Course Overview:

The Product and the Principles

CSCE 740 - Lecture 1 - 08/24/2015

Today's Goals

Introduce Software Engineering

- AKA: What the heck is going on in this class?
- Go over syllabus
- What you should already know
- Clarify course expectations
- Assignments/grading
- Answer any questions

Cover the basics:

- What is software?
- What are the principles governing software engineering?

What is Software Engineering?

The development and evolution of **high-quality** (large) software systems in a **systematic, controlled, and efficient** manner.

How does this differ from CS?

Most CS courses teach you how to solve particular types of problems.

SE is about building these solutions in a rigorous manner:

- Three D's: designing, developing, and documenting
- Solutions should be reliable, functionally complete, and usable.

Typical Development Process

Concept Formation

Requirements Specification

Design

Implementation and Testing

Are we any good at building software?

Software is Expensive

The economies of **all** developed nations are dependent on software.

Software development expenditure represents a significant fraction of GNP of all developed countries.

The Scope

Software is expensive to develop and maintain.

- (1994 Standish chaos report) In the US, $220 Billion spent each year on 170,000 software projects.
  - Imagine that figure today!
- Software costs often dominate system costs.
  - The costs of software often exceed the costs of hardware.
- For systems with a long life, maintenance costs may be several times development costs.

The Problems

We don’t deliver finished products on time

- Only 16.1% of projects delivered on time and within budget, with all planned features complete as specified.
- 31.1% of projects are cancelled before delivery.
  - $81 billion spent per year on cancelled projects
- The rest were completed and operational,
  - over budget, behind schedule, poor quality

What if you mess up?

If you screw up during design, development, or testing…

- **Best Case:** Software bugs hurt profits.
- **Worst Case:** Software bugs hurt people.

Software Bugs Hurt Profits

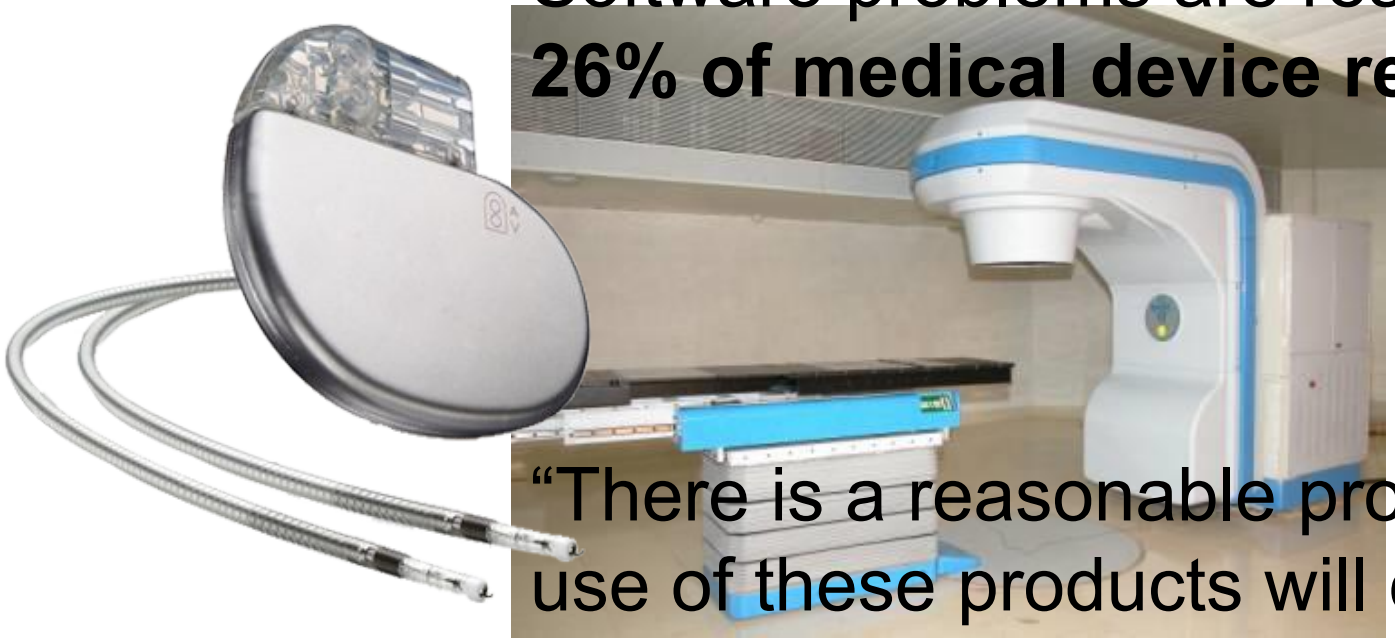
"Bugs cost the U.S. economy $60 billion annually… and testing would relieve one-third of the cost."

