

# What is Software Engineering?

Martin Kellogg

# Reading quiz: SE + research

Q1: In the reading, “Redwine-Riddle” refers to:

- A. a model of technology maturation for software
- B. a pro-forma paper abstract
- C. a software engineering research conference
- D. a famous software system used as a subject in early SE research

Q2: When and where is this class’ final exam? (Give the date, start time, and room.)

# Reading quiz: SE + research

Q1: In the reading, “Redwine-Riddle” refers to:

- A. a model of technology maturation for software
- B. a pro-forma paper abstract
- C. a software engineering research conference
- D. a famous software system used as a subject in early SE research

Q2: When and where is this class’ final exam? (Give the date, start time, and room.)

# Reading quiz: SE + research

Q1: In the reading, “Redwine-Riddle” refers to:

- A. a model of technology maturation for software
- B. a pro-forma paper abstract
- C. a software engineering research conference
- D. a famous software system used as a subject in early SE research

Q2: When and where is this class’ final exam? (Give the date, start time, and room.)

**December 15, 8:30am, CULM LECT 2**

# Announcements

- Final demo signups open
  - See Discord for link
- Engineer panel on Monday
  - You must submit at least one question by Sunday AoE
    - but you can submit more than one
- Oral exam on Wednesday 12/10
  - Optional, replaces your IP1 grade if you choose to take it
  - You must sign up by Monday AoE
  - Exam itself in GITC 4402
  - First 2 signups didn't have names attached (my mistake)

# What is Software Engineering?

Today's agenda:

- **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

# 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 fields, 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 (CS 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 (CS 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?

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

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

# 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?

- Most computer science research
  - 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.

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

# 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 actually done by **students**, working
  - professor supplies funding 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 undergraduates like “**more school**”.
  - This is a long way from being 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 usually 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 usually 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:

- 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

# 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
  - (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 quickly, skipping the details of”
  - in academic papers: abstract, introduction (and conclusion)
- Be careful, though: **not all papers are created equal!**
  - as a dev, you’re not looking for **high-quality!**
    - 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](http://csrankings.org)

# What is Software Engineering?

Today's agenda:

- 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**

# 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 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 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 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 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: Differential Testing?

14:00 - 15:30 Testing & Analysis 1 at Grand Hall 1

Chair(s): Rohan Padhye Carnegie Mellon University

14:00 10m ★ **Mokav: Execution-driven Differential Testing with LLMs**

Talk Journal-First

Khashayar Etemadi ETH Zurich, Bardia Mohammadi Sharif University of Technology, Zhendong Su ETH Zurich, Martin Monperrus KTH Royal Institute of Technology

14:10 10m ★ **Validity-Preserving Delta Debugging via Generator Trace Reduction**

Talk Journal-First

Luyao Ren Peking University, Xing Zhang Peking University, Ziyue Hua Peking University, Yanyan Jiang Nanjing University, Xiao He ByteDance, Yingfei Xiong Peking University, Tao Xie Peking University

14:20 10m ★ **Execution-Aware Program Reduction for WebAssembly via Record and Replay**

Talk Research Papers

Doehyun Baek University of Stuttgart, Daniel Lehmann Google, Germany, Ben L. Titzer Carnegie Mellon University, Sukyoung Ryu KAIST, Michael Pradel CISPA Helmholtz Center for Information Security

🔗 Pre-print

14:30 10m ★ **DebCovDiff: Differential Testing of Coverage Measurement Tools on Real-World Projects**

Talk Research Papers

Wentao Zhang University of Illinois Urbana-Champaign, Jinghao Jia University of Illinois Urbana-Champaign, Erkai Yu University of Illinois Urbana-Champaign, Darko Marinov University of Illinois at Urbana-Champaign, Tianyin Xu University of Illinois at Urbana-Champaign

📎 Media Attached

14:40 10m ★ **DRIFT: Debug-based Trace Inference for Firmware Testing**

Talk Research Papers

Changming Liu Northeastern University, Alejandro Mera Northeastern University, Meng Xu University of Waterloo, Engin Kirda Northeastern University

14:50 10m ★ **Enhancing Differential Testing With LLMs For Testing Deep Learning Libraries**

