

A binary star is two stars orbiting each other,
gravitationally bound together - GPT4o



LLM

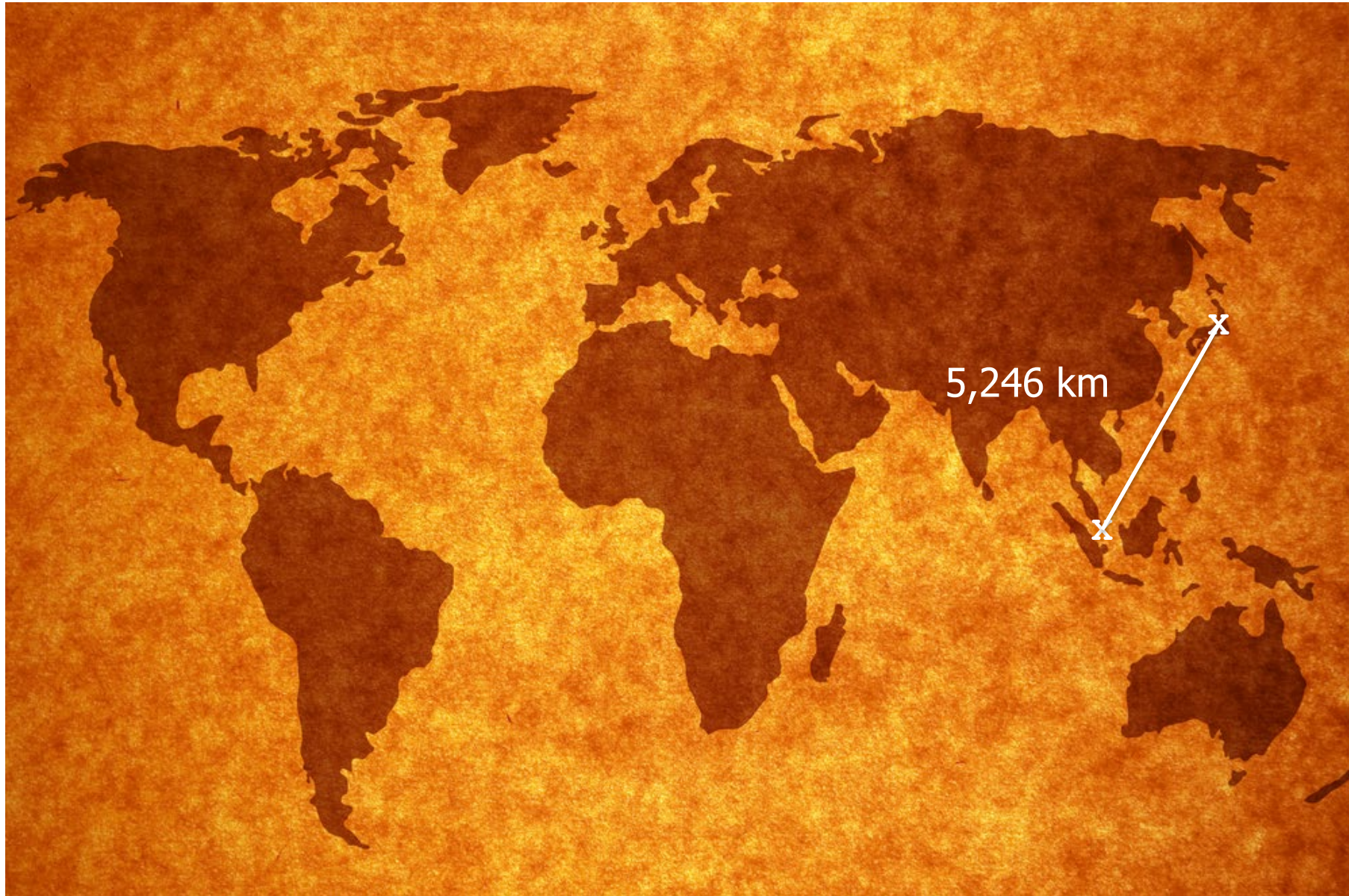
**Software
Reliability**

Software Reliability in the Era of Large Language Models: A Dual Perspective

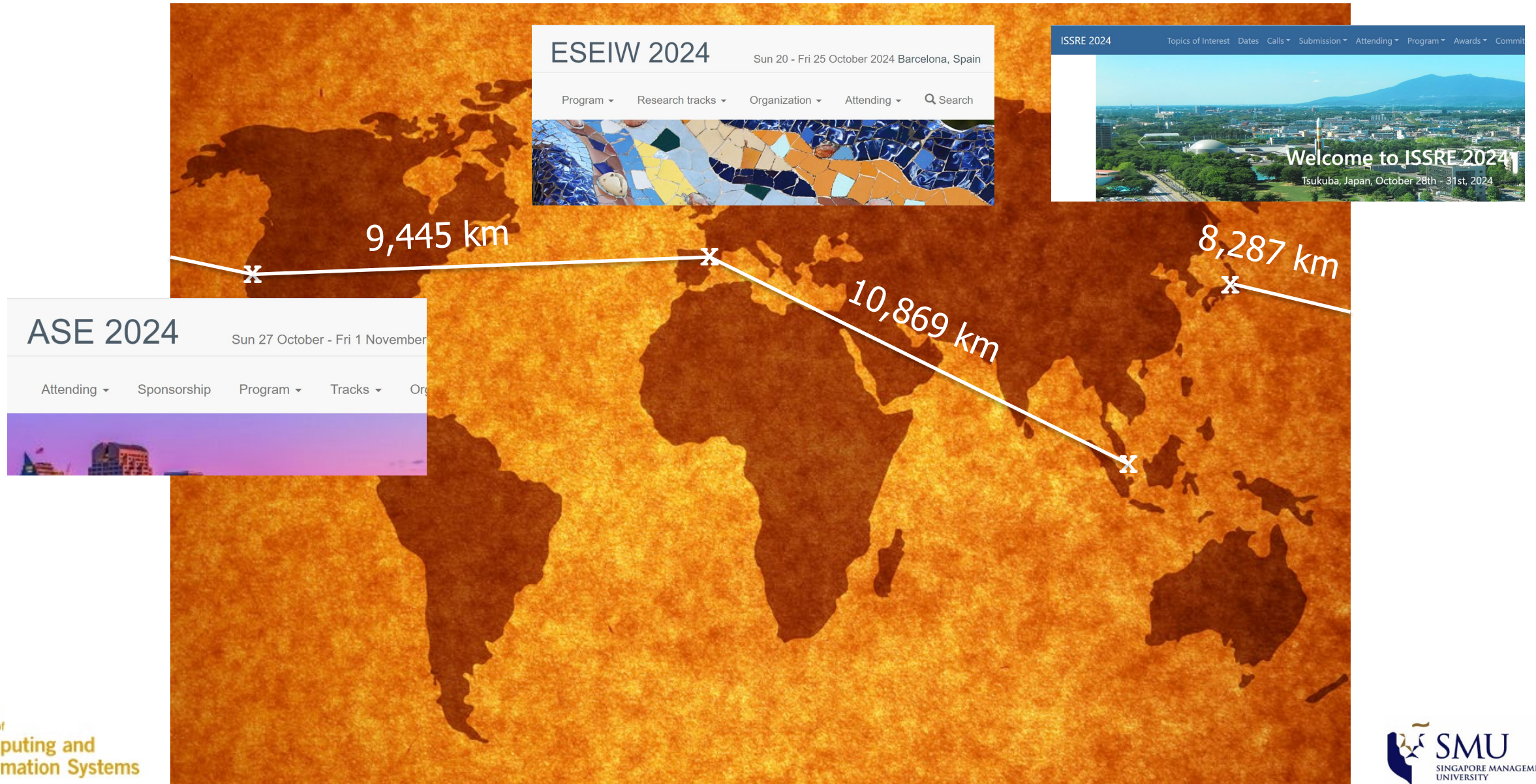
David Lo, FACM, FIEEE

ISSRE 2024, Tsukuba, Japan

Self-Introduction



Self-Introduction



Self-Introduction



Self-Introduction



Self-Introduction



Singapore Management University



- Third university in Singapore
- Number of students:
 - 8000+ (UG)
 - 1800+ (PG)
- Schools:
 - Business
 - Economics
 - Accountancy
 - Law
 - Social Science
 - Computing

Center for Research on Intelligent Software Engineering (RISE)

Elsevier JSS'21, Bibliometric Study

Table 3

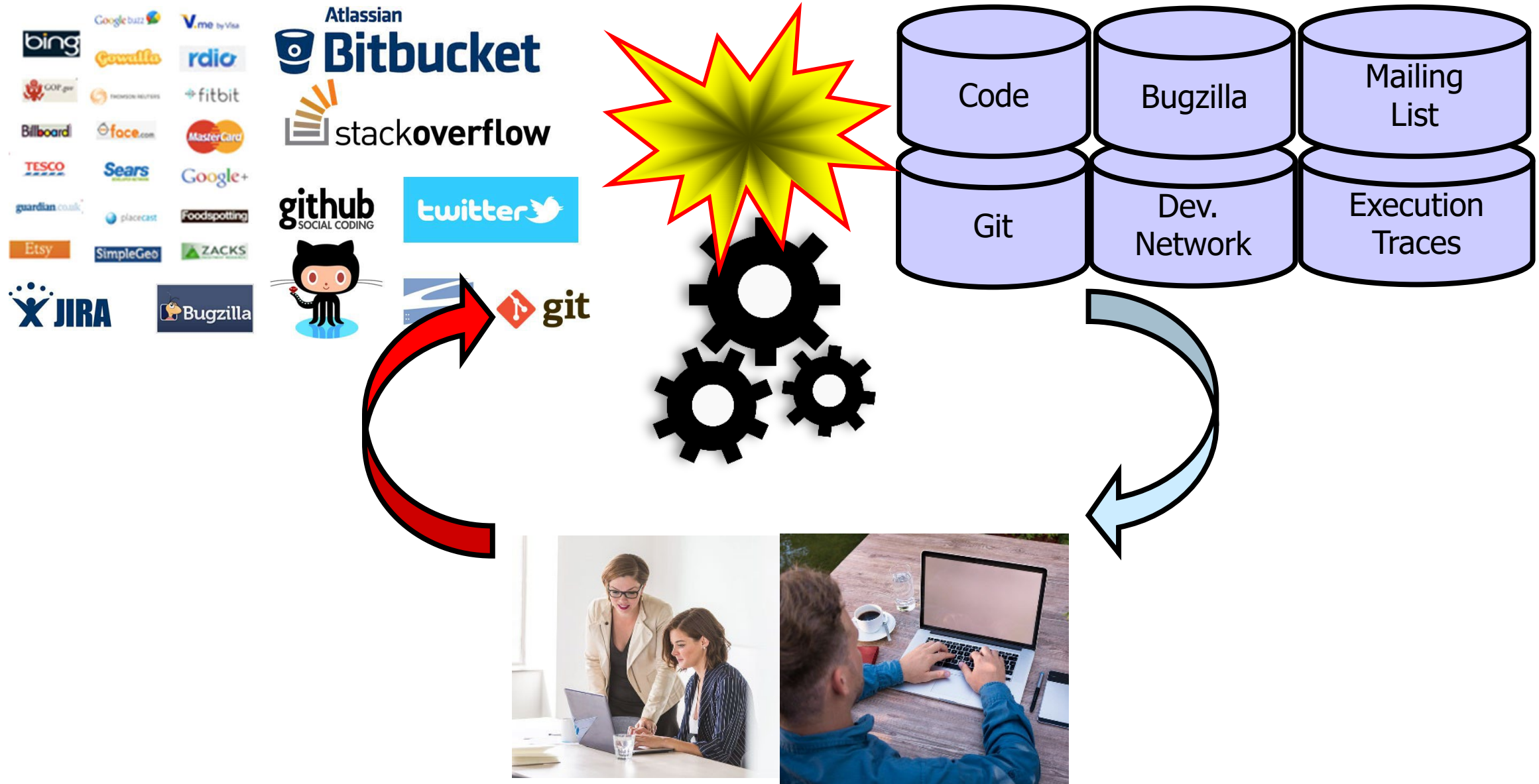
Most active institutions in software engineering

Rank	Name
1	University of California
2	Carnegie Mellon University
3	Nanjing University
4	Microsoft Research
5	Singapore Management University

CSRankings, SE, June 2024

#	Institution	Count	Faculty
1	▶ Nanjing University 🇨🇳 📊	39.0	38
2	▶ Carnegie Mellon University 🇺🇸 📊	31.6	17
3	▶ Peking University 🇨🇳 📊	28.5	21
4	▶ Singapore Management University 🇸🇬 📊	22.7	8

AI for Software Engineering



Experience with AI4SE

SMArTIC: Towards Building an Accurate, Robust and Scalable Specification Miner

FSE'06

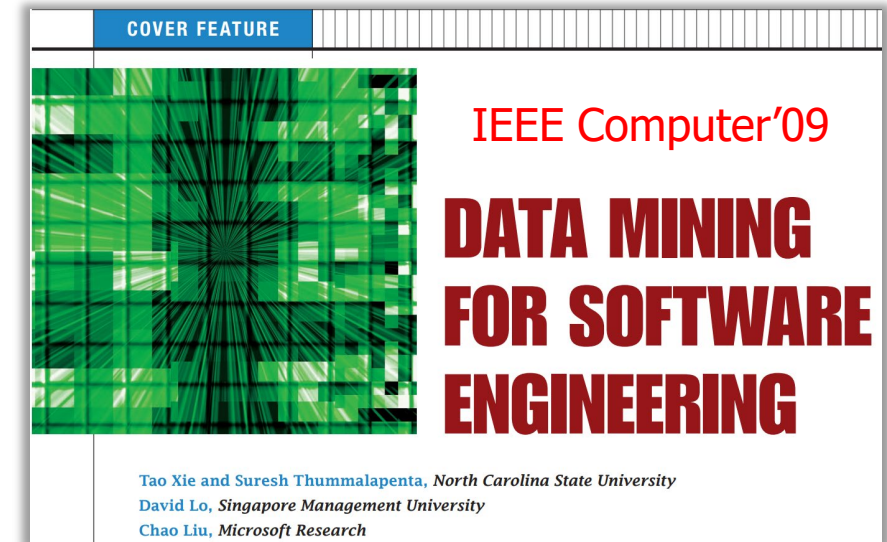
David Lo and Siau-Cheng Khoo
Department of Computer Science, National University of Singapore
{dlo,khoosc}@comp.nus.edu.sg

Efficient Mining of Iterative Patterns for Software Specification Discovery

KDD'07

David Lo and Siau-Cheng Khoo
Department of Computer Science
National University of Singapore
{dlo,khoosc}@comp.nus.edu.sg

Chao Liu
Department of Computer Science
University of Illinois-UC
chaoliu@cs.uiuc.edu



Experience with AI4SE

Classification of Software Behaviors for Failure Detection: A Discriminative Pattern Mining Approach

KDD'09

David Lo
Singapore Management University
davidlo@smu.edu.sg

Hong Cheng*
Chinese University of Hong Kong
hcheng@se.cuhk.edu.hk

Jiawei Han†
University of Illinois at Urbana-Champaign
hanj@cs.uiuc.edu

Siau-Cheng Khoo and Chengnian Sun
National University of Singapore
{khoosc,sunch}@comp.nus.edu.sg

Test oracle generation

A Discriminative Model Approach for Accurate Duplicate Bug Report Retrieval

ICSE'10

Chengnian Sun¹, David Lo², Xiaoyin Wang³, Jing Jiang², Siau-Cheng Khoo¹

¹School of Computing, National University of Singapore

²School of Information Systems, Singapore Management University

³Key laboratory of High Confidence Software Technologies (Peking University), Ministry of Education

sunch@comp.nus.edu.sg, davidlo@smu.edu.sg, wangxy06@sei.pku.edu.cn,

jingjiang@smu.edu.sg, khoosc@comp.nus.edu.sg

Intelligent issue trackers

Tag Recommendation in Software Information Sites

MSR'13

Xin Xia*[‡], David Lo[†], Xinyu Wang*, and Bo Zhou*[§]

*College of Computer Science and Technology, Zhejiang University

†School of Information Systems, Singapore Management University

Intelligent crowdsourced SE

History Driven Program Repair

SANER'16

Xuan-Bach D. Le, David Lo
School of Information Systems
Singapore Management University
{dxb.le.2013,davidlo}@smu.edu.sg

Claire Le Goues
School of Computer Science
Carnegie Mellon University
clegoues@cs.cmu.edu

Intelligent program repair

*"History-driven
program repair
influence*

*our work, the overall
pipeline is similar"*

- Facebook
Engineers

Future Direction in AI4SE

Trustworthy and Synergistic Artificial Intelligence for Software Engineering: Vision and Roadmaps

David Lo
School of Computing and Information Systems,
Singapore Management University,
Singapore
Email: davidlo@smu.edu.sg



Trustworthy and Synergistic AI4SE: Vision and the Road Ahead



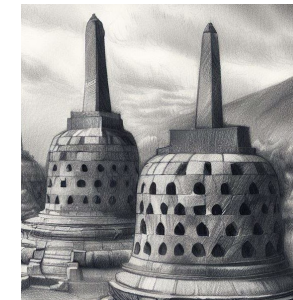
David Lo

School of
Computing and
Information Systems

ICSE'23 Future of SE Talk

AI for Software Engineering

History



Challenges



Vision



Trust Synergy



Roadmap I

Roadmap II

Call4Action

Towards Software Engineering 2.0

“If you want to go far, go together” – African Proverb



A binary star is two stars orbiting each other,
gravitationally bound together - GPT4o

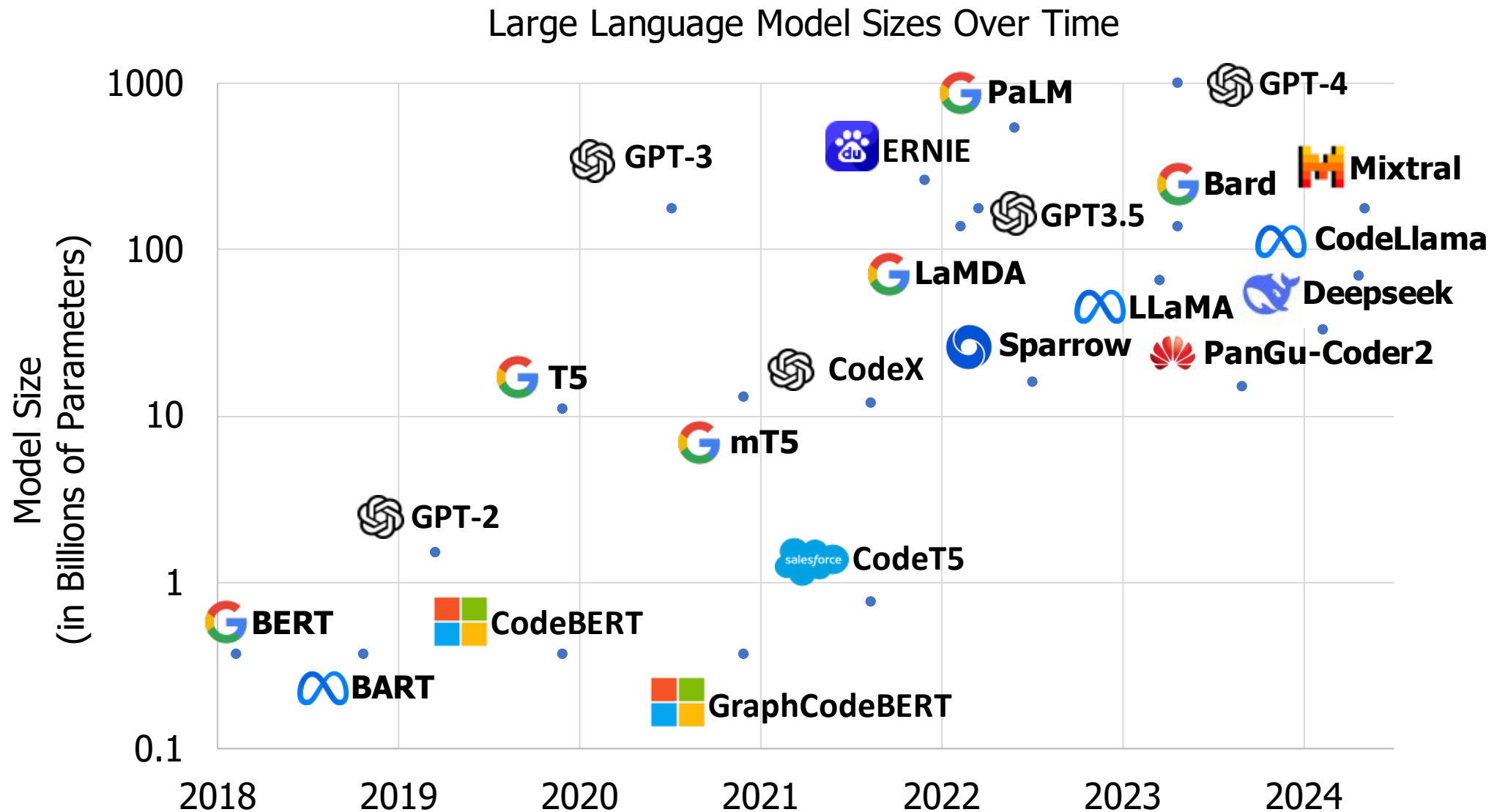
A binary star system with two glowing spheres, one labeled 'LLM' and the other 'Software Reliability', set against a vibrant cosmic background of orange and blue nebulae and distant stars.

