CSE 403

Software Engineering

Winter 2023

Course introduction

Today

- The CSE 403 team
- Logistics and resources
- What is Software Engineering
- Course overview and expectations

The CSE 403 team

Instructor

- René Just (rjust@cs.washington.edu)
- Office hours: After class and by appointment

Teaching assistants/project managers

- Jesse Hu
- Ben Kushigian
- Edward Misback
- Reshabh Sharma
- Apollo Zhu

Logistics: meetings

- **Lectures**: M/W/F 12:30pm – 1:20pm (G10)
- **Team meetings**: Tue 1:30pm – 2:20pm (ECE 125)
- **Project meetings**: Thu 1:30pm – 2:20pm (G10)

This Thursday only:
Work on project proposal with your assigned partner.

Logistics: resources

- **Course website**:
  https://homes.cs.washington.edu/~rjust/courses/CSE403 (cs.uw.edu/403)
- Submission of assignments via **Canvas**:
  https://canvas.uw.edu
- Discussions on **Slack**:
  https://cse403-wi23.slack.com

Logistics: communication

Communication guidelines

- We use Slack for all **non-sensitive** communication.
- See the Slack guidelines for this course.

Resources

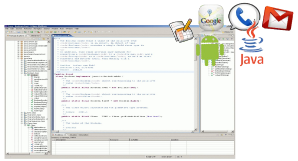
- The go-to page for this course is the course web site.
- All relevant information is on the website, or linked from it.
- Canvas for assignments and non-public materials.

Today

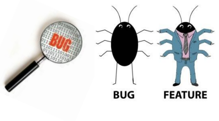
- The CSE 403 team
- Logistics and Background
- What is Software Engineering
- Course overview and expectations

What is Software Engineering?

- Developing in an IDE
  and software ecosystem?



- Debugging and maintaining a software system?



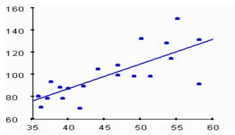
- Deploying and running
  a software system?

```
rjust@gator:/tmp/Closure-9$ tail -20 ~/projects/defects4j/init.sh
                && wget -nv $EVOSUITE_URL/$EVOSUITE_RT_JAR
# Set symlinks for the supported version of EvoSuite
ln -sf $DIR_LIB_GEN/$EVOSUITE_JAR $DIR_LIB_GEN/evosuite-current.jar
ln -sf $DIR_LIB_RT/$EVOSUITE_RT_JAR $DIR_LIB_RT/evosuite-rt.jar

#
# Download Randoop
#
echo
echo "Setting up Randoop ... "
RANDOOP_VERSION="2.1.0"
RANDOOP_URL="https://github.com/randoop/randoop/releases/download/v${RANDOOP_VERSION}"
RANDOOP_JAR="randoop-${RANDOOP_VERSION}.jar"
cd $DIR_LIB_GEN && [ ! -f $RANDOOP_JAR ] \
                && wget -nv $RANDOOP_URL/$RANDOOP_JAR
# Set symlink for the supported version of Randoop
ln -sf $DIR_LIB_GEN/$RANDOOP_JAR $DIR_LIB_GEN/randoop-current.jar

echo
echo "Defects4J successfully initialized."
rjust@gator:/tmp/Closure-9$ defects4j test -r
Running ant (compile.tests)............................................. OK
Running ant (run.dev.tests)............................................. OK
Failing tests: 0
rjust@gator:/tmp/Closure-9$
```

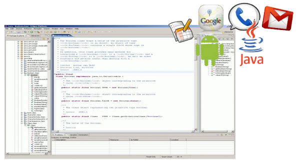
- Empirically evaluating a software system?



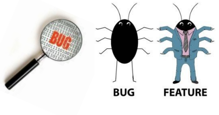
- Writing (design) docs?

What is Software Engineering?

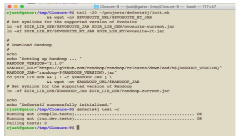
- Developing in an IDE and software ecosystem?



- Debugging and maintaining a software system?



- Deploying and running a software system?

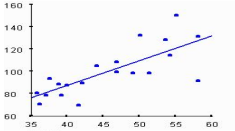


```
rjust@gator:/tmp/Closure-9$ tail -20 ~/projects/defects4j/init.sh
                && wget -nv $EVOSUITE_URL/$EVOSUITE_RT_JAR
# Set symlinks for the supported version of EvoSuite
ln -sf $DIR_LIB_GEN/$EVOSUITE_JAR $DIR_LIB_GEN/evosuite-current.jar
ln -sf $DIR_LIB_RT/$EVOSUITE_RT_JAR $DIR_LIB_RT/evosuite-rt.jar

#
# Download Randoop
#
echo
echo "Setting up Randoop ... "
RANDOOP_VERSION="2.1.0"
RANDOOP_URL="https://github.com/randoop/randoop/releases/download/v${RANDOOP_VERSION}"
RANDOOP_JAR="randoop-${RANDOOP_VERSION}.jar"
cd $DIR_LIB_GEN && [ ! -f $RANDOOP_JAR ] \
                && wget -nv $RANDOOP_URL/$RANDOOP_JAR
# Set symlink for the supported version of Randoop
ln -sf $DIR_LIB_GEN/$RANDOOP_JAR $DIR_LIB_GEN/randoop-current.jar

echo
echo "Defects4J successfully initialized."
rjust@gator:/tmp/Closure-9$ defects4j test -r
Running ant (compile.tests)............................................. OK
Running ant (run.dev.tests)............................................. OK
Failing tests: 0
rjust@gator:/tmp/Closure-9$
```

- Empirically evaluating a software system?



