What is Software Engineering?

Martin Kellogg

What is Software Engineering?

Today's agenda:

- Reading Quiz
- What is research? How is it similar/different from SE generally?
- Your relationship to researchers, as a developer
- What sort of problems does SE research solve
- Course Evaluation

What is Software Er

Today's agenda:

- Reading Quiz
- What is research? How is
- Your relationship to rese
- What sort of problems d
- Course Evaluation

Announcements:

- Canvas assignment for SE panel is up (you must submit at least one question before lecture on Tuesday)
- All remaining assignments are up on Canvas (incl. optional #2)
- Individual reflection due date is EoD May 5; all other project parts are due at the beginning of the final exam slot (8:30am on May 5)
- still working on practice exam; it'll be up before the weekend

What is Software Engineering?

Today's agenda:

- **Reading Quiz**
- What is research? How is it similar/different from SE generally?
- Your relationship to researchers, as a developer
- What sort of problems does SE research solve
- Course Evaluation

Reading Quiz: What is SE?

Q1: the author says that which of the following are common in software engineering research, based on the ICSE papers that she examined (list all that apply):

A. improved method or means of developing software
B. methods for analysis of correctness (testing and verification)
C. neuroimaging studies of humans reading code
D. ethnographic studies of software engineers “in the wild”

Q2: **TRUE** or **FALSE**: the key problem that the author identifies in software engineering research is that it lacks a good “layperson’s” explanation of the field (like physics or biology has)

Reading Quiz: What is SE?

Q1: the author says that which of the following are common in software engineering research, based on the ICSE papers that she examined (list all that apply):

A. improved method or means of developing software
B. methods for analysis of correctness (testing and verification)
C. neuroimaging studies of humans reading code
D. ethnographic studies of software engineers “in the wild”

Q2: **TRUE** or **FALSE**: the key problem that the author identifies in software engineering research is that it lacks a good “layperson’s” explanation of the field (like physics or biology has)

Reading Quiz: What is SE?

Q1: the author says that which of the following are common in software engineering research, based on the ICSE papers that she examined (list all that apply):

A. improved method or means of developing software
B. methods for analysis of correctness (testing and verification)
C. neuroimaging studies of humans reading code
D. ethnographic studies of software engineers “in the wild”

Q2: **TRUE** or **FALSE**: the key problem that the author identifies in software engineering research is that it lacks a good “layperson’s” explanation of the field (like physics or biology has)

What is Software Engineering?

Today's agenda:

- Reading Quiz
- **What is research? How is it similar/different from SE generally?**
- Your relationship to researchers, as a developer
- What sort of problems does SE research solve
- Course Evaluation

What is research?

What is research?

- ***Research*** is the process of innovation: creating or discovering something that has never been built/known before

What is research?

- ***Research*** is the process of innovation: creating or discovering something that has never been built/known before
- All software development is to some extent **innovative**

What is research?

- ***Research*** is the process of innovation: creating or discovering something that has never been built/known before
- All software development is to some extent **innovative**
  - the cost of copying software is zero, so any new software has **by definition** not been created before

What is research?

- ***Research*** is the process of innovation: creating or discovering something that has never been built/known before
- All software development is to some extent **innovative**
  - the cost of copying software is zero, so any new software has **by definition** not been created before
  - this contrasts with many other fields, where practitioners ("engineers" or otherwise) are **not** doing anything fundamentally novel

What is research?

- ***Research*** is the process of innovation: creating or discovering something that has never been built/known before
- All software development is to some extent **innovative**
  - the cost of copying software is zero, so any new software has **by definition** not been created before
  - this contrasts with many other fields, where practitioners (“engineers” or otherwise) are **not** doing anything fundamentally novel
    - in those field, anyone doing something new is doing “research”

What is research?

- If all software development is innovative, what distinguishes **computer science research** from just doing software engineering?

What is research?

- If all software development is innovative, what distinguishes **computer science research** from just doing software engineering?
  - the key difference is that most computer science research is **meta** in some way

What is research?

- If all software development is innovative, what distinguishes **computer science research** from just doing software engineering?
  - the key difference is that most computer science research is **meta** in some way
    - e.g., it might explore how to build **classes** of programs, like operating systems (OS) or compilers (PL)

What is research?

- If all software development is innovative, what distinguishes **computer science research** from just doing software engineering?
  - the key difference is that most computer science research is **meta** in some way
    - e.g., it might explore how to build **classes** of programs, like operating systems (OS) or compilers (PL)
    - or, it might explore **foundational notions** of what computers can and cannot do (theory)

What is research?

