Introduction

Idaho State University | Computer Science

Isaac Griffith

CS 2263
Department of Informatics and Computer Science
Idaho State University

ROAR

Outcomes

After today's lecture you should be able to:

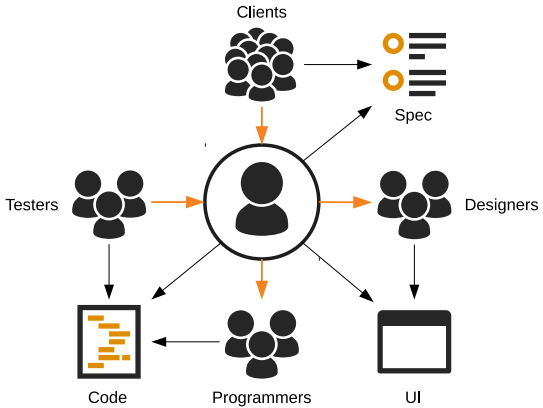
- Understand the reason for this course
- Understand the requirements of this course and policies of the course as laid out in thy syllabus
- Describe the need for teamwork and what makes for a good team
- Describe the need for software engineering in a larger context

Inspiration

“Nine women cannot make a baby in one month.” – Fred Brooks

Syllabus Review

Why care about Software Engineering?



- Interact with clients to determine their system requirements
- Translate user requirements into technical specifications
- Interact with designers to convey the possible interface of the software
- Interact/guide the coders/developers to keep track of system development
- Perform system testing with sample/live data with the help of testers
- Implement the new system

Defining Software

What is software?

Defining Software

What is software?

1. instructions (computer programs) that when executed provide desired features, function, and performance;

Defining Software

What is software?

1. instructions (computer programs) that when executed provide desired features, function, and performance;
2. data structures that enable the programs to adequately manipulate information, and

Defining Software

What is software?

1. instructions (computer programs) that when executed provide desired features, function, and performance;
2. data structures that enable the programs to adequately manipulate information, and
3. descriptive information in both hard copy and virtual forms that describes the operation and use of the programs

Software Deterioration

85%

Software Project Failure Rate

UK, USA, and Norway Surveys

50% Total Failures

40% Partial Failures

Standish Group (Corporate Projects) – 1994

31% Canceled

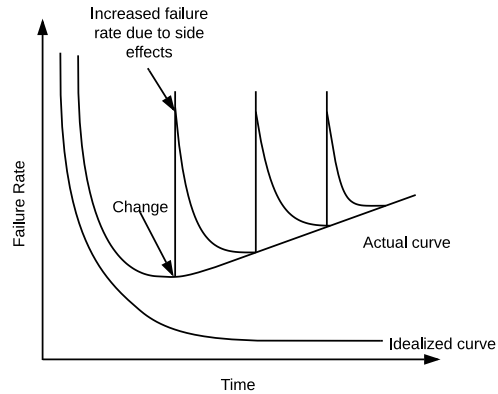
53% Challenged

180% Average Cost overrun

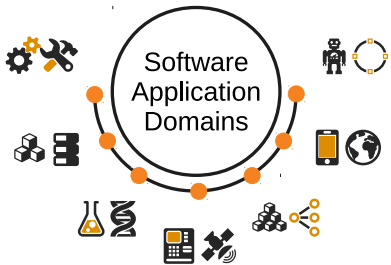
Standish Group 2007

46% Cost or Time Overruns

13% Outright Failures



Software Domains and Challenges



- Open-world computing
  - Creating software to allow machines of all sizes to communicate with each other across vast networks
- Netsourcing
  - Architect simple and sophisticated applications that benefit targeted end-user markets worldwide
- Open Source
  - Distributing source code for computing applications so customers can make local modifications easily and reliably

Legacy Software

Legacy Software Systems:

Systems that were developed decades ago and have been continually modified to meet changes in business requirements and computing platforms. The proliferation of such systems is causing headaches for large organizations who find them costly to maintain and risky to evolve.

Reasons Legacy Systems Evolve:

- Meet the needs of new computing environments/technologies
- To implement new business requirements
- To inter-operate with other more modern systems or databases
- To become viable within an evolving computing environment

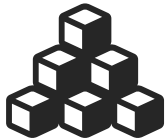
Changing Nature of Software

The goal of modern software engineering is to “devise methodologies that are founded on the notion of evolution;”

Software systems continually change and all must interoperate and cooperate with each other.

Four broad categories are evolving to dominate:



Software Engineering

Software Engineering (IEEE):

1. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
2. The study of approaches as in (1)

- Software engineering encompasses a process, a collection of methods, and an array of tools that allow professionals to build high quality software.
- Software engineers view computer software, as being made up of the programs, documents, and data required to design and build the system.
- Software users are only concerned with whether or not software products meet their expectations and make their tasks easier to complete.