- Writing (design) docs?

All of the above and much more!

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!
  - Software testing and debugging
  - Maintenance and refactoring

Why is Software Engineering important?

Software is eating the world!

BBC NEWS
News Front Page
Africa
Americas
cnet
Search CNET
Reviews
News
CNN
U.S. Edition
News Video TV Opinions More...
LATEX
R Git
facebook

Last Updated: Friday, 15 August, 2003, 08:43 GMT 09:43 UK

E-mail this to a friend   Printable version

Blackouts cause N America chaos

Apple finally fixes 'gotofail' OS X

released a system software ... known as the "gotofail"

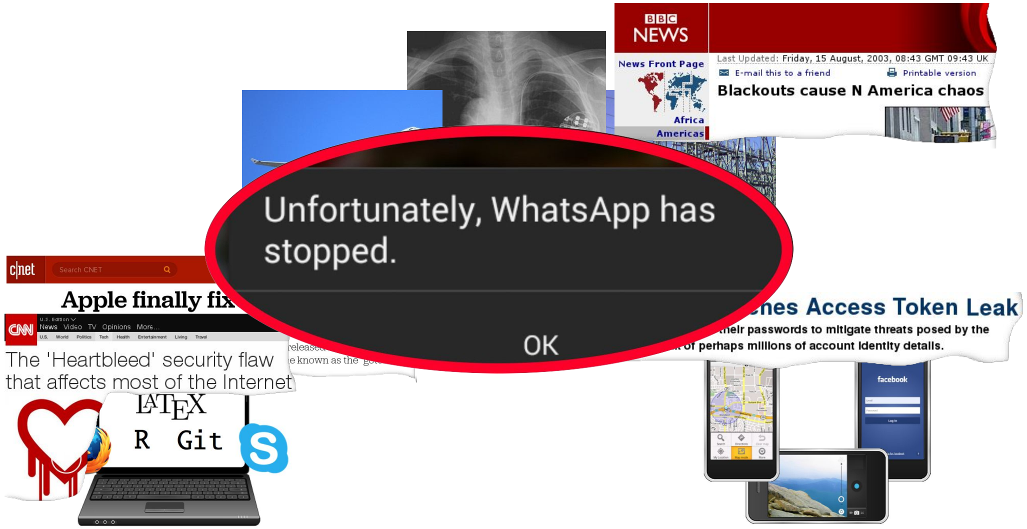
The 'Heartbleed' security flaw that affects most of the Internet

Facebook Patches Access Token Leak

Users should change their passwords to mitigate threats posed by the accidental leak of perhaps millions of account identity details.

Why is Software Engineering important?

Software is eating the world!



Last Updated: Friday, 15 August, 2003, 08:43 GMT 09:43 UK

E-mail this to a friend   Printable version

Blackouts cause N America chaos

Apple finally fix

release... e known as the "go...

The 'Heartbleed' security flaw that affects most of the Internet

...nes Access Token Leak

...heir passwords to mitigate threats posed by the ... of perhaps millions of account identity details.

Unfortunately, WhatsApp has stopped.

Summary: Software Engineering

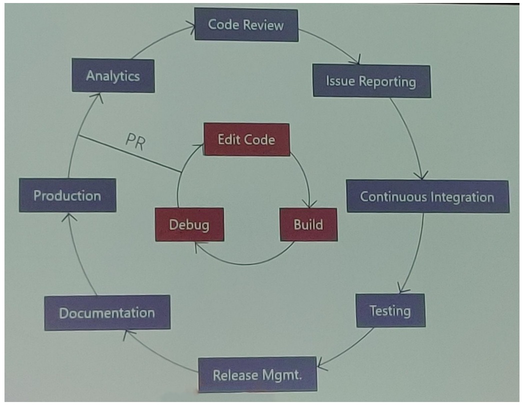
What is Software Engineering?

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

Why is it important?

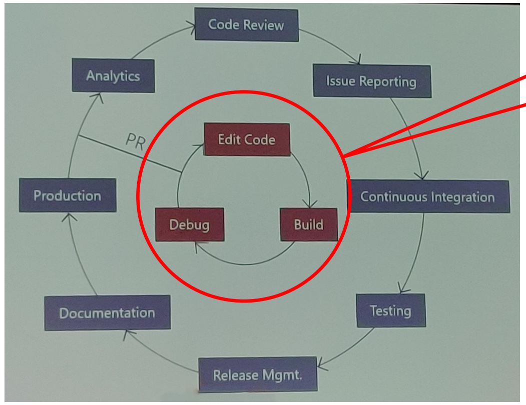
- Decomposes a complex engineering problem.
- Organizes processes and effort.
- Improves software reliability.
- Improves developer productivity.