- NIST

"Finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it before."

- Barry Boehm **(TRW Emeritus Professor, USC)**

Software Bugs Hurt People



Software problems are responsible for **26% of medical device recalls** in 2010.

"There is a reasonable probability that use of these products will cause serious adverse health consequences or death."

- **US Food and Drug Administration**

Why Software Engineering?

Software engineering is concerned with theories, methods, and tools for professional software development.

Better engineering practices lead to **better** software, **faster** development, and **cheaper** production costs.

The Need for Disciplined Practices

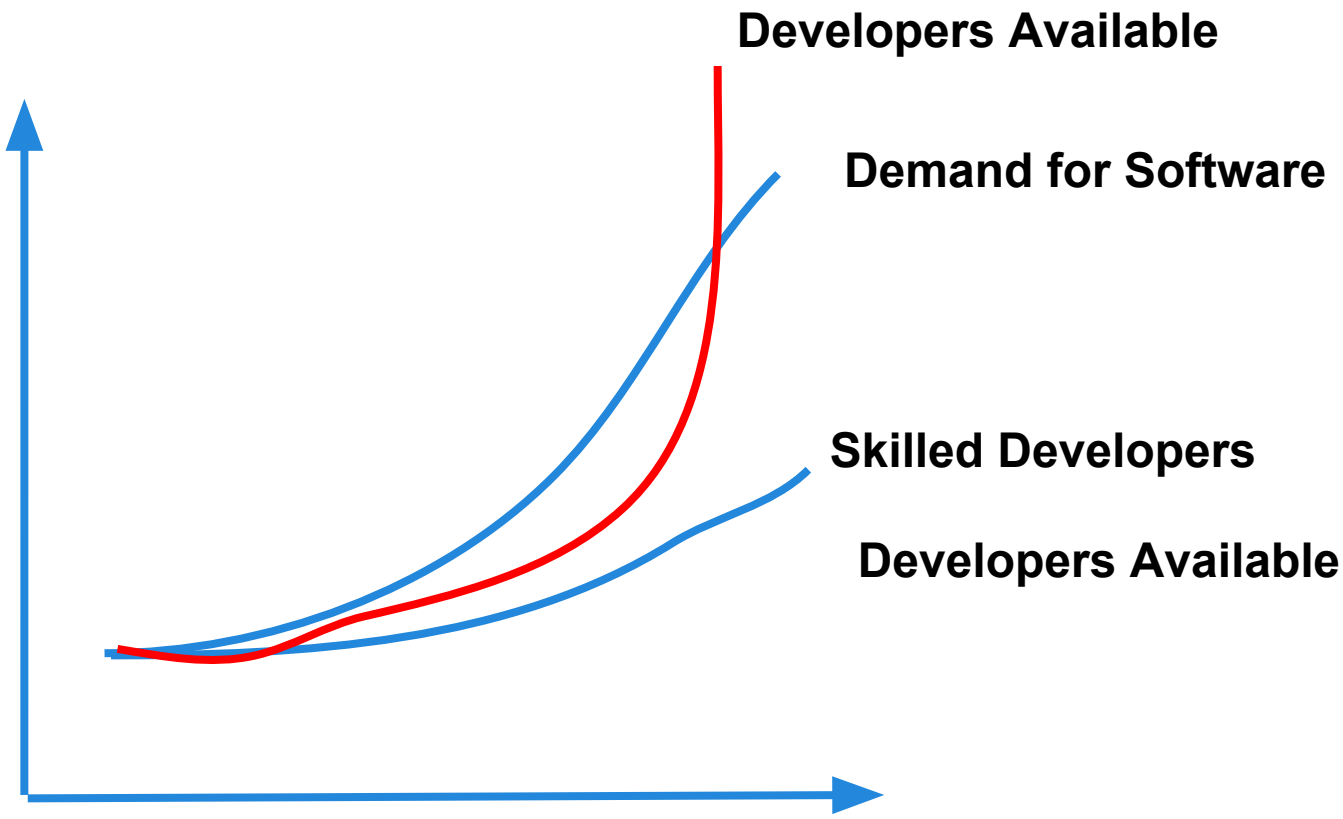
The job of software engineers is to:

