Northeastern

CS5500 MANAGING SOFTWARE DEVELOPMENT

INTRODUCTION

Northeastern University
College of Computer and Information Science
440 Huntington Avenue . 202 West Village H . Boston, MA 02115 . T 617.373.2462 . ccis.northeastern.edu

CS5500 “MANAGING SOFTWARE DEVELOPMENT”

Section 1 meets Tuesdays and Fridays from 9:50am - 11:30am in Behrakis 325.
Section 2 meets Tuesdays and Fridays from 1:35 pm - 3:15 pm in Behrakis 320.

Instructor for Section 1: Michael Weintraub
Email: m.weintraub@northeastern.edu
Office Hours: 2-4pm Mondays and Thursdays and by appointment
Office: 302-d WVH 617.373.2301

Instructor for Section 2: Frank Tip
Email: f.tip@northeastern.edu
Office Hours: 3:30-4:30pm Tuesdays and Fridays and by appointment
Office: 322 WVH 617.373.2642

Teaching Assistants

Aravind Chinta chinta.a@husky.neu.edu
Harsha Jakkappanavar jakkappanavar.h@husky.neu.edu
Joyesh Kakkar joyeshk@ccs.neu.edu
Ashish Kumar kumar.as@husky.neu.edu

COURSE OBJECTIVE

Understand what it takes and develop your knowledge and skills to deliver successful software systems.

- Writing good code is a necessary condition
  - *But it's not enough to guarantee success*

Success or failure in software development depends on more than just writing good code and this is what the course covers.

When you co-op or graduate, you are going to work on projects valued at a few dollars up into the millions (or more). You may be a staff member or a leader. We want you to be successful.

SOFTWARE DISASTERS: DENVER AIRPORT

- ambitious new automated system for baggage handling
- one of the most notorious examples of project failure
- resulted in the newly complete airport sitting idle for 16 months while engineers worked on getting the system to work
- added approximately $560M to cost of the airport

- feature article in Scientific American titled the “Software’s Chronic Crisis”

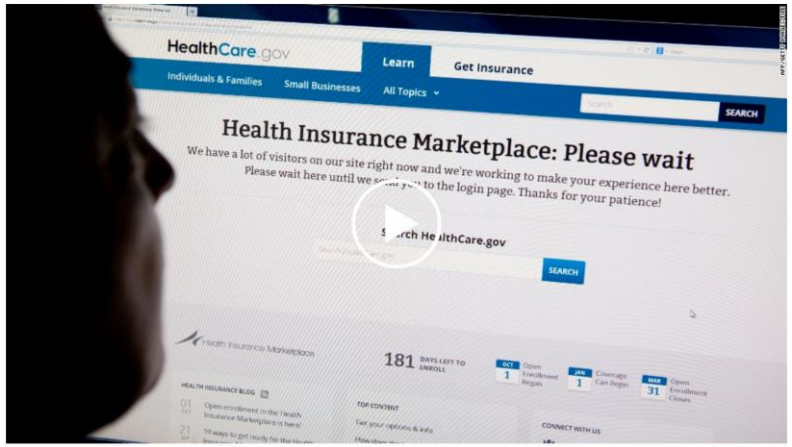
- http://calleam.com/WTPF/?page_id=2086
- http://users.csc.calpoly.edu/~jdalbey/SWE/Papers/SciAmGibbs/SciAmGibbs.html

SOFTWARE DISASTERS: OBAMACARE

Contractors blame government for Obamacare website woes

By Tom Cohen, CNN

Updated 6:41 AM ET, Fri October 25, 2013



- A more recent example of software-related problems…
- CNN: “Federal official says time frame didn't allow for enough testing of HealthCare.gov”
- Wall Street Journal: “Software, Design Defects Cripple Health-Care Website”

