CSC301
Introduction to Software Engineering

Mohamed A. Mansour

Fall 2014

UNIVERSITY OF TORONTO
FACULTY OF ARTS & SCIENCE

Instructor

- Mohamed A. Mansour
- Email: mmansour@cs.toronto.edu
- Office: BA 4268

- Office Hours: Thursday 2:00-4:00pm

CSC301
Motivation

What do you think?

- Software Engineering == Coding/Programming
- Software Engineering != Coding/Programming
- Why?

Timetabe

	Day	Time	Room
Lectures	Tue	10:00 -- 11:00	BA1200
	Thu	10:00 --11:00	BA1200
Tutorials	Thu	11:00 -- 12:00	BA1200 BA2165 BA2185

No tutorials for the 1st week

Introduction to Software Engineering
Course Description

- Introduction to software development methodologies with an emphasis on *agile development methods* appropriate for rapidly-moving projects. Basic software development infrastructure; *requirements elicitation and tracking; prototyping; basic project management; basic UML; introduction to software architecture; design patterns; testing*.

Marking Scheme

Exercises **Mini-exercise 1%** **Exercise 9%**	10%
Project Work (4 phases)	45%
Midterm	10%
Final Exam	35%

Course Conduct

- Come to ALL the lectures and come prepared
  - *If asked to do any readings or other preparations, do it. Also try to think about it.*
- Take notes during lectures.
  - *Don't hesitate to ask questions to clarify material if you are not 100% clear on.*
- Use the Piazza Discussion Board for course discussions and general discussions regarding Software Engineering
  - *I really want to engage you guys on the board*
- Closed book test.
  - *The objective is not memorized answers. Demonstrate to us that you understand in your answers!*

Instructor Industrial Background

- About 15 years of industrial experience.
  - *4 years at a startup*
  - *6 years at IBM (mainly developing one of the WebSphere products)*
  - *1 year in a medium sized company*
  - *Independent consultant/mentor (for startup companies)*
- Played different roles: Software Engineer/Designer/Architect/Team Lead/Project Manager/Business Analyst
- Worked using Agile techniques and in *CMMi level 5* certified development center

Industrial Research

Worked for IBM Zurich Research Lab (ZRL) in Switzerland

- ZRL widely known for achieving two Nobel prizes (4 researchers)

Certifications

SUN Certified programmer for the JAVA 2 Platform

IBM Certified Business Process Analyst

IBM Certified Solution Designer - IBM Rational Unified Process V7.0

Academic

- Best Paper Award in 2007
- Published a book chapter
- Organized and chaired a workshop in 2009
- Was the principal investigator for a consulting engagement with one of the largest financial institutions in Canada investigating their Requirements Engineering practices

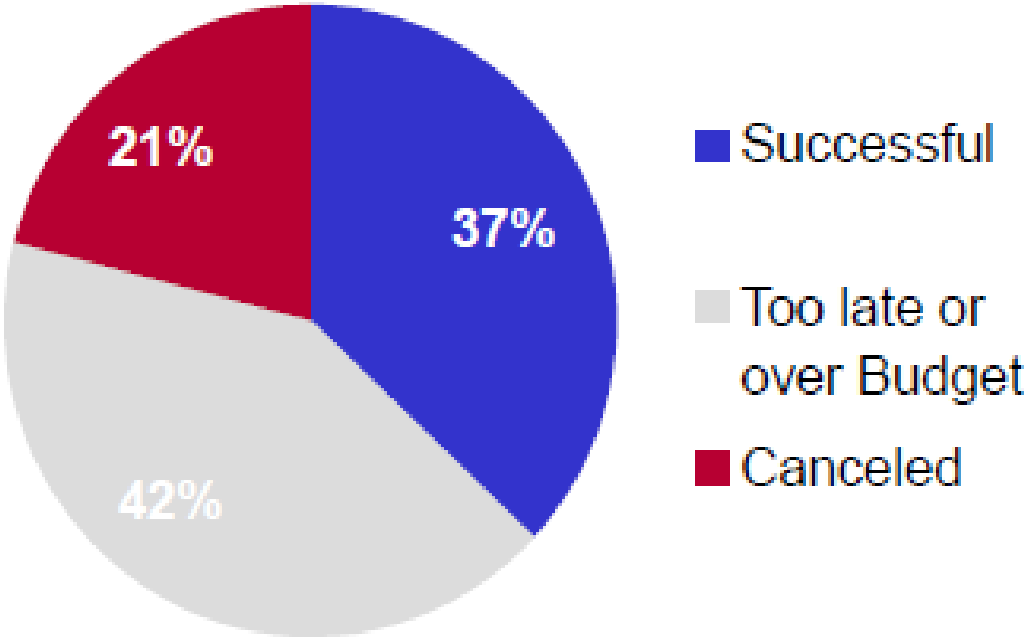
Why is Software Engineering Important?