Talk Journal-First

Meiziniu Li The Hong Kong University of Science and Technology, Dongze Li The Hong Kong University of Science and Technology, Jianmeng Liu The Hong Kong University of Science and Technology, Jialun Cao Hong Kong University of Science and Technology, Yongqiang Tian Monash University, Shihui Wang The Hong Kong University of Science and Technology

15:00 10m ★ **Unit Test Update through LLM-Driven Context Collection and Error-Type-Aware Refinement**

Talk Research Papers

Yuanhe Zhang Zhejiang University, Zhiqian Yang Zhejiang University, Shengyi Pan Zhejiang University, Zhongxin Liu Zhejiang University

15:10 10m ★ **Metamorphic Testing for Audio Content Moderation Software**

Talk Research Papers

Wenxuan Wang Hong Kong University of Science and Technology, Yongjiang Wu The Chinese University of Hong Kong, Junyuan Zhang The Chinese University of Hong Kong, Shuqing Li The Chinese University of Hong Kong, Yun Peng The Chinese University of Hong Kong, Wenting Chen City University of Hong Kong, Michael Lyu The Chinese University of Hong Kong

15:20 10m ★ **Comprehend, Imitate, and then Update: Unleashing the Power of LLMs in Test Suite Evolution**

Talk Research Papers

# What's Hot: Differential Testing?

14:00 - 15:30 Testing & Analysis 1 at Grand Hall 1

Chair(s): Rohan Padhye Carnegie Mellon University

14:00 10m ★ **Mokav: Execution-driven Differential Testing with LLMs**

Talk Journal-First

Khashayar Etemadi ETH Zurich, Bardia Mohammadi Sharif University of Technology, Zhendong Su ETH Zurich, Martin Monperrus KTH Royal Institute of Technology

14:10 10m ★ **Validity-Preserving Delta Debugging via Generator Trace Reduction**

Talk Journal-First

Luyao Ren Peking University, Xing Zhang Peking University, Ziyue Hua Peking University, Yanyan Jiang Nanjing University, Xiao He ByteDance, Yingfei Xiong Peking University, Tao Xie Peking University

14:20 10m ★ **Execution-Aware Program Reduction for WebAssembly via Record and Replay**

Talk Research Papers

Doehyun Baek University of Stuttgart, Daniel Lehmann Google, Germany, Ben L. Titzer Carnegie Mellon University, Sukyoung Ryu KAIST, Michael Pradel CISPA Helmholtz Center for Information Security

🔗 Pre-print

14:30 10m ★ **DebCovDiff: Differential Testing of Coverage Measurement Tools on Real-World Projects**

Talk Research Papers

Wentao Zhang University of Illinois Urbana-Champaign, Jinghao Jia University of Illinois Urbana-Champaign, Erkai Yu University of Illinois Urbana-Champaign, Darko Marinov University of Illinois at Urbana-Champaign, Tianyin Xu University of Illinois at Urbana-Champaign

📎 Media Attached

14:40 10m ★ **DRIFT: Debug-based Trace Inference for Firmware Testing**

Talk Research Papers

Changming Liu Northeastern University, Alejandro Mera Northeastern University, Meng Xu University of Waterloo, Engin Kirda Northeastern University

14:50 10m ★ **Enhancing Differential Testing With LLMs For Testing Deep Learning Libraries**

Talk Journal-First

Meiziniu Li The Hong Kong University of Science and Technology, Dongze Li The Hong Kong University of Science and Technology, Jianmeng Liu The Hong Kong University of Science and Technology, Jialun Cao Hong Kong University of Science and Technology, Yongqiang Tian Monash University, Shihui Wang The Hong Kong University of Science and Technology

15:00 10m ★ **Unit Test Update through LLM-Driven Context Collection and Error-Type-Aware Refinement**

Talk Research Papers

Yuanhe Zhang Zhejiang University, Zhiqian Yang Zhejiang University, Shengyi Pan Zhejiang University, Zhongxin Liu Zhejiang University