Software Engineering Realities

- Problem should be understood before software solution is developed
- Design is a pivotal activity
- Software should exhibit high quality
- Software should be maintainable

Think-Pair-Share

Take 2 minutes and think about the following question:

Given the nature of software and the changes that have happened in your lifetime, what can we say about the changes we may see in the future? How do you think we will be building software in the next 5 years?

Pair up with your neighbor and take the next few minutes to discuss your thoughts.

Programming in the Large

Programming in the Small vs Large

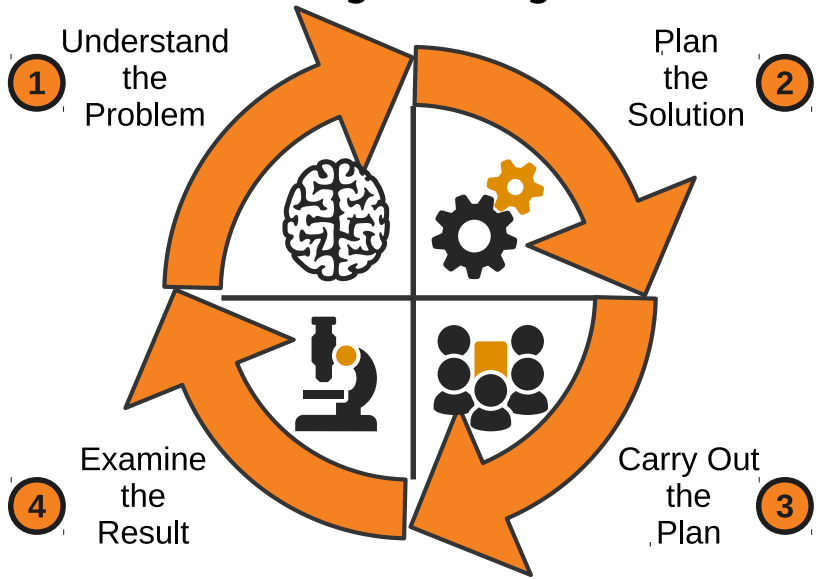
Programming in the Small:

- Programs do one relatively simple task well
- Team is typically small, even one engineer
- Program created is complete within itself
- Ready to run on system in which developed
- Typically, short-lived projects

Programming in the Large:

- Programs typically perform man complicated tasks
- Medium - Large-sized industrial teams
- Programs are split into several or hundreds of modules
  - Each module is similar in complexity to a single program
- Requires ongoing testing, maintenance, and documentation
- Work requires many months to several years

Software Engineering Practice



Software Engineering Practice

Understand the Problem

- Who are the stakeholders?
- What functions and features are required to solve the problem?
- Is it possible to create smaller problems that are easier to understand?
- Can a graphic analysis model be created?

Plan the Solution

- Have you seen similar problems before?
- Has a similar problem been solved?
- Can readily solvable subproblems be defined?
- Can a design model be created?

Creating a “Jelled” Team

- A team of people so strongly knit that the whole is greater than the sum of its parts

- Characteristics of a jelled team:
  - Very low turnover rate
  - Strong sense of identity
  - A feeling of eliteness
  - Team vs. individual ownership of the project
  - Team members enjoy their work

Motivating People

- Motivation is the greatest influence on performance
- Monetary rewards usually do not motivate
- Suggested motivating techniques:
  - 20% time rule
  - Peer-to-peer recognition awards
  - Team ownership (refer to the team as “we”)
  - Allow members to focus on what interests them
  - Utilize equitable compensation
  - Encourage group ownership
  - Provide for autonomy, but trust the team to deliver

Handling Conflict

- Preventing or mitigating conflict:
  - Cohesiveness has the greatest effect
  - Clearly defining roles and holding team members accountable
  - Establish work & communications rules in the project charter

- Additional techniques:
  - Clearly define plans for the project
  - Make sure the team understands the importance of the project
  - Develop detailed operating procedures
  - Develop a project charter
  - Develop a schedule of commitments in advance
  - Forecast other priorities and their impact on the project

Environment & Infrastructure

- Environment–Choose the right set of tools
  - Use appropriate CASE tools to:
    - Increase productivity and centralize information (repository)
    - Utilize diagrams–more easily understood
  - Establish standards to reduce complexity
- Infrastructure–Document the project appropriately
  - Store deliverables & communications in a project binder
  - Use Unified Process standard documents
  - Don't put off documentation to the last minute

Think Pair Share

Take 2 minutes and think about the following question:

Consider the notion of a jelled team. Today, have multiple teams and team members working on a project separated by distance is a common practice. How would distance affect the ability of maintaining a Jelled team?

Pair up with your neighbor and take the next few minutes to discuss your thoughts.

Are there any questions?