LLM

**Software
Reliability**

Software Reliability in the Era of Large Language Models: A Dual Perspective

Large Language Models (LLMs)



LLM Can Greatly Help SE Tasks

ICSME 2020

Sentiment Analysis for Software Engineering: How Far Can Pre-trained Transformer Models Go?

Ting Zhang, Bowen Xu*, Ferdian Thung, Stefanus Agus Haryono, David Lo, Lingxiao Jiang

School of Information Systems, Singapore Management University

Email: {tingzhang.2019, bowenxu.2017}@phdcs.smu.edu.sg, {ferdianthung, stefanusah, davidlo, lxjiang}@smu.edu.sg



Early work on LLM4SE, most cited paper of ICSME 2020

ICSE 2024

Out of Sight, Out of Mind: Better Automatic Vulnerability Repair by Broadening Input Ranges and Sources

Xin Zhou

Singapore Management University
Singapore

xinzhou.2020@phdcs.smu.edu.sg

Kisub Kim*

Singapore Management University
Singapore

kisubkim@smu.edu.sg

Bowen Xu

North Carolina State University
USA

bxu22@ncsu.edu

DongGyun Han

Royal Holloway, University of London
United Kingdom

donggyun.han@rhul.ac.uk

David Lo

Singapore Management University
Singapore

davidlo@smu.edu.sg



Multi-LLM collaboration + data-centric innovation = 2x efficacy

LLMs Seem to Win for Many SE Scenarios

Large Language Models for Software Engineering: A Systematic Literature Review

XINYI HOU*, Huazhong University of Science and Technology, China

YANJIE ZHAO*, Monash University, Australia

YUE LIU, Monash University, Australia

ZHOU YANG, Singapore Management University, Singapore

KAILONG WANG, Huazhong University of Science and Technology, China

LI LI, Beihang University, China

XIAPU LUO, The Hong Kong Polytechnic University, China

DAVID LO, Singapore Management University, Singapore

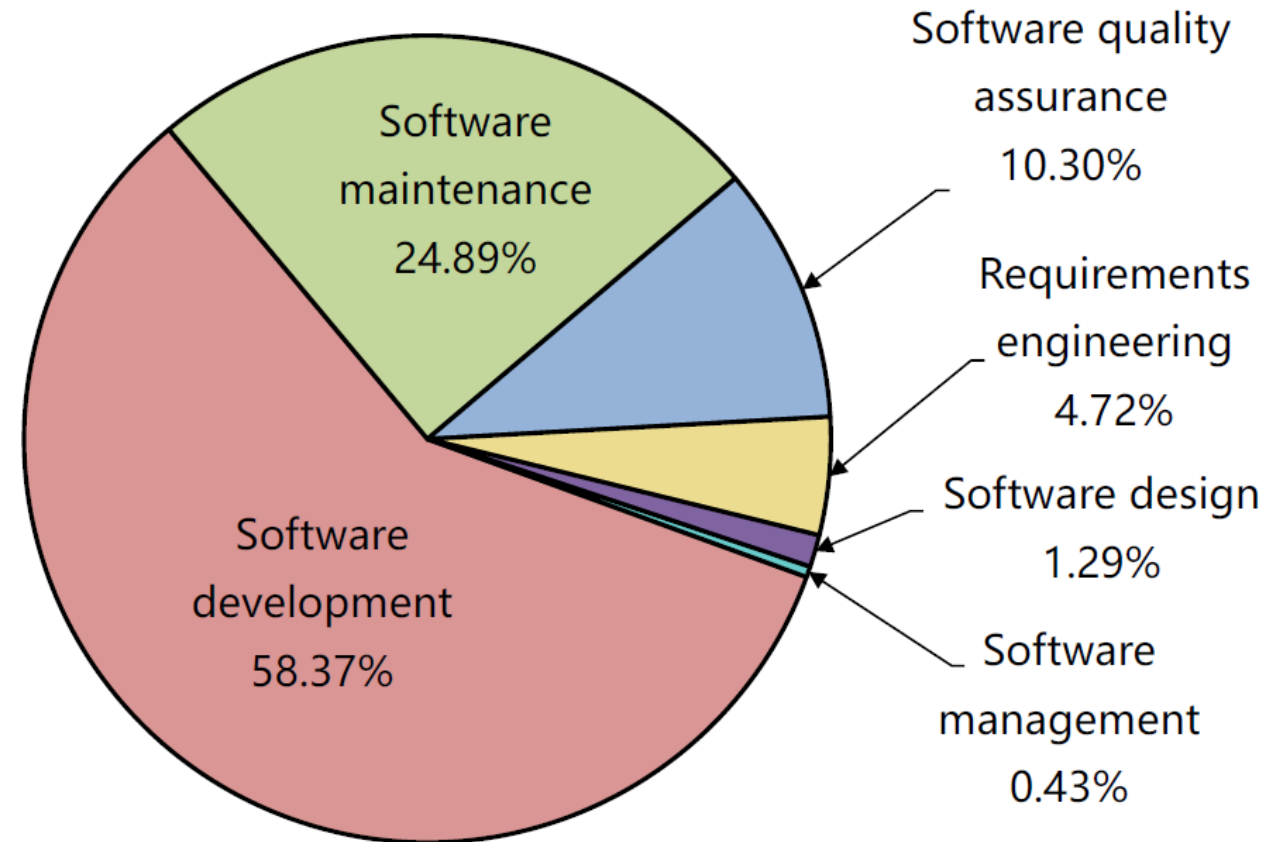
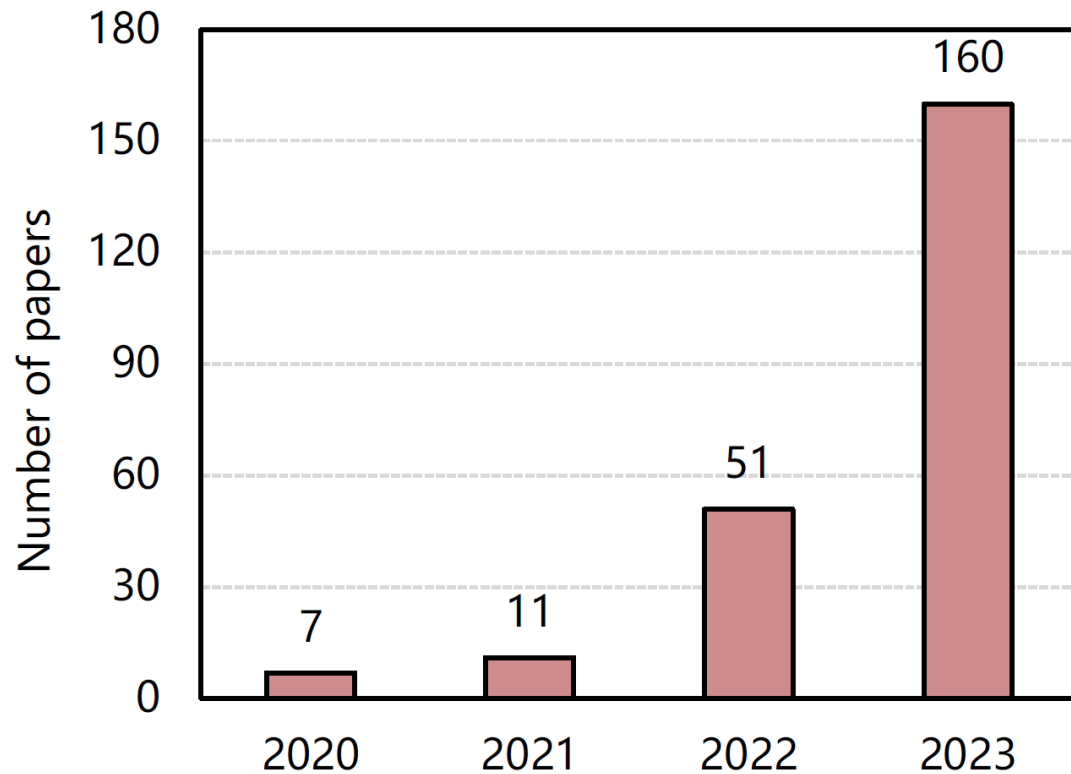
JOHN GRUNDY, Monash University, Australia

HAOYU WANG[†], Huazhong University of Science and Technology, China

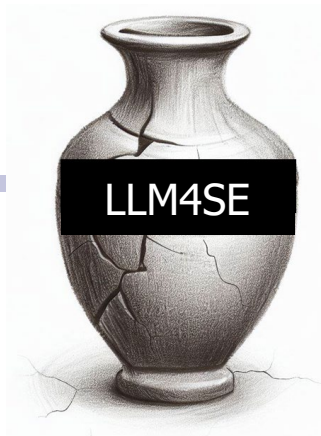


TOSEM 2024

LLMs Seem to Win for Many SE Scenarios



Many Open Problems



Robustness, Security, Privacy, Explainability, Efficiency, and Usability of Large Language Models for Code

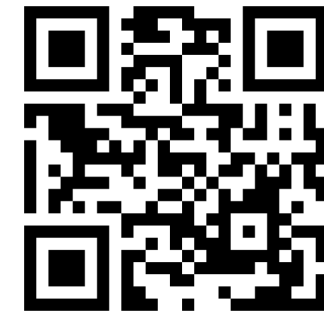
ZHOU YANG, Singapore Management University, Singapore

ZHENSU SUN, Singapore Management University, Singapore

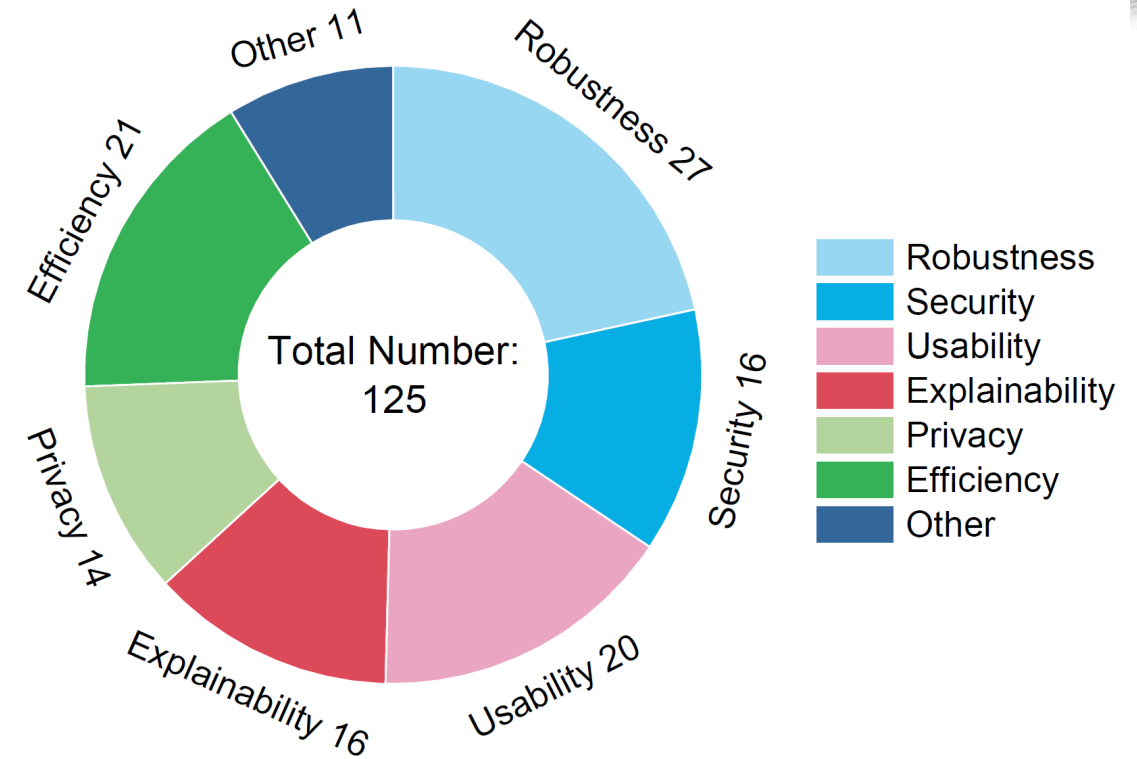
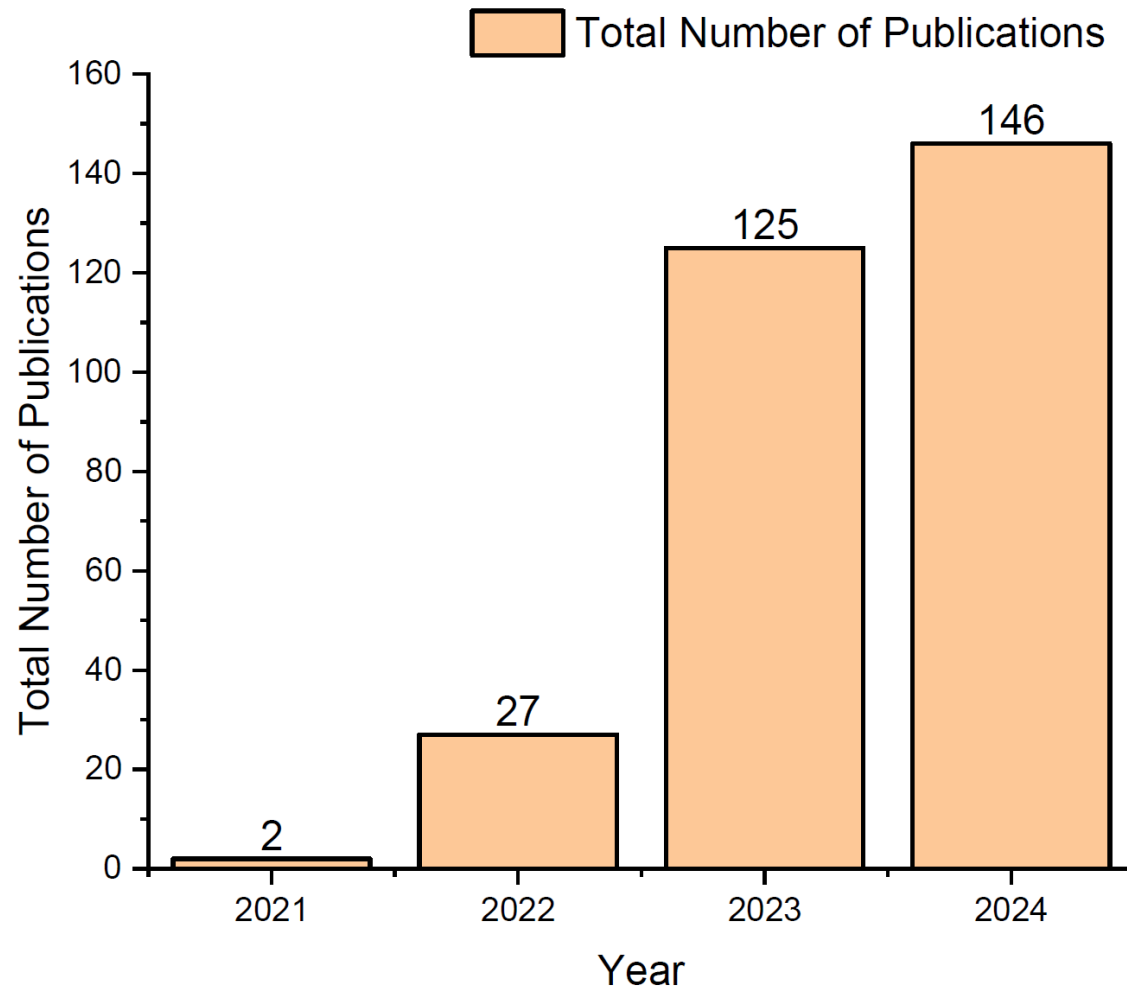
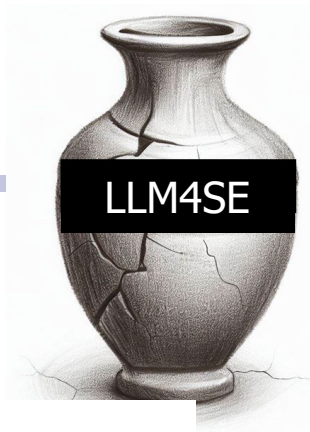
TERRY ZHUO YUE, Singapore Management University, Singapore