15:10 10m ★ **Metamorphic Testing for Audio Content Moderation Software**

Talk Research Papers

Wenxuan Wang Hong Kong University of Science and Technology, Yongjiang Wu The Chinese University of Hong Kong, Junyuan Zhang The Chinese University of Hong Kong, Shuqing Li The Chinese University of Hong Kong, Yun Peng The Chinese University of Hong Kong, Wenting Chen City University of Hong Kong University of Science and Technology, Michael Lyu The Chinese University of Hong Kong

15:20 10m ★ **Comprehend, Imitate, and then Update: Unleashing the Power of LLMs in Test Suite Evolution**

Talk Research Papers

# What's Hot: Differential Testing?

- Many of the topics that we covered in this class are still **actively being researched**
  - Differential testing is one example, but top conferences in SE also discuss delta debugging, static analysis, etc.

# What's Hot: Differential Testing?

- Many of the topics that we covered in this class are still **actively being researched**
  - Differential testing is one example, but top conferences in SE also discuss delta debugging, static analysis, etc.
- I have only covered the **tip of the iceberg** in this class with respect to most of these topics
  - If you want to, you can learn a lot more about any of them!

# What's Hot: Using LLMs in SE

14:00 - 15:30 **Code Generation 1 at Vista**

Chair(s): Zhongxin Liu Zhejiang University

14:00 10m ★ **QuanBench: Benchmarking Quantum Code Generation with Large Language Models**

Talk

Research Papers  
Xiaoyu Guo Kyushu University, Minggu Wang Kyushu University, Jianjun Zhao Kyushu University

14:10 10m ★ **Token Sugar: Making Source Code Sweeter for LLMs through Token-Efficient Shorthand**

Talk

Research Papers  
Zhenwu Sun Singapore Management University, Chengran Yang Singapore Management University, Singapore, Xiaoning Du Monash University, Zhou Yang University of Alberta, Alberta Mar

14:20 10m ★ **FGIT: Fault-Guided Fine-Tuning for Code Generation**

Talk

Research Papers  
Lishui Fan Zhejiang University, Zhongxin Liu Zhejiang University, Haoye Wang Hangzhou City University, Lingfeng Bao Zhejiang University, Xin Xia Zhejiang University, Shaping Li Zhej

14:30 10m ★ **Mixture-of-Experts Low-Rank Adaptation for Multilingual Code Summarization**

Talk

Research Papers  
Tianchen Yu School of Software Engineering, South China University of Technology, Li Yuan School of Software Engineering, South China University of Technology, Guangzhou, China, Hailin Hu University of Technology, Guangzhou, China

14:40 10m ★ **EfficientEdit: Accelerating Code Editing via Edit-Oriented Speculative Decoding**

Talk

Research Papers  
Peidong Wang Beihang university, Li Zhang Beihang University, Fang Liu Beihang University, Yinghao Zhu Beihang University, Wang Xu Tsinghua University, Lin Shi Beihang University, Xi Technologies, n.n.  
⌚ Pre-print

14:50 10m ★ **Bias Testing and Mitigation in LLM-based Code Generation**

Talk

Journal-First  
Dong Huang The University of Hong Kong, Jie M. Zhang King's College London, Qingwen Bu Shanghai Jiao Tong University, Xiaofei Xie Singapore Management University, Junjie Chen Ti

15:00 10m ★ **FastCoder: Accelerating Repository-level Code Generation via Efficient Retrieval and Verification**

Talk

Research Papers  
Qianhui Zhao Beihang University, Li Zhang Beihang University, Fang Liu Beihang University, Xiaoli Lian Beihang University, China, Meng Qiaoyuanhe Beihang University, Ziqian Jiao Be  
⌚ Pre-print

15:10 10m ★ **AlignCoder: Aligning Retrieval with Target Intent for Repository-Level Code Completion**

Talk

Research Papers  
Tianyue Jiang Sun Yat-sen University, Yanlin Wang Sun Yat-sen University, Yanlin Wang Sun Yat-sen University, Daya Guo , Ensheng Shi Huawei, Yuchi Ma Huawei Cloud Computing Tec

15:20 10m ★ **Effectiveness of symmetric metamorphic relations on validating the stability of code generation LLM**

Talk

Journal-First  
Chan Pak Yuen Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, China, Jacky Keung City University of Hong Kong, Zhen Yang Shandong University

