

The COS 333 Project

Copyright © 2018 by
Robert M. Dondero, Ph.D.
Princeton University

Overview

- A simulation of reality
- In groups of 3-5 people...
- Build a substantial networked *three tier* application

Three-Tier App

- **Three tier** app
 - User interface tier
 - Usually graphical, often browser, sometimes mobile
 - Data management tier
 - Usually a DBMS
 - Processing tier
 - “Business logic”

Three-Tier Apps

- Examples:
 - Many websites: stores (Amazon, Ebay, ...), news services, maps, ...
 - Many Internet apps: Email, chat, ...
 - Many mobile apps
- Your app should be similar
 - But smaller, simpler, defined by your interests or the needs of others

Decisions

- User interface tier
 - Browser, desktop, tablet, phone, ...
 - HTML, CSS, JavaScript, AJAX, jQuery, AngularJS, React, Java, Objective-C, Swift, ...
- Data management tier
 - Relational vs. NoSQL
 - SQLite, MySQL, PostgreSQL, MongoDB, flat files, ...

Decisions

- Processing tier
 - **Languages:** Java, Python, PHP, C, C++, C#, Perl, Ruby, JavaScript...
 - **Data comm formats:** text, serialized objects, XML, JSON, ...
 - **Dev frameworks:** Bottle, Flask, Django, Spark, Spring, Rails, Android SDK, ...
 - **Dev tools:** Eclipse, Visual Studio, Android Studio, XCode, ...
 - **Hosting services:** your computer, CS Dept computer, OIT computer, Heroku, AWS, ...

Decision Making Strategies

- Do informal thinking and exploring early
 - So you have time to let ideas gel
- Do many simple experiments early
 - So you learn what works or doesn't
- Do *least-risk design*
 - Minimize risk
 - The module to be developed next should be the one which, if problematic, will have the largest negative impact on the system as a whole

Getting Started

- Now!
- Better: Two weeks ago!
- Think about potential projects
 - Check out *Previous Projects* web page
 - Check out *Project Ideas* web page
 - Talk to CS faculty and students
 - Talk to faculty and students in other depts
 - Look around you!

Potential Projects

- **Option 1:** Develop an app to fulfill your needs => **maybe a simulation of reality**
- **Option 2:** Develop an app to fulfill the needs of others => **reality!**
- I strongly recommend option 2
 - Again, see *Project Ideas* web page

Working with Instructors

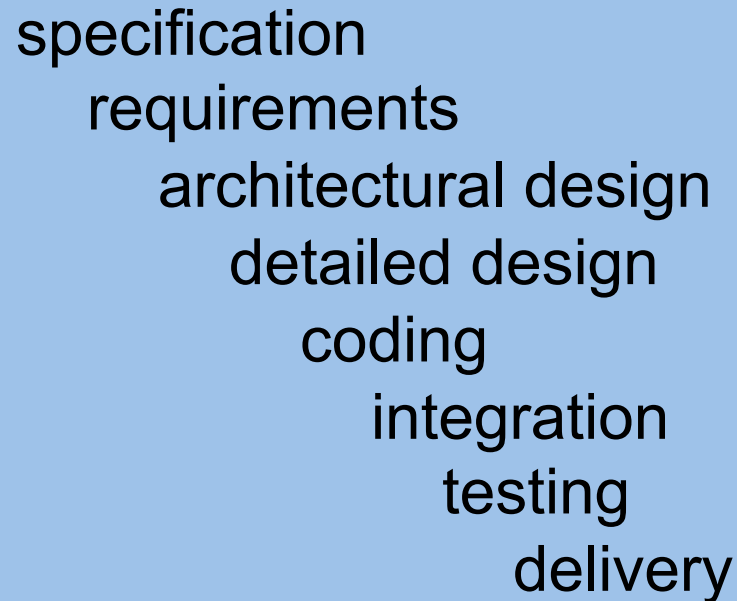
- You will manage your project
- We will help
 - **TA adviser:** first-level “manager”
 - More **mentoring** and **monitoring** than **managing**
 - It’s your project, not the TA’s
 - **Lead instructor:** second-level “manager”
 - Again, more **mentoring** and **monitoring** than **managing**

Process

- Use an orderly process
- This is **NOT** a process:
 - Talk about the app at dinner
 - Hack some code together
 - Test it a bit
 - Do some debugging
 - Fix the obvious bugs
 - Repeat until the semester ends

Process

- Classic *waterfall* model
 - Overkill for 333, but some process is essential



The diagram illustrates the classic waterfall model of software development. It consists of a light blue rectangular box containing a list of steps arranged in a descending staircase pattern from top-left to bottom-right. The steps are: specification, requirements, architectural design, detailed design, coding, integration, testing, and delivery.

```
graph TD; A[specification] --> B[requirements]; B --> C[architectural design]; C --> D[detailed design]; D --> E[coding]; E --> F[integration]; F --> G[testing]; G --> H[delivery];
```

specification
requirements
architectural design
detailed design
coding
integration
testing
delivery

An Informal Process

- **Step 1: Form a team**
 - For match-making:
 - Use ProjectFinder app (required)
 - Use Piazza (optional)

