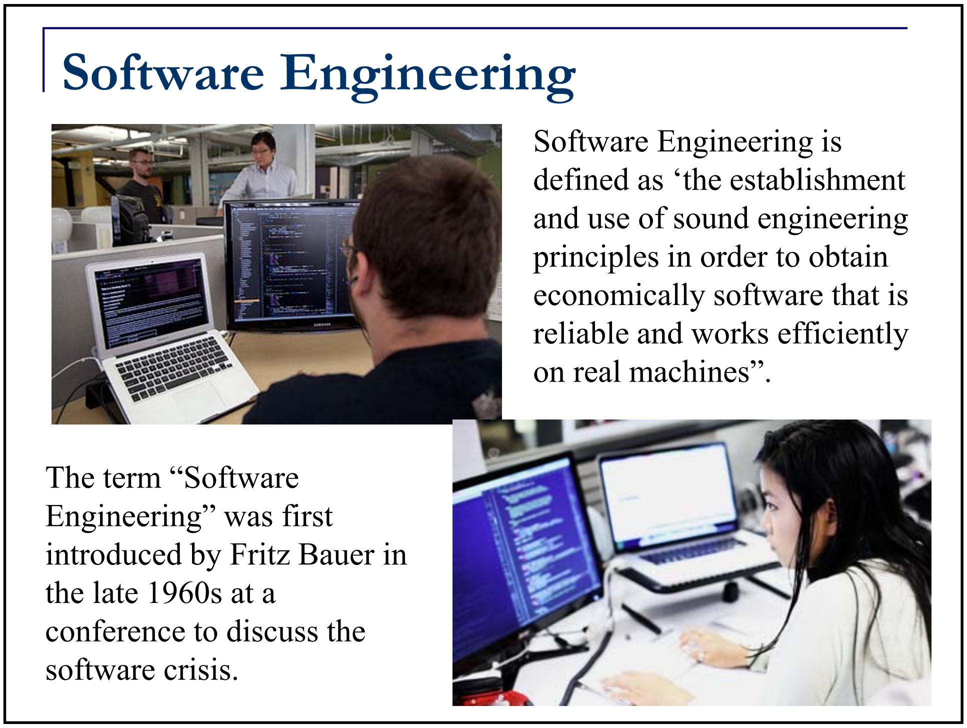
CSC40232: SOFTWARE ENGINEERING

Professor: Jane Cleland-Huang
Lecture 1: Introduction
Wednesday, January 18th
sarec.nd.edu/courses/SE2017

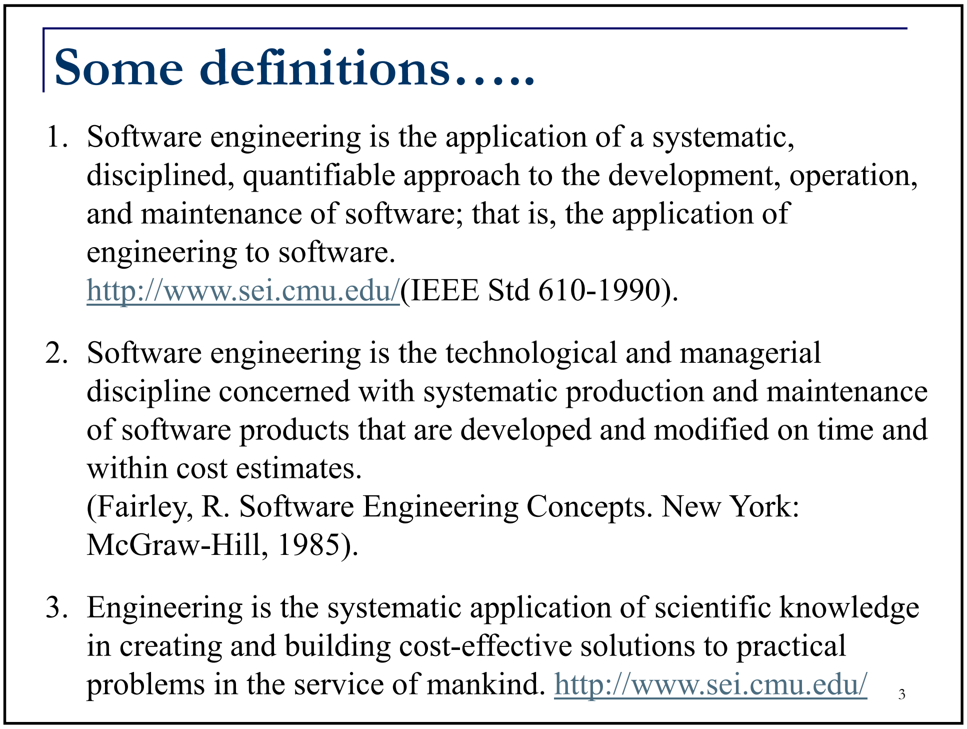
Department of
Computer Science and
Engineering



