Introduction

Idaho State University | Computer Science

Isaac Griffith

CS 4422 and CS 5599
Department of Computer Science
Idaho State University

ROAR

Idaho State University
Computer Science

Outcomes

At the end of Today's Lecture you will be able to:

- Understand the course goals
- Understand the place of maintenance in software engineering

ROAR

Idaho State University
Computer Science

Introduction

What we will do today

- Course Introductions
- Syllabus Review
- Introduction to Software Testing
  - When Software Goes bad
  - Goals of Testing Software

ROAR

Idaho State University
Computer Science

Who Am I?

Isaac Griffith

Assistant Professor

Education at Montana State University:

- PhD, MS, BS in Computer Science
- Graduate Certificate in Applied Statistics
- BA in Philosophy

Research Interests:

- Empirical Software Engineering
- eXtended Reality & Computer Graphics
- Software Quality
- Software Architecture

ROAR

Idaho State University
Computer Science

A Bit More About Me

I have been told

- I come off as condescending
- I come off as rude
- I come off as angry
- I don't take criticism well

How to deal with this

- I am a confrontational and forceful speaker, in class
- If you are having a problem with me, please do not hesitate to speak with me individually either after class, via email, or in my office hours so that we can address and resolve the issue.
- I am very dedicated to being the best instructor I can, but without your help I cannot improve myself or my courses.

Pet Peeves

- Students coming late to class and causing an interruption to the flow.
- Students arguing a tangential point or perceived error, which should be taken offline

ROAR

Idaho State University
Computer Science

Student Introductions

Round-Robin Style

- Your (preferred) Name
- Your year in school
- Your major
- Something interesting about you

ROAR

Prerequisite Review

Course Prereqs

- CS 3321 Introduction to Software Engineering

What you should be familiar with:

- Software Engineering Processes
- Requirements Engineering
- Basic concept of testing

Idaho State University
Computer Science

Academic Integrity

- ISU Academic Integrity and Dishonesty Policy can be found at: http://coursecat.isu.edu/undergraduate/academic_integrity_and_dishonesty_policy/
- Academic Dishonesty is broken down into two groups:
  - Cheating
  - Plagiarism
- Instructor-Level Penalties:
  - Written Warning
  - Re-submission of work
  - Grade reduction
  - Fail the course
- My Policy
  - **First-time: Fail the Course**
  - Second-time: There isn't one!
- University-Level Penalties:
  - Suspension
  - Expulsion

ROAR

Idaho State University
Computer Science

Responsibilities of Professor

- Prepare **useful** and **interesting knowledge** for you
- Post materials on course website **before** class
- Come to class **on time**, **prepared** to teach
- Offer **challenging** but **reasonable** homework and tests
- Grade **fairly** without bias
- **Return graded work promptly** with educational comments
- Goals:
  - Have **interesting** lectures
  - Make the class **fun**
  - Use **technology appropriately**

ROAR

Idaho State University
Computer Science

Responsibilities of Students

- Come to class **on time**
- If you miss a class, **learn material** on your own
  - **Never miss the first meeting of any class!**
- **Listen** to all instructions
- Turn in assignments **on time**
- **Ask for help** when you're confused
- **Read** the material
- If you disagree with my policies, **disagree politely**
- **Goals**:
  - Read **before** class
  - **Learn** enough to earn a good grade

ROAR

Taking Notes

- The **slides summarize** the material
- My **words** emphasize **highlights**
- We **learn** a lot by **transferring** information
  - Through our ears
  - To our brains
  - To our pencils
  - Onto paper
- Unless you have a **perfect memory**, I expect you to take notes on what I say.