OTHER SOFTWARE DISASTERS

- Mariner 1 (1962)
  Rocket crash due to missing dash
- Eole 1 (1971)
  72 weather balloons get wrong cmd
- Nimbus 7 (1978)
  Satellite misses ozone hole for 6 yrs
- HMS Sheffield (1982)
  Exocet rocket id'ed as "friend"
- Stanislaw Petrow (1983)
  Russia detects global nuclear attack
- Therac 25 (1985)
  Radiation overdose kills six
- Stock crash (1987)
  Dow Jones loses 22% in one day
- Vincennes (1988)
  Passenger jet mistaken to be F-14
- Patriot (1991)
  Misses to shoot down Iraqi Scud
- Climate Orbiter (1999)
  Confuses metrics and imperial
- US Blackout (2003)
  50 mln affected for 5 days
- Apple SSL bug (2012)
  18 months w/o SSL authentication
- 3200 prisoners released early (2015)
- Nest Thermostat users left in the cold (2016)
- HSBC major outage (2016)
- Delta Airlines: power outage causes system-wide failure worldwide (2016)
- …

QUESTIONS

- Why does it take so long to get software finished?
- Why are the development costs so high?
- Why can’t we find all errors?
- Why do we spend so much time and effort maintaining existing programs?
- Why is it difficult to measure progress?

ONE SIZE DOES NOT FIT ALL

Domains Vary widely

Each area has its own lingo, history, drivers, funding, regulations,…

TECHNOLOGY is constantly in motion.

New stuff appears. Some of it is truly new. Some is old stuff with new wrappers. Some is an adaptation of old stuff.

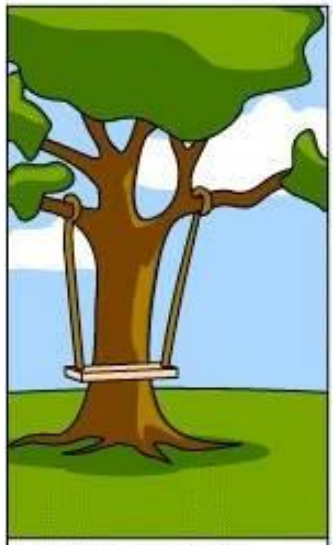
There is a belief that new is always better.

Old stuff evolves or stagnates.

There is a belief that old is always not as good as new.

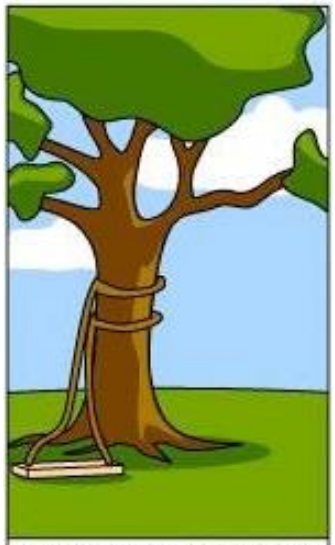
What you call new/old or good/bad often depends on where you sit, what you know, and whom you know.