Software Engineering

Software Engineering is defined as 'the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines".

The term "Software Engineering" was first introduced by Fritz Bauer in the late 1960s at a conference to discuss the software crisis.

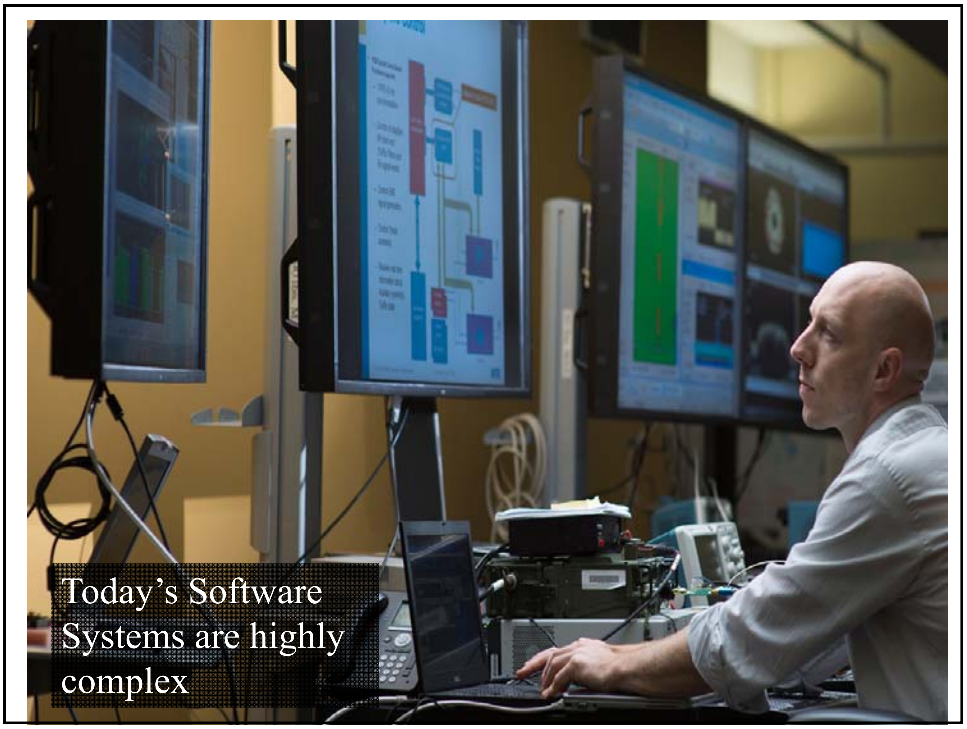


Some definitions…..

1. Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
   http://www.sei.cmu.edu/(IEEE Std 610-1990).

2. Software engineering is the technological and managerial discipline concerned with systematic production and maintenance of software products that are developed and modified on time and within cost estimates.
   (Fairley, R. Software Engineering Concepts. New York: McGraw-Hill, 1985).

3. Engineering is the systematic application of scientific knowledge in creating and building cost-effective solutions to practical problems in the service of mankind. http://www.sei.cmu.edu/



Today's Software Systems are highly complex

Ariane 5 – Flight 501

Background

- European Space Agency's reusable launch vehicle.
- Ariane-4 a major success.
- Ariane-5 developed for larger payloads.

Events

- Launched 4th June, 1996.
- $500 million payload
- Veered off course during launch.
- Self-destructed 40 seconds after take-off.

