CS 520

Theory and Practice of Software Engineering
Fall 2023

Course introduction

September 5, 2023

The CS 520 team

Instructor

- Heather Conboy
- Lectures: Tu/Th 10-11:15 AM will be recorded
- Office hours: TBD and by appointment
- hconboy@cs.umass.edu

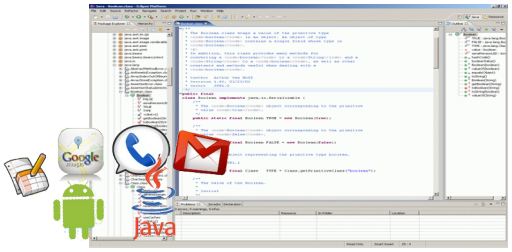
Course support

- Teaching Assistants: Mahbuba Tasmin, Dilara Tekinoglu, Pratheba Selvaraju
- 4 graders

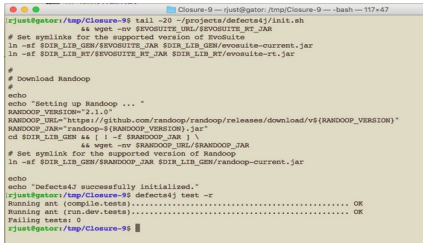
Today

- What is Software Engineering and why is it important
- Course expectations, topics, and logistics

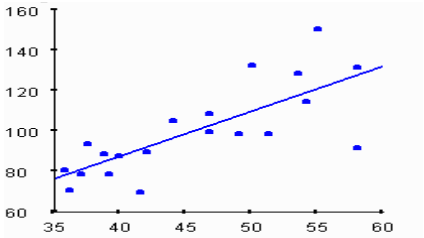
What is Software Engineering?



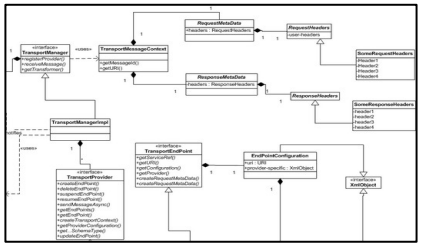
- Developing in an integrated development environment?



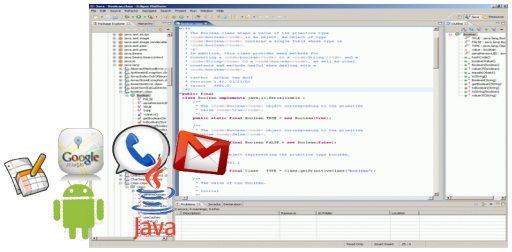
- Coding and debugging?
- Deploying and running a software system?



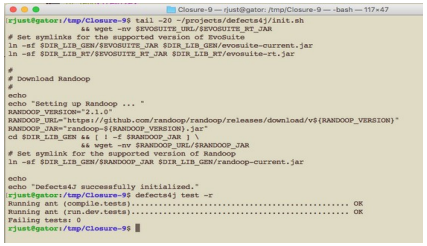
- Empirical evaluations?
- Modeling and designing?



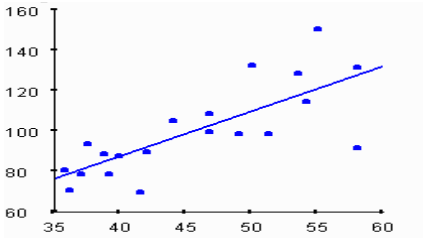
What is Software Engineering?



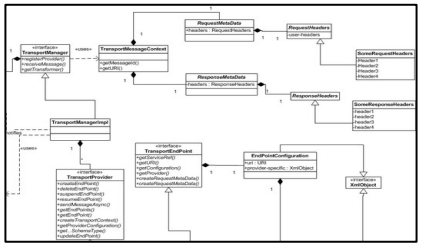
- Developing in an integrated development environment?



- Coding and debugging?
- Deploying and running a software system?



- Empirical evaluations?
- Modeling and designing?



All of the above -- much more than just writing code!

What is Software Engineering?

More than just writing code

The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

What is Software Engineering?

More than just writing code

The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

- Common Software Engineering tasks include:
  - Requirements engineering
  - Specification writing and documentation
  - Software architecture and design
  - Programming
  - Verification & Validation (e.g., manual reviewing, testing, model checking)
  - Software debugging and repair

What is Software Engineering?

More than just writing code

The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

- Common Software Engineering tasks include:
  - Requirements engineering
  - Specification writing and documentation
  - Software architecture and design
  - Programming — Just one out of many important tasks!
  - Verification & Validation (e.g., manual reviewing, testing, model checking)
  - Software debugging and repair

What is Software Engineering?