How the customer explained it



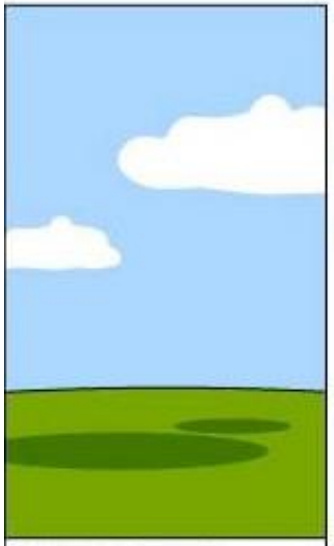
How the Project Leader understood it

How the Analyst designed it

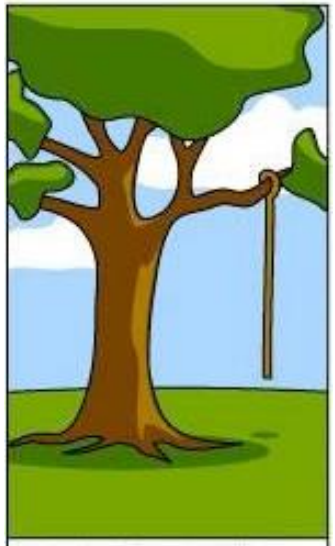


How the Programmer wrote it

How the Business Consultant described it

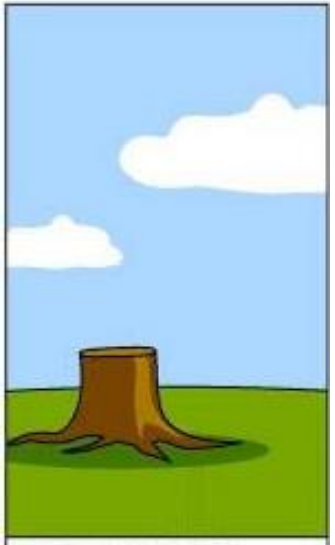


How the project was documented



What operations installed

How the customer was billed



How it was supported

What the customer really needed

WHAT'S THE LIFETIME OF YOUR WORK?

Look at engineering and the lifetime of many objects

How many bridges, roads, and buildings are 10's, 100's or possibly 1000 years old?

Can they be renovated, updated, or changed?

Think about software.

How much software lasts more than five years? 10 years? 20?
What is the common opinion of these systems?

Are these systems usually easily extended, updated, or renovated – directly and predictably?

YOUR OBJECTIVE

Given

- *Client requirements vary*
- *There is no standard process for software development/ engineering*
- *There is no "best" language, operating system, file store, database, environment, …*

Your job is to learn what is needed to deliver software successfully.

1. techniques,
2. technologies,
3. processes

The art of software development is to choose the right tools, methods, and procedures for a project and to deliver effective solutions.

LET'S LEVEL SET: WHAT DOES *SUCCESSFULLY* REALLY MEAN?

Most common answer: Delivering a quality solution

This is defined typically as:

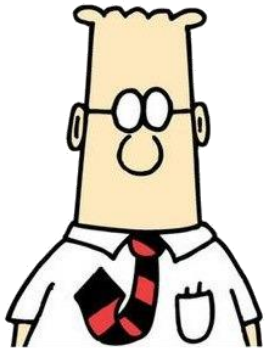
1. Functionality
   - ✦ The system has to do what it needs to do.
2. Usability
   - ✦ The users of the system need to be able to use it effectively and efficiently.
3. Maintainability
   - ✦ Changes or repairs must be able to be made directly and implemented without untoward impacts
4. Efficiency
   - ✦ The system should make effective use of resources
5. Dependability/ Reliability/Resiliency
   - ✦ The system should be available and remain working for a given period of time. When faced with failures or bad data, performance should not be impacted, or at least degrade gracefully.

So *successfully* is some achievement or measurement against these five factors…

WHAT DOES *DELIVER SOLUTIONS SUCCESSFULLY* REALLY MEAN?

Programmer View

1. Functionality
   Build what you asked for
2. Usability
3. Maintainability
4. Efficiency
5. Dependability/ Reliability/ Resiliency



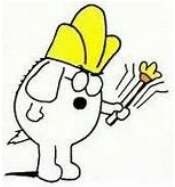
Management View

1, 2, 3,5	Actually work If it's a billed service/product, can we collect the data so we can collect?
1,5	Adapt to changes in market conditions
1,2,?	Satisfy any regulators (if any)
2	Be seen by users as a net plus rather than net negative
3	Be able to have whoever runs the system be able to do so Be extensible or repairable
??	Be defined, constructed, and deployed in a reasonable timeframe within an acceptable budget
4, ??	Be able to be run cost-effectively
	Does this put us ahead, even, or closer to the competition?
	How much revenue depends on this?