- If all software development is innovative, what distinguishes **computer science research** from just doing software engineering?
  - the key difference is that most computer science research is **meta** in some way
    - e.g., it might explore how to build **classes** of programs, like operating systems (OS) or compilers (PL)
    - or, it might explore **foundational notions** of what computers can and cannot do (theory)
    - or explore what computers we can **physically build** (arch)

What is research?

- So then what's meta about **software engineering** research?

What is research?

- So then what's meta about **software engineering** research?
- Software engineering researchers study:

What is research?

- So then what's meta about **software engineering** research?
- Software engineering researchers study:
  - **what** developers do
    - e.g., studies of developers, what makes them more or less productive, etc.

What is research?

- So then what's meta about **software engineering** research?
- Software engineering researchers study:
  - **what** developers do
    - e.g., studies of developers, what makes them more or less productive, etc.
  - **how** they do it
    - e.g., software architecture, design patterns

What is research?

- So then what's meta about **software engineering** research?
- Software engineering researchers study:
  - **what** developers do
    - e.g., studies of developers, what makes them more or less productive, etc.
  - **how** they do it
    - e.g., software architecture, design patterns
  - better ways to improve **software quality**
    - e.g., new kinds of testing, static analysis, etc.

What is research?

- So then what's meta about **software engineering** research?
- Software engineering researchers study:
  - **what** developers do
    - e.g., studies of developers, what makes them more or less productive, etc.
  - **how** they do it
    - e.g., software architecture, design patterns
  - better ways to improve **software quality**
    - e.g., new kinds of testing, static analysis, etc.
  - and anything else related to improving **developer productivity**

What is research?

We'll come back to this stuff later in the lecture in a bit more detail, with some examples.

- So then what's meta abou
- Software engineering rese
  - **what** developers do
    - e.g., studies of developers, what makes them more or less productive, etc.
  - **how** they do it
    - e.g., software architecture, design patterns
  - better ways to improve **software quality**
    - e.g., new kinds of testing, static analysis, etc.
  - and anything else related to improving **developer productivity**

Who does research?

Who does research?

- Most computer science research occurs in **universities**
  - including NJIT!

Who does research?

- Most computer science research occurs in **universities**
  - including NJIT!
- Most research is actually done by **students** (especially **PhD students**), working under a professor

Who does research?

- Most computer science research occurs in **universities**
  - including NJIT!
- Most research is actually done by **students** (especially **PhD students**), working under a professor
  - professor supplies high-level research vision + experience and training

Who does research?

- Most computer science research occurs in **universities**
  - including NJIT!
- Most research is actually done by **students** (especially **PhD students**), working under a professor
  - professor supplies high-level research vision + experience and training
  - student does the grunt work of writing code, gather data, etc.

Who does research?

Not just PhD students: as an **undergraduate** you can get involved in research too (I did!)

- Most computer science researc
  - including NJIT!
- Most research is actually done by **students** (especially **PhD students**), working under a professor
  - professor supplies high-level research vision + experience and training
  - student does the grunt work of writing code, gather data, etc.

Who does research?

- Most computer science research occurs in **universities**
  - including NJIT!
- Most research is actually done by **students** (especially **PhD students**), working under a professor
  - professor supplies high-level research vision + experience and training
  - student does the grunt work of writing code, gather data, etc.
- Some research is done in industry

Who does research?

- Most computer science research occurs in **universities**
  - including NJIT!
- Most research is actually done by **students** (especially **PhD students**), working under a professor
  - professor supplies high-level research vision + experience and training
  - student does the grunt work of writing code, gather data, etc.
- Some research is done in industry
  - e.g., Microsoft has MSR, AWS has ARG, etc.

Who does research?

- Most computer science research occurs in **universities**
  - including NJIT!
- Most research is actually done by **students** (especially **PhD students**), working under a professor
  - professor supplies high-level research vision + experience and training
  - student does the grunt work of writing code, gather data, etc.
- Some research is done in industry
  - e.g., Microsoft has MSR, AWS has ARG, etc.
  - sometimes developers do research by accident, too!

Who does research?

- Most computer science research occurs in **universities**
  - including NJIT!
- Most research is ac
  students), working
  - professor suppl ce and training
  - student does the grunt work of writing code, gather data, etc.
- Some research is done in industry
  - e.g., Microsoft has MSR, AWS has ARG, etc.
  - sometimes developers do research by accident, too!

However, developers rarely **publish** their research, which is important if you want it to be a part of the **total sum of human knowledge**.

