333 Project

• a simulation of reality:
  - building a substantial system
  - in groups of 3 or 4 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
  - email, chat, ...

• your project:
  - make something of roughly this structure
  - but smaller, simpler, defined by your interests

Getting started

• right now
  - think about potential projects
  - talk to TA’s, bwk; look at previous ones; look around you
  - form a group of 3 or 4

• by Wed Mar 9 meet with bwk (earlier is better)
  - to be sure it’s generally ok

• Fri Mar 11: design document draft (before break)
  - ~3 pages of text, pictures, etc.
  - overview
    - project name / title, short paragraph on what it is
    - names, email addresses, primary role(s)
  - components & interfaces
    - major pieces, how they fit together
    - major design choices
    - web vs. standalone, languages, tools, environment, ...
  - schedule
  - 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
Project proposal

• discussion by Wed Mar 9 (earlier is good)
  - discuss project with bwk to be sure it's generally ok
• design document draft Fri Mar 11 (before break)
  - ~3 pages of text, pictures, etc.

• overview
  - a short paragraph on what it is
  - project name / title
  - people names, email addresses, primary role(s)
  - 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, ...
• schedule
• risks

• these are not binding commitments but should be
  your best guess based on significant thought and
  discussion among team members
  - 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 ...
COS 333 informal process

• conceptual design
  - roughly, what are we doing?
  - blackboard sketches

• requirements definition ("what")
  - gather ideas about what it should do
  - specify with written docs, prototypes, scenarios
  - potential users, competitive analysis, prototyping
  - this should not change much 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: 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
- critical thing is to hide design 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 (if any)
  - specific algorithms
- "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 based or standalone? Unix or Windows or Mac?
    - build the GUI in Java or VB or .NET or ...
    - what kinds of windows will be visible?
    - what do individual screens and menus look like?
  - Java or PHP or Perl or C# or ...
- 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
  - what do users see and do?
    - scenarios are very helpful (storyboards)
    - sketches of screen shots
    - diagrams of how information, commands, etc., will flow
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?
    - 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 always 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**
  - CVS 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!

---

**2005 Schedule**

<table>
<thead>
<tr>
<th>February</th>
<th>March</th>
</tr>
</thead>
<tbody>
<tr>
<td>S  M  Tu  W  Th  F  S</td>
<td>1  2  3  4  5</td>
</tr>
<tr>
<td>6  7  8  9 10 11 12</td>
<td>6  7  8  9 10 11 12</td>
</tr>
<tr>
<td>13  14  15  16  17  18  19</td>
<td>13  14  15  16  17  18  19</td>
</tr>
<tr>
<td>&lt;- you are here</td>
<td>spring break - enjoy</td>
</tr>
<tr>
<td>20  21  22  23  24  25  26</td>
<td>20  21  22  23  24  25  26</td>
</tr>
<tr>
<td>27  28</td>
<td>27  28  29  30  31</td>
</tr>
<tr>
<td>March</td>
<td>design reviews</td>
</tr>
<tr>
<td>S  M  Tu  W  Th  F  S</td>
<td>April</td>
</tr>
<tr>
<td>1  2  3  4  5  meet with bwk by 9th</td>
<td>1  2</td>
</tr>
<tr>
<td>6  7  8  9 10 11 12 design doc draft</td>
<td>3  4  5  6  7  8  9</td>
</tr>
<tr>
<td>13  14  15  16  17  18  19 spring break - enjoy</td>
<td>10  11  12  13  14  15  16</td>
</tr>
<tr>
<td>20  21  22  23  24  25  26 design doc; TA mtg</td>
<td>17  18  19  20  21  22  23</td>
</tr>
<tr>
<td>27  28  29  30  31 design reviews</td>
<td>24  25  26  27  28  29  30</td>
</tr>
<tr>
<td>May</td>
<td>beta test</td>
</tr>
<tr>
<td>S  M  Tu  W  Th  F  S</td>
<td>May</td>
</tr>
<tr>
<td>1  2  3  4  5  6  7 project demos</td>
<td>1</td>
</tr>
<tr>
<td>8  9  10  11  12  13  14 Dean’s date</td>
<td>17  18  19  20  21  22  23</td>
</tr>
<tr>
<td></td>
<td>alpha test</td>
</tr>
</tbody>
</table>
Some mechanics

• groups of 3 or 4
  - find your own partners

• Chris and Aquinas will be first-level managers

• weekly meeting of your whole group with your manager each week after break
  - everyone must attend essentially all of these

• be prepared:
  - what have we accomplished
  - what didn’t 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