# What's Hot: Using LLMs in SE

14:00 - 15:30 Code Generation 1 at Vista						
Chair(s)	Chair(s): <a href="#">Jia Li</a> Tsinghua University					
14:00	10m	★ Qu	<a href="#">Coverage-Based Harmfulness Testing for LLM Code Transformation</a>	Research Papers	Hongtao Tan Concordia University, Haibo Wang Concordia University, Diany Pressato Concordia University, Yisen Xu Software PErformance, Analysis, and Reliability (SPEAR) Group, Jacky Keung City University of Hong Kong	
14:10	10m	★ Tol	<a href="#">RealisticCodeBench: Towards More Realistic Evaluation of Large Language Models for Code Generation</a>	Research Papers	Xiao Yu Zhejiang University, Haixuan Chen Wuhan University of Technology, Lei Liu Xi'an Jiaotong University, Xing Hu Zhejiang University, Jacky Keung City University of Hong Kong	
14:20	10m	★ FG	<a href="#">Code-DiTing: Automatic Evaluation of Code Generation without References or Test Cases</a>	Research Papers	Guang Yang , Yu Zhou Nanjing University of Aeronautics and Astronautics, Xiang Chen Nantong University, Wei Zheng Northwestern Polytechnical University, Xing Hu Zhejiang University, Jacky Keung City University of Hong Kong	
14:30	10m	★ Mi	<a href="#">An Agent-based Evaluation Framework for Complex Code Generation</a>	Research Papers	Xinchen Wang Harbin Institute of Technology, Pengfei Gao ByteDance, Chao Peng ByteDance, Ruida Hu Harbin Institute of Technology, Shenzhen, Cuiyun Gao Harbin Institute of Technology, Nenghai Yu School of Cyber Security, University of Science and Technology of China	
14:40	10m	★ Eff	<a href="#">PseudoFix: Refactoring Distorted Structures in Decompiled C Pseudocode</a>	Research Papers	Gangyang Li University of Science and Technology of China, Xiuwei Shang University of Science and Technology of China, Shaoyin Cheng University of Science and Technology of China, Nenghai Yu School of Cyber Security, University of Science and Technology of China	
14:50	10m	★ Bla	<a href="#">Evaluating and Improving Framework-based Parallel Code Completion with Large Language Models</a>	Research Papers	Ke Liu , Qinglin Wang Shandong Normal University, Xiang Chen Nantong University, Guang Yang , YiGui Feng National University of Defense Technology, Gencheng Li Harbin Institute of Technology	
15:00	10m	★ Fa	<a href="#">Variational Prefix Tuning for diverse and accurate code summarization using pre-trained language models</a>	Journal-First	Junda Zhao Department of Mechanical and Industrial Engineering, University of Toronto, Yuliang Song Department of Mechanical and Industrial Engineering, University of Toronto	
15:10	10m	★ Ali	<a href="#">Effective Code Membership Inference for Code Completion Models via Adversarial Prompts</a>	Research Papers	Yuan Jiang Harbin Institute of Technology, Zehao Li Harbin Institute of Technology, Shan Huang Harbin Institute of Technology, Christoph Treude Singapore Management University	
15:20	10m	★ Eff	<a href="#">LongCodeZip: Compress Long Context for Code Language Models</a>	Research Papers	Yuling Shi Shanghai Jiao Tong University, Yichun Qian Stanford University, Hongyu Zhang Chongqing University, Beijun Shen Shanghai Jiao Tong University, Xiaodong Wang University of Texas at Dallas	Pre-print Media Attached