An Informal Process

- **Step 2: Choose a leader**
 - Goal: *conceptual integrity* (Brooks)
 - Make system so coherent that it appears to be the product of a single mind
 - Implications:
 - Everyone has to pull together
 - Someone has to be in charge
 - **Successful software development projects are not democracies**

An Informal Process

- **Step 3: Define requirements (who, what)**
 - **Who** are the users?
 - Decide what user need(s) the system will fulfill
 - Involve users
 - Conduct interviews, watch them work, ...
 - **What** should the system do?
 - Create scenarios
 - Low fidelity early: paper-and-pencil screen sketches, story boards, ...
 - Maybe high fidelity later: static screens, ...
 - Involve users
 - Show scenarios, solicit feedback

An Informal Process

- **Step 4: Design (how)**
 - **How** will the system work?
 - Decide pervasive design issues: languages, dev environment, DBMS, ...
 - Determine how to get data
 - Partition into major subsystems
 - **Specify interfaces between subsystems**
 - **Make “build versus buy” decisions**

Design: Interfaces

- Interface
 - The “public” part of a module
 - A module’s “advertisement” to clients
 - A module’s contract with clients
 - What are the inputs?
 - What are the outputs?
 - Who manages resources?
 - Who detects/reports errors?

Design: Interfaces

- Hide design & implementation decisions behind interfaces
 - So they can be changed later without affecting the rest of the program
- Common comment: “I wish we had done interfaces better”
- Less common comment: “We thought hard about the interfaces so it was easy to make changes without breaking anything”
- **Try to stay friendly!**

Design: Build vs. Buy

- OK to use modules/code from elsewhere
 - E.g., copy or adapt open source
- However:
 - Must identify what you have used, and where it came from
 - Overall project design must be your work
 - Selection and assembly of components must be your work
 - Most of the code must be your work

An Informal Process

- **Step 5: Implement**
 - Involve all team members
 - Every team member must do a significant part of the coding
 - Code in stages
 - Do not build something that requires a “big merge” where nothing works until everything works
 - **Always be able to stop and declare success**

An Informal Process

- **Step 6: Test**
 - Does the system work as **you** intend?
 - Integrated with Implement step
 - Make sure it always works
 - Fix bugs before adding features
 - Additional distinct step at the end

An Informal Process

- **Step 7: Evaluate**
 - Does the system work as its **users** intend?
 - Does the system fulfill the users' needs?
 - Approaches:
 - Involve users
 - Do it yourself using formal evaluation techniques

An Informal Process

- **Step 8: Document**
 - Integrated with previous steps
 - Additional distinct step at the end
 - User's guide, programmer's guide, ...

An Informal Process

- Count on iteration
 - Revisit Requirements and Design infrequently
 - Iterate between Implement and Test frequently

An Informal Process

- Use version control system for all code
 - **Git** (or, by permission, some other VCS) **is mandatory**

An Informal Process

- Leave lots of room for “overhead” activities
 - **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!
 - **Deliverables:** You must package your system for delivery...

Deliverables

- See *Project* web page for details
- See *Schedule* web page for due dates
- All deliverables are graded

Deliverables

- Pre-project deliverables:
 - Entry in *ProjectFinder Application*
 - Goal: Help you form a project team
 - Goal: Inform others what you're doing
 - Add row and repeatedly update it to indicate your:
 - Technical interests, project interests, project name (blank if still looking), project descrip (blank if still looking)
 - <http://www.cs.princeton.edu/~rdondero/ProjectFinder>

Deliverables

- Pre-project deliverables:
 - *Approval Meeting*
 - Team meets with lead instructor, to be sure your project idea is OK
 - Team presents one reasonably firm consensus idea, not several vague ones

Deliverables

- Early-project deliverables:
 - *Team Directory*
 - On Google drive of team leader
 - *Project Overview* document
 - In Team Directory
 - Elevator speech, overview, requirements, functionality, design, milestones, risks

Deliverables

- Mid-project deliverables:
 - *Timeline* document
 - In Team Directory
 - **Updated weekly**
 - Weekly status meetings
 - Attendance is mandatory
 - Be prepared to describe what you accomplished, what you didn't, what you plan to do next, anticipated risks

Deliverables

- Late-project deliverables:
 - Demonstration of prototype
 - During team meeting
 - Demonstration of alpha version
 - During team meeting
 - Demonstration of beta version
 - During team meeting
 - Presentation
 - Slides in Team Directory

Deliverables

- Late-project deliverables (cont.)
 - *User's Guide* document
 - In Team Directory
 - Description of **what your app does**
 - Audience: (non-technical) end user
 - **Probably the most important document**
 - *Programmer's Guide* document
 - In Team Directory
 - Description of **how your app works**
 - Audience: maintenance programmer

Deliverables

- Late-project deliverables (cont.)
 - *Product Evaluation* document
 - In Team Directory
 - Results of testing
 - Does the app work as you intend?
 - Results of formal evaluation
 - Does the app fulfill its users' needs?
 - *Project Evaluation* document
 - In Team Directory
 - Surprises? Lessons learned? Advice to future teams?

Deliverables

- Late-project deliverables (cont.)
 - Source code
 - In Team Directory
 - The application
 - Give instructors a URL, installation instructions, a computer, a phone,...