or ...?
    what kinds of windows will be visible?
    what do individual screens and menus look like?
  - Java or PHP or Perl or C# or ...?
    mix & match, or all the same?
- **think through decisions at each stage so you know enough to make decisions at next stage**
- **but this is still very iterative**
  - don't make binding decisions until you are all fairly comfortable with them
  - do simple experiments to test what works or doesn't
  - what do users see and do?
    scenarios are very helpful (storyboards, "use cases")
    sketches of screen shots
    diagrams of how information, commands, etc., will flow
  - what data is stored and retrieved
    how is it organized

# 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?
  - what would your web page look like?
- **talk/demo outline**
  - how would you organize a talk and demo to give at the end of the semester?
  - what would you want working for the demo?
- **business plan**
  - how would you pitch it to an angel or venture capitalist or Google?
    what does it do for who?
    who would want it?
    what's the competition?
    what are the stages of evolution or major releases?
- **job talk / interview**
  - what did we do that's really cool?

## Things to keep in mind

- **project management**
  - everyone has to pull together
  - someone has to be in charge
- **architecture**
  - how do the pieces fit together?
  - make it work like the product of a single mind
  - but with multiple developers
    "Good interfaces make good neighbors"?
- **user interface**
  - what does it look like?
  - make it look like the product of a single mind
- **development**
  - everyone has to do a significant part of the coding
- **quality assurance / testing**
  - make sure it <u>always</u> works
    should always be able to compile and run it
    fix bugs before adding features
- **documentation**
  - internals doc, web page, advertising, presentation,
  - final report
- **risks**
  - what could go wrong?
  - what are you dependent on that might not work out?

## Things to do from the beginning

- **think about schedule**
  - keep a timeline of what you intend and what you did
- **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
    (but do not try to do it all at the end -- that's disaster)
- **use source code control for everything**
  - SVN or equivalent is mandatory
- **leave lots of room for "overhead" activities**
  - testing: build quality in from the beginning
  - documentation: you have to provide 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 unavoidable reasons
  - health: there is more to life than this project!

# 2009 Project Schedule

```
       February
 S  M Tu  W Th  F  S
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21   <- you are here
22 23 24 25 26 27 28

       March
 S  M Tu  W Th  F  S
 1  2  3  4  5  6  7   initial talk with bwk
 8  9 10 11 12 13 14   design doc due before break
15 16 17 18 19 20 21   spring break - enjoy
22 23 24 25 26 27 28   TA meetings begin this week
29 30 31

       April
 S  M Tu  W Th  F  S
          1  2  3  4
 5  6  7  8  9 10 11   prototype
12 13 14 15 16 17 18
19 20 21 22 23 24 25   alpha test
26 27 28 29 30

       May
 S  M Tu  W Th  F  S
                1  2   beta test
 3  4  5  6  7  8  9   demo days
10 11 12 13 14 15 16   Dean's date: all done
17 18 19 20 21 22 23
```

# Some mechanics

- **groups of 3 to 5**
  - find your own partners
  - use the newsgroup for match-making
- **TA's will be your first-level managers**
  - more mentoring and monitoring than managing
  - it's your project, not the TA's
- **weekly meeting of your whole group with your manager each week after spring break**
  - everyone must attend all of these
- **be prepared:**
  - what did we accomplish
  - what didn't we get done
  - what do we plan to do next
- **these meetings are a graded component**
- **this is my attempt to make sure that things don't get left to the last week of the semester**