Taking notes will make a difference in your performance

Electronic Communication Devices

- Mobile phones, laptops, …
- Texting, IMming, Email, web surfing …
- These are all great tools – **out of the classroom**
- In the classroom, they
  - **Distract** the professor
  - **Annoy** your classmates
  - **Interfere** with your ability to learn
- **Laptops** can only be used during in-class exercises
- Other gadgets (and laptops) should be **silent** and **put away**
  - If you cannot keep your devices quite, you will be asked to leave

Reading

- Books have knowledge
- Professors are simply guides
- **Information**: comes from lectures
- **Knowledge**: comes from books and homework
- **Wisdom**: comes from experience

Read, Read, Read

Software Testing and Maintenance – Introduction

Idaho State University
Computer Science

“Traditional” Quality Attributes

1. Efficiency of process (time-to-market)
2. Efficiency of execution (performance)

This is what we teach is important in undergraduate CS classes...
... It was true in 1980

ROAR

Idaho State University
Computer Science

Modern Quality Attributes

1. Reliability
2. Usability
3. Security
4. Availability
5. Scalability
6. Maintainability
7. Performance & Time to market

All of these factors (sometimes called ”- ilities”) are important in the 2000s

ROAR

1960s Software Projects

- In the **1960s** we built tiny log cabins ...
- **Single**-programmer
- Not much **complexity**
- **No process** needed
- Design could be kept in short term **memory**

1970s Software Projects

- In the **1970s** we build bigger houses ...
- Still **single**-programmer - focus on **algorithms** and **programming**
- A little **more complex**
- We had to start **thinking harder**
- The lack of process led to some **disasters**

- For most of the industry, **quality** did not affect the bottom line
- But **costs** were starting to **increase**

1980s Software Projects

- In the **1980s** we built office buildings ...
- We needed **teamwork** - and **communication**
- A lot more **complex** - **data abstraction**
- We needed to write down **requirements** and **design**
- poor process and ignorance of need for process created spectacular **failures**

- We no longer had the skills and knowledge for **successful engineering**

1990s Software Projects

- In the **1990s** we build skyscrapers ...
- We needed **more** than teamwork and communication
- We needed totally **new technologies** - languages, modeling techniques, processes
- Software development **changed** completely
- New languages (Java, UML, etc.) led to **revolutionary procedures**
- Education fell **behind** ...

2000s Software Projects

- In the **2000s** we build integrated collections of continuously evolving cities …
- Algorithm design and programming is **no longer the primary focus** of software development
- **CS education** fell so far behind it is almost **obsolete**

- New applications (web, embedded) is making **quality crucial**
- Developers learn more from **training courses** than they did in college
- Not much **new development**

Idaho State University
Computer Science

Pace of Change is Exhilarating

- We have **gone from** …
  - **Log cabins** … to **houses** … to **office buildings** … to **skyscrapers** … to building the most **complicated engineering systems** in human history.
- In just **half a career**!!
- **Civil engineers** took **thousands** of years for this kind of change
  - And the most complicated civil engineering products pale in comparison to the complexity of a modern IT system
- **Electrical Engineers** took a couple of **centuries**

No way researchers, educators, or engineers could keep up!

ROAR

Theory, Practice and Education

- What have you learned in college?

How to build houses

- General software engineering courses (CS 3321) introduces a few concepts about buildings

THe way we build software has changed dramatically since the CS curriculum stabilized in 1980!!

- Very little new development is being done
- Maintenance … evolution … re-engineering … maintainability … being “agile”

Idaho State University
Computer Science

What Can You Do?

- As a **developer** ..
  - Program very **neatly**
  - **Design** to make change easy
  - Follow **processes** that make change easy
- As a **professional**...
  - **Listen** to your colleagues when they teach you things you didn't learn in college
  - Take **training classes** eagerly (in the next 20 years you should spend more time in training than you spent in college CS courses)
  - Further your **education** (MS degree)

Idaho State University
Computer Science

Goals of This Class

1. **Reliability & Testing**
2. Usability
3. Security
4. Availability
5. Scalability
6. **Maintainability**
7. Performance & Time to market

ROAR

Current Reality

- Most software development is actually maintenance
- Maintenance is no longer as boring as it was in the 1980s
- “We have as many testers as we have developers. And developers spend half their time testing. We’re more of a testing organization than we’re a software organization” – Bill Gates of Microsoft

This class teaches modern methods for the two dominant portions of software development

Are there any questions?