- Modern economies highly depend on software
- Software expenses represent a significant fraction of GNP in all developed countries
- More and more systems are software controlled
- Almost every aspect of our life has some software
  - *Power plants, nuclear stations*
  - *Medical equipment*
  - *Banks, ATMs, Stock Market*
  - *Home Appliances*
  - *Cars, aeroplanes*
  - *Etc…*
- Software Engineering is concerned with theories, methods and tools for professional software development

Why is Software Engineering Important?

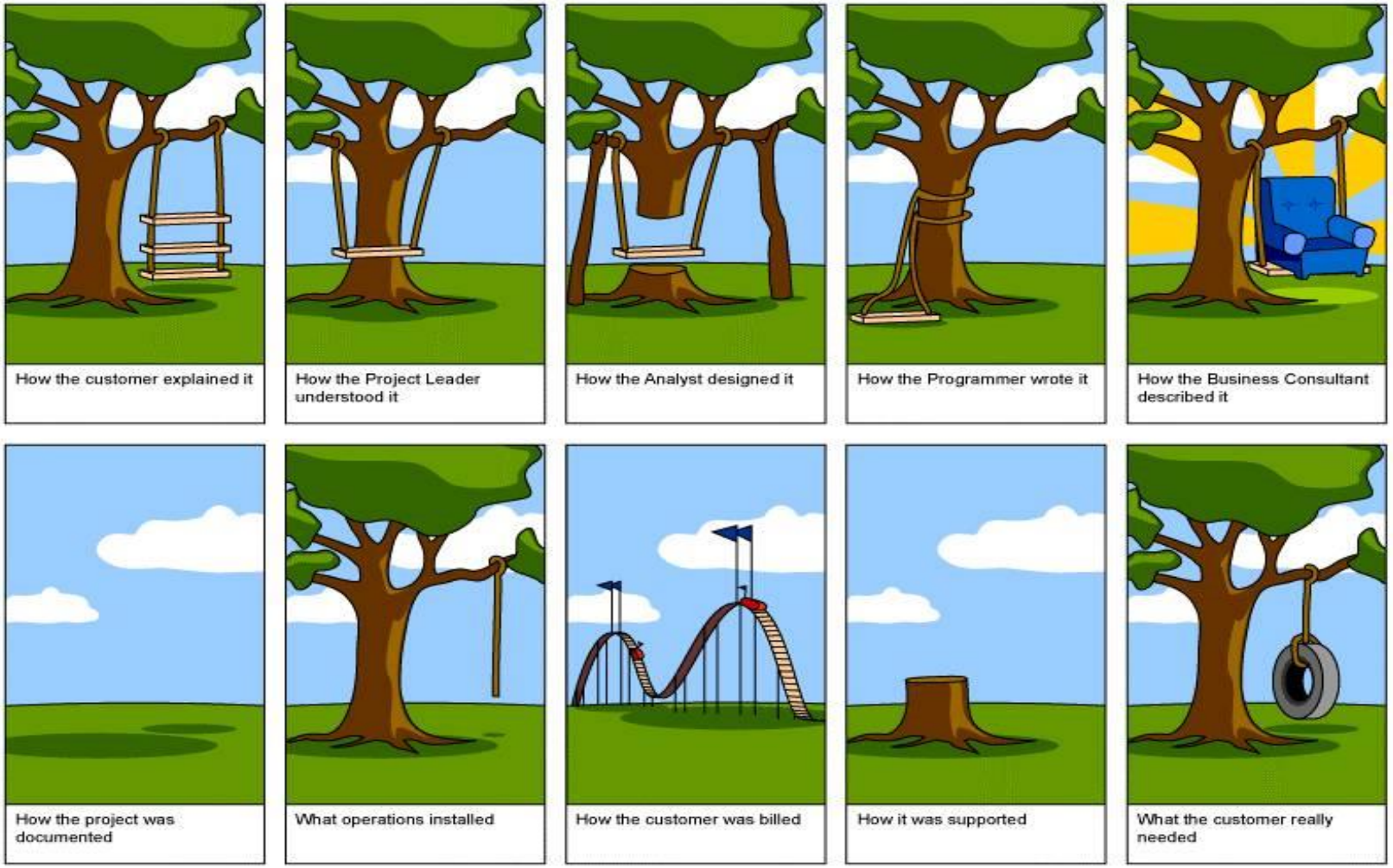
- The complexity of programs is increasing
- Increasing system lifetime
- New applications for the use of computers emerge.
  - *Over 10 Billion microprocessors are produced annually*
  - *Only 2% ended up in PCs, the rest is embedded in systems*
  - *About 1/3 of car cost is in its electrical/software equipment*
    - *Most of the innovation and value comes from these systems*
- Software development is an integral component of system development