Why?

- Unhandled floating point exception.

Therac-25

- One of the worst accidents in the computer industry.
- A medical device used to treat patients for tumors and cancers.
- Small amounts of radiation are applied to a patient through a set of filters that reduces the actual amount of radiation the patient receives.
- Resulted in 6 deaths.
- No safety interlocks programmed into the software system.

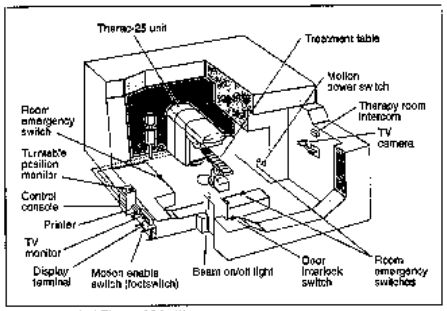


Figure 1. Typical Therac-25 facility.

Recent Failures:

http://www.computerworlduk.com/galleries/infrastructure/top-10-software-failures-of-2014-3599618/

In December 2015 more than 3,200 US prisoners were released early due to errors in the way a prisoner's sentence is computed according to good or bad behavior.

Another one…

http://www.computerworlduk.com/galleries/infrastructure/top-10-software-failures-of-2014-3599618/

In 2014-2015 Nissan recalled airbags from over 1 million cars because of software errors in the way sensor information was used to determine whether an adult or child was in the front seat.

And one final one…

http://www.computerworlduk.com/galleries/infrastructure/top-10-software-failures-of-2014-3599618/

A Software glitch in the F-35 Joint Strike Fighter meant that its sensors couldn't differentiate between singular and multiple threats.

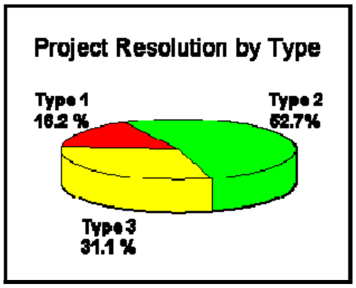
Chaos Report

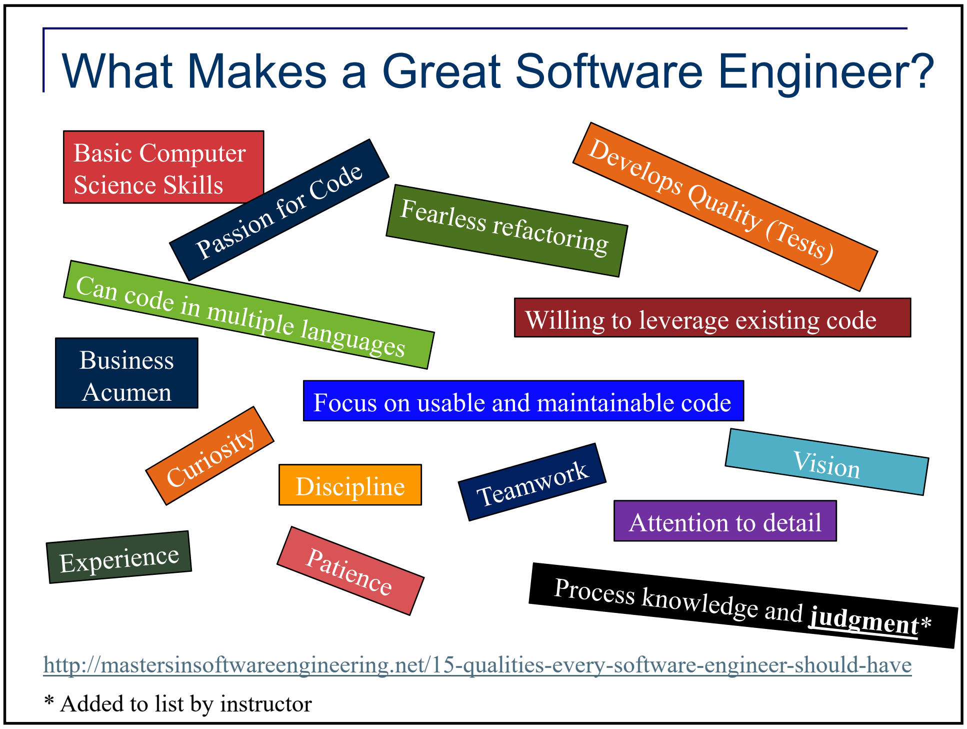
Standish Research Group Report

Project Success: Type 1. The project is completed on-time and on-budget, with all features and functions as initially specified. **(2000: 28%)**