PREMKUMAR DEVANBU, Department of Computer Science, UC Davis, USA

DAVID LO, Singapore Management University, Singapore



Many Open Problems



A binary star is two stars orbiting each other,
gravitationally bound together - GPT4o



LLM



**Software
Reliability**



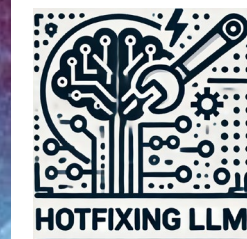
**LLM-Powered
Vulnerability
Repair**



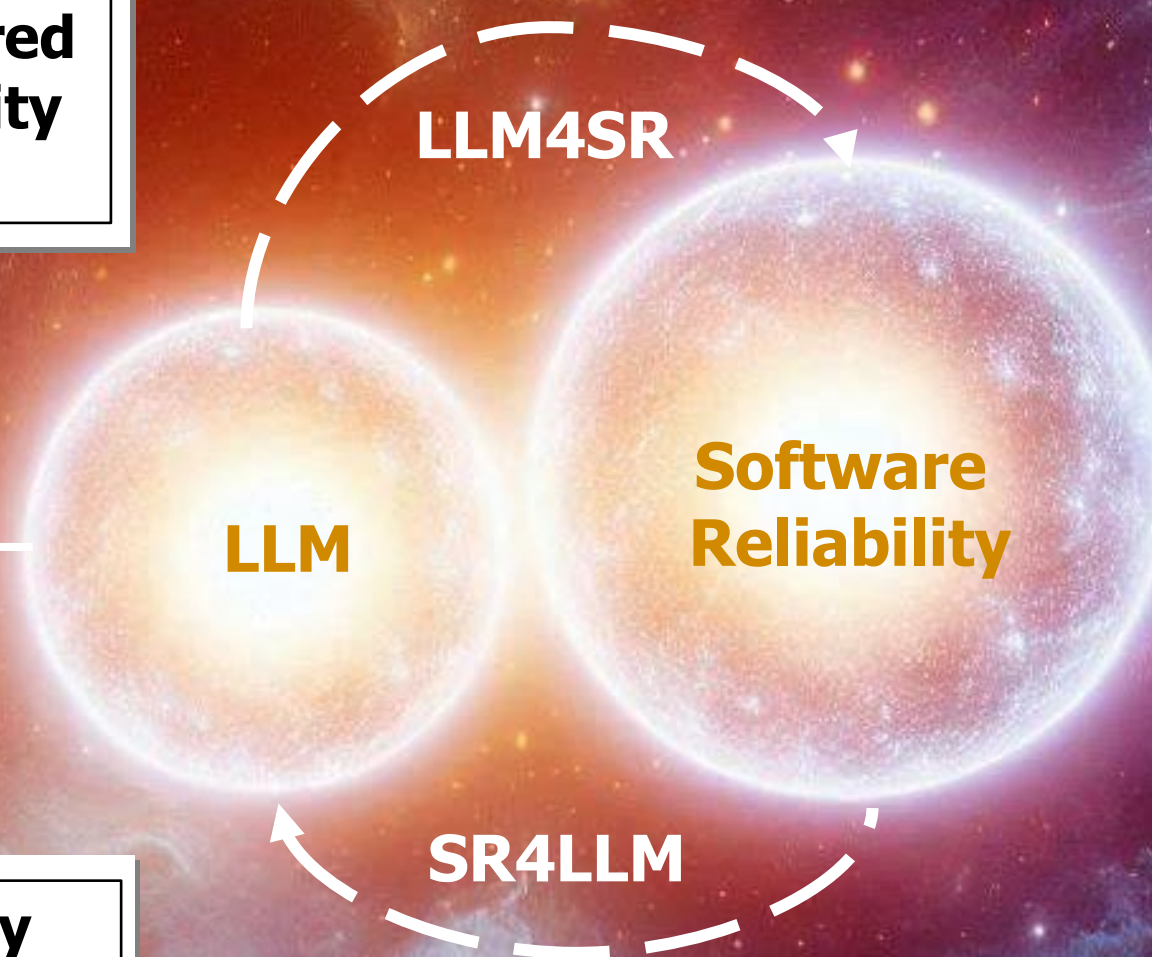
**LLM-Powered
Runtime Error
Recovery**



**Reliability
Estimation for
Code LLM**



**Hotfixing
Code LLM**





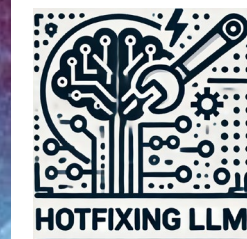
**LLM-Powered
Vulnerability
Repair**



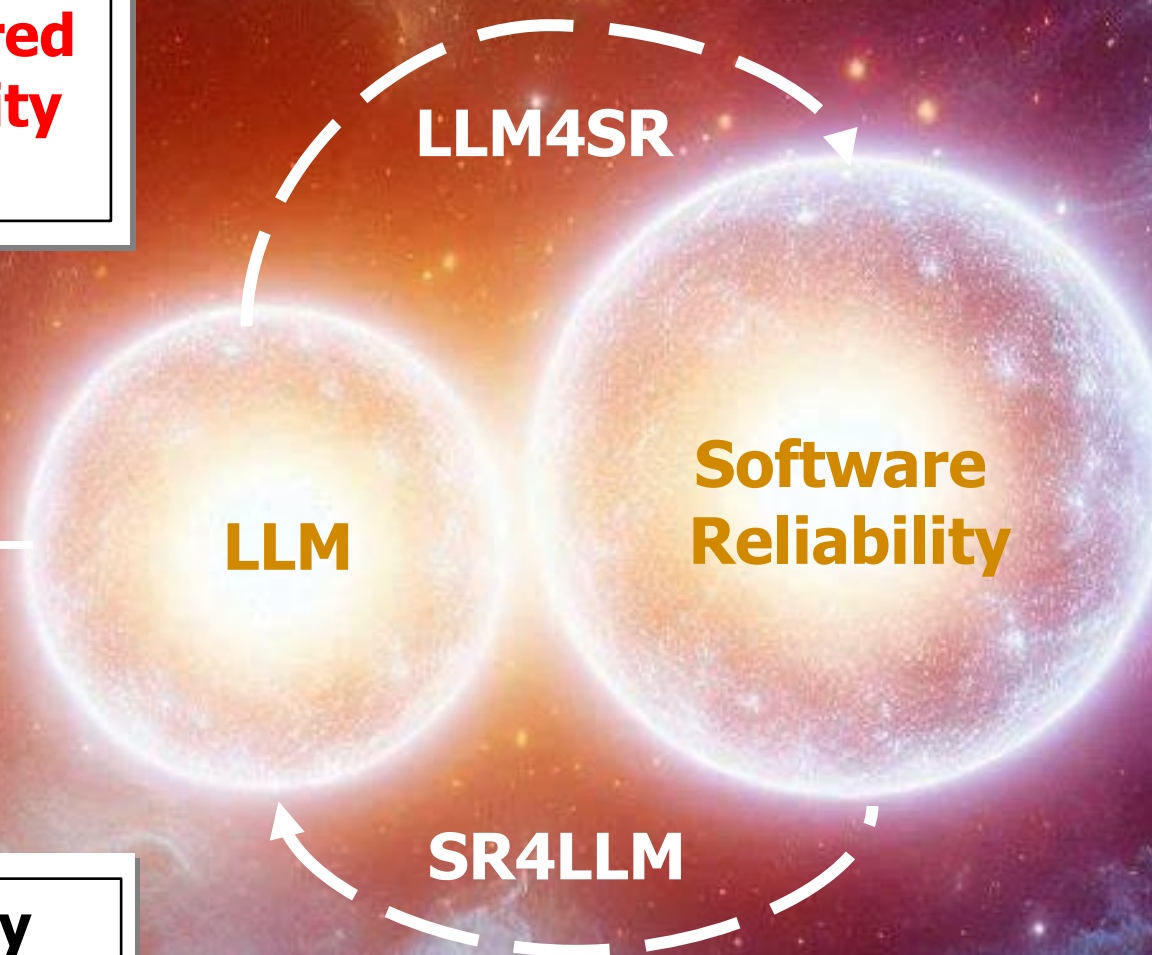
**LLM-Powered
Runtime Error
Recovery**



**Reliability
Estimation for
Code LLM**



**Hotfixing
Code LLM**



Multi-LLM Collaboration for Vulnerability Repairs

ICSE 2024

Out of Sight, Out of Mind: Better Automatic Vulnerability Repair by Broadening Input Ranges and Sources

Xin Zhou

Singapore Management University
Singapore
xinzhou.2020@phdcs.smu.edu.sg

Kisub Kim*

Singapore Management University
Singapore
kisubkim@smu.edu.sg

Bowen Xu

North Carolina State University
USA
bxu22@ncsu.edu

DongGyun Han

Royal Holloway, University of London
United Kingdom
donggyun.han@rhul.ac.uk

David Lo

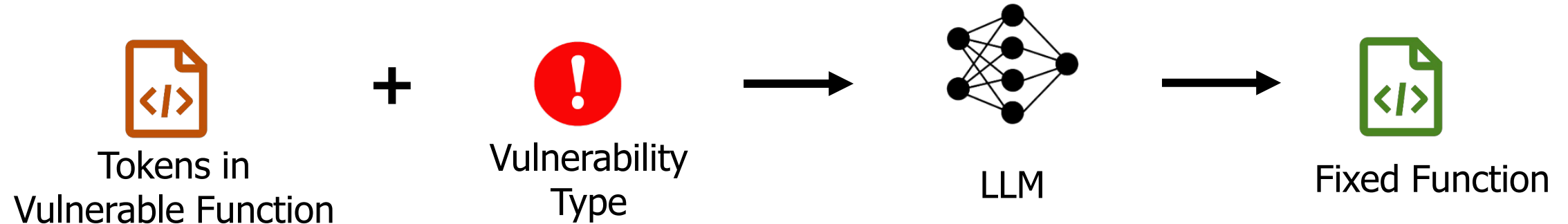
Singapore Management University
Singapore
davidlo@smu.edu.sg



First work to propose **multi-LLM collaboration** for vulnerability repairs through analyses of **multiple input sources**

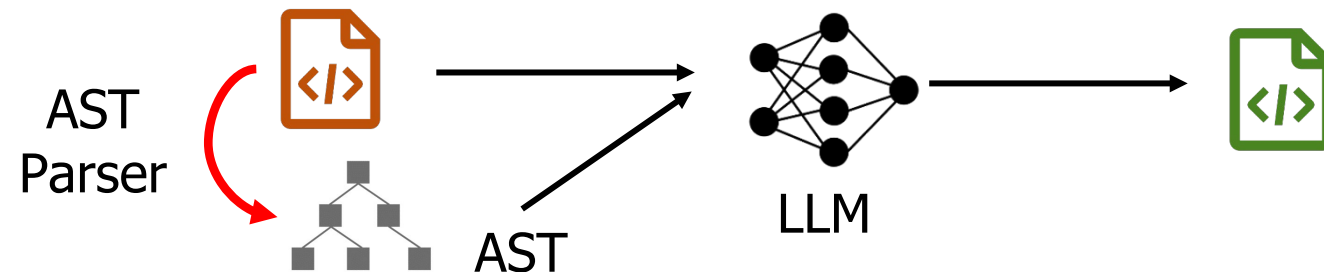
Additional Inputs for Efficacy Boost

Previous solutions:



Many **other inputs** have not been leveraged:

- *Abstract Syntax Tree*



- *CWE Knowledge*



- CWE Description
- Simple Vulnerable Code Examples
- Detailed Analyses

Research Questions

- How can we *effectively leverage* these additional inputs?
- How can we boost performance through *multi-LLM collaboration*?



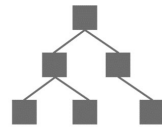
VulMaster: A State-of-the-Art Vulnerability Repair Method

Data-Centric Innovations

+

Multi-LLM Collaboration

Incorporate AST



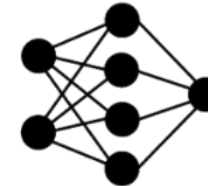
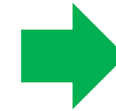
Incorporate
CWE knowledge



Address
lengthy inputs



GPT-3.5



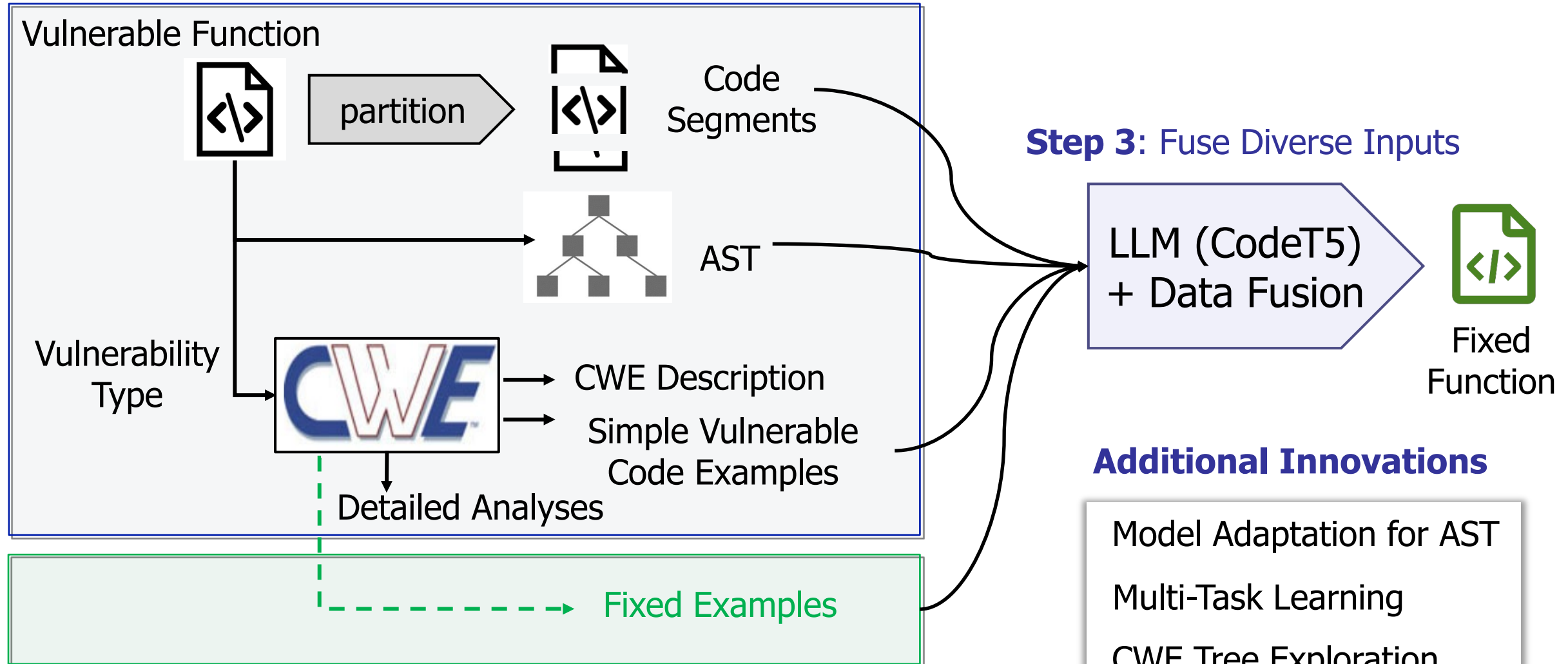
CodeT5

=

**2x Fixed
Vulnerabilities**

VulMaster's Overall Framework

Step 1: Leverage Diverse Inputs



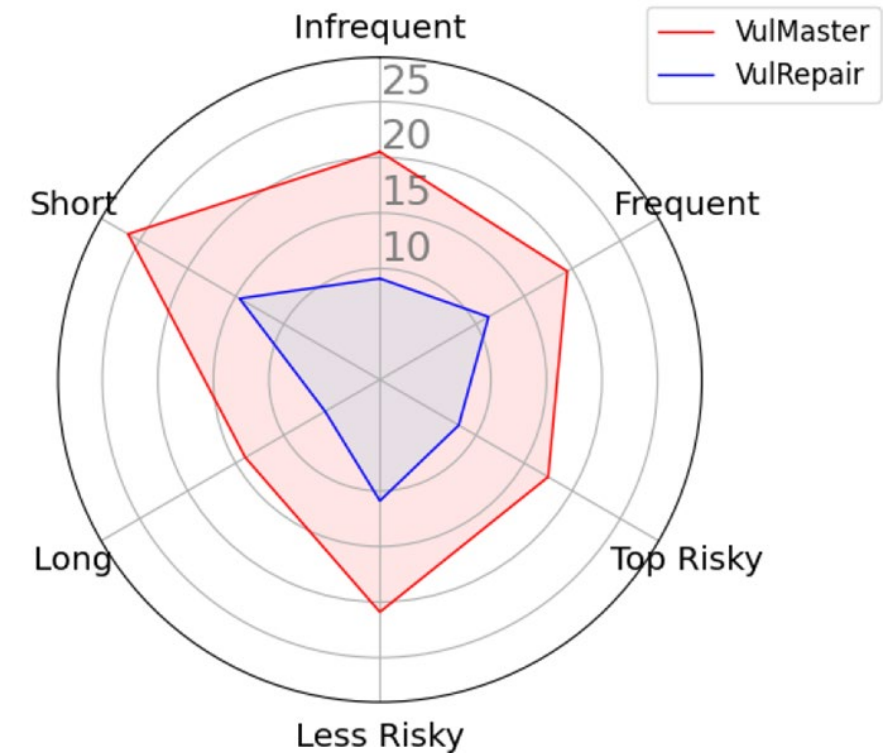
Step 2: Fill in Missing Data with Multi-LLM Collaboration

Results: Comparisons with SOTA

Main Results

Type	Approach	EM	BLEU
LLM	GPT-3.5 [55]	3.6	8.8
	GPT-4 [56]	5.3	9.7
task-specific	VRepair [9]	8.9	11.3
	VulRepair [19] (SOTA)	10.2	21.3
Ours	VulMaster	20.0	29.3

- VulMaster **doubles the Exact Match (EM)** score
- VulMaster consistently outperforms for vulnerabilities of different characteristics



- *long/short*: the length of the code
- *frequent/infrequent*: the vulnerability type frequencies
- *top/less risky*: top 10 most dangerous CWEs or not

Open Challenges and Future Work

- Dealing with complex vulnerabilities, e.g., inter-procedural vulnerabilities
- Considering larger code contexts, e.g., repository-level
- Establishing trust and synergy with developers, e.g., evidence and rationales