# What's Hot: Using LLMs in SE

14:00 - 15:30 Code Generation 1 at Vista

## Applications of LLM and Other AI Technologies

- Liuqing Chen, Yunnong Chen, Shuhong Xiao, Yaxuan Song, Lingyun Sun, Yankun Zhen, Tingting Zhou, Yanfang Chang:  
**EGFE: End-to-end Grouping of Fragmented Elements in UI Designs with Multimodal Learning.** 11:1-11:12
- Cuiying Gao, Gaozhun Huang, Heng Li, Bang Wu, Yueming Wu, Wei Yuan:  
**A Comprehensive Study of Learning-based Android Malware Detectors under Challenging Environments.** 12:1-12:13
- Antonio Mastropaoletti, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta:  
**Toward Automatically Completing GitHub Workflows.** 13:1-13:12
- Junjielong Xu, Ziang Cui, Yuan Zhao, Xu Zhang, Shilin He, Pinjia He, Liquan Li, Yu Kang, Qingwei Lin, Yingnong Dang, Saravan Rajmohan, Dongmei Zhang:  
**UniLog: Automatic Logging via LLM and In-Context Learning.** 14:1-14:12
- Yutong Wang, Cindy Rubio-González:  
**Predicting Performance and Accuracy of Mixed-Precision Programs for Precision Tuning.** 15:1-15:13
- Benjamin Steenhoek, Hongyang Gao, Wei Le:  
**Dataflow Analysis-Inspired Deep Learning for Efficient Vulnerability Detection.** 16:1-16:13
- Aidan Z. H. Yang, Claire Le Goues, Ruben Martins, Vincent J. Hellendoorn:  
**Large Language Models for Test-Free Fault Localization.** 17:1-17:12

10:10	TUM	All!	Res	Tier	Talk	Research Papers	
					Yuan Jiang	Harbin Institute of Technology, Zehao Li Harbin Institute of Technology, Shan Huang Harbin Institute of Technology, Christoph Treude Singapore Management University	
15:20	10m	★	Eff	Jou	12:20	10m	<b>LongCodeZip: Compress Long Context for Code Language Models</b>
							Research Papers Yuling Shi Shanghai Jiao Tong University, Yichun Qian Stanford University, Hongyu Zhang Chongqing University, Beijun Shen Shanghai Jiao Tong University, Xiaodong

# What's Hot: Using LLMs in SE

14:00 - 15:30 Code Generation 1 at Vista

## Applications of LLM and Other AI Technologies

- 📄 ↴ 🔍 ↵ Luqing Chen ⓘ, Yunnong Chen ⓘ, Shuhong Xiao ⓘ, Yaxuan Song ⓘ, Lingyun Sun ⓘ, Yankun Zhen ⓘ, Tingting Zhou ⓘ, Yanfang Chang ⓘ:  
**EGEE: End-to-end Grouping of Fragmented Elements in UI Designs with Multimodal Learning.** 11:1-11:12

### DNN and Language Models for Code

- 📄 ↴ 🔍 ↵ Binhang Qi ⓘ, Hailong Sun ⓘ, Hongyu Zhang ⓘ, Ruobing Zhao ⓘ, Xiang Gao ⓘ:  
**Modularizing while Training: A New Paradigm for Modularizing DNN Models.** 31:1-31:12
- 📄 ↴ 🔍 ↵ Lipeng Ma ⓘ, Weidong Yang ⓘ, Bo Xu ⓘ, Sihang Jiang ⓘ, Ben Fei ⓘ, Jiaqing Liang ⓘ, Mingjie Zhou ⓘ, Yanghua Xiao ⓘ:  
**KnowLog: Knowledge Enhanced Pre-trained Language Model for Log Understanding.** 32:1-32:13
- 📄 ↴ 🔍 ↵ Changan Niu ⓘ, Chuanyi Li ⓘ, Vincent Ng ⓘ, David Lo ⓘ, Bin Luo ⓘ:  
**FAIR: Flow Type-Aware Pre-Training of Compiler Intermediate Representations.** 33:1-33:12
- 📄 ↴ 🔍 ↵ Qi Guo ⓘ, Junming Cao ⓘ, Xiaofei Xie ⓘ, Shangqing Liu ⓘ, Xiaohong Li ⓘ, Biuhan Chen ⓘ, Xin Peng ⓘ:  
**Exploring the Potential of ChatGPT in Automated Code Refinement: An Empirical Study.** 34:1-34:13
- 📄 ↴ 🔍 ↵ Boxi Yu ⓘ, Jiayi Yao ⓘ, Qiuai Fu ⓘ, Zhiqing Zhong ⓘ, Haotian Xie ⓘ, Yaoliang Wu ⓘ, Yuchi Ma ⓘ, Pinjia He ⓘ:  
**Deep Learning or Classical Machine Learning? An Empirical Study on Log-Based Anomaly Detection.** 35:1-35:13
- 📄 ↴ 🔍 ↵ Yangruibo Ding ⓘ, Benjamin Steenhoeck ⓘ, Kexin Pei ⓘ, Gail E. Kaiser ⓘ, Wei Le ⓘ, Baishakhi Ray ⓘ:  
**TRACED: Execution-aware Pre-training for Source Code.** 36:1-36:12
- 📄 ↴ 🔍 ↵ Hao Yu ⓘ, Bo Shen ⓘ, Dezhil Ran ⓘ, Jiaxin Zhang ⓘ, Qi Zhang ⓘ, Yuchi Ma ⓘ, Guangtai Liang ⓘ, Ying Li ⓘ, Qianxiang Wang ⓘ, Tao Xie ⓘ:  
**CoderEval: A Benchmark of Pragmatic Code Generation with Generative Pre-trained Models.** 37:1-37:12
- 📄 ↴ 🔍 ↵ Shibbir Ahmed ⓘ, Hongyang Gao ⓘ, Hridesh Rajan ⓘ:  
**Inferring Data Preconditions from Deep Learning Models for Trustworthy Prediction in Deployment.** 38:1-38:13