Project Challenged: Type 2. The project is completed and operational but over-budget, over the time estimate, and offers fewer features and functions than originally specified. **(2000: 49%)**

Project Impaired: Type 3. The project is canceled at some point during the development cycle. **(2000: 23%)** (Are ALL impaired projects failures???)





What Makes a Great Software Engineer?

Basic Computer Science Skills

Passion for Code

Fearless refactoring

Develops Quality (Tests)

Can code in multiple languages

Willing to leverage existing code

Business Acumen

Focus on usable and maintainable code

Curiosity

Discipline

Teamwork

Vision

Attention to detail

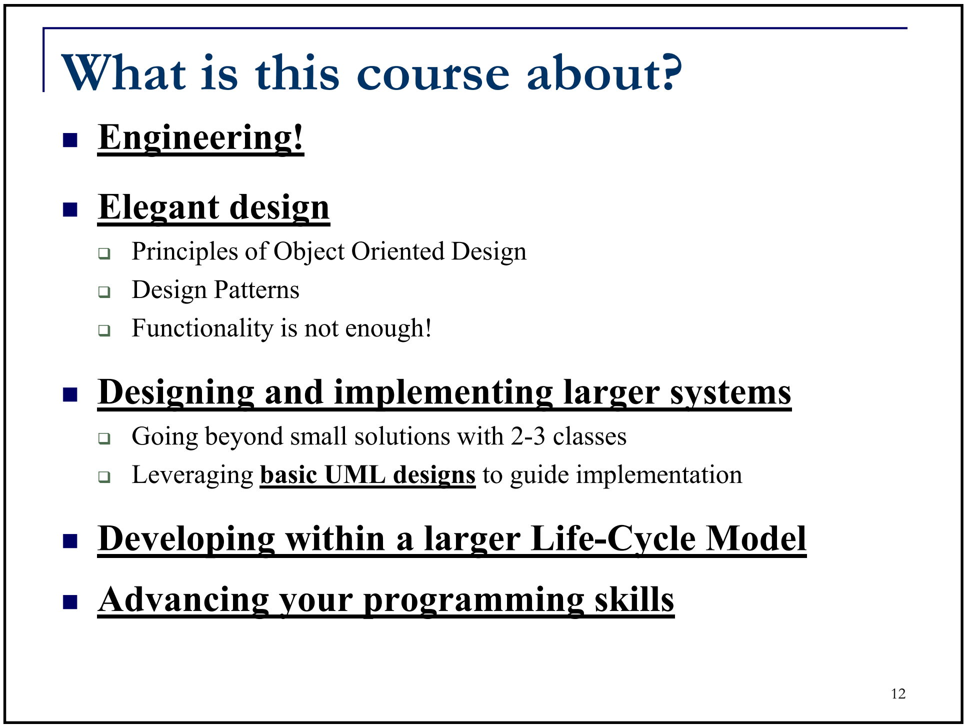
Experience

Patience

Process knowledge and **judgment***

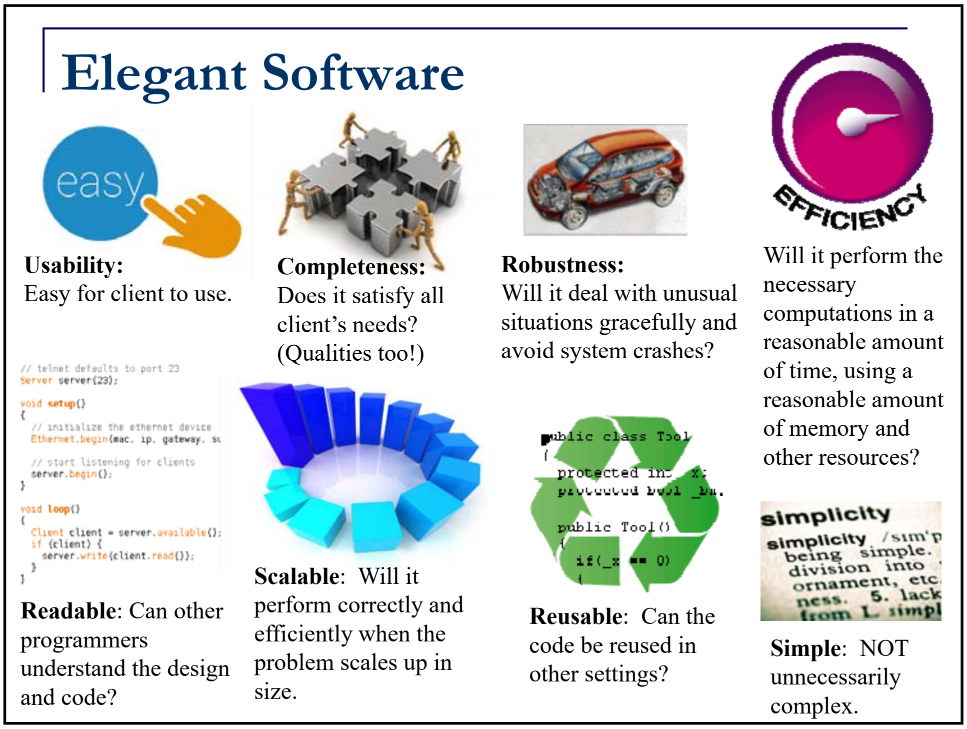
http://mastersinsoftwareengineering.net/15-qualities-every-software-engineer-should-have

* Added to list by instructor



What is this course about?

- **Engineering!**
- **Elegant design**
  - Principles of Object Oriented Design
  - Design Patterns
  - Functionality is not enough!
- **Designing and implementing larger systems**