Aside: should you do a PhD?

Aside: should you do a PhD?

- In my experience, most undergrads think that doing a PhD is just like “**more school**”.

Aside: should you do a PhD?

- In my experience, most undergrads think that doing a PhD is just like “**more school**”.
  - This is a long way from the truth: being a PhD student is more like a **job** that gives you a PhD when you do it long enough

Aside: should you do a PhD?

- In my experience, most undergrads think that doing a PhD is just like “**more school**”.
  - This is a long way from the truth: being a PhD student is more like a **job** that gives you a PhD when you do it long enough
    - for example, PhD students in CS are typically **paid**, although not very much (“stipends”)

Aside: should you do a PhD?

- In my experience, most undergrads think that doing a PhD is just like "**more school**".
  - This is a long way from the truth: being a PhD student is more like a **job** that gives you a PhD when you do it long enough
    - for example, PhD students in CS are typically **paid**, although not very much ("stipends")
    - the PhD student's **advisor** (a professor) is their boss

Aside: should you do a PhD?

- In my experience, most und
  like "**more school**".
  - This is a long way from
    like a **job** that gives you
    - for example, PhD students in CS are typically **paid**, although not very much ("stipends")
    - the PhD student's **advisor** (a professor) is their boss

Another misconception: in the US, you usually **do not** need a master's degree to start a PhD program!

Aside: should you do a PhD?

- In my experience, most undergrads think that doing a PhD is just like "**more school**".
  - This is a long way from the truth: being a PhD student is more like a **job** that gives you a PhD when you do it long enough
    - for example, PhD students in CS are typically **paid**, although not very much ("stipends")
    - the PhD student's **advisor** (a professor) is their boss
- For this reason, in my opinion more undergraduates should at least **consider** doing a PhD

Aside: should you do a PhD?

- In my experience, most undergrads think that doing a PhD is just like "**more school**".
  - This is a long way from the truth: being a PhD student is more like a **job** that gives you a PhD when you do it long enough
    - for example, PhD students in CS are typically **paid**, although not very much ("stipends")
    - the PhD student's **advisor** (a professor) is their boss
- For this reason, in my opinion more undergraduates should at least **consider** doing a PhD
  - it might be more affordable than you think!

Aside: should you do a PhD?

- Pros of doing a PhD:

Aside: should you do a PhD?

- Pros of doing a PhD:
  - you become a **world expert** in a topic

Aside: should you do a PhD?

- Pros of doing a PhD:
  - you become a **world expert** in a topic
  - push forth the **bounds of human knowledge**

Aside: should you do a PhD?

- Pros of doing a PhD:
  - you become a **world expert** in a topic
  - push forth the **bounds of human knowledge**
  - some jobs are **only accessible** to people with PhDs:

Aside: should you do a PhD?

- Pros of doing a PhD:
  - you become a **world expert** in a topic
  - push forth the **bounds of human knowledge**
  - some jobs are **only accessible** to people with PhDs:
    - professor
      - although you can **teach** without a PhD, you can't get tenure without one

Aside: should you do a PhD?

- Pros of doing a PhD:
  - you become a **world expert** in a topic
  - push forth the **bounds of human knowledge**
  - some jobs are **only accessible** to people with PhDs:
    - professor
      - although you can **teach** without a PhD, you can't get tenure without one
    - industrial researcher
      - e.g., static analysis designer, ML architecture developer, etc.

Aside: should you do a PhD?

- Cons of doing a PhD:

Aside: should you do a PhD?

- Cons of doing a PhD:
  - it's a **bad financial decision** (due to opportunity cost)
    - PhD students get paid, but much less than e.g., software engineer salaries

Aside: should you do a PhD?

- Cons of doing a PhD:
  - it's a **bad financial decision** (due to opportunity cost)
    - PhD students get paid, but much less than e.g., software engineer salaries
  - it takes a **long time**
    - typically 4 to 6 years, sometimes longer

Aside: should you do a PhD?

- Cons of doing a PhD:
  - it's a **bad financial decision** (due to opportunity cost)
    - PhD students get paid, but much less than e.g., software engineer salaries
  - it takes a **long time**
    - typically 4 to 6 years, sometimes longer
  - it's **mentally taxing**
    - you're working on only one thing for 4-6 years!
    - rates of mental health problems among PhD students are much higher than the general population

Aside: should you do a PhD?

- If despite those cons, you think a PhD is something you might be interested in, come **talk to me** (or another professor in the department)

Aside: should you do a PhD?