# What's Hot: Using LLMs in SE

14:00 - 15:30 Code Generation 1 at Vista

## Applications of LLM and Other AI Technologies

- [Download](#) [View](#) [Share](#) [Copy](#) Liqing Chen, Yunnong Chen, Shuhong Xiao, Yaxuan Song, Lingyun Sun, Yankun Zhen, Tingting Zhou, Yanfang Chang:  
**EGEE: End-to-end Grouping of Fragmented Elements in UI Designs with Multimodal Learning.** 11:1-11:12
- [Download](#) [View](#) DNN and Language Models for Code
- [Download](#) [View](#) Binhang Qi, Hailong Sun, Hongyu Zhang, Ruobing Zhao, Xiang Gao:  
**Modularizing while Training: A New Paradigm for Modularizing DNN Models.** 31:1-31:12
- [Download](#) [View](#) Testing with and for AI
- [Download](#) [View](#) [Share](#) [Copy](#) Reload this page Sidong Feng, Chunyang Chen:  
**Prompting Is All You Need: Automated Android Bug Replay with Large Language Models.** 67:1-67:13
- [Download](#) [View](#) Neelofar, Aldeida Aleti:  
**Towards Reliable AI: Adequacy Metrics for Ensuring the Quality of System-level Testing of Autonomous Vehicles.** 68:1-68:12
- [Download](#) [View](#) [Share](#) [Copy](#) Yakun Zhang, Wenjie Zhang, Dezhi Ran, Qihao Zhu, Chengfeng Dou, Dan Hao, Tao Xie, Lu Zhang:  
**Learning-based Widget Matching for Migrating GUI Test Cases.** 69:1-69:13
- [Download](#) [View](#) [Share](#) [Copy](#) Yinlin Deng, Chunqiu Steven Xia, Chenyuan Yang, Shizhuo Dylan Zhang, Shujing Yang, Lingming Zhang:  
**Large Language Models are Edge-Case Generators: Crafting Unusual Programs for Fuzzing Deep Learning Libraries.** 70:1-70:13
- [Download](#) [View](#) [Share](#) [Copy](#) Yuanhong Lan, Yifei Lu, Zhong Li, Minxue Pan, Wenhua Yang, Tian Zhang, Xuandong Li:  
**Deeply Reinforcing Android GUI Testing with Deep Reinforcement Learning.** 71:1-71:13

# Advice: Large Language Models (LLMs) in SE

# Advice: Large Language Models (LLMs) in SE

- Current trends suggest that LLMs are going to be a **major part** of software engineering (and many other disciplines) going forward
  - great at writing boilerplate/tests, “fancy autocomplete”, etc.