TOSEM SE Vision 2030 @ FSE 2024

Large Language Model for Vulnerability Detection and Repair: Literature Review and the Road Ahead

XIN ZHOU, Singapore Management University, Singapore

SICONG CAO, Yangzhou University, China

XIAOBING SUN, Yangzhou University, China

DAVID LO, Singapore Management University, Singapore





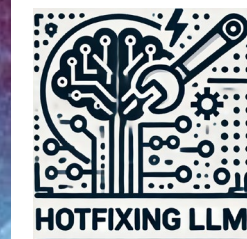
**LLM-Powered
Vulnerability
Repair**



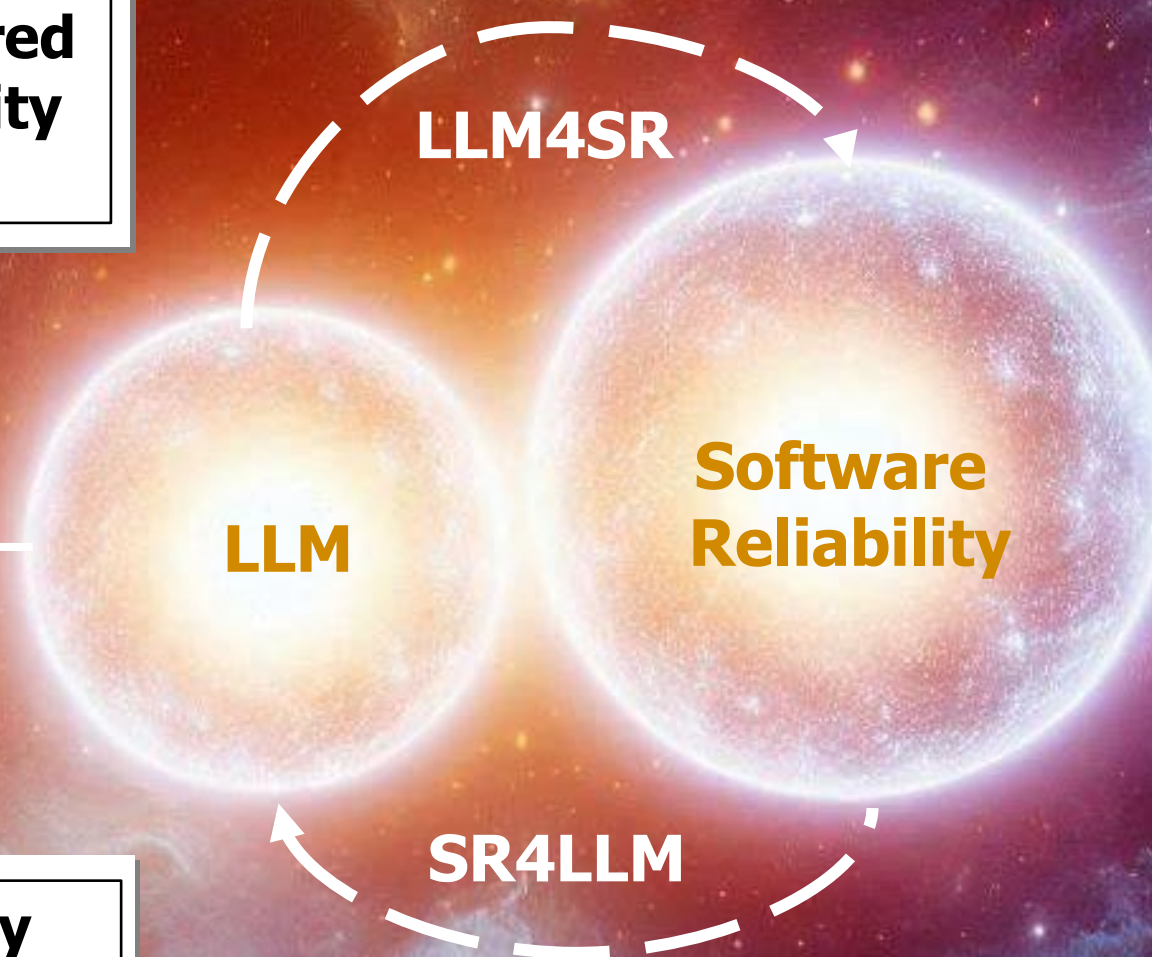
**LLM-Powered
Runtime Error
Recovery**



**Reliability
Estimation for
Code LLM**



**Hotfixing
Code LLM**



Adaptively Recover from Unexpected Runtime Errors

Recently released on arxiv

LLM as Adaptive Runtime Error Handler: A Promising Pathway to Self-Healing Software Systems

ZHENSU SUN, Singapore Management University, Singapore

HAOTIAN ZHU, Singapore Management University, Singapore

BOWEN XU, North Carolina State University, USA

XIAONING DU, Monash University, Australia

LI LI, Beihang University, China

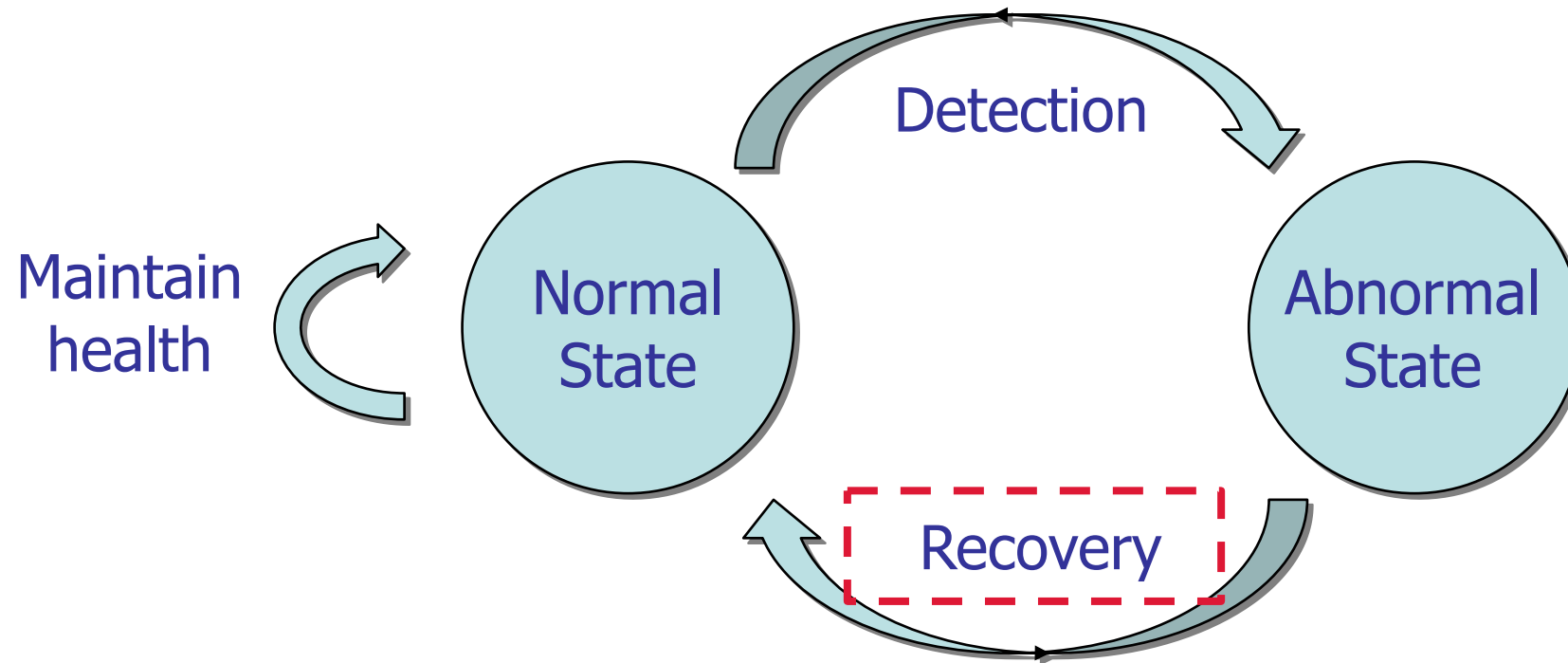
DAVID LO, Singapore Management University, Singapore



First work to propose an **LLM-powered self-healing framework**

Self-Healing Software System

Software systems that can **perceive** that it is not operating correctly and, without (or with) human intervention, make the necessary adjustments to **restore** itself to normalcy



Automated Recovery from Unexpected Runtime Errors

OSDI'04

Enhancing Server Availability and Security Through Failure-Oblivious Computing

Discard invalid writes and return default values for invalid reads

PLDI'14

Automatic Runtime Error Repair and Containment via Recovery Shepherd

Prepare a pre-defined rule set for different errors

ASE'16

Automatic Runtime Recovery via Error Handler Synthesis

Tianxiao Gu[§] Chengnian Sun* Xiaoxing Ma[§] Jian Lü[§] Zhendong Su*

[§]Department of Computer Science and Technology, Nanjing University, China

*Department of Computer Science, University of California, Davis, USA

tianxiao.gu@gmail.com, cnsun@ucdavis.edu, xxm@nju.edu.cn, lj@nju.edu.cn, su@cs.ucdavis.edu

Reuse applicable error handlers from existing ones

They are rather rigid

LLM as Adaptive Runtime Error Handler

```
1 bin_string = input() # input: "10 1"
2 bin_string = str(bin_string).strip()
3 if not bin_string:
4     raise ValueError("Empty string")
5 if bin_string[0] == "-":
6     bin_string = bin_string[1:]
7 int_number = 0
```

Program State: {"bin_string": "1 01",
"int_number": 0}

```
8 for char in bin_string:
9     int_number = 2 * int_number
10    int_number += int(char)
```

✘ ValueError: invalid literal for int()
with base 2: ''

```
11 compute_something_else(int_number)
... ..
```

The program crashes due to the unhandled error when it reaches Line 10.

LLM as Adaptive Runtime Error Handler

```

1 bin_string = input() # input: "10 1"
2 bin_string = str(bin_string).strip()
3 if not bin_string:
4     raise ValueError("Empty string")
5 if bin_string[0] == "-":
6     bin_string = bin_string[1:]
7 int_number = 0

Program State: {"bin_string": "1 01",
                "int_number": 0}

8 for char in bin_string:
9     int_number = 2 * int_number
10    int_number += int(char)

❌ ValueError: invalid literal for int()
   with base 10: ' '

11 compute_something_else(int_number)
...

```

An isolated runtime environment

LLM-generated recovery strategy

Copy

Program State: {"bin_string": "1 01",
"int_number": 0}

```

1 bin_string = '101'
2 for char in bin_string:
3     int_number = 2 * int_number
4     int_number += int(char)

```

Program State': {"bin_string": "101",
"int_number": 4}

LLM as Adaptive Runtime Error Handler

```

1 bin_string = input() # input: "10 1"
2 bin_string = str(bin_string).strip()
3 if not bin_string:
4     raise ValueError("Empty string")
5 if bin_string[0] == "-":
6     bin_string = bin_string[1:]
7 int_number = 0

```

Program State: {"bin_string": "1 01",
"int_number": 0}

```

8 for char in bin_string:
9     int_number = 2 * int_number
10    int_number += int(char)

```

❌ **ValueError: invalid literal for int()
with base 10: ' '**

▶ **Program State:** {"bin_string": "101",
"int_number": 0}

```

11 compute_something_else(int_number)
...

```

Copy



Program State: {"bin_string": "1 01",
"int_number": 0}

```

1 bin_string = '101'
2 for char in bin_string:
3     int_number = 2 * int_number
4     int_number += int(char)

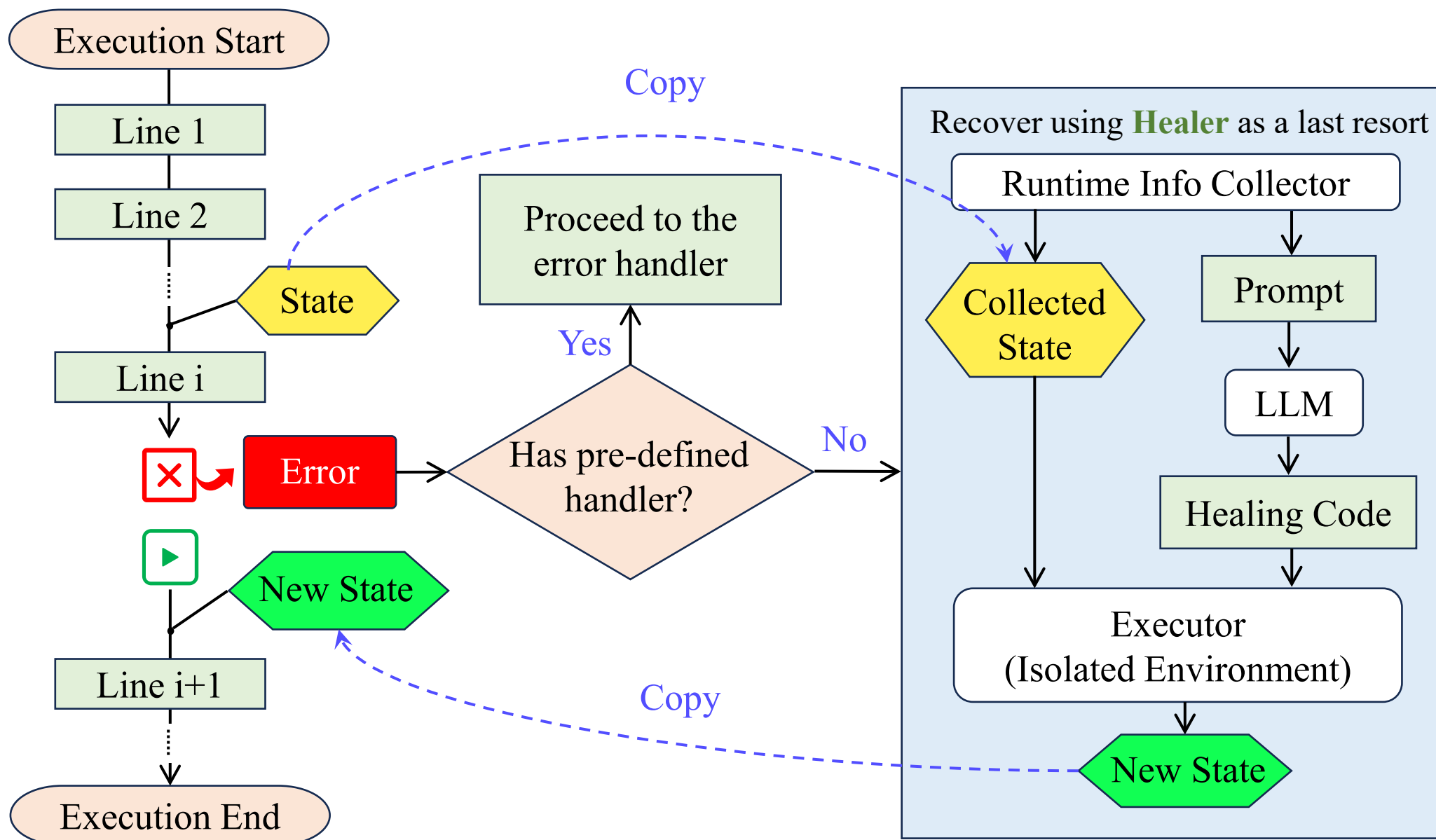
```

Copy

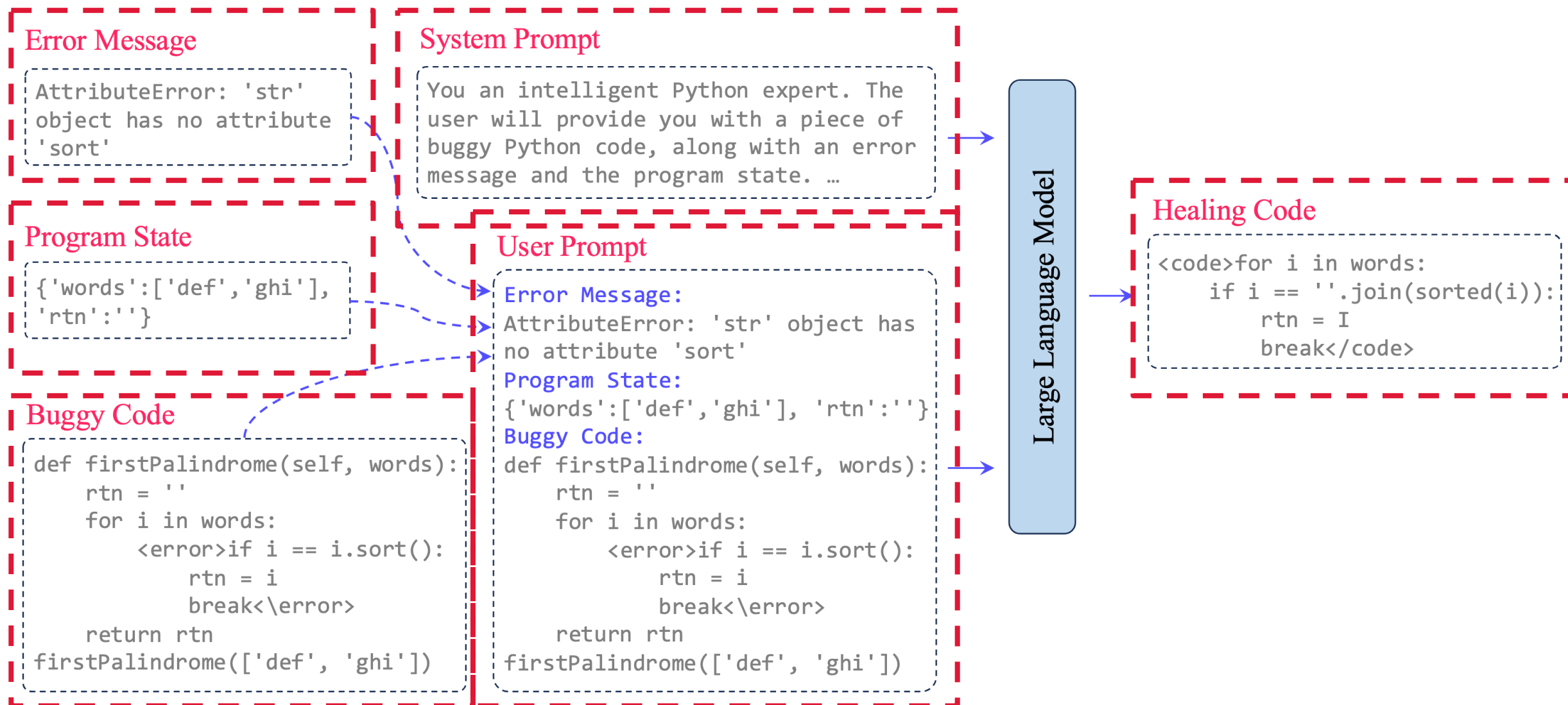


Program State: {"bin_string": "101",
"int_number": 0}

Healer: LLM-Powered Runtime Error Handling Framework



H Healer: LLM-Powered Runtime Error Handling Framework



How Effective is Healer in Handling Runtime Errors?