The Role of Software Engineering in Practice



(Engineering workflow at Microsoft, Big Code summit 2019)

The Role of Software Engineering in Practice



Intro-level courses focus on the inner loop.

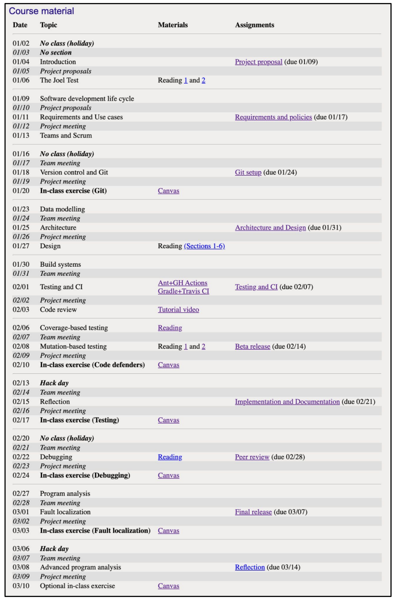
(Engineering workflow at Microsoft, Big Code summit 2019)

CSE 403 largely focuses on the outer loop.

Today

- The CSE 403 team
- Logistics and Background
- What is Software Engineering
- Course overview and expectations

Course overview: grading



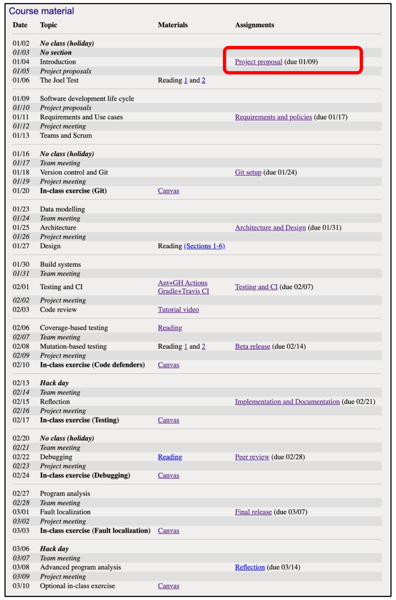
Course material

Date	Topic	Materials	Assignments
01/02	***No class (holiday)***		
01/03	***No section***		
01/04	Introduction		Project proposal (due 01/09)
01/05	*Project proposals*		
01/06	The Joel Test	Reading 1 and 2	
01/09	Software development life cycle		
01/10	*Project proposals*		
01/11	Requirements and Use cases		Requirements and policies (due 01/17)
01/12	*Project meeting*		
01/13	Teams and Scrum		
01/16	***No class (holiday)***		
01/17	*Team meeting*		
01/18	Version control and Git		Git setup (due 01/24)
01/19	*Project meeting*		
01/20	**In-class exercise (Git)**	Canvas	
01/23	Data modelling		
01/24	*Team meeting*		
01/25	Architecture		Architecture and Design (due 01/31)
01/26	*Project meeting*		
01/27	Design	Reading (Sections 1-6)	
01/30	Build systems		
01/31	*Team meeting*		
02/01	Testing and CI	Ant+GH Actions Gradle+Travis CI	Testing and CI (due 02/07)
02/02	*Project meeting*		
02/03	Code review	Tutorial video	
02/06	Coverage-based testing	Reading	
02/07	*Team meeting*		
02/08	Mutation-based testing	Reading 1 and 2	Beta release (due 02/14)
02/09	*Project meeting*		
02/10	**In-class exercise (Code defenders)**	Canvas	
02/13	***Hack day***		
02/14	*Team meeting*		
02/15	Reflection		Implementation and Documentation (due 02/21)
02/16	*Project meeting*		
02/17	**In-class exercise (Testing)**	Canvas	
02/20	***No class (holiday)***		
02/21	*Team meeting*		
02/22	Debugging	Reading	Peer review (due 02/28)
02/23	*Project meeting*		
02/24	**In-class exercise (Debugging)**	Canvas	
02/27	Program analysis		
02/28	*Team meeting*		
03/01	Fault localization		Final release (due 03/07)
03/02	*Project meeting*		
03/03	**In-class exercise (Fault localization)**	Canvas	
03/06	***Hack day***		
03/07	*Team meeting*		
03/08	Advanced program analysis		Reflection (due 03/14)
03/09	*Project meeting*		
03/10	Optional in-class exercise	Canvas	

Grading

- 55%: Course project
  - 70% project milestones
  - 30% final project review
- 35%: In-class exercises and individual assignments
- 10%: Participation
  - Engagement in project meetings
  - In-class discussions and activities (polls, small-group activities, etc.)
  - Slack contributions
- **No final exam!**