- produce high-quality products
- produce them on schedule
- and do this within planned costs

In this class, we'll learn some useful skills.

(You'll want the practice)

Developers in Demand



Desired Course Outcomes

Distinguish between different development processes and design techniques.

Be familiar with requirements elicitation and be able to create a requirements document.

Be familiar with the fundamentals of black-box (functional) and white-box (structural) testing and test selection methods.

Lecture Plan (approximate)

- Introduction and Processes (2 weeks)
- Requirements Specification (3 weeks)
- Design Fundamentals (1 week)
- Software Architecture (1 week)
- Design (OO) (2 weeks)
- Implementation (1 week)
- Testing (3 weeks)
- Reliability and Maintenance (1 week)

Contact Info

- Instructor: Greg Gay (Dr, Professor, $#*%)
  - E-mail: greg@greggay.com
  - Office Hours: T/Th, 4:00-5:00 PM, 3A66 Swearingen Engineering Center
- Website:
  - http://dropbox.cse.sc.edu/course/view.php?id=351
  - (Moodle - will be used for course material and assignment submission)

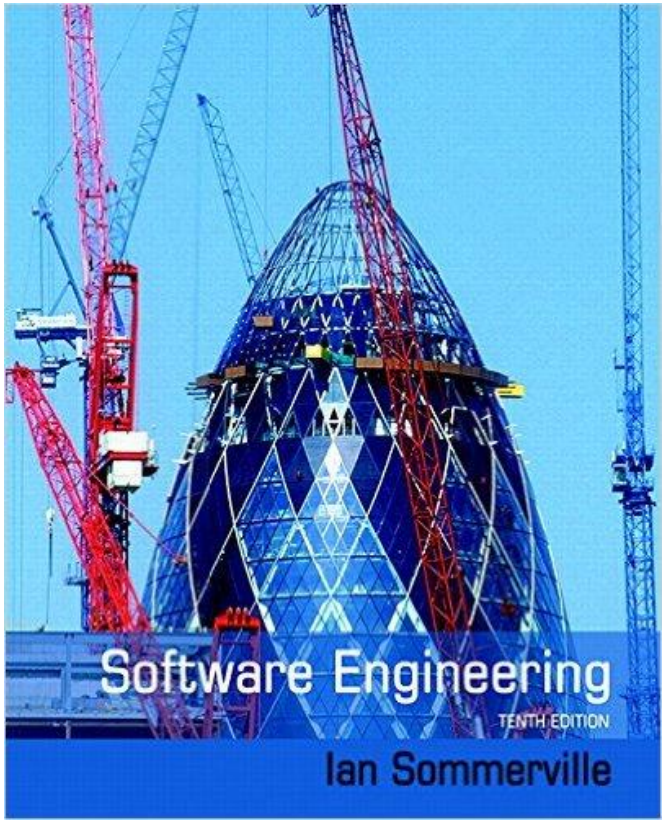
Textbook

Required:

- *Software Engineering*, Ian Sommerville.
  - Ninth **or** Tenth Edition

Recommended:

- *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Martin Fowler. Third Edition.
- *Head First Design Patterns*. Eric Freeman, Bert Bates, Kathy Sierra, Elisabeth Robson.