  - Going beyond small solutions with 2-3 classes
  - Leveraging **basic UML designs** to guide implementation
- **Developing within a larger Life-Cycle Model**
- **Advancing your programming skills**



Elegant Software

Usability:
Easy for client to use.

Completeness:
Does it satisfy all client's needs? (Qualities too!)

Robustness:
Will it deal with unusual situations gracefully and avoid system crashes?

Will it perform the necessary computations in a reasonable amount of time, using a reasonable amount of memory and other resources?

```
// telnet defaults to port 23
Server server(23);

void setup()
{
  // initialize the ethernet device
  Ethernet.begin(mac, ip, gateway, su

  // start listening for clients
  server.begin();
}

void loop()
{
  Client client = server.available();
  if (client) {
    server.write(client.read());
  }
}
```

Readable: Can other programmers understand the design and code?

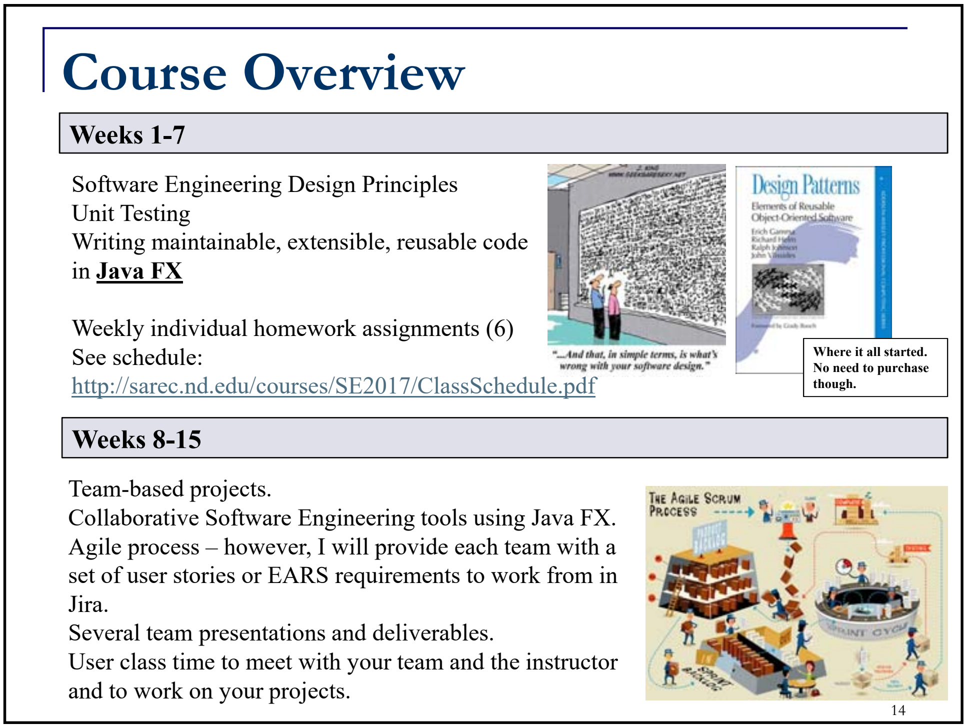
Scalable: Will it perform correctly and efficiently when the problem scales up in size.

```
public class Tool
{
  protected int  x;

  public Tool()
  {
    if(_x == 0)
    {
```