# Advice: Large Language Models (LLMs) in SE

- Current trends suggest that LLMs are going to be a **major part** of software engineering (and many other disciplines) going forward
  - great at writing boilerplate/tests, “fancy autocomplete”, etc.
- My view: the current generation of LLMs have a lot in common with **mediocre junior engineers**:

# Advice: Large Language Models (LLMs) in SE

- Current trends suggest that LLMs are going to be a **major part** of software engineering (and many other disciplines) going forward
  - great at writing boilerplate/tests, “fancy autocomplete”, etc.
- My view: the current generation of LLMs have a lot in common with **mediocre junior engineers**:
  - can program to a specification reasonably well, but:

# Advice: Large Language Models (LLMs) in SE

- Current trends suggest that LLMs are going to be a **major part** of software engineering (and many other disciplines) going forward
  - great at writing boilerplate/tests, “fancy autocomplete”, etc.
- My view: the current generation of LLMs have a lot in common with **mediocre junior engineers**:
  - can program to a specification reasonably well, but:
    - need a lot of supervision from senior engineers to be useful

# Advice: Large Language Models (LLMs) in SE

- Current trends suggest that LLMs are going to be a **major part** of software engineering (and many other disciplines) going forward
  - great at writing boilerplate/tests, “fancy autocomplete”, etc.
- My view: the current generation of LLMs have a lot in common with **mediocre junior engineers**:
  - can program to a specification reasonably well, but:
    - need a lot of supervision from senior engineers to be useful
    - make a lot of mistakes that senior engineers wouldn’t

# Advice: Large Language Models (LLMs) in SE

- Current trends suggest that LLMs are going to be a **major part** of software engineering (and many other disciplines) going forward
  - great at writing boilerplate/tests, “fancy autocomplete”, etc.
- My view: the current generation of LLMs have a lot in common with **mediocre junior engineers**:
  - can program to a specification reasonably well, but:
    - need a lot of supervision from senior engineers to be useful
    - make a lot of mistakes that senior engineers wouldn’t
- Unlike junior engineers, though, LLMs **don’t eventually grow** into senior engineers (pipeline disruption?)

# Advice: Large Language Models (LLMs) in SE

- Current trends suggest that LLMs are going to be a **major part** of software engineering (and many other disciplines) going forward
  - great at writing boilerplate/tests, “fancy autocomplete”, etc.
- My view: the current generation of LLMs have a lot in common with **mediocre junior engineers**:
  - can program to some extent
    - need a lot of examples
    - make a lot of mistakes
- Unlike junior engineers, LLMs don't know what they don't know, so how do we validate their output?

# What's Hot: Automated Program Repair

## AI&SE Program Repair

---

-     Julian Aron Prenner , Romain Robbes :  
**Out of Context: How important is Local Context in Neural Program Repair?** 83:1-83:13
-     Hadeel Eladawy , Claire Le Goues , Yuriy Brun :  
**Automated Program Repair, What Is It Good For? Not Absolutely Nothing!** 84:1-84:13
-     Wenzhang Yang , Linhai Song , Yinxing Xue :  
**Rust-lancet: Automated Ownership-Rule-Violation Fixing with Behavior Preservation.** 85:1-85:13
-     Fairuz Nawer Meem , Justin Smith , Brittany Johnson :  
**Exploring Experiences with Automated Program Repair in Practice.** 86:1-86:11
-     Yiu Wai Chow , Luca Di Grazia , Michael Pradel :  
**PyTy: Repairing Static Type Errors in Python.** 87:1-87:13
-     Xin Zhou , Kisub Kim , Bowen Xu , DongGyun Han , David Lo :  
**Out of Sight, Out of Mind: Better Automatic Vulnerability Repair by Broadening Input Ranges and Sources.** 88:1-88:13
-     Changhua Luo , Wei Meng , Shuai Wang :  
**Strengthening Supply Chain Security with Fine-grained Safe Patch Identification.** 89:1-89:12
-     Shaoheng Cao , Minxue Pan , Yu Pei , Wenhua Yang , Tian Zhang , Linzhang Wang , Xuandong Li :  
**Comprehensive Semantic Repair of Obsolete GUI Test Scripts for Mobile Applications.** 90:1-90:13
-     Zunchen Huang , Chao Wang :  
**Constraint Based Program Repair for Persistent Memory Bugs.** 91:1-91:12