- If despite those cons, you think a PhD is something you might be interested in, come **talk to me** (or another professor in the department)

Which professor to approach? Choose a **research professor** whose work sounds interesting to you (or who you know already from class).

Aside: should you do a PhD?

- If despite those cons, you think a PhD is something you might be interested in, come **talk to me** (or another professor in the department)

to find out about a professor's work, google "their name NJIT" and read their website

Which professor to approach? Choose a **research professor** whose **work** sounds interesting to you (or who you know already from class).

Aside: should you do a PhD?

- If despite those cons, you think a PhD is something you might be interested in, come **talk to me** (or another professor in the department)

Which professor to approach? Choose a **research professor** whose work sounds interesting to you (or who you know already from class).

- at NJIT, research professors all have “professor” in the title
- teaching professors are “lecturers”

Aside: should you do a PhD?

- If despite those cons, you think a PhD is something you might be interested in, come **talk to me** (or another professor in the department)
  - high-quality PhD programs require **letters of recommendation** from professors you've worked with, so you should work with a professor :)

Aside: should you do a PhD?

- If despite those cons, you think a PhD is something you might be interested in, come **talk to me** (or another professor in the department)
  - high-quality PhD programs require **letters of recommendation** from professors you've worked with, so you should work with a professor :)
  - it's best to approach professors about joining their research group when you're a **sophomore or junior**

Aside: should you do a PhD?

- If despite those cons, you think a PhD is something you might be interested in, come **talk to me** (or another professor in the department)
  - high-quality PhD programs require **letters of recommendation** from professors you've worked with, so you should work with a professor :)
  - it's best to approach professors about joining their research group when you're a **sophomore or junior**
    - at this stage, you know enough to be useful, but you'll be around long enough that you can ramp up on a project

What is Software Engineering?

Today's agenda:

- Reading Quiz
- What is research? How is it similar/different from SE generally?
- **Your relationship to researchers, as a developer**
- What sort of problems does SE research solve
- Course Evaluation

Research to a developer

- Assuming you’re not going to do a PhD, why should you care about research in software engineering (or CS in general)?

Research to a developer

- Assuming you're not going to do a PhD, why should you care about research in software engineering (or CS in general)?
  - CS is a very **fast-changing**, young field
    - implying best practices change a lot: what we've covered in 490 might not be true anymore in 5/10/20 years

Research to a developer

- Assuming you're not going to do a PhD, why should you care about research in software engineering (or CS in general)?
  - CS is a very **fast-changing**, young field
    - implying best practices change a lot: what we've covered in 490 might not be true anymore in 5/10/20 years
  - Many developers are also working in fast-changing **domains** within CS
    - e.g., if you're working on ML, you'll want to keep up with the latest ML research

Research to a developer

- You may also have **industrial researchers** embedded in your company

Research to a developer

- You may also have **industrial researchers** embedded in your company
  - if you're at a "big tech" company, you definitely do; other places, it's a maybe

Research to a developer

- You may also have **industrial researchers** embedded in your company
  - if you're at a "big tech" company, you definitely do; other places, it's a maybe
- Especially if you're working on something **cutting edge** and you're considering trying to keep up with the latest research yourself, finding an industrial researcher in your company is a good idea
  - they can keep up with the research so you don't have to!

Keeping up with research

Keeping up with research

- **Industry-focused** academic publications
  - e.g., CACM (“Communications of the ACM”) is great for this

Keeping up with research

- **Industry-focused** academic publications
  - e.g., CACM (“Communications of the ACM”) is great for this
- Find some **technology bloggers** that you like
  - common tech blog entry: a review of a recent paper by the blogger (they read it so you don’t have to!)

Keeping up with research

- **Industry-focused** academic publications
  - e.g., CACM (“Communications of the ACM”) is great for this
- Find some **technology bloggers** that you like
  - common tech blog entry: a review of a recent paper by the blogger (they read it so you don’t have to!)
- Attend **industry conferences** (at your employer’s expense…)

Keeping up with research

- **Industry-focused** academic publications
  - e.g., CACM (“Communications of the ACM”) is great for this
- Find some **technology bloggers** that you like
  - common tech blog entry: a review of a recent paper by the blogger (they read it so you don’t have to!)
- Attend **industry conferences** (at your employer’s expense…)
- Keep up with research areas you’re particularly interested in directly, by reading (or, more likely, **skimming**) papers
  - more advice on this next