Reusable: Can the code be reused in other settings?

Simple: NOT unnecessarily complex.


"...And that, in simple terms, is what's wrong with your software design."
Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
THE AGILE SCRUM PROCESS

Course Overview

Weeks 1-7

Software Engineering Design Principles
Unit Testing
Writing maintainable, extensible, reusable code in **Java FX**

Weekly individual homework assignments (6)
See schedule:
http://sarec.nd.edu/courses/SE2017/ClassSchedule.pdf

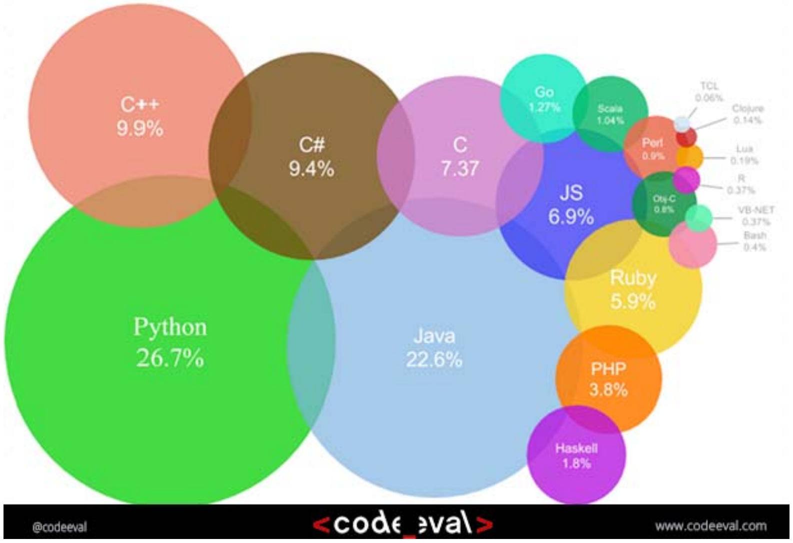
Where it all started.
No need to purchase though.

Weeks 8-15

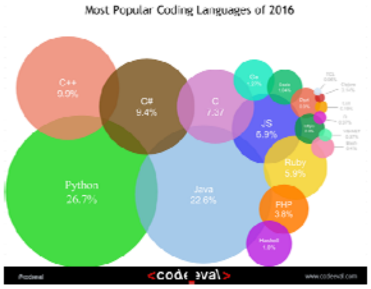
Team-based projects.
Collaborative Software Engineering tools using Java FX.
Agile process – however, I will provide each team with a set of user stories or EARS requirements to work from in Jira.
Several team presentations and deliverables.
User class time to meet with your team and the instructor and to work on your projects.

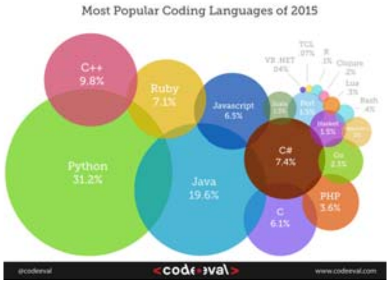
14

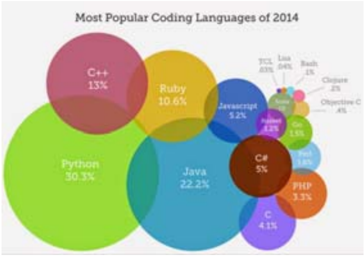
Why Java?