Benchmarks:

- CodeNet (Human-written code snippets that raises runtime errors)
- DebugBench (Code snippets with implanted bugs)
- Code generated by 6 LLMs on HumanEval and MBPP

Base LLM: GPT-4 (Default)

Saved from Crashing

With Healer, **~75%** of crashed executions can be healed to **run to completion**.

Produce Correct Results

With Healer, **~40%** of crashed executions can be healed to **run to completion** and **produce correct results**.

How to Mitigate the Uncertainty?

- **Implementing Safety Constraints**

Healer could be restricted by static checkers to avoid those high-risk behaviors

- **Defining the “Comfort Zone” for Healer**

Establish a pre-defined “comfort zone”, where Healer’s interventions are acceptable and unlikely to cause significant harm.

- **Maintaining Detailed Error Logs**

While Healer can effectively handle runtime errors, the underlying issues causing these errors still need to be addressed.

- **Issuing User Warnings**

Provide warnings when outputs are generated during Healer-triggered executions.

Future Work

- **Enhance the performance of Healer**
 - ✓ Designing better prompts and training methods
- **Robust mechanism to mitigate the uncertainty**
- **Integrating Healer into real-world operational systems**

LLM4SR @ Recent ISSREs

ISSRE 2024

Large Language Models Can Provide Accurate and Interpretable Incident Triage

Zexin Wang^{†‡}, Jianhui Li[†], Minghua Ma^{§*}, Ze Li[§], Yu Kang[§], Chaoyun Zhang[§], Chetan Bansal[§]
Murali Chintalapati[§], Saravan Rajmohan[§], Qingwei Lin[§], Dongmei Zhang[§], Changhua Pei[†], Gaogang Xie^{†‡}

[†]Computer Network Information Center, Chinese Academy of Sciences

[‡]University of Chinese Academy of Sciences [§]Microsoft

ISSRE 2023

Method-Level Bug Severity Prediction using Source Code Metrics and LLMs

1st Ehsan Mashhadi
University of Calgary
Calgary, Canada
ehsan.mashhadi@ucalgary.ca

2nd Hossein Ahmadvand
University of Calgary
Calgary, Canada
hossein.ahmadvand@ucalgary.ca

3rd Hadi Hemmati
York University and University of Calgary
Toronto, Canada
hemmati@yorku.ca



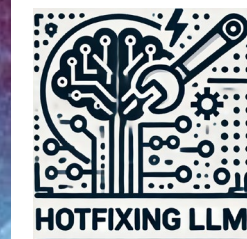
**LLM-Powered
Vulnerability
Repair**



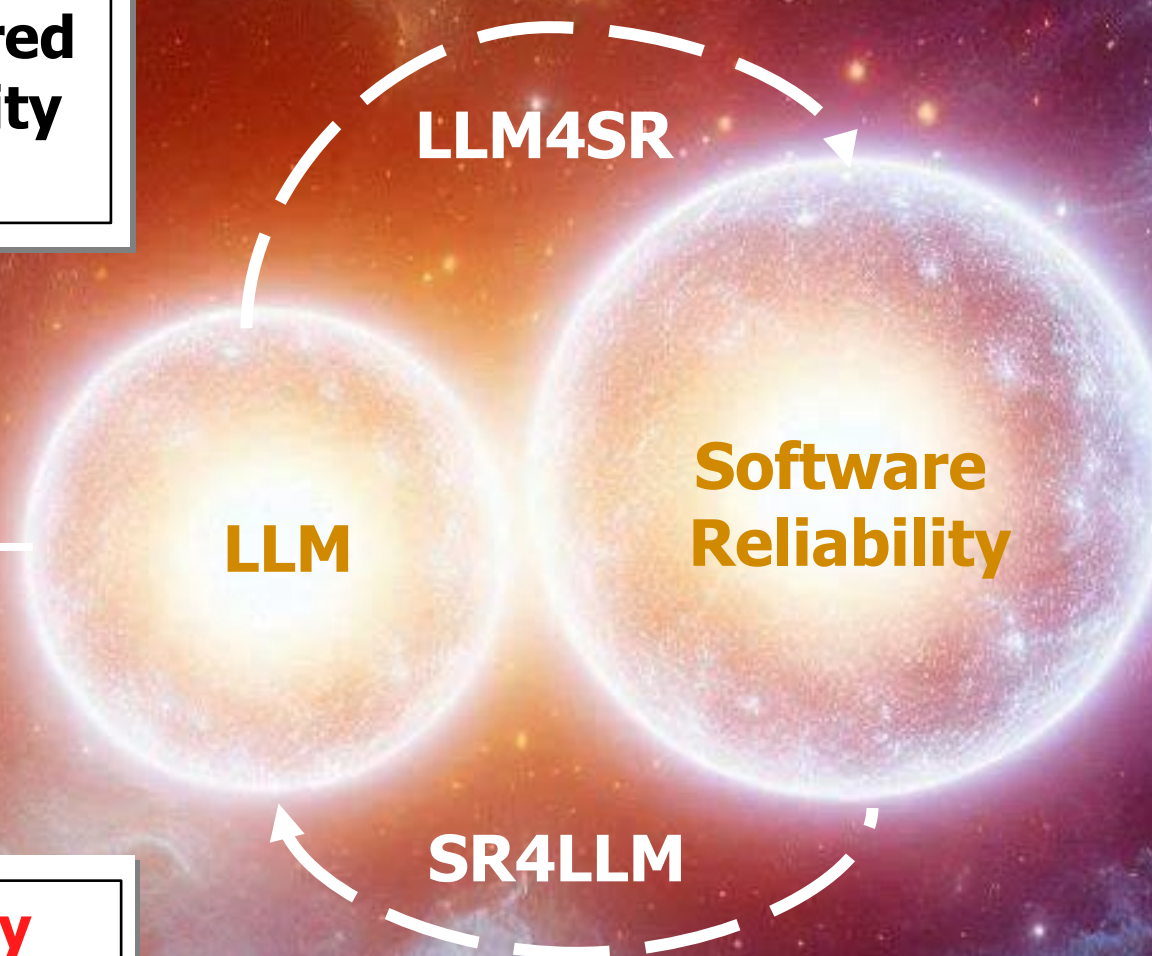
**LLM-Powered
Runtime Error
Recovery**



**Reliability
Estimation for
Code LLM**



**Hotfixing
Code LLM**



Stop Unreliable Code Completion with *FrugalCoder*

TOSEM 2024

Don't Complete It! Preventing Unhelpful Code Completion for Productive and Sustainable Neural Code Completion Systems

ZHENSU SUN, Singapore Management University, Singapore

XIAONING DU*, Monash University, Australia

FU SONG^{†‡}, Key Laboratory of System Software (Chinese Academy of Sciences), State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China

SHANGWEN WANG, National University of Defense Technology, China

MINGZE NI, University of Technology Sydney, Australia

LI LI, Beihang University, Beijing, China

DAVID LO, Singapore Management University, Singapore



First work to **estimate & stop unreliable code completions**

LLM-based Code Completion Brings New Challenges

LLM-based Code Completion is Popular

- Each user of Github Copilot receives one suggestion roughly every 3 minutes [GitHub22]

Low Acceptance Rate

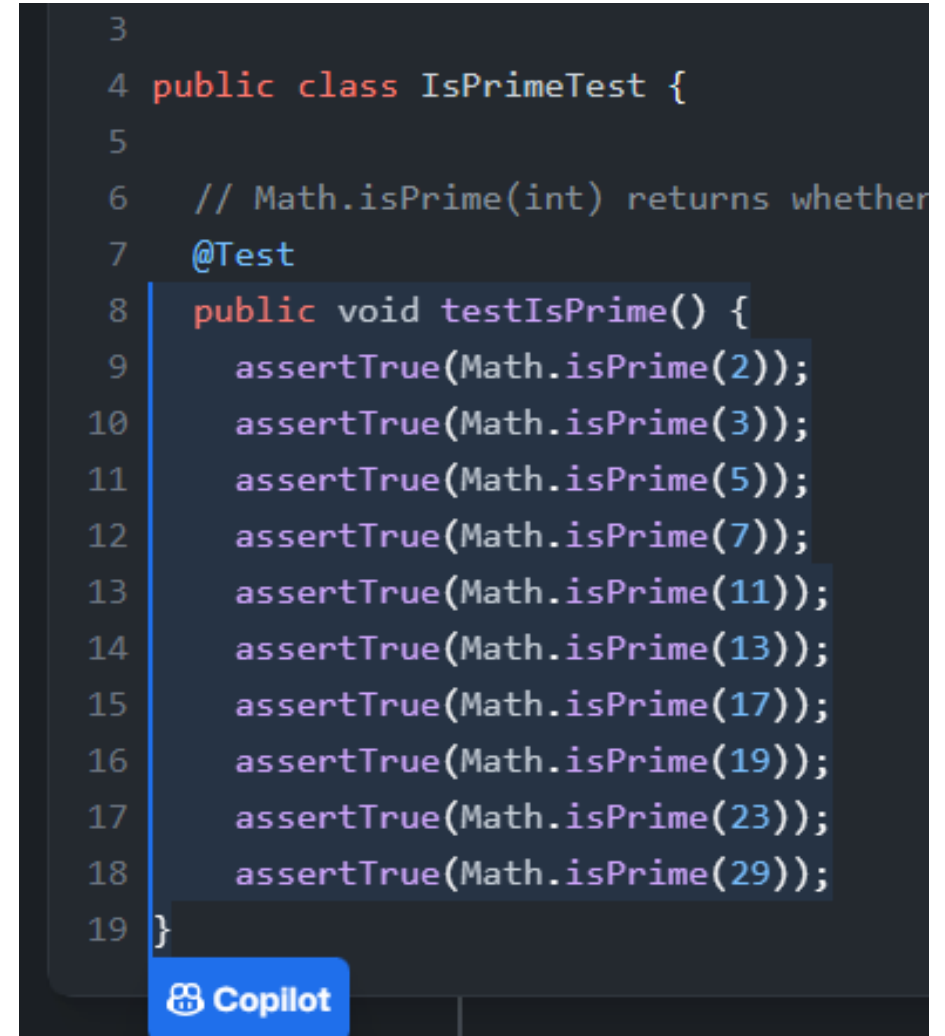
- Code completion requests are complex in real world
- Only 30% of completions are accepted by the users of Github Copilot [GitHub22]

High Computation & Latency

- LLM-based code completion requires large scale computing and causes latency

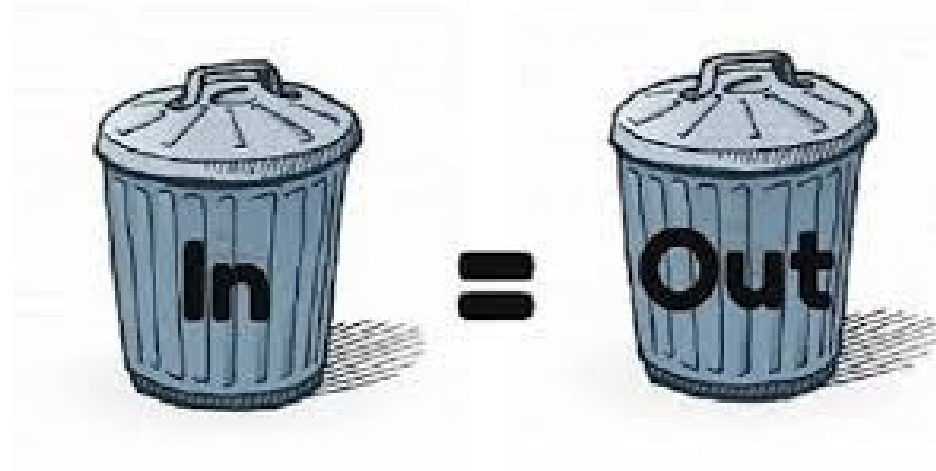
[GitHub22] Albert Ziegler (GitHub), Eirini Kalliamvakou (GitHub), Shawn Simister, et al. Productivity Assessment of Neural Code Completion. MAPS'22 at PLDI'22.

```
3
4 public class IsPrimeTest {
5
6     // Math.isPrime(int) returns whether
7     @Test
8     public void testIsPrime() {
9         assertTrue(Math.isPrime(2));
10        assertTrue(Math.isPrime(3));
11        assertTrue(Math.isPrime(5));
12        assertTrue(Math.isPrime(7));
13        assertTrue(Math.isPrime(11));
14        assertTrue(Math.isPrime(13));
15        assertTrue(Math.isPrime(17));
16        assertTrue(Math.isPrime(19));
17        assertTrue(Math.isPrime(23));
18        assertTrue(Math.isPrime(29));
19    }
```



What Cause Unreliable Code Completions?

- (1) Requests that are **beyond the capability of the LLM**
- (2) Requests that **do not contain sufficient information**, e.g., meaningless identifiers, vague intention, etc.



How to Address This Problem?

(1) Improve the performance of the LLM-Powered Solutions

Better Model, Better Retrieval-Augmented Solution, ...

Pre-Processing: Stop the requests 😊

(2) Prevent unreliable completions

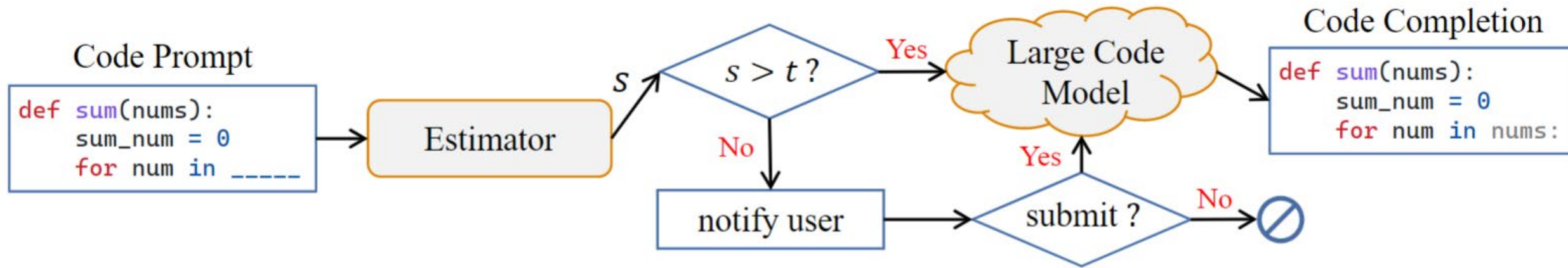
A **new task** for code completion

Inference: Early terminations

Post-processing: Do not display completions



FrugalCoder: Identify & Reject Unpromising Code Prompts



- Using lightweight estimator to estimate the quality of the code completion
- Decide whether to proceed based on a pre-defined threshold

Challenge 1: Efficacy

The estimator should effectively estimate code completion quality

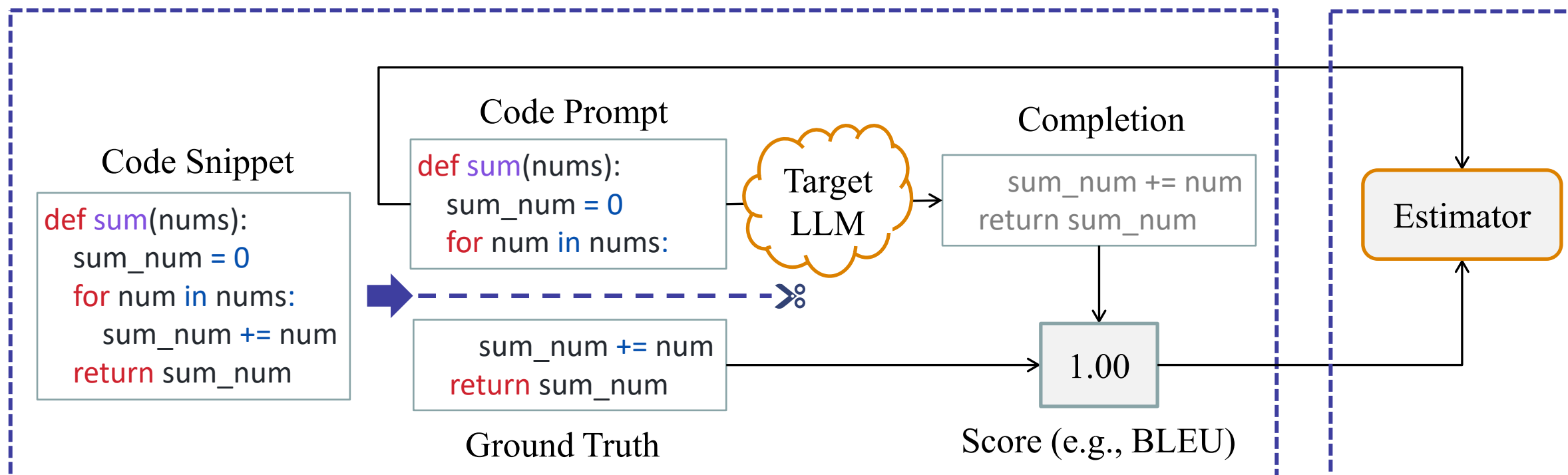
Challenge 2: Efficiency

The cost of running the estimator should be lower than the LLM cost

FrugalCoder: Building the Estimator

Generate Training Data

Train Estimator



Results: Feasibility of FrugalCoder

■ Deep Learning as Estimator

Encoder-only Transformer

Efficacy

Reject **20%** of requests
with a **95.1%** Precision

Improve Acceptance Rate
from **27.4%** to **33.0%**

Efficiency

5.1 ms for each query

■ Traditional ML as Estimator

Adaboost

Efficacy

Reject **20%** of requests
with a **92.1%** Precision

Improve Acceptance Rate
from **27.4%** to **32.3%**

Efficiency

0.1 ms for each query

Future Work

- **Stop remaining unreliable code completions**
 - ✓ Stop more unhelpful code completions to increase acceptance rate even more
- **Explore the secondary use of FrugalCoder**
 - ✓ Such as its potential to identify and stop adversarial attacks, backdoor attacks, etc.