Reading papers

- I strongly recommend that you **skim** papers as a developer

Reading papers

- I strongly recommend that you **skim** papers as a developer
  - if you're going to read them at all

Reading papers

- I strongly recommend that you **skim** papers as a developer
  - if you're going to read them at all
- "**skimming**" = "reading only the **most important results**, and skipping the details of how those results were reached"

Reading papers

- I strongly recommend that you **skim** papers as a developer
  - if you're going to read them at all
- "**skimming**" = "reading only the **most important results**, and skipping the details of how those results were reached"
  - in academic papers, this usually means reading just the abstract and introduction (and maybe the conclusion)

Reading papers

- I strongly recommend that you **skim** papers as a developer
  - if you're going to read them at all
- "**skimming**" = "reading only the **most important results**, and skipping the details of how those results were reached"
  - in academic papers, this usually means reading just the abstract and introduction (and maybe the conclusion)
- Be careful, though: **not** all academic papers are **equally high-quality**!

Reading papers

- I strongly recommend that you **skim** papers as a developer
  - if you're going to read them at all
- "**skimming**" = "reading only the **most important results**, and skipping the details of how those results were reached"
  - in academic papers, this usually means reading just the abstract and introduction (and maybe the conclusion)
- Be careful, though: **not** all academic papers are **equally high-quality**!
  - as a dev, you're not trained to judge this, so relying on peer review + recommendations from e.g., tech bloggers is smart

Reading papers

- I strongly recommend that you **skim** papers as a developer
  - if you're going to read them at all
- "**skimming**" = "reading
  skipping the details of
  - in academic paper
    and introduction (
- Be careful, though: **no**
  high-quality!
  - as a dev, you're not
    review + recommendations from e.g., tech bloggers is smart

Exception: papers published by **industrial research labs** (e.g., Google Research, MSR) are almost always written in a style closer to what developers are trained to read. These are often the ones you want to focus on as a developer, anyway!

Reading papers: finding papers

Reading papers: finding papers

- In computer science, new research is usually published in **conferences** (not journals)

Reading papers: finding papers

- In computer science, new research is usually published in **conferences** (not journals)
  - conferences have shorter **publication lag**, often < 6 months

Reading papers: finding papers

- In computer science, new research is usually published in **conferences** (not journals)
  - conferences have shorter **publication lag**, often < 6 months
- If you want to get a feel for the latest research in a part of CS, you need to find the **best conferences** for that field
  - usually, fields have many conferences, of which only 2-4 are high-quality

Reading papers: finding papers

- In computer science, new research is usually published in **conferences** (not journals)
  - conferences have shorter **publication lag**, often < 6 months
- If you want to get a feel for the latest research in a part of CS, you need to find the **best conferences** for that field
  - usually, fields have many conferences, of which only 2-4 are high-quality
- To find the best conferences, you could:

Reading papers: finding papers

- In computer science, new research is usually published in **conferences** (not journals)
  - conferences have shorter **publication lag**, often < 6 months
- If you want to get a feel for the latest research in a part of CS, you need to find the **best conferences** for that field
  - usually, fields have many conferences, of which only 2-4 are high-quality
- To find the best conferences, you could:
  - ask a peer in industrial research (if you have one)

Reading papers: finding papers

- In computer science, new research is usually published in **conferences** (not journals)
  - conferences have shorter **publication lag**, often < 6 months
- If you want to get a feel for the latest research in a part of CS, you need to find the **best conferences** for that field
  - usually, fields have many conferences, of which only 2-4 are high-quality
- To find the best conferences, you could:
  - ask a peer in industrial research (if you have one)
  - use a website like csrankings.org

What is Software Engineering?

Today's agenda:

- Reading Quiz
- What is research? How is it similar/different from SE generally?
- Your relationship to researchers, as a developer
- **What sort of problems does SE research solve**
- Course Evaluation

Software Engineering Research

Software Engineering Research

- Some research areas in CS are united by **methodology**
  - e.g., most PL papers are “compilers for X”

Software Engineering Research

- Some research areas in CS are united by **methodology**
  - e.g., most PL papers are “compilers for X”
- Other areas are united by **application**
  - e.g., most OS papers are about operating systems

Software Engineering Research

- Some research areas in CS are united by **methodology**
  - e.g., most PL papers are “compilers for X”
- Other areas are united by **application**
  - e.g., most OS papers are about operating systems
- Software engineering research is united by an application: **developer productivity**

Software Engineering Research

- Some research areas in CS are united by **methodology**
  - e.g., most PL papers are “compilers for X”
- Other areas are united by **application**
  - e.g., most OS papers are about operating systems
- Software engineering research is united by an application: **developer productivity**
  - as a developer, this is an application you will probably care about

Software Engineering Research

- Some research areas in CS are united by **methodology**
  - e.g., most PL papers are “compilers for X”
- Other areas are united by **application**
  - e.g., most OS papers are about operating systems
- Software engineering research is united by an application: **developer productivity**
  - as a developer, this is an application you will probably care about
  - so SE research is particularly important to developers!