Course overview: workload



Course material

Date	Topic	Materials	Assignments
01/02	***No class (holiday)***		
01/03	***No section***		
01/04	Introduction		Project proposal (due 01/09)
01/05	*Project proposals*		
01/06	The Joel Test	Reading 1 and 2	
01/09	Software development life cycle		
01/10	*Project proposals*		
01/11	Requirements and Use cases		Requirements and policies (due 01/17)
01/12	*Project meeting*		
01/13	Teams and Scrum		
01/16	***No class (holiday)***		
01/17	*Team meeting*		
01/18	Version control and Git		Git setup (due 01/24)
01/19	*Project meeting*		
01/20	**In-class exercise (Git)**	Canvas	
01/23	Data modelling		
01/24	*Team meeting*		
01/25	Architecture		Architecture and Design (due 01/31)
01/26	*Project meeting*		
01/27	Design	Reading (Sections 1-6)	
01/30	Build systems		
01/31	*Team meeting*		
02/01	Testing and CI	Ant+GH Actions Gradle+Travis CI	Testing and CI (due 02/07)
02/02	*Project meeting*		
02/03	Code review	Tutorial video	
02/06	Coverage-based testing	Reading	
02/07	*Team meeting*		
02/08	Mutation-based testing	Reading 1 and 2	Beta release (due 02/14)
02/09	*Project meeting*		
02/10	**In-class exercise (Code defenders)**	Canvas	
02/13	***Hack day***		
02/14	*Team meeting*		
02/15	Reflection		Implementation and Documentation (due 02/21)
02/16	*Project meeting*		
02/17	**In-class exercise (Testing)**	Canvas	
02/20	***No class (holiday)***		
02/21	*Team meeting*		
02/22	Debugging	Reading	Peer review (due 02/28)
02/23	*Project meeting*		
02/24	**In-class exercise (Debugging)**	Canvas	
02/27	Program analysis		
02/28	*Team meeting*		
03/01	Fault localization		Final release (due 03/07)
03/02	*Project meeting*		
03/03	**In-class exercise (Fault localization)**	Canvas	
03/06	***Hack day***		
03/07	*Team meeting*		
03/08	Advanced program analysis		Reflection (due 03/14)
03/09	*Project meeting*		
03/10	Optional in-class exercise	Canvas	

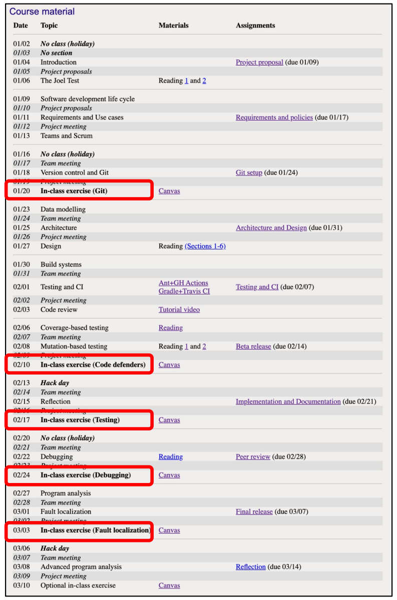
Grading

- 55%: Course project
- 35%: In-class exercises and individual assignments
- 10%: Participation
- **No final exam!**

Workload

- One project assignment each week

Course overview: workload



Course material

Date	Topic	Materials	Assignments
01/02	***No class (holiday)***		
01/03	***No section***		
01/04	Introduction		Project proposal (due 01/09)
01/05	*Project proposals*		
01/06	The Joel Test	Reading 1 and 2	
01/09	Software development life cycle		
01/10	*Project proposals*		
01/11	Requirements and Use cases		Requirements and policies (due 01/17)
01/12	*Project meeting*		
01/13	Teams and Scrum		
01/16	***No class (holiday)***		
01/17	*Team meeting*		
01/18	Version control and Git		Git setup (due 01/24)
01/19	*Project meeting*		
01/20	**In-class exercise (Git)**	Canvas	
01/23	Data modelling		
01/24	*Team meeting*		
01/25	Architecture		Architecture and Design (due 01/31)
01/26	*Project meeting*		
01/27	Design	Reading (Sections 1-6)	
01/30	Build systems		
01/31	*Team meeting*		
02/01	Testing and CI	Ant+GH Actions Gradle+Travis CI	Testing and CI (due 02/07)
02/02	*Project meeting*		
02/03	Code review	Tutorial video	
02/06	Coverage-based testing	Reading	
02/07	*Team meeting*		
02/08	Mutation-based testing	Reading 1 and 2	Beta release (due 02/14)
02/09	*Project meeting*		
02/10	**In-class exercise (Code defenders)**	Canvas	
02/13	***Hack day***		
02/14	*Team meeting*		
02/15	Reflection		Implementation and Documentation (due 02/21)
02/16	*Project meeting*		
02/17	**In-class exercise (Testing)**	Canvas	
02/20	***No class (holiday)***		
02/21	*Team meeting*		
02/22	Debugging	Reading	Peer review (due 02/28)
02/23	*Project meeting*		
02/24	**In-class exercise (Debugging)**	Canvas	
02/27	Program analysis		
02/28	*Team meeting*		
03/01	Fault localization		Final release (due 03/07)
03/02	*Project meeting*		
03/03	**In-class exercise (Fault localization)**	Canvas	
03/06	***Hack day***		
03/07	*Team meeting*		
03/08	Advanced program analysis		Reflection (due 03/14)
03/09	*Project meeting*		
03/10	Optional in-class exercise	Canvas	