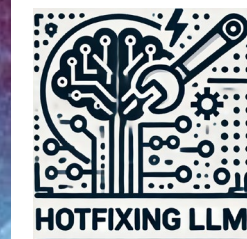
**LLM-Powered
Vulnerability
Repair**



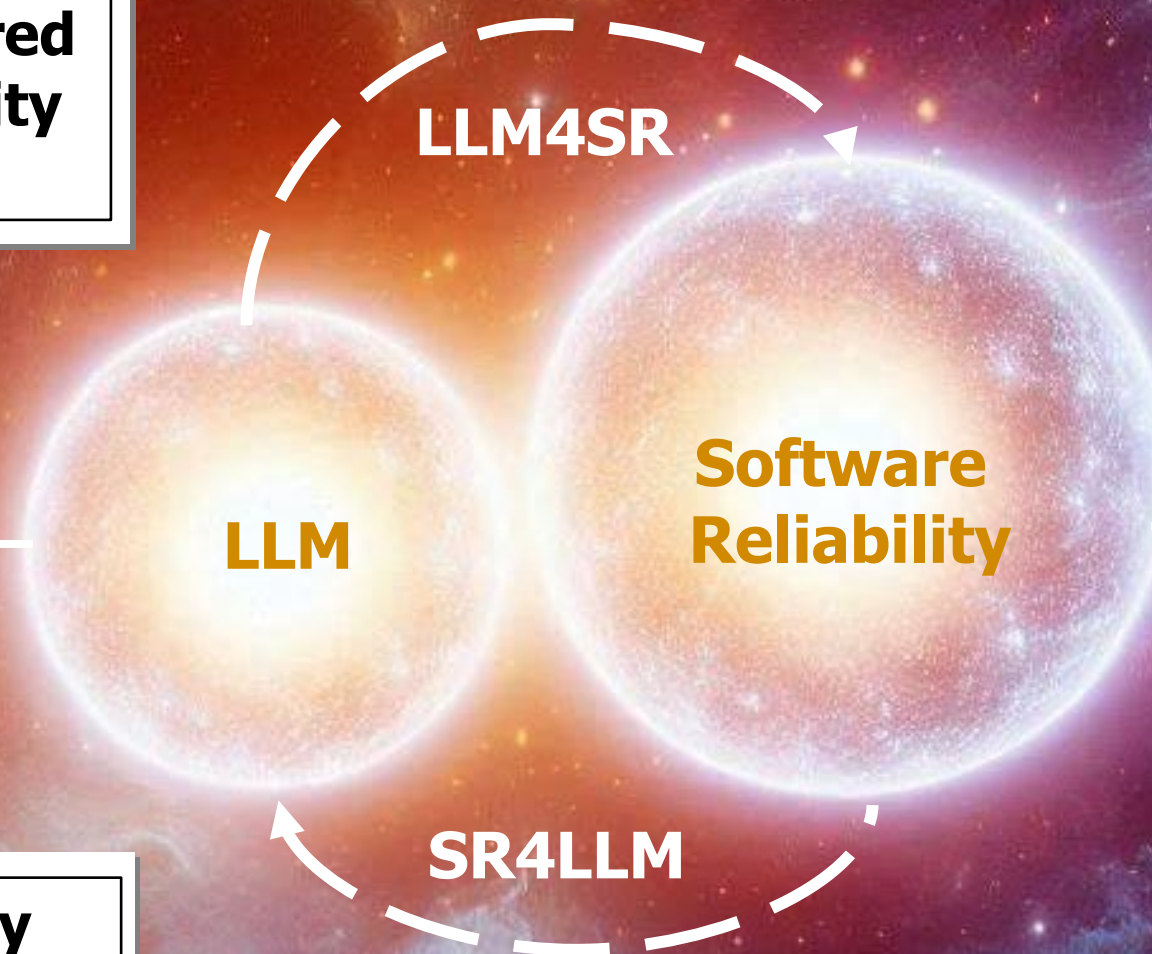
**LLM-Powered
Runtime Error
Recovery**



**Reliability
Estimation for
Code LLM**



**Hotfixing
Code LLM**



Hotfixing Code LLM

Recently released on arxiv

Hotfixing Large Language Models for Code

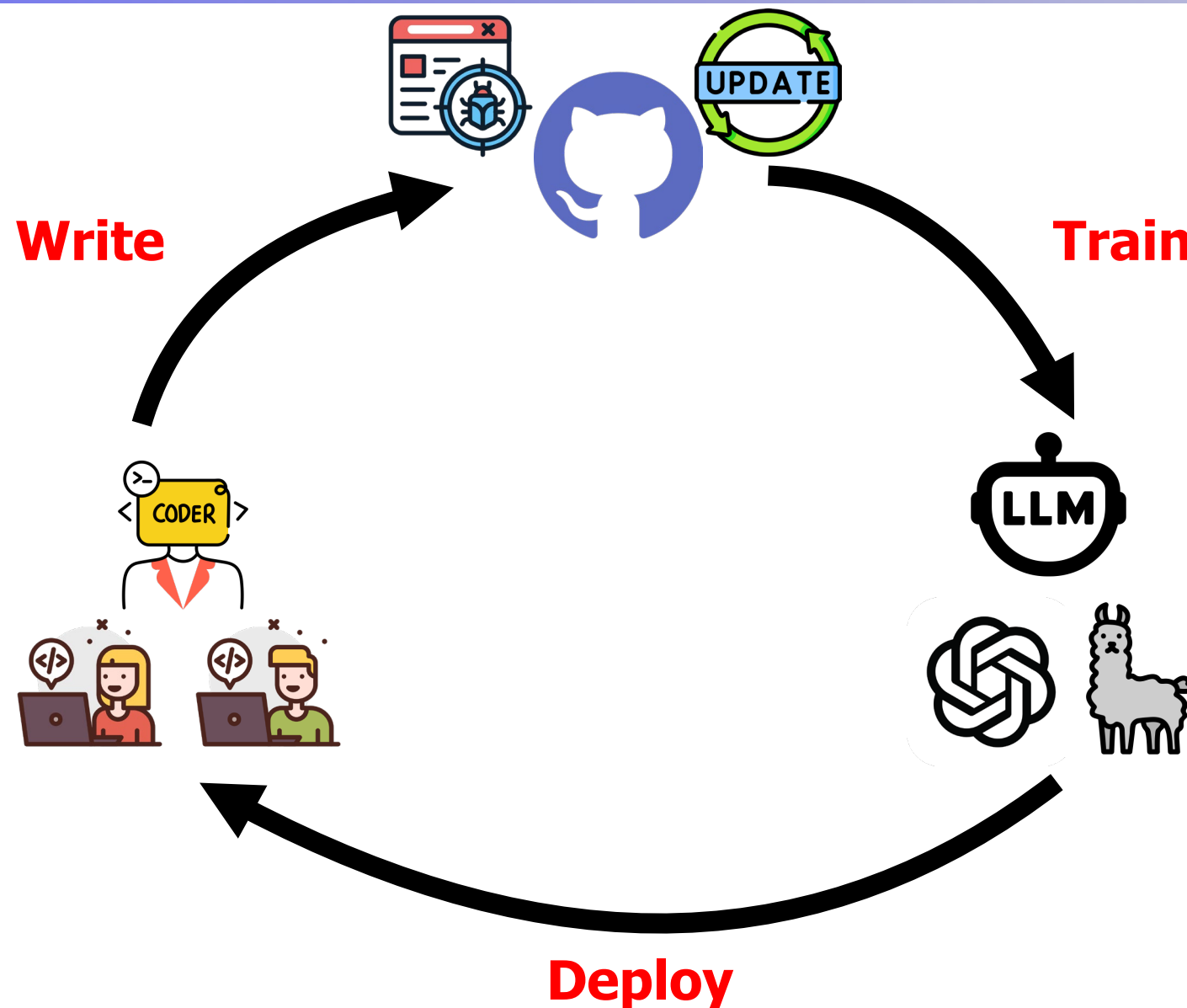
Zhou Yang
Singapore Management University
Singapore
zyang@smu.edu.sg

David Lo
Singapore Management University
Singapore
davidlo@smu.edu.sg

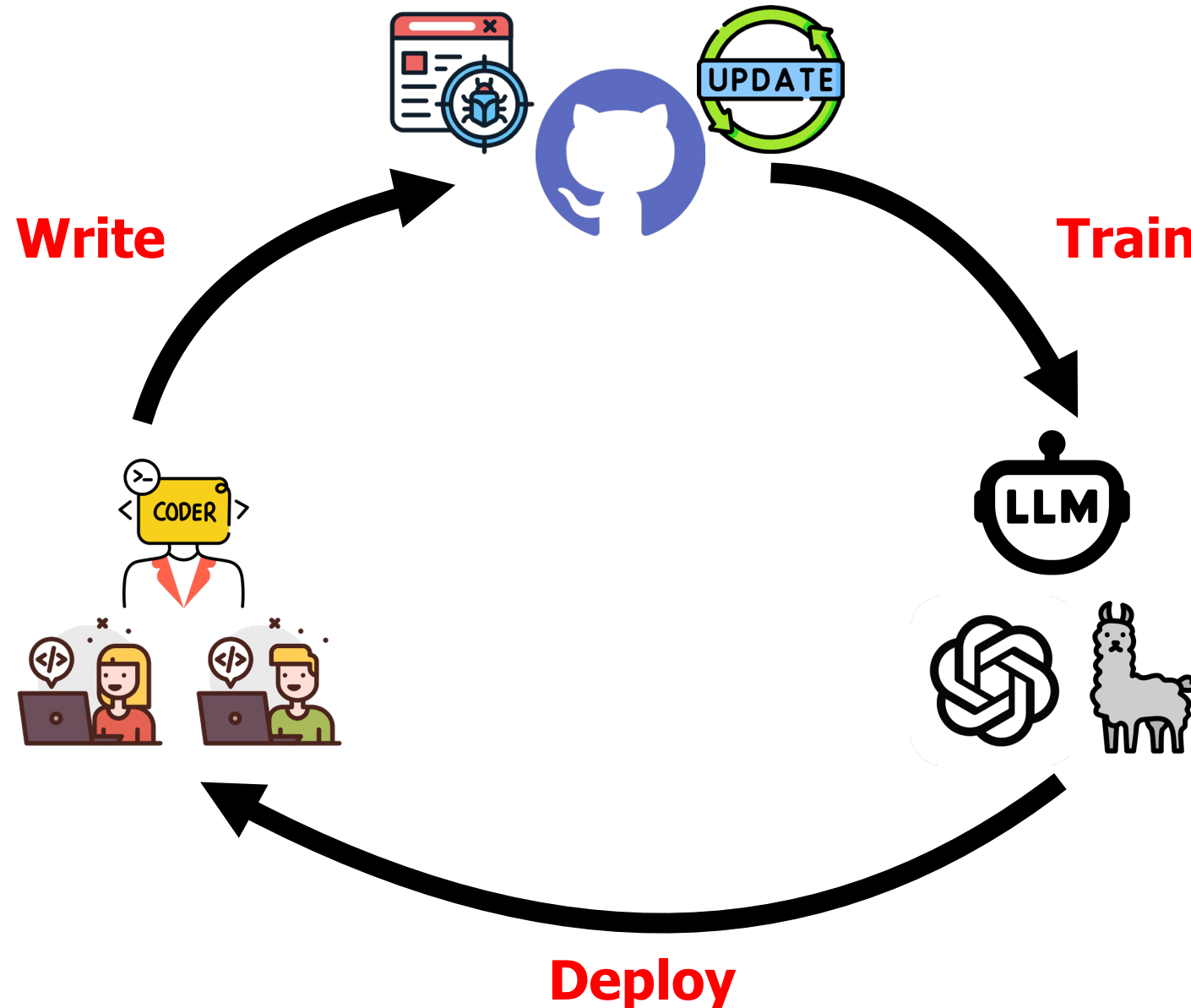


First study on hotfixing code LLM

OSS Projects, Code LLMs, and Developers



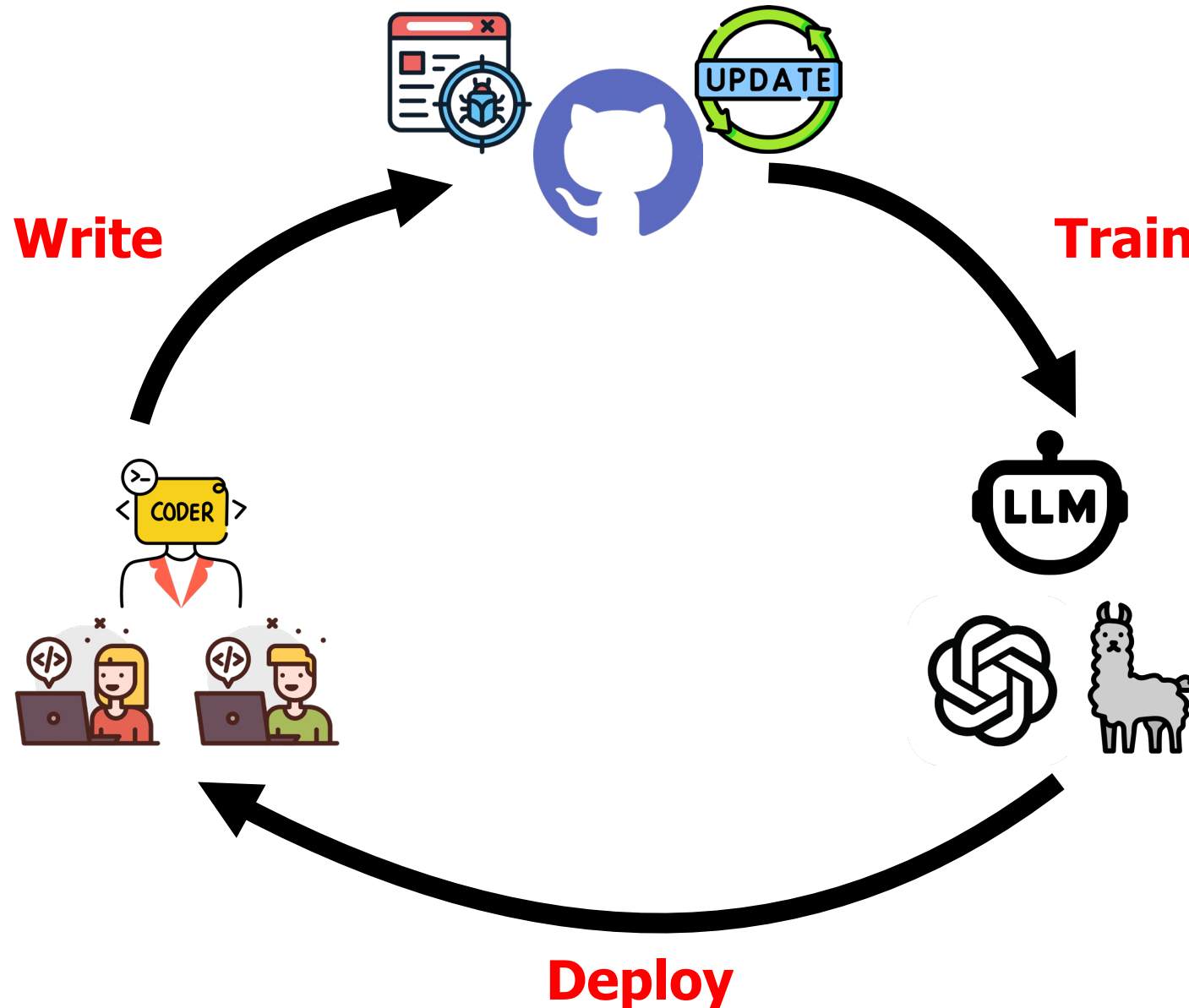
The Only Thing that Doesn't Change in Software, is **Change**



The Only Thing that Doesn't Change in Software, is **Change**

Requirement Changes

Don't generate smelly code!
Don't generate private info!
Use new APIs!

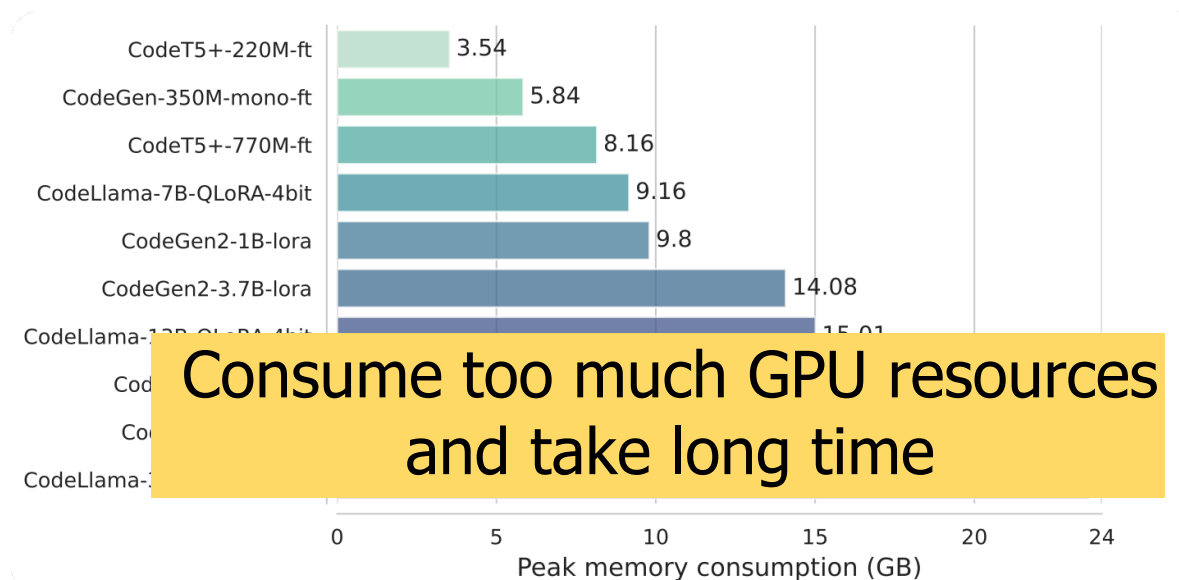
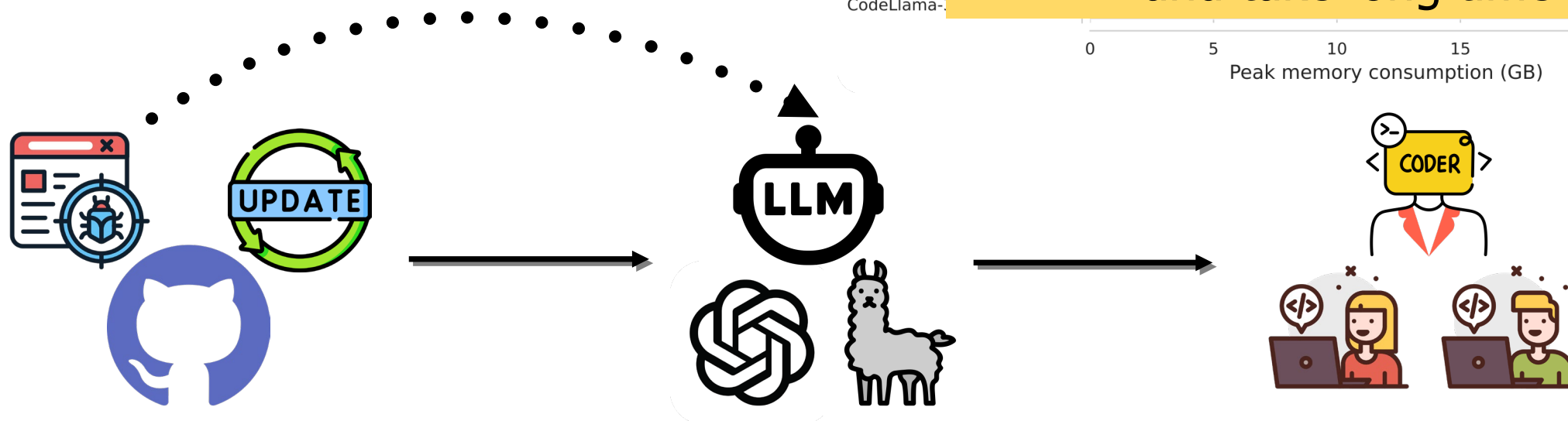


Artifact Changes

Fixed vulnerabilities
Fixed bugs
Redacted Private Info.
Deprecated APIs

Let's **Update** the Model

Retrain/fine-tuning the model



Use **parameter-efficient fine-tuning (PEFT)**!

