333 Project

- **a simulation of reality**
  - building a substantial system
  - in groups of 3 to 5 people

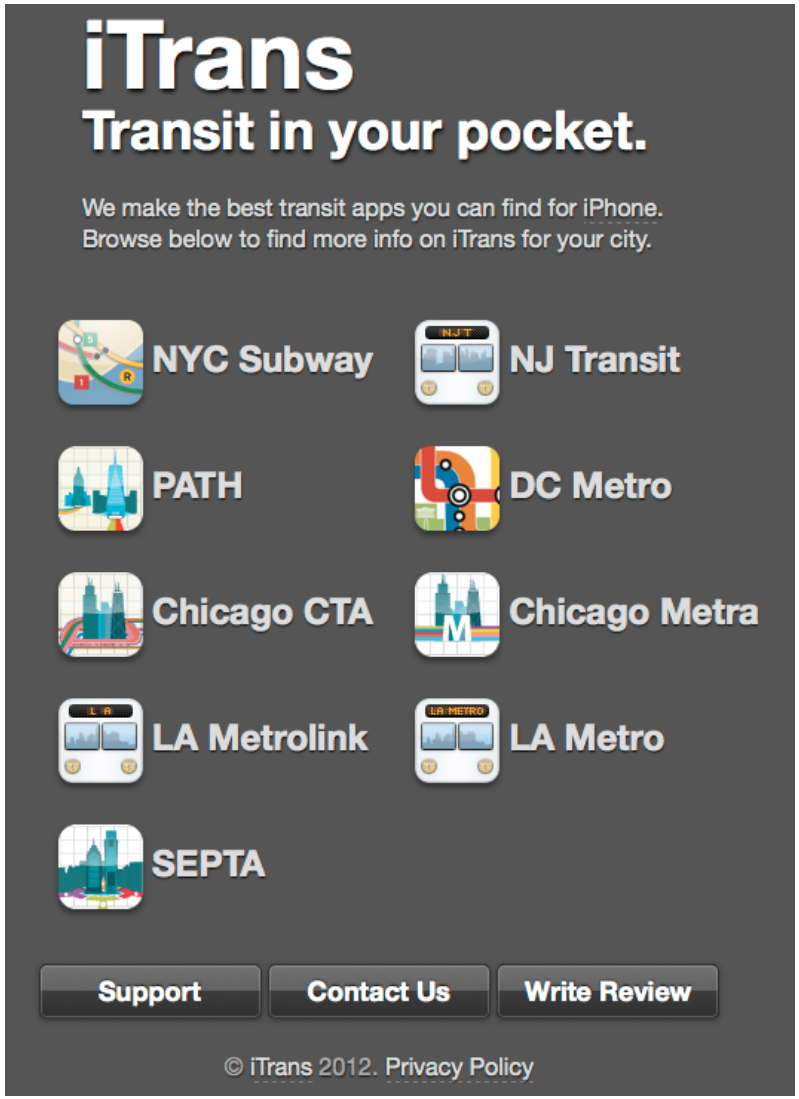
- **"three-tier" system for any application you like**