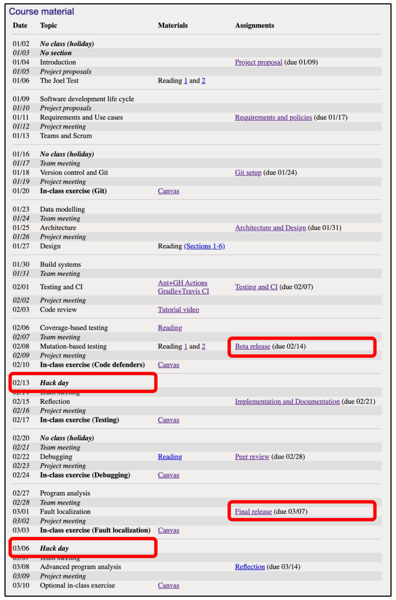
Grading

- 55%: Course project
- 35%: In-class exercises and individual assignments
- 10%: Participation
- **No final exam!**

Workload

- One project assignment each week
- 5 (+1 optional) in-class exercises

Course overview: workload



Course material

Date	Topic	Materials	Assignments
01/02	***No class (holiday)***		
01/03	***No section***		
01/04	Introduction		Project proposal (due 01/09)
01/05	*Project proposals*		
01/06	The Joel Test	Reading 1 and 2	
01/09	Software development life cycle		
01/10	*Project proposals*		
01/11	Requirements and Use cases		Requirements and policies (due 01/17)
01/12	*Project meeting*		
01/13	Teams and Scrum		
01/16	***No class (holiday)***		
01/17	*Team meeting*		
01/18	Version control and Git		Git setup (due 01/24)
01/19	*Project meeting*		
01/20	**In-class exercise (Git)**	Canvas	
01/23	Data modelling		
01/24	*Team meeting*		
01/25	Architecture		Architecture and Design (due 01/31)
01/26	*Project meeting*		
01/27	Design	Reading (Sections 1-6)	
01/30	Build systems		
01/31	*Team meeting*		
02/01	Testing and CI	Ant+GH Actions Gradle+Travis CI	Testing and CI (due 02/07)
02/02	*Project meeting*		
02/03	Code review	Tutorial video	
02/06	Coverage-based testing	Reading	
02/07	*Team meeting*		
02/08	Mutation-based testing	Reading 1 and 2	Beta release (due 02/14)
02/09	*Project meeting*		
02/10	**In-class exercise (Code defenders)**	Canvas	
02/13	***Hack day***		
02/14	*Team meeting*		
02/15	Reflection		Implementation and Documentation (due 02/21)
02/16	*Project meeting*		
02/17	**In-class exercise (Testing)**	Canvas	
02/20	***No class (holiday)***		
02/21	*Team meeting*		
02/22	Debugging	Reading	Peer review (due 02/28)
02/23	*Project meeting*		
02/24	**In-class exercise (Debugging)**	Canvas	
02/27	Program analysis		
02/28	*Team meeting*		
03/01	Fault localization		Final release (due 03/07)
03/02	*Project meeting*		
03/03	**In-class exercise (Fault localization)**	Canvas	
03/06	***Hack day***		
03/07	*Team meeting*		
03/08	Advanced program analysis		Reflection (due 03/14)
03/09	*Project meeting*		
03/10	Optional in-class exercise	Canvas	

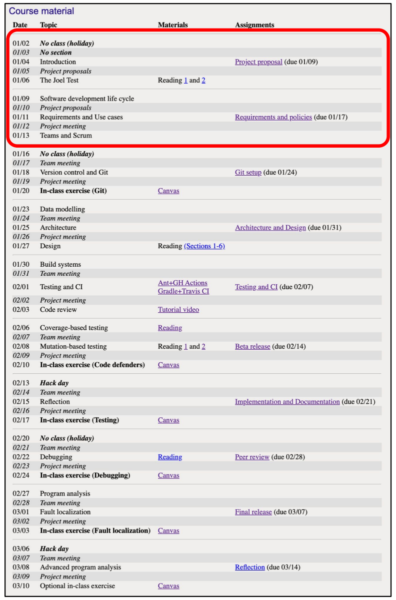
Grading

- 55%: Course project
- 35%: In-class exercises and individual assignments
- 10%: Participation
- **No final exam!**

Workload

- One project assignment each week
- 5 (+1 optional) in-class exercises
- Extra time allocated for crunch time