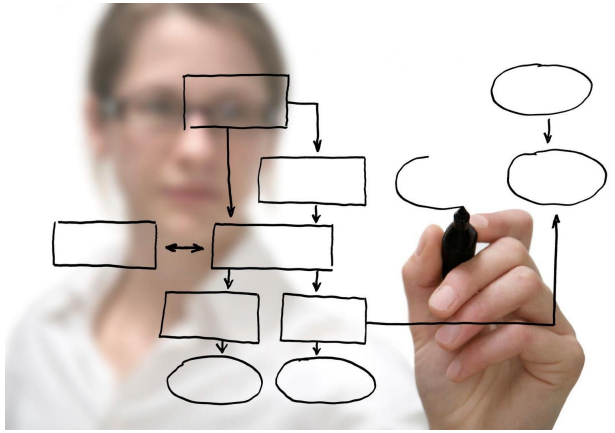


Learning Modes

Lectures/Textbook

Class Discussions

Group Project



Prerequisites

You need to be proficient in Java

- (and, ideally, C++)
- You should be able to read and write programs without additional instruction.
- This is **not** a programming language class.

You need a basic understanding of algorithms, logic, and sets.

Assignments and Grading

- Midterm and Final (30% each)
  - **Closed book**
  - Midterm: In-class, October 7th.
  - Final: December 7th.
    - Check this independently!
- Projects (35% in total)
  - Groups of three-four.
  - Frequent peer evaluations.
- Participation (5%)
  - Many in-class activities.
  - Group participation.
  - Answering questions.

Expected Workload

This class can be time consuming.

- A lot of assigned (relatively easy) reading
- Project work requires team coordination

Do not underestimate the project work.

- Good engineering is hard.
- Planning and scheduling your time is essential.
- Do NOT delay getting started.

Feedback

Problems with assignments, course questions, feedback?

- Contact me!

Problem with instructor

- Also contact me
- Contact CS front office

Other Policies

Integrity and Ethics:

The homework and programs you submit for this class must be entirely your own. If this policy is not absolutely clear, then please contact me. Any other collaboration of any type on any assignment is not permitted. It is your responsibility to protect your work from unauthorized access.

Classroom Climate:

All students are expected to behave as scholars at a leading institute of technology. This includes arriving on time, not talking during lecture (unless addressing the instructor), and not leaving the classroom before the end of lecture. Disruptive students will be warned and potentially dismissed from the classroom.

Other Policies

Make-Up and Late Homework

- The midterm and final are required.
- If a test fall on a religious holiday, they will be rescheduled.
- Make-ups for graded activities may be arranged if your absence is caused by a documented illness or personal emergency.
- Homework assignments are due at the time noted on the assignment handout. Late work is not accepted without prior approval. Any assignment turned in after the due date will be considered late and will be subject to a penalty of 10% per day, including weekends and holidays.

Other Policies

Diversity

Students in this class are expected to respectfully work with all other students, regardless of gender, race, sexuality, religion, or any other protected criteria. There is a zero-tolerance policy for any student that discriminates against other students.

Special Needs

We will provide, on a flexible and individual basis, reasonable accommodations to students that have disabilities that may affect their ability to participate in course activities or to meet course requirements Students with disabilities should contact their instructor early in the semester to discuss their individual needs.

Why is software development so %$##% hard?

software ≟ computer program

Software As Product

Software is more than the executable

- executable, installation manual, user manual, requirement documentation, design documentation, source code documentation, testing code

Development is Human-Intensive

Creation of software is labor-intensive

- Trivial manufacturing process (copying)
- But, difficult building process:
  - May involve more than just programmers:
    - Designers
    - Management
    - Testers
    - Customers
    - Legal/Government Entities

Development is Human-Intensive

Software is complex and **intangible**.