More than just writing code

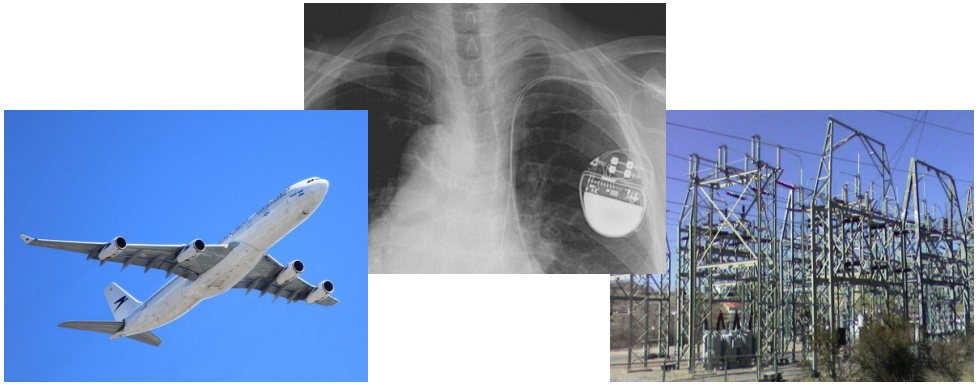
The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

- Common Software Engineering tasks include:
  - Requirements engineering
  - Specification writing and documentation
  - Software architecture and design
  - Programming
  - Validation & Verification (e.g., manual reviewing, testing, model checking)
  - Software debugging and repair

Why is Software Engineering important?

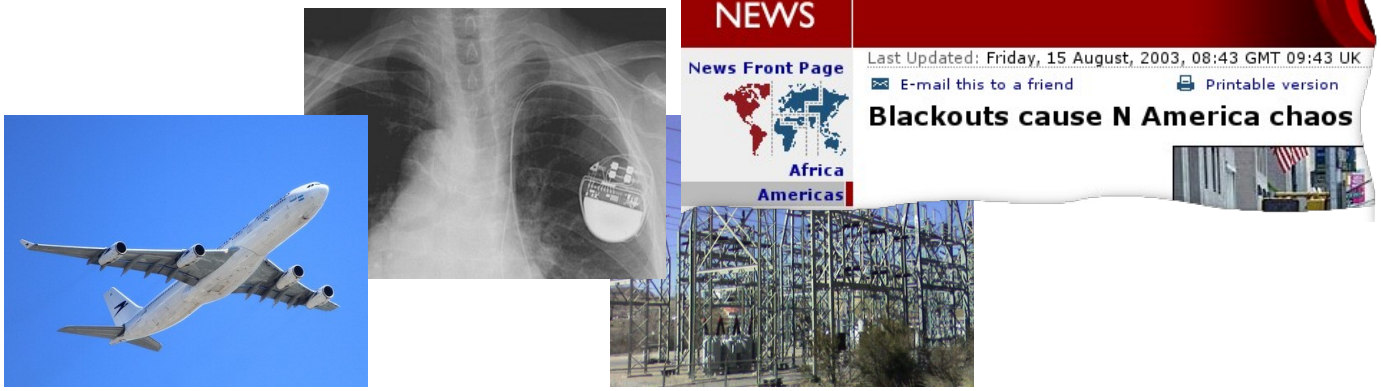
Why is Software Engineering important?

Software is everywhere...



Why is Software Engineering important?

Software is everywhere...and buggy!



How complex is software (e.g., lines of code or LoC)

Computer scientist Margaret Hamilton poses with the Apollo guidance software she and her team . Credit: Courtesy MIT Museum

Screenshot

- Debian 5.0: 324 MSLoC
  - Four times the height of the CS building
  - 5 words/LoC @ 50 wpm ⇒ 32M min ≈ 61 years

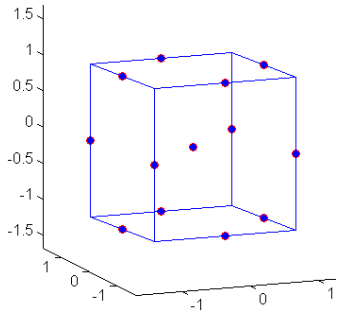
How complex is software?

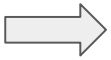
- Measures of complexity:
  - lines of code (LoC)
  - number of classes
  - number of modules
  - module interconnections and dependencies
  - time to understand
  - # of authors
  - … many more

Why is Software Engineering important?

Infrastructure is software, too!