Course overview: topics

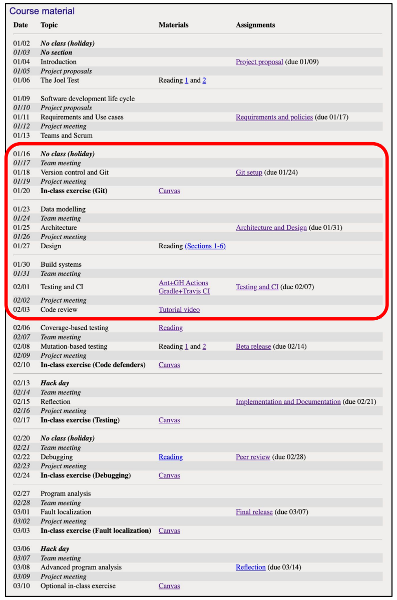


Course material

Date	Topic	Materials	Assignments
01/02	***No class (holiday)***		
01/03	***No section***		
01/04	Introduction		Project proposal (due 01/09)
01/05	*Project proposals*		
01/06	The Joel Test	Reading 1 and 2	
01/09	Software development life cycle		
01/10	*Project proposals*		
01/11	Requirements and Use cases		Requirements and policies (due 01/17)
01/12	*Project meeting*		
01/13	Teams and Scrum		
01/16	***No class (holiday)***		
01/17	*Team meeting*		
01/18	Version control and Git		Git setup (due 01/24)
01/19	*Project meeting*		
01/20	**In-class exercise (Git)**	Canvas	
01/23	Data modelling		
01/24	*Team meeting*		
01/25	Architecture		Architecture and Design (due 01/31)
01/26	*Project meeting*		
01/27	Design	Reading (Sections 1-6)	
01/30	Build systems		
01/31	*Team meeting*		
02/01	Testing and CI	Ant+GH Actions Gradle+Travis CI	Testing and CI (due 02/07)
02/02	*Project meeting*		
02/03	Code review	Tutorial video	
02/06	Coverage-based testing	Reading	
02/07	*Team meeting*		
02/08	Mutation-based testing	Reading 1 and 2	Beta release (due 02/14)
02/09	*Project meeting*		
02/10	**In-class exercise (Code defenders)**	Canvas	
02/13	***Hack day***		
02/14	*Team meeting*		
02/15	Reflection		Implementation and Documentation (due 02/21)
02/16	*Project meeting*		
02/17	**In-class exercise (Testing)**	Canvas	
02/20	***No class (holiday)***		
02/21	*Team meeting*		
02/22	Debugging	Reading	Peer review (due 02/28)
02/23	*Project meeting*		
02/24	**In-class exercise (Debugging)**	Canvas	
02/27	Program analysis		
02/28	*Team meeting*		
03/01	Fault localization		Final release (due 03/07)
03/02	*Project meeting*		
03/03	**In-class exercise (Fault localization)**	Canvas	
03/06	***Hack day***		
03/07	*Team meeting*		
03/08	Advanced program analysis		Reflection (due 03/14)
03/09	*Project meeting*		
03/10	Optional in-class exercise	Canvas	

- **Software processes, requirements, and specification**
  - Different software development processes.
  - Precise writing (requirements and specifications).

Course overview: topics

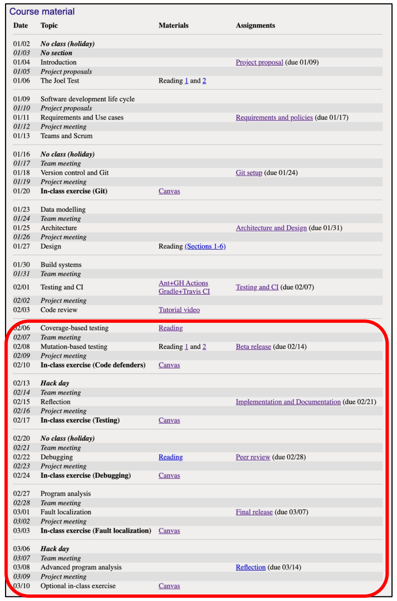


Course material

Date	Topic	Materials	Assignments
01/02	***No class (holiday)***		
01/03	***No section***		
01/04	Introduction		Project proposal (due 01/09)
01/05	*Project proposals*		
01/06	The Joel Test	Reading 1 and 2	
01/09	Software development life cycle		
01/10	*Project proposals*		
01/11	Requirements and Use cases		Requirements and policies (due 01/17)
01/12	*Project meeting*		
01/13	Teams and Scrum		
01/16	***No class (holiday)***		
01/17	*Team meeting*		
01/18	Version control and Git		Git setup (due 01/24)
01/19	*Project meeting*		
01/20	**In-class exercise (Git)**	Canvas	
01/23	Data modelling		
01/24	*Team meeting*		
01/25	Architecture		Architecture and Design (due 01/31)
01/26	*Project meeting*		
01/27	Design	Reading (Sections 1-6)	
01/30	Build systems		
01/31	*Team meeting*		
02/01	Testing and CI	Ant+GH Actions Gradle+Travis CI	Testing and CI (due 02/07)
02/02	*Project meeting*		
02/03	Code review	Tutorial video	
02/06	Coverage-based testing	Reading	
02/07	*Team meeting*		
02/08	Mutation-based testing	Reading 1 and 2	Beta release (due 02/14)
02/09	*Project meeting*		
02/10	**In-class exercise (Code defenders)**	Canvas	
02/13	***Hack day***		
02/14	*Team meeting*		
02/15	Reflection		Implementation and Documentation (due 02/21)
02/16	*Project meeting*		
02/17	**In-class exercise (Testing)**	Canvas	
02/20	***No class (holiday)***		
02/21	*Team meeting*		
02/22	Debugging	Reading	Peer review (due 02/28)
02/23	*Project meeting*		
02/24	**In-class exercise (Debugging)**	Canvas	
02/27	Program analysis		
02/28	*Team meeting*		
03/01	Fault localization		Final release (due 03/07)
03/02	*Project meeting*		
03/03	**In-class exercise (Fault localization)**	Canvas	
03/06	***Hack day***		
03/07	*Team meeting*		
03/08	Advanced program analysis		Reflection (due 03/14)
03/09	*Project meeting*		
03/10	Optional in-class exercise	Canvas	