- Can't see it, touch it, hear it, or smell it.
- Hard to see mistakes until it is done.
  - (and you still might miss them)
- Components highly interconnected, hard to visualize the "structure" of a system.

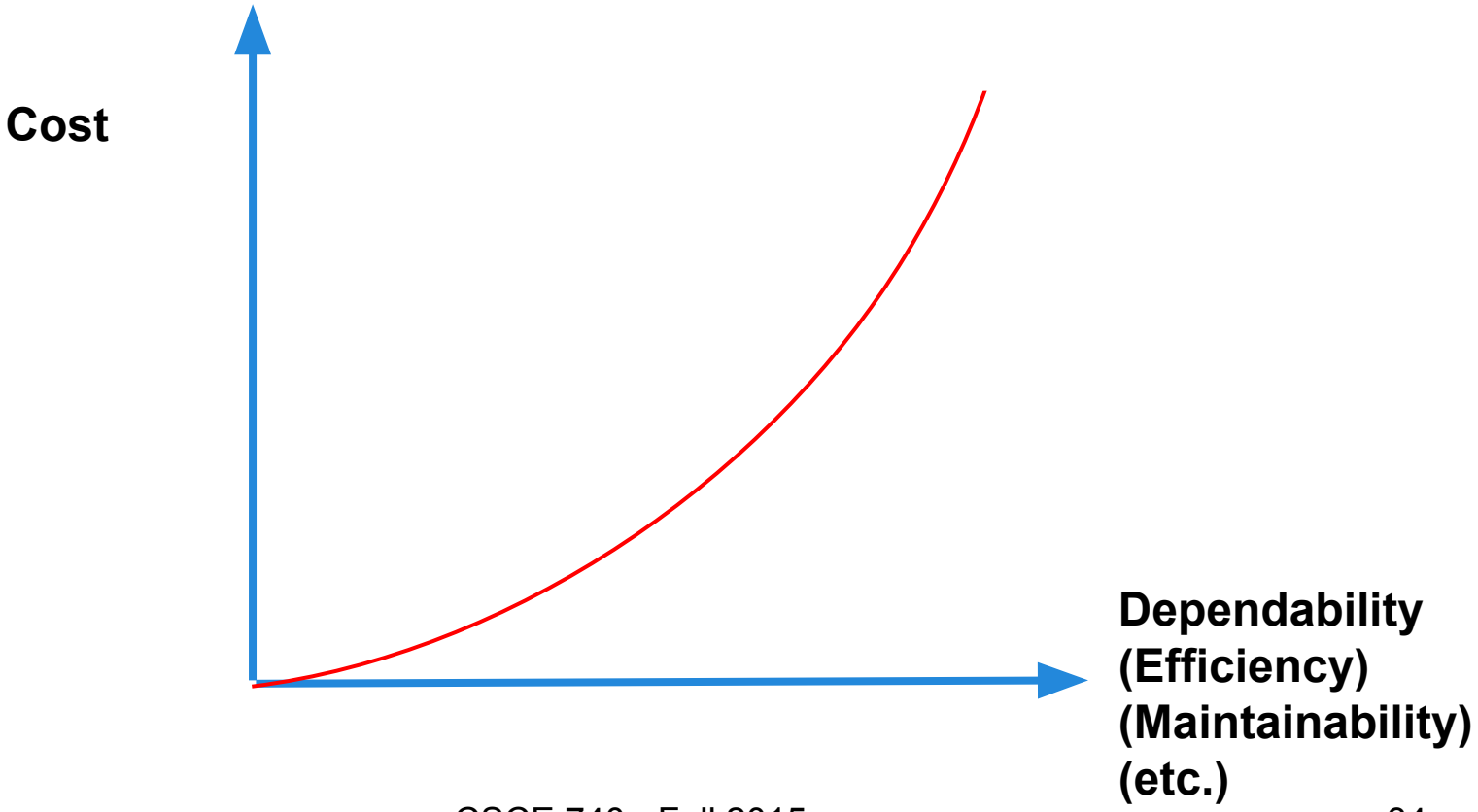
Software Product Attributes

High-quality software doesn't "just" function, but fulfills certain attributes and goals.