What's Hot in Software Engineering Research

- My goal in this section is to give you a **taste** of some of the most exciting research going on in the software engineering community right now
  - these slides aren't exhaustive

What's Hot in Software Engineering Research

- My goal in this section is to give you a **taste** of some of the most exciting research going on in the software engineering community right now
  - these slides aren't exhaustive
- If you **want to know more** about any of this, come by my office hours or make an appointment with me - I love to talk about this stuff!

What's Hot: Testing Self-driving Cars

Software Testing I

- Guannan Lou, Yao Deng, Xi Zheng, Mengshi Zhang, Tianyi Zhang:
  Testing of autonomous driving systems: where are we and where should we go? 31-43
- Yinlin Deng, Chenyuan Yang, Anjiang Wei, Lingming Zhang:
  Fuzzing deep-learning libraries via automated relational API inference. 44-56
- Penghui Li, Wei Meng, Kangjie Lu:
  SEDiff: scope-aware differential fuzzing to test internal function models in symbolic execution. 57-69
- Ali Reza Ibrahimzada, Yigit Varli, Dilara Tekinoglu, Reyhaneh Jabbarvand:
  Perfect is the enemy of test oracle. 70-81
- Yao Deng, Xi Zheng, Mengshi Zhang, Guannan Lou, Tianyi Zhang:
  Scenario-based test reduction and prioritization for multi-module autonomous driving systems. 82-93
- Haoxiang Tian, Yan Jiang, Guoquan Wu, Jiren Yan, Jun Wei, Wei Chen, Shuo Li, Dan Ye:
  MOSAT: finding safety violations of autonomous driving systems using multi-objective genetic algorithm. 94-106

What's Hot: Testing Self-driving Cars

Software Testing I

all from FSE '22

Guannan Lou, Yao Deng, Xi Zheng, Mengshi Zhang, Tianyi Zhang:
Testing of autonomous driving systems: where are we and where should we go? 31-43

Yinlin Deng, Chenyuan Yang, Anjiang Wei, Lingming Zhang:
Fuzzing deep-learning libraries via automated relational API inference. 44-56

Penghui Li, Wei Meng, Kangjie Lu:
SEDiff: scope-aware differential fuzzing to test internal function models in symbolic execution. 57-69

Ali Reza Ibrahimzada, Yigit Varli, Dilara Tekinoglu, Reyhaneh Jabbarvand:
Perfect is the enemy of test oracle. 70-81

Yao Deng, Xi Zheng, Mengshi Zhang, Guannan Lou, Tianyi Zhang:
Scenario-based test reduction and prioritization for multi-module autonomous driving systems. 82-93

Haoxiang Tian, Yan Jiang, Guoquan Wu, Jiren Yan, Jun Wei, Wei Chen, Shuo Li, Dan Ye:
MOSAT: finding safety violations of autonomous driving systems using multi-objective genetic algorithm. 94-106

What's Hot: Devs + the brain

ICSE '21

- Madeline Endres, Zachary Karas, Xiaosu Hu, Ioulia Kovelman, Westley Weimer: **Relating Reading, Visualization, and Coding for New Programmers: A Neuroimaging Study.** 600-612

FSE '22

- Norman Peitek, Annabelle Bergum, Maurice Rekrut, Jonas Mucke, Matthias Nadig, Chris Parnin, Janet Siegmund, Sven Apel: **Correlates of programmer efficacy and their link to experience: a combined EEG and eye-tracking study.** 120-131

What's Hot: Testing + Analysis of Android

Software Testing II

- Cong Li, Yanyan Jiang, Chang Xu:
  Cross-device record and replay for Android apps. 395-407
- Alberto Martin-Lopez, Sergio Segura, Antonio Ruiz-Cortés:
  Online testing of RESTful APIs: promises and challenges. 408-420
- Yixue Zhao, Saghar Talebipour, Kesina Baral, Hyojae Park, Leon Yee, Safwat Ali Khan, Yuriy Brun, Nenad Medvidovic, Kevin Moran:
  Avgust: automating usage-based test generation from videos of app executions. 421-433
- Jue Wang, Yanyan Jiang, Ting Su, Shaohua Li, Chang Xu, Jian Lu, Zhendong Su:
  Detecting non-crashing functional bugs in Android apps via deep-state differential analysis. 434-446
- Seulbae Kim, Taesoo Kim:
  RoboFuzz: fuzzing robotic systems over robot operating system (ROS) for finding correctness bugs. 447-458