- **Software processes, requirements, and specification**
  - Different software development processes.
  - Precise writing (requirements and specifications).
- **Software development**
  - Decompose a complex problem and build abstractions.
  - Improve your coding skills.
  - Effectively use version control, build systems, and code review.
  - Continuous integration (CI).

Course overview: topics

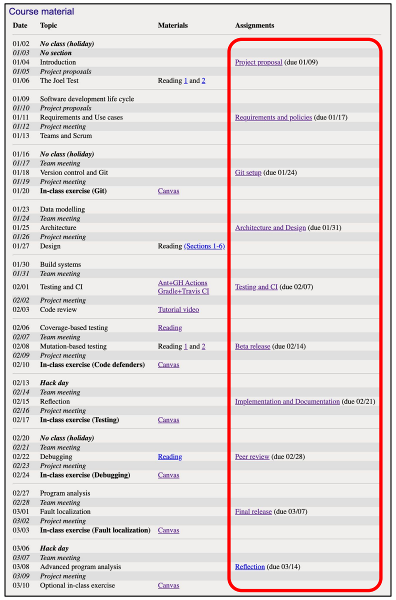


Course material

Date	Topic	Materials	Assignments
01/02	***No class (holiday)***		
01/03	***No section***		
01/04	Introduction		Project proposal (due 01/09)
01/05	*Project proposals*		
01/06	The Joel Test	Reading 1 and 2	
01/09	Software development life cycle		
01/10	*Project proposals*		
01/11	Requirements and Use cases		Requirements and policies (due 01/17)
01/12	*Project meeting*		
01/13	Teams and Scrum		
01/16	***No class (holiday)***		
01/17	*Team meeting*		
01/18	Version control and Git		Git setup (due 01/24)
01/19	*Project meeting*		
01/20	**In-class exercise (Git)**	Canvas	
01/23	Data modelling		
01/24	*Team meeting*		
01/25	Architecture		Architecture and Design (due 01/31)
01/26	*Project meeting*		
01/27	Design	Reading (Sections 1-6)	
01/30	Build systems		
01/31	*Team meeting*		
02/01	Testing and CI	Ant+GH Actions Gradle+Travis CI	Testing and CI (due 02/07)
02/02	*Project meeting*		
02/03	Code review	Tutorial video	
02/06	Coverage-based testing	Reading	
02/07	*Team meeting*		
02/08	Mutation-based testing	Reading 1 and 2	Beta release (due 02/14)
02/09	*Project meeting*		
02/10	**In-class exercise (Code defenders)**	Canvas	
02/13	***Hack day***		
02/14	*Team meeting*		
02/15	Reflection		Implementation and Documentation (due 02/21)
02/16	*Project meeting*		
02/17	**In-class exercise (Testing)**	Canvas	
02/20	***No class (holiday)***		
02/21	*Team meeting*		
02/22	Debugging	Reading	Peer review (due 02/28)
02/23	*Project meeting*		
02/24	**In-class exercise (Debugging)**	Canvas	
02/27	Program analysis		
02/28	*Team meeting*		
03/01	Fault localization		Final release (due 03/07)
03/02	*Project meeting*		
03/03	**In-class exercise (Fault localization)**	Canvas	
03/06	***Hack day***		
03/07	*Team meeting*		
03/08	Advanced program analysis		Reflection (due 03/14)
03/09	*Project meeting*		
03/10	Optional in-class exercise	Canvas	

- **Software processes, requirements, and specification**
  - Different software development processes.
  - Precise writing (requirements and specifications).
- **Software development**
  - Decompose a complex problem and build abstractions.
  - Improve your coding skills.
  - Effectively use version control, build systems, and code review.
  - Continuous integration (CI).
- **Software testing and debugging**
  - Write effective (unit) tests.
  - Hands-on experience, using testing and debugging techniques.
  - (Advanced) program analysis.