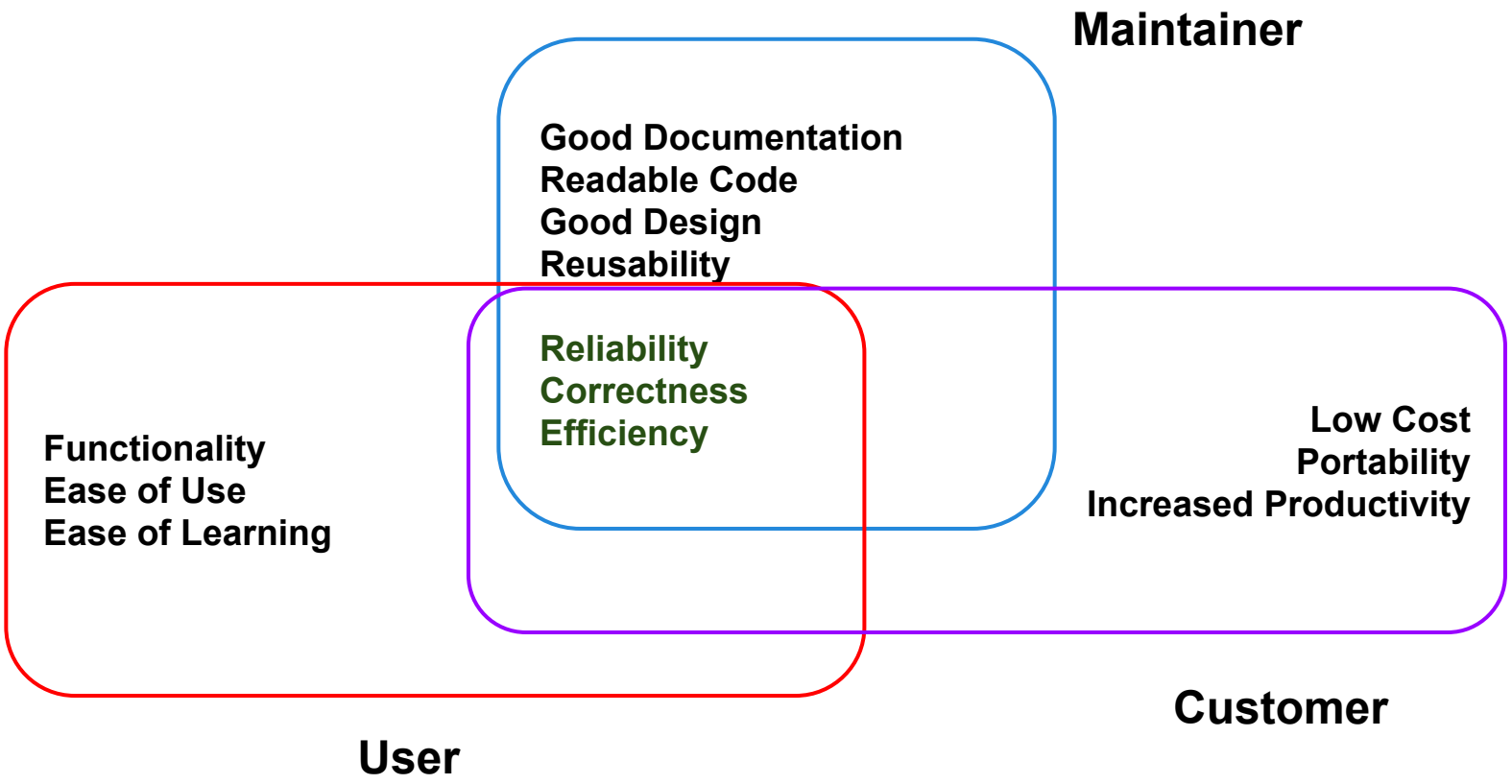
- **Maintainability:** Should be possible for the software to evolve to meet changing requirements.
- **Dependability:** Software should not cause physical or economic damage in the event of failure.
- **Efficiency:** Software should not make wasteful use of system resources.
- **Usability:** Software should have an appropriate user interface and documentation.