EFFECTIVE PROJECTS INVOLVE MULTIPLE PLAYERS

Product/Marketing

Bosses



Sponsors

Clients

Users

Tech Teams

Operations

Finance

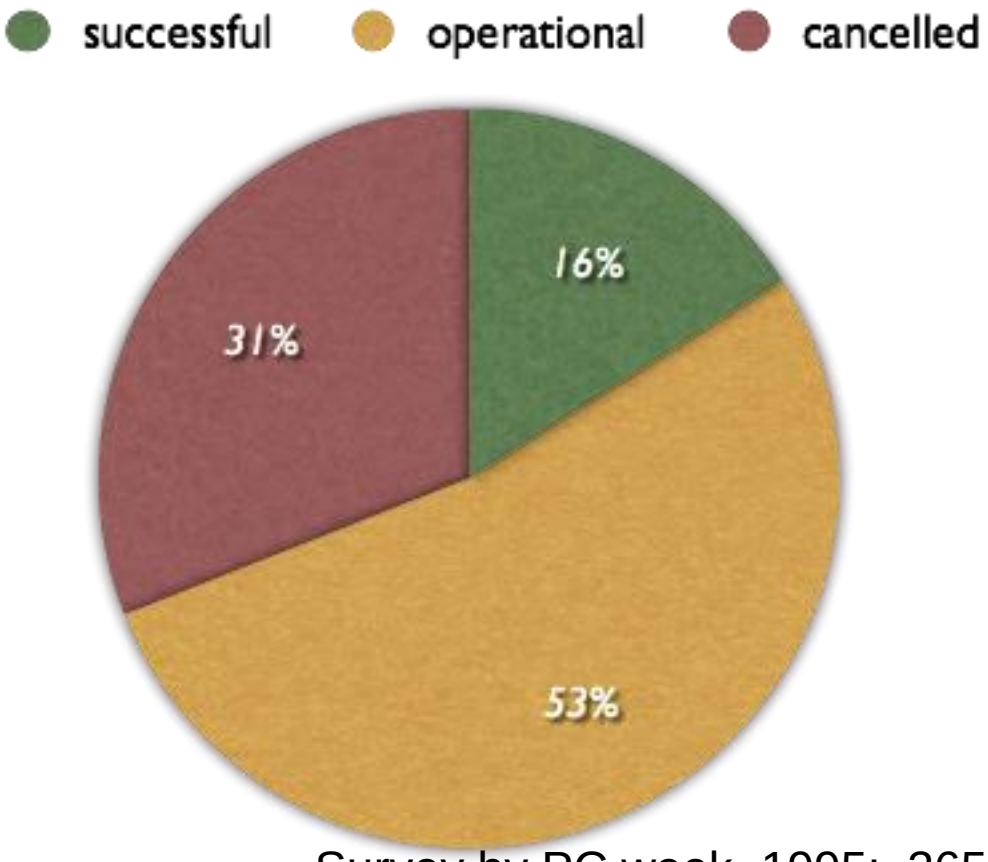
Lawyers

Executives

Business

Competitors

MEASURING SUCCESS



Survey by PC week, 1995: 365 information systems professionals on success of software development projects

ROADMAP TO THE COURSE

We will explore each constituent with respect to people, processes and technology

We will get our hands dirty with a group project

You will come away with an understanding of what is needed for professional software development

- √ Gain an understanding of the different pieces that all have to come together to deliver successful software
- √ Learn what programming within a team involves
- √ Get an exposure to some tools
- √ Continue to enhance your programming skills and/or broader experience base

COURSE PROJECT

- Designed so that you can experience the Software Engineering Lifecycle:
  - Requirements Gathering & Use Cases
  - OO Design using UML
  - Implementation
  - Testing
  - Extension and Refactoring (in response to changed requirements)

- Organized in 4 Phases
  - Phase 1: Requirements (individual)
  - Phase 2: Design (team)
  - Phase 3: Implementation (team)
  - Phase 4: Modified Design & Implementation (team)

COURSE PROJECT

- The course project will be a collaborative effort
  - Teams of 4 people
  - Sub-teams for different components of the system
- Intended to reflect realistic development practice:
  - Requirements intentionally somewhat vague
  - You don't get to pick your team-mates
  - Implementation language will be Java
  - Must use prescribed tools for version control, issue tracking, ...
- To make it "more interesting":
  - At several points in the project, team composition will be changed
  - …meaning that you will have extend/adapt/test code written by others
  - …so write your code with that in mind!