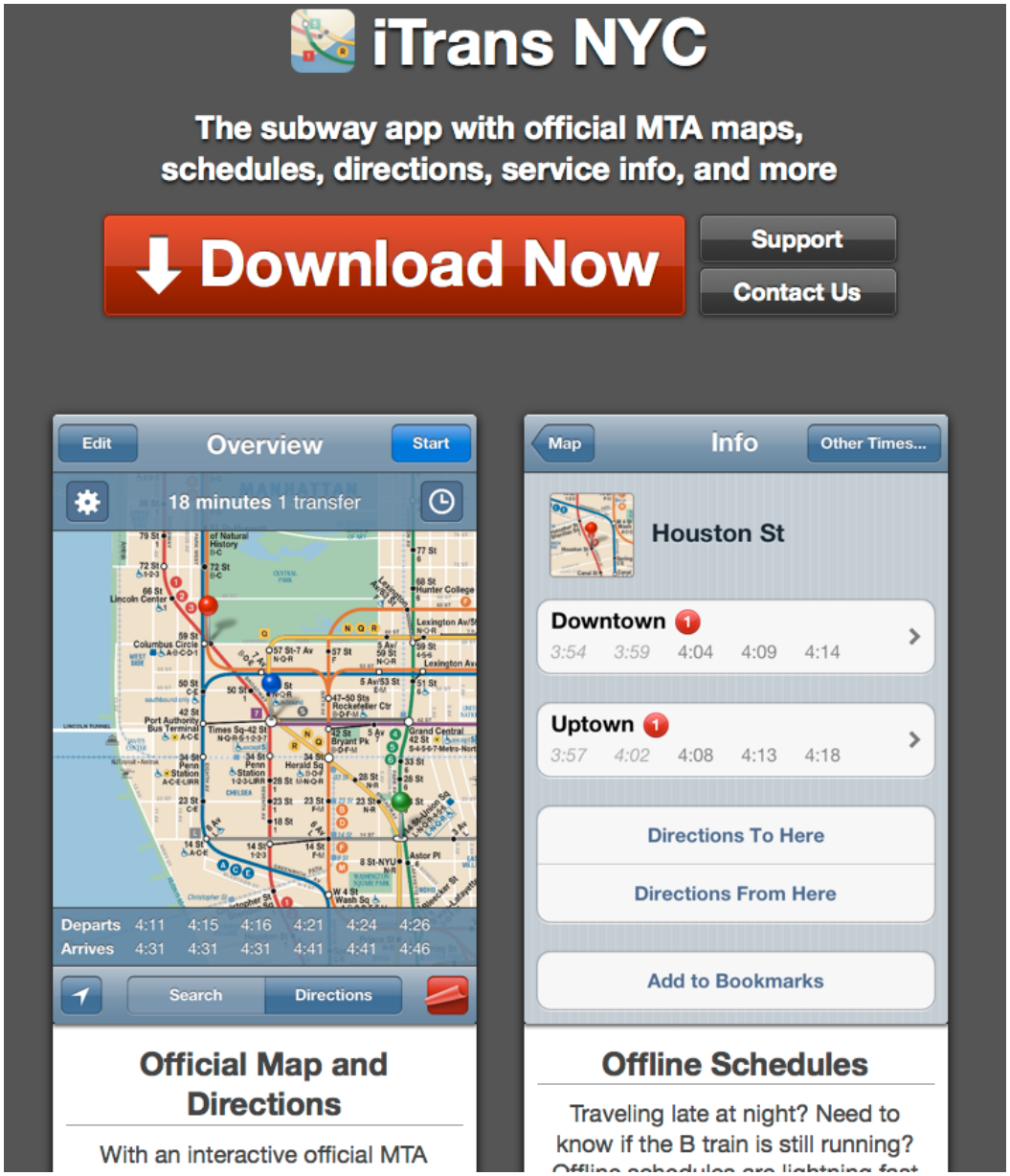
- **3 major pieces**
  - graphical user interface ("presentation layer")
  - processing in the middle ("business logic")
  - storage / data management
- **examples: many web-based services**
  - Amazon, Ebay, other web stores
  - news, information services, bots, mashups
  - email, chat, search, code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

Some local examples

- Point
- PTX
- Events
- Rooms
- ICE
- TigerFinder

itrans.info


iTrans
Transit in your pocket.
We make the best transit apps you can find for iPhone.
Browse below to find more info on iTrans for your city.
NYC Subway
NJ Transit
PATH
DC Metro
Chicago CTA
Chicago Metra
LA Metrolink
LA Metro
SEPTA
Support
Contact Us
Write Review
© iTrans 2012. Privacy Policy


iTrans NYC
The subway app with official MTA maps,
schedules, directions, service info, and more
Download Now
Support
Contact Us
Edit
Overview
Start
18 minutes 1 transfer
Departs 4:11 4:15 4:16 4:21 4:24 4:26
Arrives 4:31 4:31 4:31 4:41 4:41 4:46
Search
Directions
Official Map and Directions
With an interactive official MTA
Map
Info
Other Times...
Houston St
Downtown 1
3:54 3:59 4:04 4:09 4:14
Uptown 1
3:57 4:02 4:08 4:13 4:18
Directions To Here
Directions From Here
Add to Bookmarks
Offline Schedules
Traveling late at night? Need to
know if the B train is still running?

Choices (a small and incomplete list)

- **user interface**
  - browser, desktop, phone, game console, 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, ...
- **development tools and frameworks**
  - Django, Rails, Google Web Toolkit, XCode, Eclipse, Visual Studio, ...