Expensive to Maximize Attributes

Costs rise exponentially if very high levels of an attribute are required.



Quality is in the Eyes of Beholders



Typical Development Process

Concept Formation

Requirements Specification

Design

Implementation and Testing

Typical Development Process

Concept Formati

(Ok, I had an idea, and I think it's a good one.)

Implementation ~~and Testing~~

The Trade-Off Game

Software engineering is the process of designing, constructing and maintaining ***the best software possible*** given the ***available resources***.

We are always trading off between what we want, what we need, and what we've got. As a NASA engineer put it,

- **“Better, faster, or cheaper - pick any two”**

The Role of Software Engineers

Software engineers, therefore, aren't just responsible for designing, constructing, and maintaining software.

They are the people we look to ***plan***, ***make***, and ***justify well-informed decisions*** about ***trade-offs*** throughout the development process.

Principles of SE

Software Engineering is based on a collection of fundamental principles.

These principles guide the development of all aspects of software development:

- Languages
- Methods
- Tools
- Processes
- Project Management

1: Rigor and Formality

Proper engineering requires professionalism, rigorous practices, and a formal approach.

But… software development is a creative process. Creativity often implies informality and chaos.

Do rigor and formality contradict creativity?

Rigor and Formality

Do rigor and formality contradict creativity?

Not necessarily so:

- Provide structure to the process.
- Increase skill.
- Increase the confidence in the creative results.

2: Separation of Concerns

We cannot deal with all aspects of a problem simultaneously.

So, separate issues and tasks:

- Functional design from efficiency goals.
- Requirements specification before design.
- Implement behaviors one feature at a time.

Separation of Concerns

By separating concerns, won't we miss out on optimizations?

Sure, but if we take the global view, we may fail entirely!

Separation of concerns allows separation of responsibilities.

- Separation of managerial and technical issues.
- Separation of requirements and design.

3: Modularity

A complex system must be broken down into smaller modules.

Three goals:

- Decomposability
  - Break the system into understandable modules.
- Composability
  - Construct the system from smaller pieces.
- Ease of Understanding
  - System will change. We must understand it.

Modularity Properties

- Cohesion = The degree to which modules are compatible.
- Coupling = The degree of interdependence between modules.

We want **high** cohesion and **low** coupling.

4: Abstraction

Simplify a problem by identify the important aspects, focusing on those, and pretending that the other details don't exist.

- Examples of abstraction:
  - Behavior analysis
  - Design notations
- Should have multiple abstractions of the same reality (return to those other details, focus on them).

5: Anticipation of Change

Change is inevitable. Plan for it.

- Make sure artifacts are easy to change.
- Maintain many versions of all artifacts.
- Plan for personnel turnover.
- Plan for a rapidly changing market.
- Plan for rapidly changing technology.

6: Generality

In every problem, attempt to find a more general solution.

- A general problem is often easier to solve.
- A generalized solution may be reusable.
- If you are lucky, you may be able to buy instead of build.

7: Incrementality

Move towards the goal in increments.

Separation of concerns and modularity facilitate incrementality.

Process of software development is focused on incrementality (requirements, then design, etc.).

We Have Learned

- Software engineering is concerned with the theories, methods, and tools for developing, managing, and maintaining software products.
- Software engineers do all of this while planning for and making trade-offs given the resources they have.
- Seven principles guide all aspects of software development. Keep them in mind this course (and in the future).

Next Time

- Plan your team selection.
  - The earlier, the better!

- Speaking of planning…
  - Project planning, risk management, and software development processes.
  - Reading: Sommerville, chapters 2 and 3.