What's Hot: Testing + Analysis of Android

FSE '22

Software Testing II

Cong Li, Yanyan Jiang, Chang Xu:
Cross-device record and replay for Android apps. 395-407

Alberto Martin-Lopez, Sergio Segura, Antonio Ruiz-Cortés:
Online testing of RESTful APIs: promises and challenges. 408-420

Yixue Zhao, Saghar Talebipour, Kesina Baral, Hyojae Park, Leon Yee, Safwat Ali Khan, Yuriy Brun, Nenad Medvidovic, Kevin Moran:
Avgust: automating usage-based test generation from videos of app executions. 421-433

Jue Wang, Yanyan Jiang, Ting Su, Shaohua Li, Chang Xu, Jian Lu, Zhendong Su:
Detecting non-crashing functional bugs in Android apps via deep-state differential analysis. 434-446

Seulbae Kim, Taesoo Kim:
RoboFuzz: fuzzing robotic systems over robot operating system (ROS) for finding correctness bugs. 447-458

What's Hot: Testing + Analysis of Android

In our own department, **Iulian Neamtiu** (and his students) work on this!

FSE '22

Software Testing II

Cong Li, Yanyan Jiang, Chang Xu:
Cross-device record and replay for Android apps. 395-407

Alberto Martin-Lopez, Sergio Segura, Antonio Ruiz-Cortés:
Online testing of RESTful APIs: promises and challenges. 408-420

Yixue Zhao, Saghar Talebipour, Kesina Baral, Hyojae Park, Leon Yee, Safwat Ali Khan, Yuriy Brun, Nenad Medvidovic, Kevin Moran:
Avgust: automating usage-based test generation from videos of app executions. 421-433

Jue Wang, Yanyan Jiang, Ting Su, Shaohua Li, Chang Xu, Jian Lu, Zhendong Su:
Detecting non-crashing functional bugs in Android apps via deep-state differential analysis. 434-446

Seulbae Kim, Taesoo Kim:
RoboFuzz: fuzzing robotic systems over robot operating system (ROS) for finding correctness bugs. 447-458

What's Hot: Understanding Code Review

All from FSE '22

Qianhua Shan, David Sukhdeo, Qianying Huang, Seth Rogers, Lawrence Chen, Elise
Using nudges to accelerate code reviews at scale. 472-482

Enrico Fregnan, Larissa Braz, Marco D'Ambros, Gül Çalikli, Alberto Bacchelli:
First come first served: the impact of file position on code review. 483-494

Aldeida Aleti

Yang Hong, Chakkrit Tantithamthavorn, Patanamon Thongtanunam, Aldeida Aleti:
CommentFinder: a simpler, faster, more accurate code review comments recommendation. 507-519

Prahar Pandya, Saurabh Tiwari:
CORMS: a GitHub and Gerrit based hybrid code reviewer recommendation approach for modern code review. 546-557

Larissa Braz, Alberto Bacchelli:
Software security during modern code review: the developer's perspective. 810-821

What's Hot: ML + SE

All from FSE '22

Machine Learning I

- Mengdi Zhang, Jun Sun:
  Adaptive fairness improvement based on causality analysis. 6-17
- Saikat Chakraborty, Toufique Ahmed, Yangruibo Ding, Premkumar T. Devanbu, Baishakhi Ray:
  NatGen: generative pre-training by "naturalizing" source code. 18-30

What's Hot: ML + SE

All from FSE '22

Machine Learning I

Mengdi Zhang, Jun Sun:
Adaptive fairness improvement based on causality analysis. 6-17

Saikat Chakraborty, Toufique Ahmed, Yangruibo Ding, Premkumar T. Devanbu, Baishakhi Ray:
NatGen: generative pre-training by "naturalizing" source code. 18-30

Machine Learning II

Junming Cao, Bihuan Chen, Chao Sun, Longjie Hu, Shuaihong Wu, Xin Peng:
Understanding performance problems in deep learning systems. 357-369

Moshi Wei, Yuchao Huang, Junjie Wang, Jiho Shin, Nima Shiri Harzevili, Song Wang:
API recommendation for machine learning libraries: how far are we? 370-381