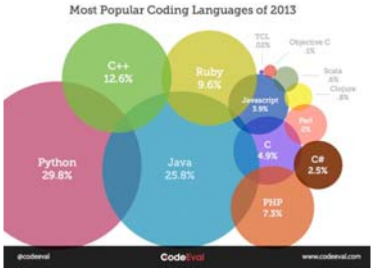


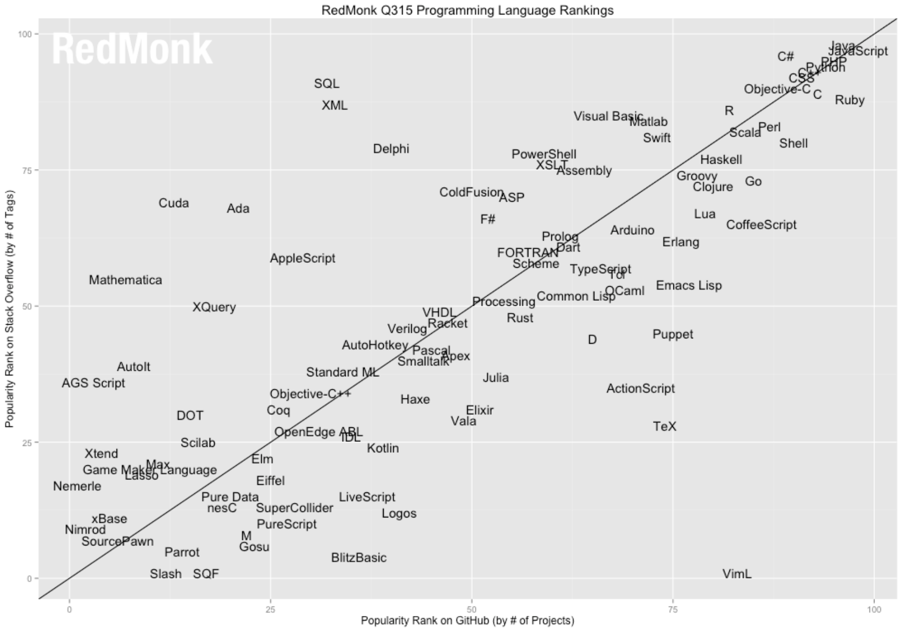
Programming Language Trends





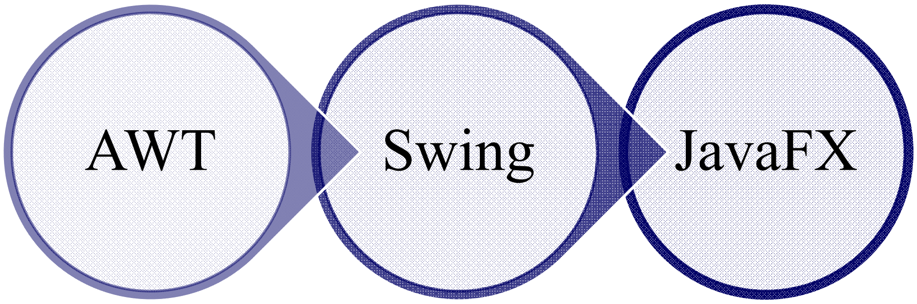






17

Why Java FX?



Abstract Window Toolkit:
Original GUI framework – but platform specific in terms of appearance.

GUI Toolkit
Behavior and appearance of Swing components is consistent across platforms. Designed for desktop Apps.

