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

Choices  (a small and incomplete list)

• user interface
  - browser, desktop, phone, API, ...
  - HTML/CSS, Javascript, Flash, Jquery, Java, ...

• main languages
  - C++, Java, C#, Perl, Python, PHP, Ruby, Objective C, ...

• server
  - own machine, OIT, CS, App Engine, Amazon EC2, hosting service, ...

• database
  - MySQL, SQLite, Postgres, flat files, ...

• information exchange formats
  - text, XML, JSON, REST/Atom, ...

• development tools and frameworks
  - Django, Rails, Google Web Toolkit, XCode, Eclipse, Visual Studio, ...
Getting started

• right now, if not sooner
  - think about potential projects; form a group
    talk to TA’s & bwk; look at previous ones;
    look around you; check out the external project ideas page
• by Fri Mar 5: short meeting with bwk (earlier is fine)
  - to be sure your project idea is generally ok
  - you should have one pretty firm consensus idea, not several vague ones
• Fri Mar 12: design document draft (before break)
  - ~3 pages of text, pictures, etc. a template will be posted
  - overview
    project name / title, paragraph on what it is, one person as project manager
  - 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
• should be based on significant thought and discussion
• 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, but 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 start

• 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 and what you can use from elsewhere
  - resolve issues of connectivity, access to information, software, etc.

• implementation ("what by when")
  - make prototype
  - deliver in stages, so that each does something and still works
  - test as you go: if your system is easy to break, it gets a lower grade

Make versus buy

• you can use components and code from elsewhere
  - copy or adapt open source

• overall project 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 and outputs?
• what is the transformation?
• who manages resources, especially memory and shared state?

• hide design & implementation decisions behind interfaces, so they can be changed later without affecting the rest of the program
  - database system, data representations and file formats
  - 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 make changes without breaking anything."

Deciding what to do

• informal thinking and exploring 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 (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?
    short attention-grabbing description without big words but good buzzwords
  
• 5-7 slides for a 5-10 minute talk or a poster session
  - 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 and demo outline for the end of the semester
  - what would you want working for the demo?

• business plan
  - how would you pitch it to a venture capitalist or Yagoosoft?
    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**
  - don’t take on too big a job
  - don’t try to do it all at the beginning, but don’t try to do it all at the end

- **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: decisions will be reversed and work will be redone
  - disaster: lost files, broken hardware, overloaded systems, ...
  - sickness: you will lose time for unavoidable reasons
  - health: there is more to life than this project!

---

**2010 Project Schedule**

<table>
<thead>
<tr>
<th>February</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>M</td>
<td>Tu</td>
<td>W</td>
<td>Th</td>
<td>F</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>March</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>April</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>May</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- initial talk with bwk this week
- design doc due **before** break
- spring break
- TA meetings begin this week
- project prototype
- alpha test
- beta test
- demo days - project presentations
- Dean's date; all done
Some mechanics

- **groups of 3 to 5**
  - find your own partners
  - use the newsgroup for match-making
  - don’t leave this to the end
- **TA’s will be your first-level managers**
  - more mentoring and monitoring than managing
  - it’s your project, not the TA’s
- **meet with your manager every week after spring break**
  - everyone in the group must attend all of these meetings
- **be prepared**
  - what we accomplished
  - what we didn’t get done
  - what we do plan to do next
- **these meetings are a graded component**
  - this is an attempt to make sure that you don’t leave it all to the end