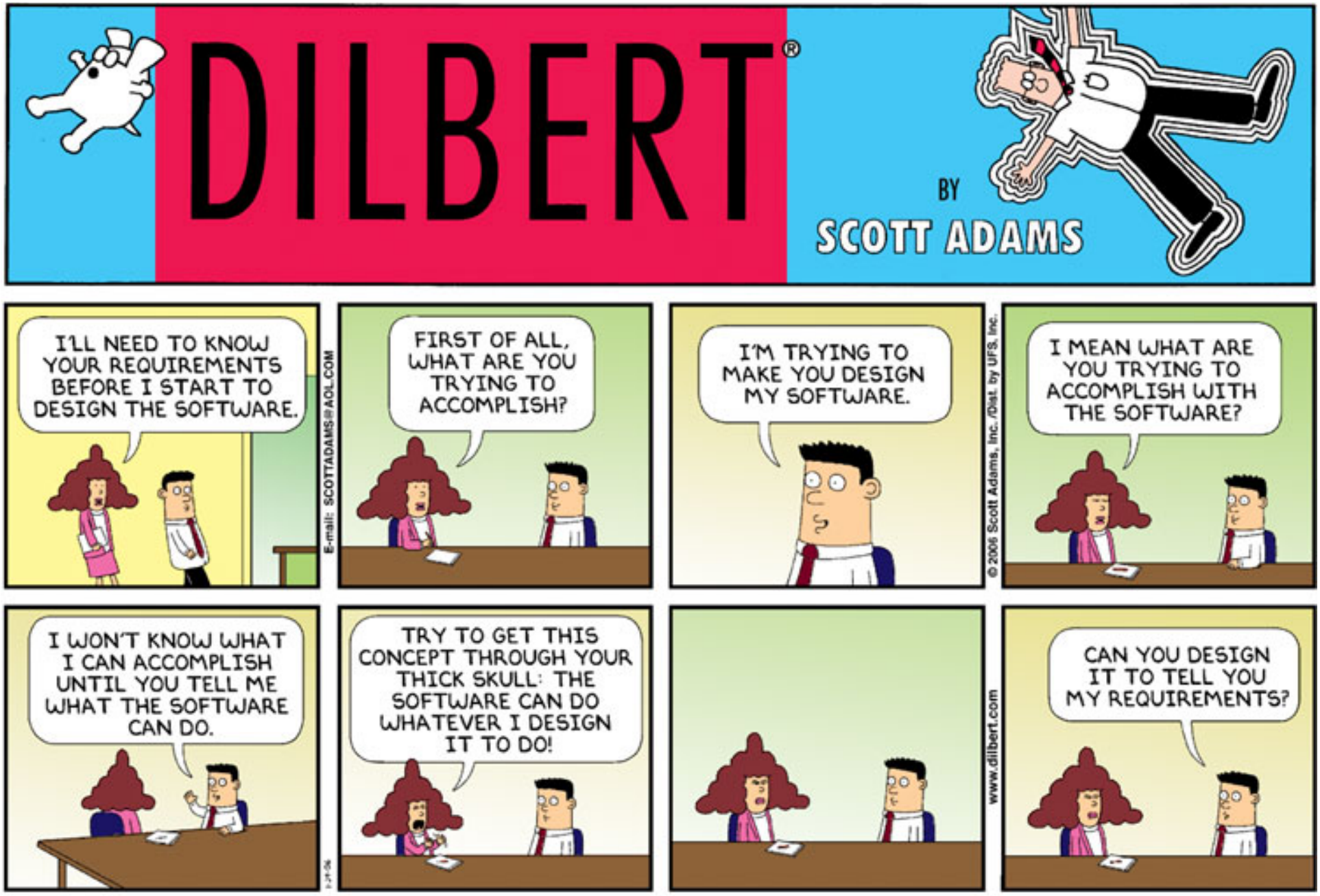
Getting started

- **right now, if not sooner**
  - think about potential projects; form a group
    talk to TA's & bwk; look at previous projects;
    look around you; check out the external project ideas page
- **by Fri Mar 9: short meeting with bwk (earlier is desirable)**
  - to be sure your project idea is generally ok
  - you should have one pretty firm consensus idea, not several vague ones
- **Fri Mar 16: design document draft (before break)**
  - ~3 pages of text, pictures, etc. a template will be posted
  - overview, initial web page, elevator speech
    project name / title, paragraph on what it is, one person as project manager
  - components & interfaces
    major design choices: web vs. standalone, languages, tools, environment, …
    major pieces, how they fit together
  - 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? make sketches, scenarios, screenshots
- **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?

For example, dating sites currently suck far worse than search did before Google. They all use the same simple-minded model. They seem to have approached the problem by thinking about how to do database matches instead of how dating works in the real world. An undergrad could build something better as a class project.

Paul Graham (co-founder of Y-Combinator)

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, Git 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!

2012 Project Schedule

	Su	Mo	Tu	We	Th	Fr	Sa	
Feb				1	2	3	4	
	5	6	7	8	9	10	11	<- you are here
	12	13	14	15	16	17	18	
	19	20	21	22	23	24	25	
	26	27	28	29				
Mar					1	2	3	
	4	5	6	7	8	9	10	initial talk with bwk
	11	12	13	14	15	16	17	design doc due before break
	18	19	20	21	22	23	24	spring break
	25	26	27	28	29	30	31	TA meetings begin this week
Apr	1	2	3	4	5	6	7	
	8	9	10	11	12	13	14	project prototype
	15	16	17	18	19	20	21	
	22	23	24	25	26	27	28	alpha test
	29	30						
May			1	2	3	4	5	beta test
	6	7	8	9	10	11	12	project presentations
	13	14	15	16	17	18	19	Dean's date: all done
	20	21	22	23	24	25	26	
	27	28	29	30	31			

Some mechanics

- **groups of 3 to 5**
  - find your own partners
  - use Piazza for match-making
  - meet potential partners before or after class
  - 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

Peer advisors

- **I am trying to arrange a group of people who have done the course before who will offer advice, perhaps give tutorials on particular topics, etc.**

- **More details to come**