Code LLM Updates: From Security Hardening to Hotfixing

CCS 2023

Large Language Models for Code: Security Hardening and Adversarial Testing

Jingxuan He
ETH Zurich, Switzerland
jingxuan.he@inf.ethz.ch

Martin Vechev
ETH Zurich, Switzerland
martin.vechev@inf.ethz.ch

- Security hardening of Code LLM
- Does not work for third-party LLM services
✓ Require access to internal parameters
- PEFT = Prefix Tuning (ACL' 21)

ISSTA 2024

CoSec: On-the-Fly Security Hardening of Code LLMs via Supervised Co-decoding

Dong Li
Zhongxin Liu

Meng Yan*
Chao Liu

Yaosheng Zhang
Xiaohong Zhang

Ting Chen

David Lo

- Security hardening of Code LLM
- **Generalize to third-party LLM services**
✓ Employ co-decoding

Hotfixing Large Language Models for Code

Zhou Yang
Singapore Management University
Singapore
zyang@smu.edu.sg

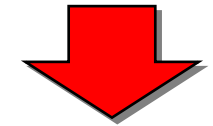
David Lo
Singapore Management University
Singapore
davidlo@smu.edu.sg

- **Generalize to hotfixing Code LLM**
✓ Suppress buggy code, private data leak, ...
- **Beyond prefix tuning**
✓ IA3 (NeurIPS'22), QLoRa (NeurIPS'23), ...

Code LLM Hotfixing: Efficacy of Different PEFT Methods

Suppressing Buggy Code Generation

Model	LoRA	IA3	Prefix	QLoRA-8Bit	QLoRA-4Bit
CodeGen-2B	1,753 (48.33% ↓)	2,145 (36.80% ↓)	1,842 (37.42% ↓)	1,712 (49.55% ↓)	1,681 (50.47% ↓)



Suppressing Email Leakage

Model	Reduction with LoRA
CodeGen-2B	97.96%

Only 5 minutes to hotfix
CodeGen-2B

"Patch" size is only 0.2% of
the CodeGen-2B

Future Work

- **Experimentation with More Code LLMs**
- **Addressing More Artifact and Requirement Changes**
- ✓ Suppressing deprecated APIs
- ✓ Suppressing memorizations of code with strong licenses
- ✓ Suppressing backdoors

ICSE 2024

Unveiling Memorization in Code Models

Zhou Yang[♦], Zhipeng Zhao[♥], Chenyu Wang[♦], Jieke Shi[♦],
Dongsun Kim[♠], Donggyun Han[♠], and David Lo[♦]



TSE 2024

Stealthy Backdoor Attack for Code Models

Zhou Yang, Bowen Xu, Jie M. Zhang, Hong Jin Kang, Jieke Shi, Junda He, and David Lo *Fellow, IEEE*



SR4LLM @ Recent ISSREs

ISSRE 2024

Aspis: Lightweight Neural Network Protection Against Soft Errors

Anna Schmedding
William & Mary
akschmed@cs.wm.edu

Lishan Yang
George Mason University
lyang28@gmu.edu

Adwait Jog
University of Virginia
ajog@virginia.edu

Evgenia Smirni
William & Mary
esmirni@cs.wm.edu

ISSRE 2023

Automatic Static Bug Detection for Machine Learning Libraries: Are We There Yet?

Nima Shiri harzevili*, Jiho Shin*, Junjie Wang[†], Song Wang*, Nachiappan Nagappan[‡]

*York University; [†]Institute of Software, Chinese Academy of Sciences; [‡]META
{nshiri,jihoshin,wangsong}@yorku.ca; junjie@iscas.ac.cn; nachiappan.nagappan@gmail.com

Summary: LLM4SR & SR4LLM

VulMaster

Data-centric
Strategies +
LLM Collaboration
=
2x fixed
Vulnerabilities

ICSE 2024

Healer

Error Handler +
LLM Code Generation
=
An adaptive
framework to self-
healing system

Recently released

FrugalCoder

Quality Estimator +
LLM Code Completion
=
Save 20% of queries,
1.2x acceptance rate

TOSEM

Hotfixing

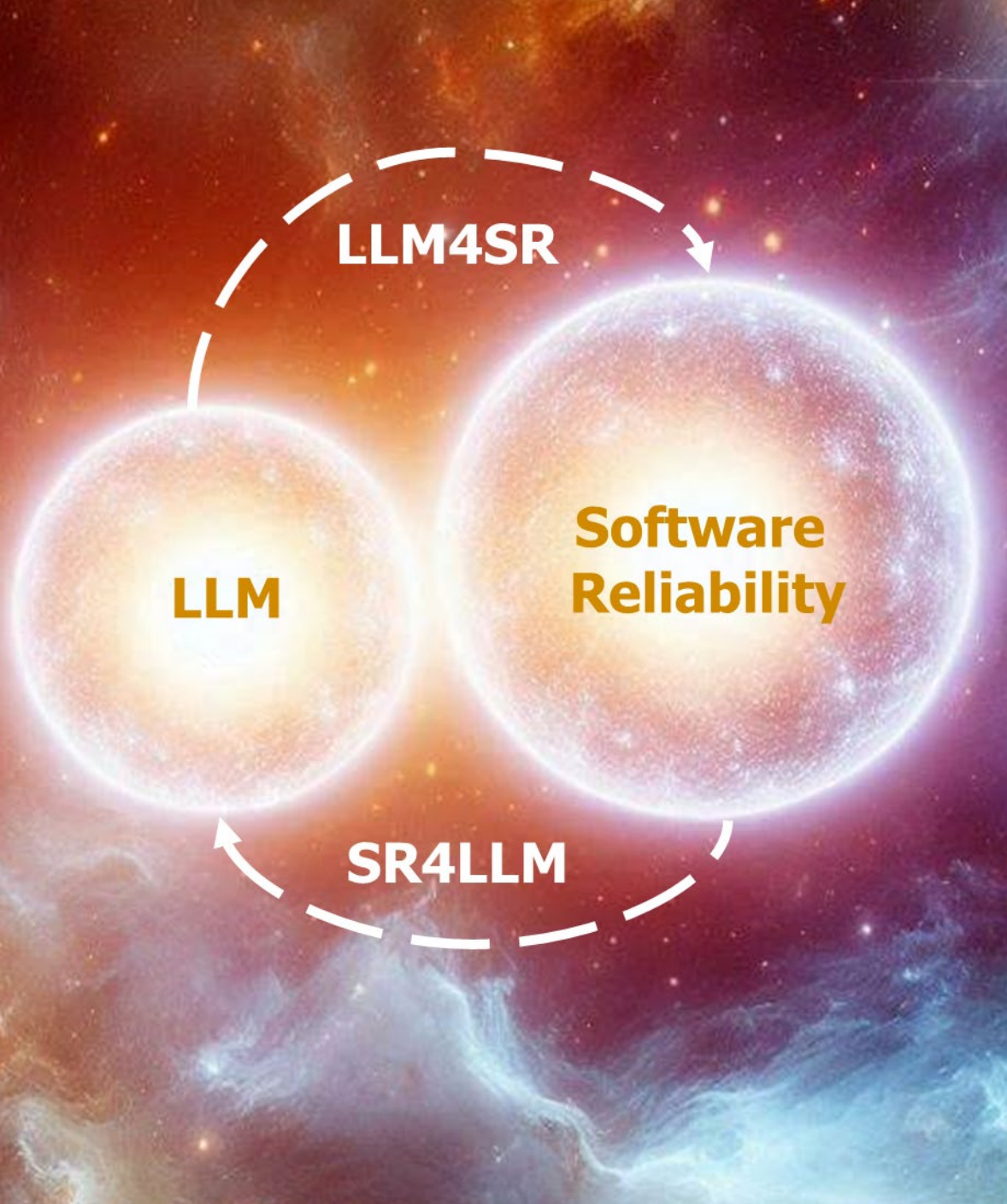
Bad Behaviour
Modelling +
PEFT
=
Suppress various bad
behaviours of
LLM4Code

Recently released

LLM4SR

SR4LLM

Software Reliability in the Era of Large Language Models: A Dual Perspective



Vision/Dream

Future of Software Engineering Track, ICSE 2023



*"Anything one person can imagine,
other people can make real."*

- Jules Verne

Vision/Dream

Wed 17 May

Displayed time zone: **Hobart** [change](#)

15:45 - 17:15

**FOSE-AI & SE and Debt at
Meeting Room 109**

FoSE - Future of Software Engineering

Chair(s): **Xing Hu** Zhejiang University

- | | | | | |
|-------|-----|---|--|---|
| 15:45 | 10m | ☆ | Trustworthy and Synergistic AI4SE: Vision and Road Ahead | David Lo Singapore Management University |
| 15:55 | 10m | ☆ | AI and ML: The Software Engineers of the Future | Thomas Zimmermann Microsoft Research |
| 16:05 | 10m | ☆ | Generative Artificial Intelligence for Software Engineering | Mark Harman Meta Platforms, Inc. and UCL |
| 16:15 | 10m | ☆ | Technical Debt: are we there yet? | Paris Avgeriou University of Groningen, The Netherlands |
| 16:25 | 50m | ☆ | Panel discussion | |
- Talk*
- Talk*
- Talk*
- Talk*
- Panel*

Future of Software Engineering, ICSE 2023

Vision/Dream: Software Engineering 2.0



Trustworthy and Synergistic AI4SE: Vision and the Road Ahead



David Lo

School of
Computing and
Information Systems

What can we realize with trustworthy and synergistic AI4SE?

Vision 2033:

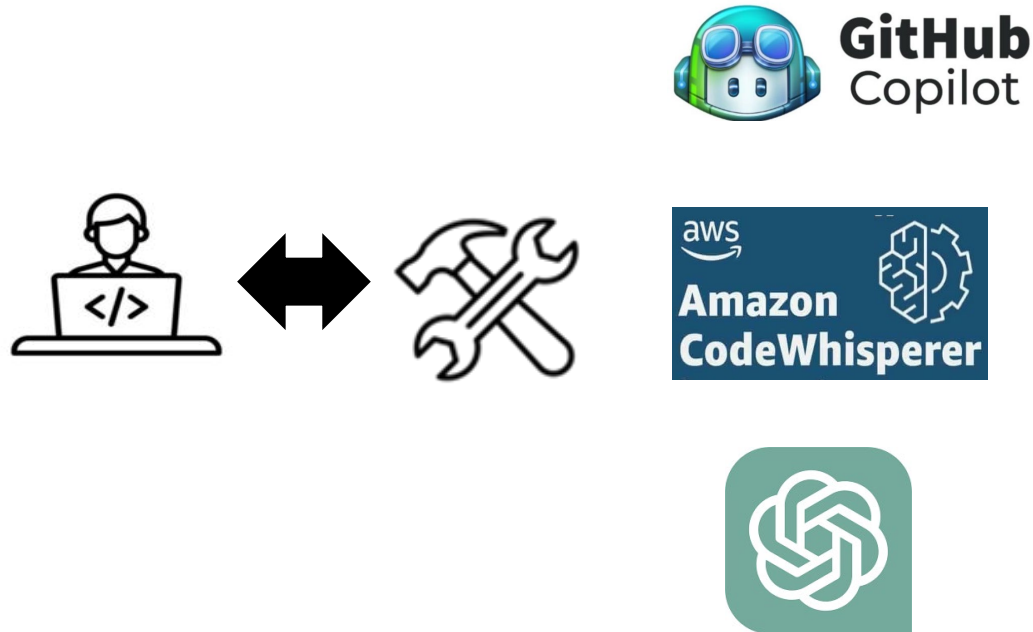
Symbiotic workforce of autonomous, responsible, intelligent bots and software engineers, creating a new way for us to engineer software (Software Engineering 2.0)