Course overview: course project



Course material

Date	Topic	Materials	Assignments
01/02	***No class (holiday)***		
01/03	***No section***		
01/04	Introduction		Project proposal (due 01/09)
01/05	*Project proposals*		
01/06	The Joel Test	Reading 1 and 2	
01/09	Software development life cycle		
01/10	*Project proposals*		
01/11	Requirements and Use cases		Requirements and policies (due 01/17)
01/12	*Project meeting*		
01/13	Teams and Scrum		
01/16	***No class (holiday)***		
01/17	*Team meeting*		
01/18	Version control and Git		Git setup (due 01/24)
01/19	*Project meeting*		
01/20	**In-class exercise (Git)**	Canvas	
01/23	Data modelling		
01/24	*Team meeting*		
01/25	Architecture		Architecture and Design (due 01/31)
01/26	*Project meeting*		
01/27	Design	Reading (Sections 1-6)	
01/30	Build systems		
01/31	*Team meeting*		
02/01	Testing and CI	Ant+GH Actions Gradle+Travis CI	Testing and CI (due 02/07)
02/02	*Project meeting*		
02/03	Code review	Tutorial video	
02/06	Coverage-based testing	Reading	
02/07	*Team meeting*		
02/08	Mutation-based testing	Reading 1 and 2	Beta release (due 02/14)
02/09	*Project meeting*		
02/10	**In-class exercise (Code defenders)**	Canvas	
02/13	***Hack day***		
02/14	*Team meeting*		
02/15	Reflection		Implementation and Documentation (due 02/21)
02/16	*Project meeting*		
02/17	**In-class exercise (Testing)**	Canvas	
02/20	***No class (holiday)***		
02/21	*Team meeting*		
02/22	Debugging	Reading	Peer review (due 02/28)
02/23	*Project meeting*		
02/24	**In-class exercise (Debugging)**	Canvas	
02/27	Program analysis		
02/28	*Team meeting*		
03/01	Fault localization		Final release (due 03/07)
03/02	*Project meeting*		
03/03	**In-class exercise (Fault localization)**	Canvas	
03/06	***Hack day***		
03/07	*Team meeting*		
03/08	Advanced program analysis		Reflection (due 03/14)
03/09	*Project meeting*		
03/10	Optional in-class exercise	Canvas	

- **Software processes, requirements, and specification**
  - Different software development processes.
  - Precise writing (requirements and specifications).
- **Software development**
  - Decompose a complex problem and build abstractions.
  - Improve your coding skills.
  - Effectively use version control, build systems, and code review.
  - Continuous integration (CI).
- **Software testing and debugging**
  - Write effective (unit) tests.
  - Hands-on experience, using testing and debugging techniques.
  - (Advanced) program analysis.
- **Course project**
  - Apply all of the above in a group project.

Course project overview

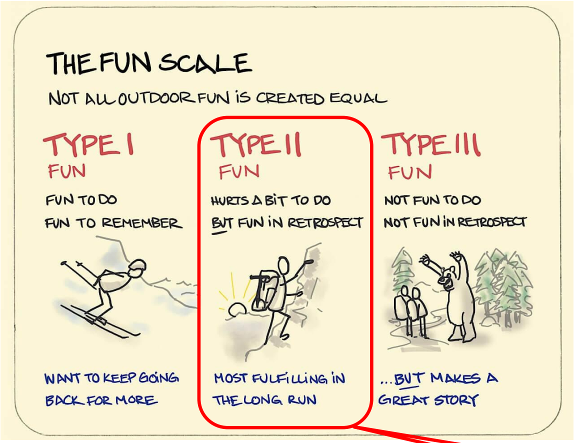
Course project proposals

Course project categories

Example categories

- Productivity and convenience apps
- Optimization problems and data science
- Gaming and making
- Extensions to open-source software
- Software Engineering research (prototypes)

CSE 403 in one picture: mostly type II fun



Sweet spot for teaching

Expectations

- Programming experience and familiarity with one programming language (Java, C++, ...).
- Active participation in discussions.
- Teamwork and communication (Slack).
- Reflecting on and improving submitted materials.

CSE 403: challenges for students

Team work

- Effective communication and coordination
- Different backgrounds, skills, and incentives

Complexity

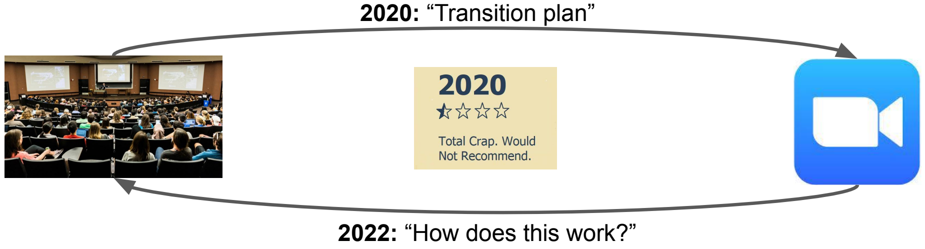
- Tooling and technology stacks
- Scale of code base

Uncertainty

- No simple check-box grading
- Trade-offs, decisions, and justifications

CSE 403: challenges for staff

In-person vs. online education



2020: “Transition plan”

2022: “How does this work?”

Enrollment

- 2020: 40 students (2 TAs)
- 2021: 85 students (5 TAs)
- 2022: 110 students (6 TAs)
- 2023: 82 students (5 TAs)

Time

- Project duration: 9 weeks
- Lecture time: 50 minutes
- Quick turnaround times

What’s next?

- *Thu: Work on project proposal (pre-assigned groups)*
- Fri: The Joel Test (or why you really should take 403)