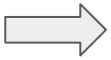
Example: Design space exploration





Infrastructure

Parallel executions of all possible configurations

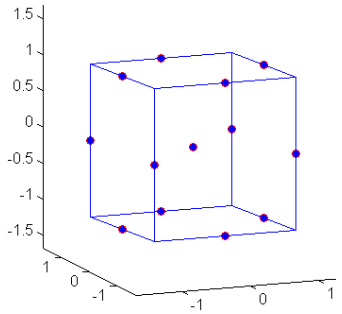


1	0.34	0.81
2	0.52	0.32
3	0.21	0.53
4	0.81	0.22
...	...	...

Why is Software Engineering important?

Infrastructure is software, too!

Example: Design space exploration



Infrastructure

Parallel executions of all possible configurations

1	0.34	0.81
2	0.52	0.32
3	0.21	0.53
4	0.81	0.22
...	...	...

- 150 configurations, 1000+ benchmarks
- 1-85 hours per execution
- 200,000+ CPU hours (~23 CPU years)

Summary: Software Engineering

What is Software Engineering?

The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

Why is it important?

- Software is everywhere and complex.
- Software defects are expensive and range from annoying to life threatening.

Common tasks include:

- Requirements engineering and documentation
- Software architecture and design
- Programming
- Validation & Verification (e.g., manual reviewing, testing, model checking)
- Software debugging and repair

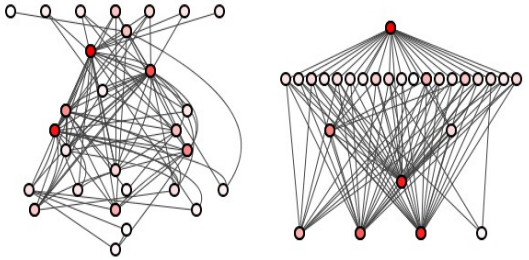
Your background and expectations

Introduction and a brief survey

- What is your background?
- What do you expect from this course?
- What are your learning goals (theory and practice)?

Course overview: the big picture

- **Software requirements, architecture, and design**
  - Requirements engineering.
  - Software modeling and UML crash course.
  - Best practices and OO design principles.
  - Architecture and design patterns.



Goal: no more spaghetti code!

Course overview: the big picture

- **Software requirements, architecture, and design**
  - Requirements engineering.
  - Software modeling and UML crash course.
  - Best practices and OO design principles.
  - Architecture and design patterns.
- **Software verification & validation (including manual reviews, testing, model checking, theorem proving) as well as debugging**
  - Learning about cutting-edge research.
  - Hands-on experience, using V&V and debugging techniques.
- **Final project**
  - Development and evaluation of a research prototype, etc.

Course overview: Rough timeline

September

- Software development processes
- Software requirements, architecture, and design
- Verification & Validation (e.g., manual reviews, testing)
- Final project: Topic selection

October/November

- High- and low-level design (e.g., principles, patterns, diagrams)
- Testing
- Debugging
- Final project: Mid-point report

November/December

- Automated program analysis (e.g., model checking)
- Reasoning about programs (e.g., theorem proving)
- Final project: Completion

Our expectations

- Programming experience
- Familiarity with an OO programming language (e.g., C++, Java, python, etc.)
- Learning to apply new SE tools
- Reading technical papers and online documentation
- Active participation in discussions and group work

Gain experience applying SE tools and techniques

- Architecture and design patterns
- Specifications as UML diagrams (e.g., class diagrams)
- Program in an OO programming language (e.g., Java, javac, java)
- Document source code (e.g., javadoc)
- xUnit testing framework (e.g., JUnit)
- Debugging techniques
- Version Control system (e.g., git)

Exposure to cutting-edge research

We will have 1 or more guest lectures on research:

- These will be held in class
- Alternatively, these will be held out of class. Videos will be available.

Assignments

- 4 homeworks [Individual or paired]
- 4 in-class exercises [Paired]
- Final project [Group]
- Participation questionnaires [Individual]

Course overview: grading

- **35%** Homeworks [Individual or Group]
- **30%** In-class exercises [Group]
- **25%** Final project [Group]
- **10%** Participation [Individual]

Logistics

- Will meet in person on Tuesday and Thursday, 10 – 11:15 AM and will also be recorded
  - Lectures, in-class exercises, project fairs
- Course schedule and policies on web site: https://people.cs.umass.edu/~hconboy/class/2023Fall/CS520/
- Course materials (e.g., slides, recorded lectures, assignments) available through Moodle:

  https://umass.moonami.com/course/view.php?id=36498
- Q&A forums for assignments via Piazza:

  https://piazza.com/umass/fall2023/CS520/home