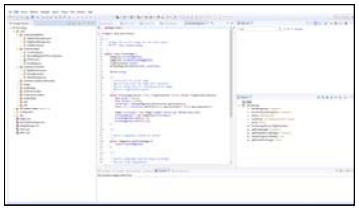
Set of **graphics** and **media packages** that enables developers to design, create, test, debug, and deploy rich client applications that **operate consistently across diverse platforms**

18

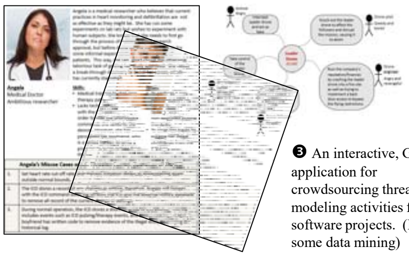
Team Projects

How can I help you today?
TIQI: A NATURAL LANGUAGE INTERFACE FOR QUERYING SOFTWARE PROJECTS
University of Notre Dame
DePaul University

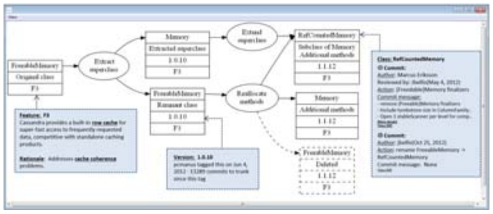
❶ A new interactive TiQi Application that utilizes our TiQi engine. http://tinyurl.com/TIQIDemo



❷ An eclipse plugin supporting impact analysis in Safety Critical Systems.



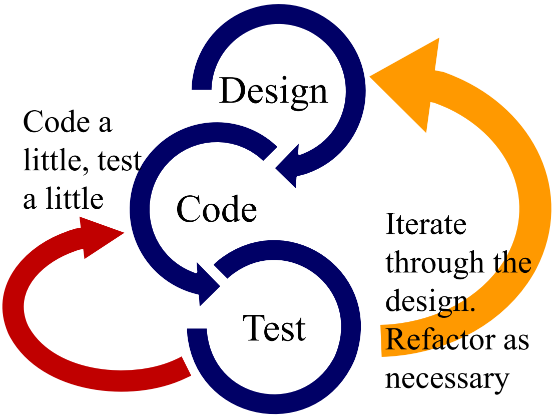
❸ An interactive, GUI based application for crowdsourcing threat modeling activities for software projects. (Requires some data mining)



❹ A utility for interactively visualizing the evolution of requirements and source code. (Some underlying code is available for detecting refactorings).

3-4 people per team.
Max three teams per project.

Program/Test Iteratively



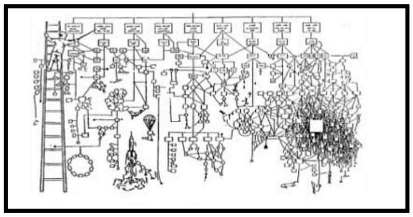
Code a little, test a little

Iterate through the design. Refactor as necessary

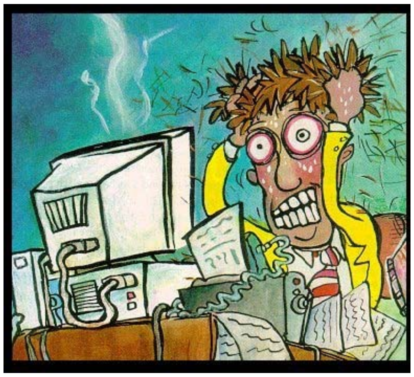
The good, the bad, and the ugly news

- **Bad news:** I will NOT help you to debug large big-bang projects. Build your projects in small pieces that are tested incrementally before you add the next piece.
- **Good news:** I will help you if you are stuck on small increments of your assignments.

Avoiding the Big Ball of Mud



Every programmer starts off with great intentions of an elegant design. However, code often doesn't end up that way!!



Big ball of mud" systems have usually been developed over a long period of time, with different individuals working on various pieces and parts. Systems developed by people with no formal architecture or programming training often fall into this pattern.

Have you ever experienced such problems?

The goal of this course is to equip you with the skills you need to produce and to maintain elegant code for non-trivially sized systems.

Homework #1 and Lecture 2

Professor Huang is out of town on January 23rd.

Lecture 2 is therefore **online only**. It consists of multiple video snippets to help you to **get started with Java and Eclipse**. Further, each video snippet is associated with a homework questions to be **coded in java** and tested using a **JUnit test case**.

Demo