Chaozheng Wang, Yuanhang Yang, Cuiyun Gao, Yun Peng, Hongyu Zhang, Michael R. Lyu:
No more fine-tuning? an experimental evaluation of prompt tuning in code intelligence. 382-394

What's Hot: ML + SE

All from FSE '22

Machine Learning I

Mengdi Zhang, Jun Sun:
Adaptive fairness improvement based on causality analysis. 6-17

Saikat Chakraborty, Toufique Ahmed, Yangruibo Ding, Premkumar T. Devanbu, Baishakhi Ray:
NatGen: generative pre-training by "naturalizing" source code. 18-30

Machine Learning II

ang:
-381

hael R. Lyu:
de intelligence. 382-394

Machine Learning III

Thanh Le-Cong, Hong Jin Kang, Truong Giang Nguyen, Stefanus Agus Haryono, David Lo, Xuan-Bach D. Le, Huynh Quyet Thang:
AutoPruner: transformer-based call graph pruning. 520-532

Xinwen Hu, Yu Guo, Jianjie Lu, Zheling Zhu, Chuanyi Li, Jidong Ge, Liguo Huang, Bin Luo:
Lighting up supervised learning in user review-based code localization: dataset and benchmark. 533-545

Prahar Pandya, Saurabh Tiwari:
CORMS: a GitHub and Gerrit based hybrid code reviewer recommendation approach for modern code review. 546-557

Moayad Alshangiti, Weishi Shi, Eduardo Lima, Xumin Liu, Qi Yu:
Hierarchical Bayesian multi-kernel learning for integrated classification and summarization of app reviews. 558-569

Liming Dong, He Zhang, Wei Liu, Zhiluo Weng, Hongyu Kuang:
Semi-supervised pre-processing for learning-based traceability framework on real-world software projects. 570-582

What's Hot: Synthesis + Repair

All from FSE '22
(note more ML!)

Program Repair/Synthesis

- Wonseok Oh, Hakjoo Oh:
  PyTER: effective program repair for Python type errors. 922-934
- Michael Fu, Chakkrit Tantithamthavorn, Trung Le, Van Nguyen, Dinh Q. Phung:
  VulRepair: a T5-based automated software vulnerability repair. 935-947
- Spandan Garg, Roshanak Zilouchian Moghaddam, Colin B. Clement, Neel Sundaresan, Chen Wu:
  DeepDev-PERF: a deep learning-based approach for improving software performance. 948-958
- Chunqiu Steven Xia, Lingming Zhang:
  Less training, more repairing please: revisiting automated program repair via zero-shot learning. 959-971
- Zhengkai Wu, Vu Le, Ashish Tiwari, Sumit Gulwani, Arjun Radhakrishna, Ivan Radicek, Gustavo Soares, Xinyu Wang, Zhenwen Li, Tao Xie:
  NL2Viz: natural language to visualization via constrained syntax-guided synthesis. 972-983

What’s Hot in Software Engineering Research

- I don’t have time to cover everything :(

What's Hot in Software Engineering Research

- I don't have time to cover everything :(
- If you want to learn more about some of these, consider doing the "optional reading activity" for today's class:
  - browse the FSE '22 proceedings
  - read 10 abstracts
  - make a list of words you don't know + look them up and submit the definitions (+ list of abstracts)
  - goal: see a bit of the breadth of the SE field and practice skimming research results

What is Software Engineering?

Today's agenda:

- Reading Quiz
- What is research? How is it similar/different from SE generally?
- Your relationship to researchers, as a developer
- What sort of problems does SE research solve
- **Course Evaluation**

Course Evaluation

- This is my first semester teaching undergrads, so I'd **really appreciate** your feedback (sorry you're guinea pigs!)

Course Evaluation

- This is my first semester teaching undergrads, so I'd **really appreciate** your feedback (sorry you're guinea pigs!)
- You can find it on Canvas or at https://blue.njit.edu/blue/

Course Evaluation

- This is my first semester teaching undergrads, so I'd **really appreciate** your feedback (sorry you're guinea pigs!)
- You can find it on Canvas or at https://blue.njit.edu/blue/
- I'd especially like feedback on:
  - How could I have helped to keep your group on track for the project? More deadlines? Earlier demos? Something else?
  - Should this class have a midterm next semester?
  - Is Covey.Town a good project? If not, what's wrong with it?
  - Did we cover any topics you already knew about from other classes? Topics you wish we'd gone into in more depth?