HOMEWORKS

- HW1: UML
- HW2: requirements
- HW3: design patterns
- HW4: to be announced…

COURSE STRUCTURE

- schedule (roughly) organized to go through the phases of software development, from requirements to implementation to refactoring/adaptation to deployment
- schedule of project deliverables (roughly) aligned with lecture content
- homework topics also (roughly) aligned with lecture content
- after a few initial lectures, we'll alternate between "lecture sessions" on Tuesdays and "project sessions" on Fridays
  - project sessions used for Q&A about project, code reviews, etc.
- final presentations about the project
  - plan is to do this in a single 5pm-10pm session for both sections of the course (food will be supplied by the instructors)
  - attendance for this session is mandatory
- no final exam!

COURSE SCHEDULE (PART 1)

		lecture topics	homeworks	project
week 1	Tue January 10	overview		
	Fri January 13	SE Life Cycle		
week 2	Tue January 17	UML	HW1 assigned	
	Fri January 20	UML		Phase 1 start
week 3	Tue January 24	requirements	HW1 due	
	Fri January 27	*project*		
week 4	Tue January 31	unit testing, JUnit	HW2 assigned	Phase 1 due
	Fri February 3	*project*	HW2 due	Phase 2 assigned
week 5	Tue February 7	system testing		
	Fri February 10	*project*		
week 6	Tue February 14	user experience (UX)		
	Fri February 17	*project*		
week 7	Tue February 21	architecture		Phase 2 due
	Fri February 24	*project*		Phase 3 assigned
week 8	Tue February 28	design principles		
	Fri March 3	*project*		

COURSE SCHEDULE (PART 2)

		lecture topics	homeworks	project
week 9	Tue March 7	*SPRING BREAK*		
	Fri March 10	*SPRING BREAK*		
week 10	Tue March 14	design patterns	HW3 assigned	
	Fri March 17	*project*		
week 11	Tue March 21	design patterns		
	Fri March 24	*project*		Phase 3 due
week 12	Tue March 28	refactoring	HW3 due	Phase 4 assigned
	Fri March 31	*project*		
week 13	Tue April 4	acceptance/release	HW4 assigned	
	Fri April 7	*project*		
week 14	Tue April 11	research topics: static analysis		
	Fri April 14	*project*	HW4 due	
week 15	Tue April 18	final presentations		Phase 4 due
	Fri April 21	final presentations		

GRADING

Project: 60%

- Four project phases - each phase counts equally.
- In each phase, separate grades will be determined for the entire project, and for each of the system's sub-components.
- In each phase, each student's grade is computed as: 0.5 * the grade for the entire project + 0.5 * the grade for the sub-component that the student worked on

Homeworks: 20%

- Four homeworks - each counts equally.

Final Presentation: 20%

- Top five rated presentations across both sections will be given a bonus.
- Presentations for both sections will happen on the same night.

GRADING

score	letter grade
100	A+
>93	A
90-93	A-
>87	B+
>83-87	B
80-83	B-
>77	C+
>73-77	C
70-73	C-
>67	D+
>63-67	D
60-63	D-
<60	F

- A 100 point scale will be in effect. There will be no rounding of scores. A curve may be introduced at the professor's discretion.

LOGISTICS

- Create CCIS id if you don't already have one
- Course Web Site: http://www.ccs.neu.edu/course/cs5500sp17/index.html
- Please enroll in the piazza board for this course at http://piazza.com/northeastern/spring2017/cs5500?token=4iplpp8j6O8
- On the course schedule, there are pointers to videos and documents on jira and git. If you want more to do, please take a look at those.
- Questions?