Challenges in Software



Source:
Standish Group 2011, ca. 10000 Projects

All what you need to know about **Software Engineering** in one slide



Software Engineering

•The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is the application of engineering to software.

•Software engineering (SE) is concerned with developing and maintaining software systems that behave **reliably** and **efficiently**, are **affordable to develop and maintain**, and **satisfy all the requirements that customers have defined for them**. –**ACM** (http://computingcareers.acm.org/?page_id=12)

Real time systems

A computer system in which not having a results available in exactly the specified timeframe will yield a failure or even the loss of the entire mission.

Correctness of the systems depends **always** on being in time.

Embedded Systems

•A software system which is embedded to a larger system where its main purpose is not computation

•Most embedded systems are real-time systems

Case Study - Ariane 5

- “A picture is worth a thousand words”

- A video is worth million words.

- http://www.youtube.com/watch?v=gp_D8r-2hwk

Case Study - Ariane 5

•European Space Agency (ESA), spent 10 years and $7 billion to produce Ariane 5

•Ariane 5's first test flight on 4 June 1996 failed, with the rocket **self-destructing** 37 seconds after launch because of a malfunction in the control software.

•It had 4 expensive uninsured scientific satellites

Case Study - Ariane 5

- These are professionals, using the best practices BUT mistakes can happen ☹

- Brainstorming -
  - *Identify potential SE errors that can lead to this?*
  - *How to avoid such errors?*

Case Study - Ariane 5

•It all started with overflow exception when converting from a 64-bit floating point value to a 16-bit signed integer

•This happened in inertial reference system (IRS), which measures altitude and trajectory of the rocket

•IRS shutdown and was sending diagnostic information to the main system which was interpreting it as if it is normal flight data

Case Study - Ariane 5

A good article to read (that is not technical) was published by

New York Times

http://www.nytimes.com/1996/12/01/magazine/little-bug-big-bang.html?pagewanted=all&src=pm

Another example

“January 15, 1990 -- AT&T Network Outage. A bug in a new release of the software that controls AT&T's #4ESS long distance switches causes these mammoth computers to crash when they receive a specific message from one of their neighboring machines -- a message that the neighbors send out when they recover from a crash.

One day a switch in New York crashes and reboots, causing its neighboring switches to crash, then their neighbors' neighbors, and so on. Soon, 114 switches are crashing and rebooting every six seconds, leaving an estimated 60 thousand people without long distance service for nine hours. The fix: engineers load the previous software release.**”**

-History's Worst Software Bugs

http://www.wired.com/software/coolapps/news/2005/11/69355?currentPage=2

A 3rd example - The Northeast blackout of 2003

Most probably you have witnessed the effect of this bug by yourself.

The Northeast blackout of 2003 was a widespread power outage that occurred throughout parts of the Northeastern and Midwestern United States and the Canadian province of Ontario on Thursday, August 14, 2003, just before 4:10 p.m. EDT

A software bug known as a race condition existed in General Electric Energy's Unix-based XA/21 energy management system. Once triggered, the bug stalled FirstEnergy's control room alarm system for over an hour. System operators were unaware of the malfunction; the failure deprived them of both audio and visual alerts for important changes in system state

http://en.wikipedia.org/wiki/Northeast_blackout_of_2003

Course Project - Tools

Github (A distributed version control system)

In CSC-207 “Software Design” you learned about SVN, in this course you will learn about Github

Next lecture, Github will be introduced