# What's Hot: Automated Program Repair

## AI&SE Program Repair

	Julian Aron Preller	Romain Robbes
	Out	<b>11:00 - 12:30 Program Repair 1 at Grand Hall 1</b>
	Had	Chair(s): Chao Peng ByteDance
	Aut	11:00 10m ★ <b>Defects4C: Benchmarking Large Language Model Repair Capability with C/C++ Bugs</b>
	Wer	Talk Research Papers
	Rus	Jian Wang Nanyang Technological University, Xiaofei Xie Singapore Management University, Qiang Hu Tianjin University, Shangqing Liu Nanjing University, Jiongchi Yu Sing
	Fair	Pre-print
	Exp	11:10 10m ★ <b>MORRepair: Teaching LLMs to Repair Code via Multi-Objective Fine-Tuning</b>
	Yiu	Talk Journal-First
	'	Boyang Yang Yanshan University, Haoye Tian Aalto University, Jiadong Ren Yanshan University, Hongyu Zhang Chongqing University, Jacques Klein University of Luxemb
	PyT	Link to publication DOI Pre-print
	Xin	11:20 10m ★ <b>Test-based Patch Clustering for Automatically-Generated Patches Assessment</b>
	Out	Talk Journal-First
	Cha	Matias Martinez Universitat Politècnica de Catalunya (UPC), Maria Kechagia National and Kapodistrian University of Athens, Anjana Perera Oracle Labs, Australia, Justyna F
	Stre	11:30 10m ★ <b>Hierarchical Knowledge Injection for Improving LLM-based Program Repair</b>
	Sha	Talk Research Papers
	Con	Ramtin Ehsani Drexel University, Esteban Parra Rodriguez Belmont University, Sonia Haiduc Florida State University, Preetha Chatterjee Drexel University, USA
	Zun	11:40 10m ★ <b>Characterizing Multi-Hunk Patches: Divergence, Proximity, and LLM Repair Challenges</b>
	Con	Talk Research Papers
		Noor Nashid University of British Columbia, Daniel Ding University of British Columbia, Keheliya Gallaba Centre for Software Excellence, Ahmed E. Hassan Queen's Univers
		Pre-print
		11:50 10m ★ <b>Reinforcement Learning for Mutation Operator Selection in Automated Program Repair</b>
		Talk Journal-First
		Carol Hanna University College London, Aymeric Blot University of Rennes, IRISA / INRIA, Justyna Petke University College London

# What's Hot: Automated Program Repair

- Basic *automated program repair* (APR) idea:
  - given a test suite with one failing test and the program source
  - make some change so that the test passes

# What's Hot: Automated Program Repair

- Basic *automated program repair* (APR) idea:
  - given a test suite with one failing test and the program source
  - make some change so that the test passes
- Modern APR revival is based on **promise of LLMs**

# What's Hot: Automated Program Repair

- Basic *automated program repair* (APR) idea:
  - given a test suite with one failing test and the program source
  - make some change so that the test passes
- Modern APR revival is based on **promise of LLMs**
- But we've been here before...
  - back in 2012, we had APR systems claiming ~50% repair rate

# What's Hot: Automated Program Repair

- Basic *automated program repair* (APR) idea:
  - given a test suite with one failing test and the program source
  - make some change so that the test passes
- Modern APR revival is based on **promise of LLMs**
- But we've been here before...
  - back in 2012, we had APR systems claiming ~50% repair rate
  - this was mostly hype + bad measurements
    - ask me for more details...
  - maybe this time will be different?

# Wrapup

- If you remember one thing from this class:
  - software engineering is all about **trade-offs!**

# Wrapup

- If you remember one thing from this class:
  - software engineering is all about **trade-offs!**
- I hope you enjoyed CS 490 this semester
  - it's not over yet, but this is my last lecture

# Wrapup

- If you remember one thing from this class:
  - software engineering is all about **trade-offs!**
- I hope you enjoyed CS 490 this semester
  - it's not over yet, but this is my last lecture
- Remaining class time: course evaluations
  - I do read them!
  - find it at [canvas.njit.edu](https://canvas.njit.edu) or [blue.njit.edu/blue](https://blue.njit.edu/blue)