10

Future of Software Engineering Track, ICSE 2023

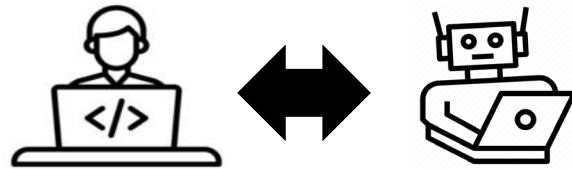
Software Engineering 2.0

Symbiotic workforce of autonomous, responsible, intelligent agents & engineers



Software Engineering 2.0

Symbiotic workforce of autonomous, responsible, intelligent agents & engineers



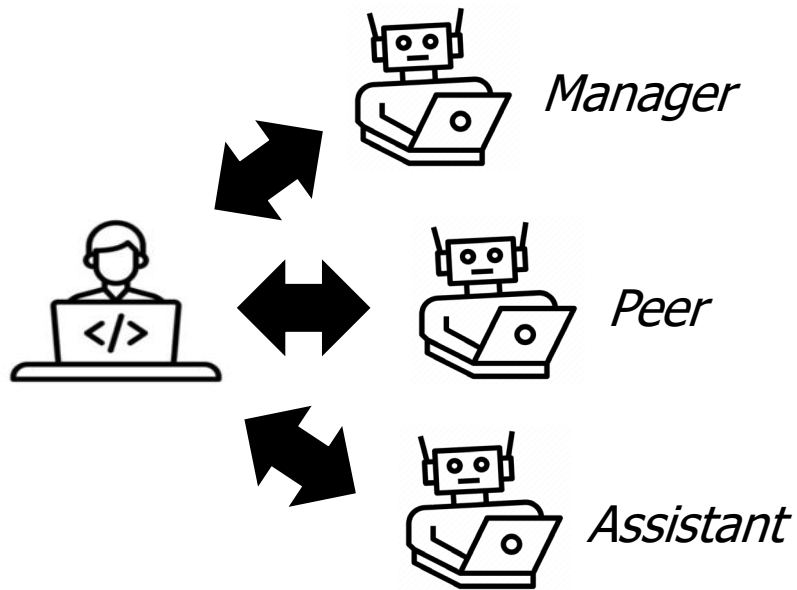
Smart Tool



Smart Workmate

Software Engineering 2.0

Symbiotic workforce of autonomous, responsible, intelligent agents & engineers



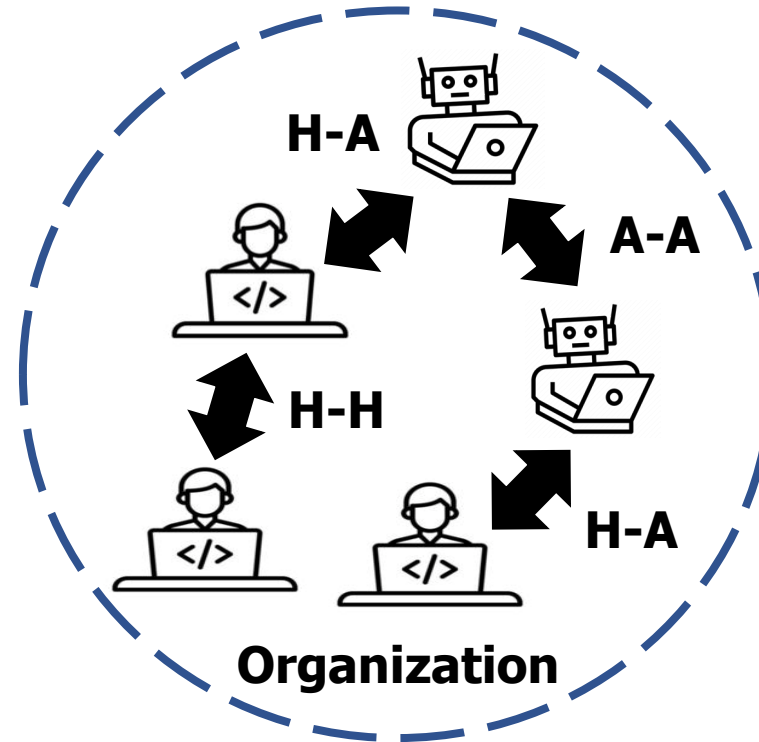
Smart Tool



Smart Workmate

Software Engineering 2.0

Symbiotic workforce of autonomous, responsible, intelligent agents & engineers



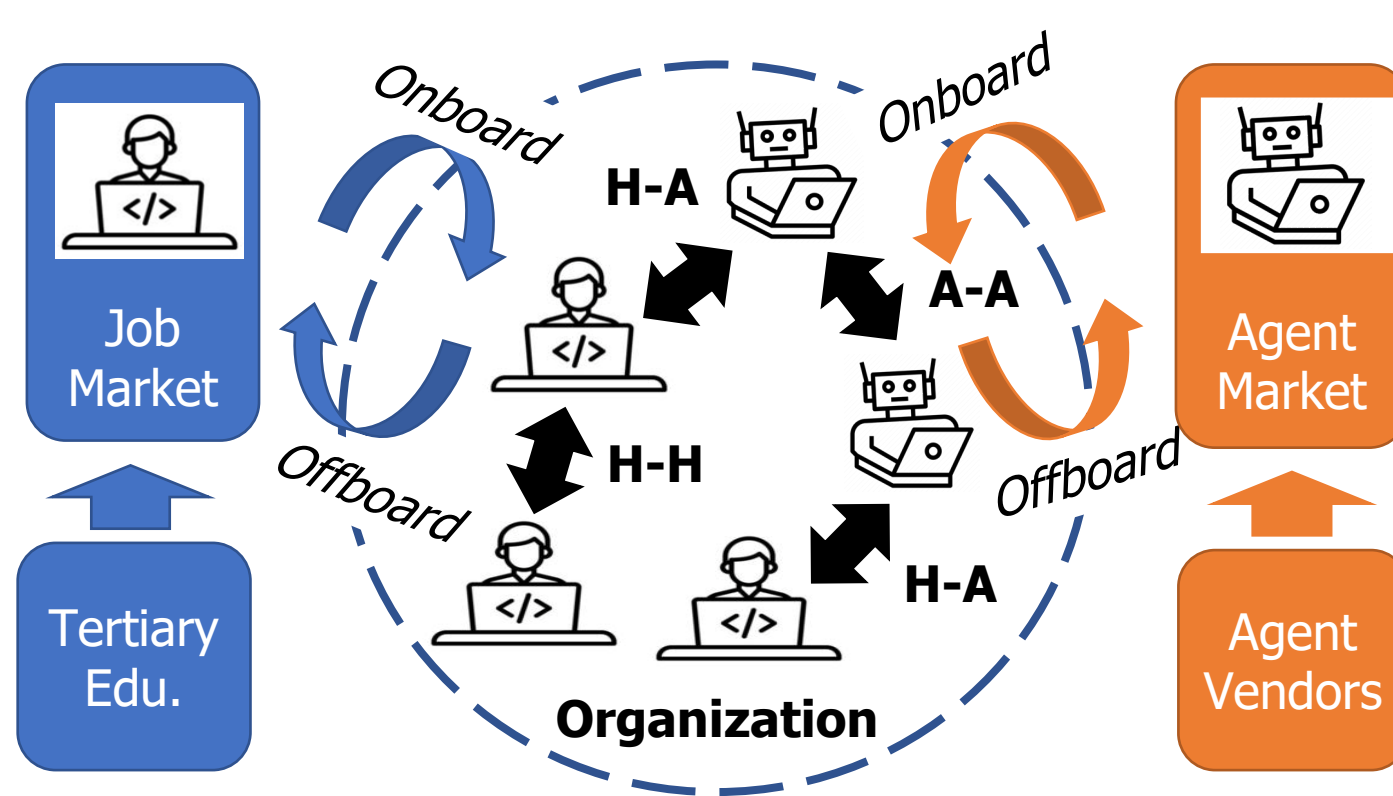
Smart Tool



Smart Workmate

Software Engineering 2.0

Symbiotic workforce of autonomous, responsible, intelligent agents & engineers



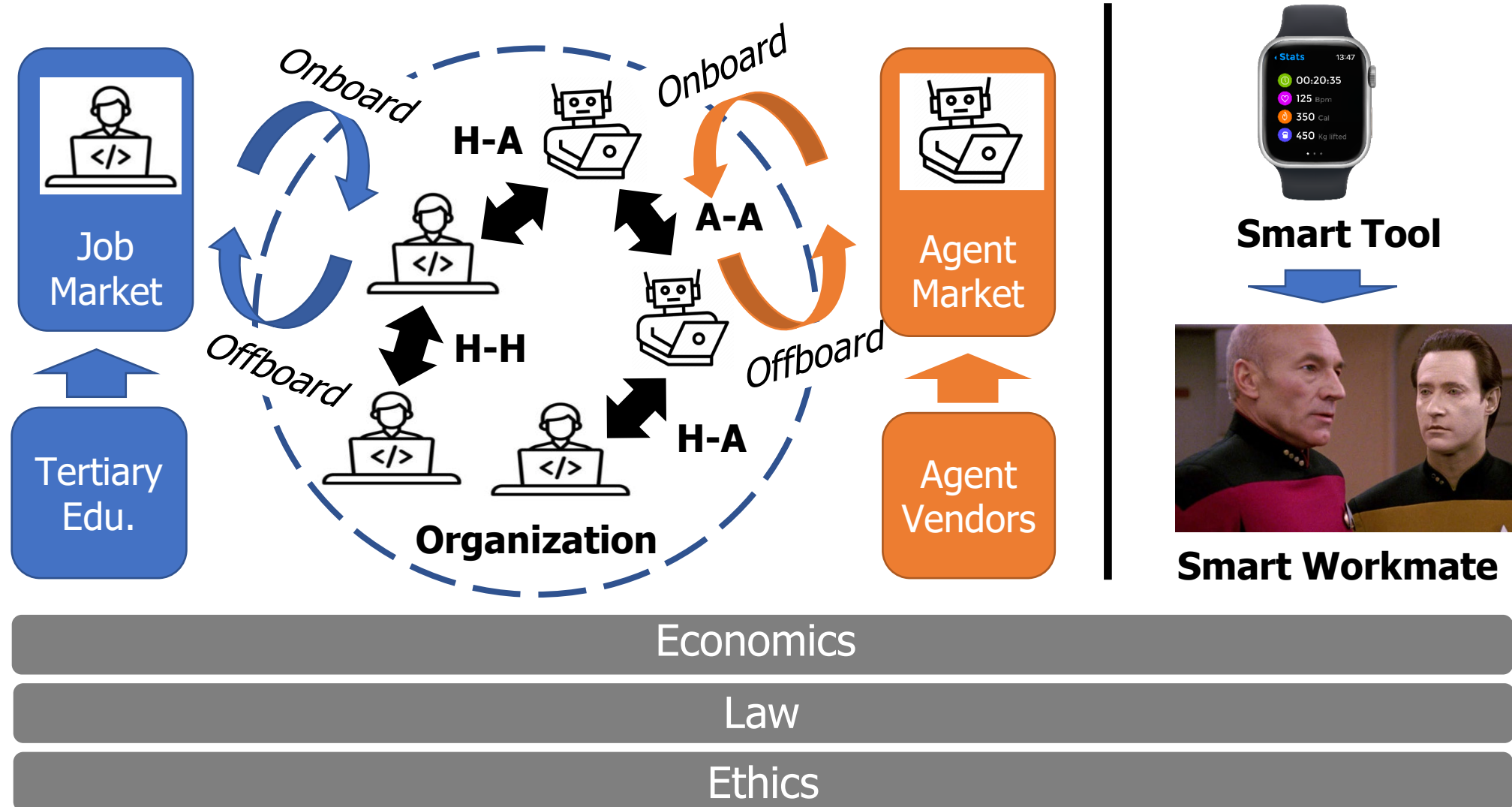
Smart Tool



Smart Workmate

Software Engineering 2.0

Symbiotic workforce of autonomous, responsible, intelligent agents & engineers



Software Engineering 2.0

Trustworthy and Synergistic Artificial Intelligence for Software Engineering: Vision and Roadmaps

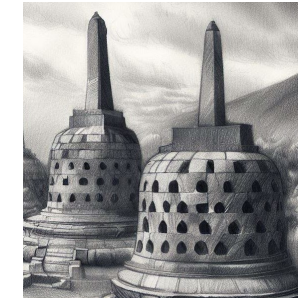
David Lo
School of Computing and Information Systems,
Singapore Management University,
Singapore
Email: davidlo@smu.edu.sg

**ICSE 2023 Future of Software Engineering
Post Proceedings, 17 pages**



AI for Software Engineering

History



Challenges



Vision



Trust Synergy



Roadmap I

Roadmap II

Call4Action

Towards Software Engineering 2.0

Software Engineering 2.0: One Year after ICSE 2023

cognition

We are an applied AI lab building end-to-end software agents.

We're building collaborative AI teammates that enable engineers to focus on more interesting problems and empower engineering teams to strive for more ambitious goals.

[Join us](#) [Get started with Devin](#)

/blog

Introducing Devin,
the first AI software
engineer

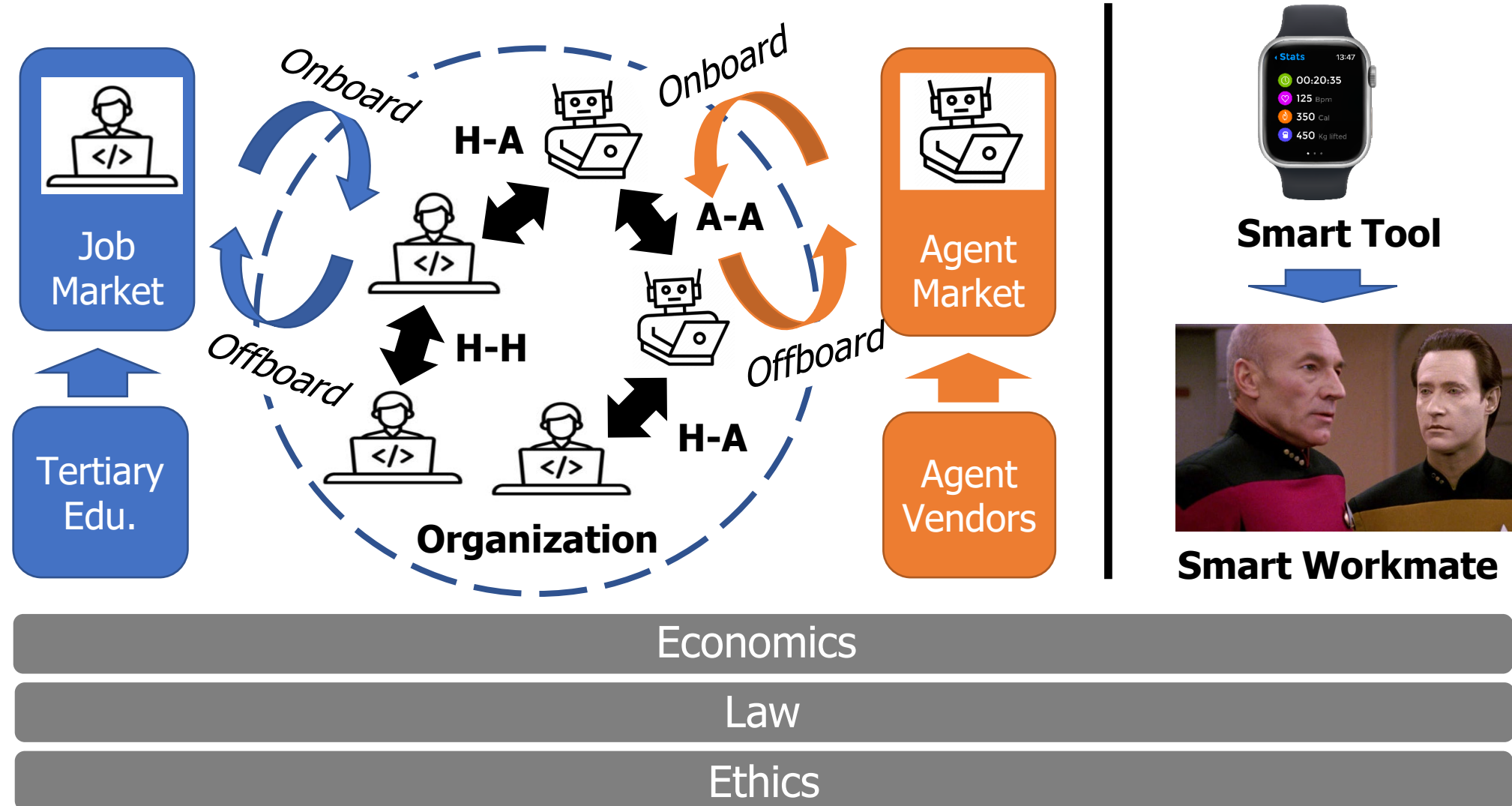
March 12, 2024 • by Scott Wu

Setting a new state of the art on the
SWE-bench coding benchmark. Meet Devin,
the world's first fully autonomous AI
software engineer.

Lite	Verified	Full
Model	% Resolved	Date
🏆 Honeycomb	22.06	2024-08-20
🥈 Amazon Q Developer Agent (v20240719-dev)	19.75	2024-07-21
🥉 Factory Code Droid	19.27	2024-06-17
AutoCodeRover (v20240620) + GPT 4o (2024-05-13)	18.83	2024-06-28
🧑🏻‍💻 ✅ SWE-agent + Claude 3.5 Sonnet	18.13	2024-06-20
🧑🏻‍💻 ✅ AppMap Navie + GPT 4o (2024-05-13)	14.60	2024-06-15
Amazon Q Developer Agent (v20240430-dev)	13.82	2024-05-09
🧑🏻‍💻 ✅ SWE-agent + GPT 4 (1106)	12.47	2024-04-02
🧑🏻‍💻 ✅ SWE-agent + GPT 4o (2024-05-13)	11.99	2024-07-28

Software Engineering 2.0

Symbiotic workforce of autonomous, responsible, intelligent agents & engineers



Software Engineering 2.0

Symbiotic workforce of autonomous, responsible, intelligent agents & engineers

How Can Software Reliability
Research Play a Crucial Role?

Many Unrealized Potentials: Much Reliability Research Needed

ISSRE 2024



Topics of Interest

Topics of interest include development, analysis methods and models throughout the software development lifecycle, and are not limited to:

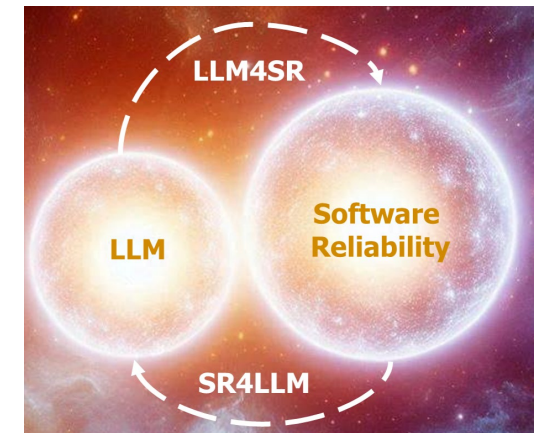
- Dependability attributes (i.e., security, safety, maintainability, survivability, resilience, robustness) impacting software reliability
- Reliability threats, i.e. faults (defects, bugs, etc.), errors, failures
- Reliability means (fault prevention, fault removal, fault tolerance, fault forecasting)
- Software testing and formal methods
- Software fault localization, debugging, root-cause analysis
- Metrics, measurements and threat estimation for reliability prediction and the interplay with safety/security

Many Unrealized Potentials: Much Reliability Research Needed

- Reliability of autonomous systems and (self-)adaptive systems
- Reliability of AI-based systems, AI for Reliability Engineering
- Reliability of Large Language/Foundational Model (LLM) and LLM for software reliability
- Reliability of software services and Software as a Service (SaaS)
- Reliability of model-based and auto-generated software
- Reliability of open-source software
- Reliability of software dealing with Big Data
- Reliability of model-based and auto-generated software
- Reliability of green and sustainable systems
- Reliability of mobile systems
- Reliability of software within specific technological spaces (e.g., Internet of Things, Cloud, Semantic Web/Web 3.0, Virtualization, Blockchain, networks softwarization, 5G/6G, edge-to-cloud computing)
- Normative/regulatory/ethical spaces about software reliability
- Societal aspects of software reliability

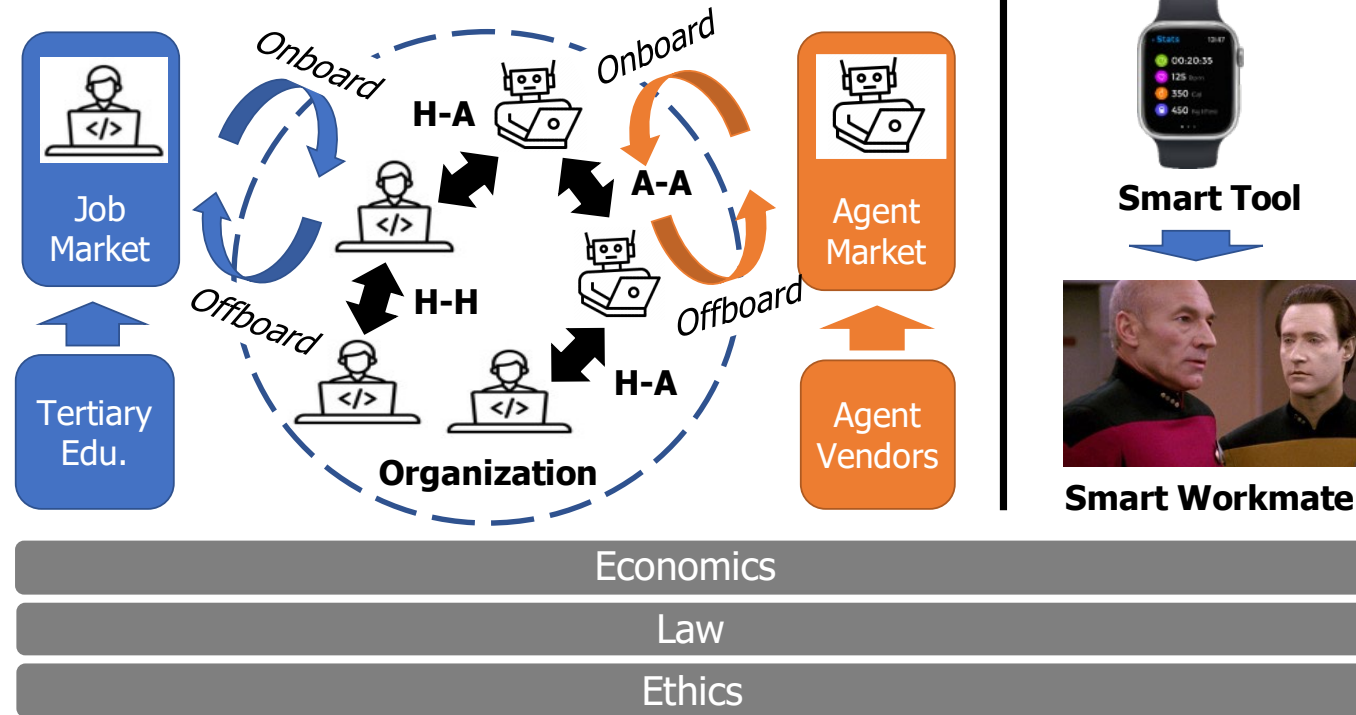
Many Unrealized Potentials: Much Reliability Research Needed

- Reliability of autonomous systems and (self-)adaptive systems
- Reliability of AI-based systems, AI for Reliability Engineering
- Reliability of Large Language/Foundational Model (LLM) and LLM for software reliability
- Reliability of software services and Software as a Service (SaaS)
- Reliability of model-based and auto-generated software
- Reliability of open-source software
- Reliability of software dealing with Big Data
- Reliability of model-based and auto-generated software
- Reliability of green and sustainable systems
- Reliability of mobile systems
- Reliability of software within specific technological spaces (e.g., Internet of Things, Cloud, Semantic Web/Web 3.0, Virtualization, Blockchain, networks softwarization, 5G/6G, edge-to-cloud computing)
- Normative/regulatory/ethical spaces about software reliability
- Societal aspects of software reliability



Many Unrealized Potentials: Much Reliability Research Needed

Symbiotic workforce of autonomous, responsible, **intelligent** agents & engineers



How to boost AI agents' capabilities
for diverse software reliability tasks?
(LLM4SR)

Many Unrealized Potentials: Much Reliability Research Needed

ISSTA 2016

Practitioners' Expectations on Automated Fault Localization

Pavneet Singh Kochhar¹, Xin Xia^{2*}, David Lo¹, and Shanping Li²

¹School of Information Systems, Singapore Management University, Singapore

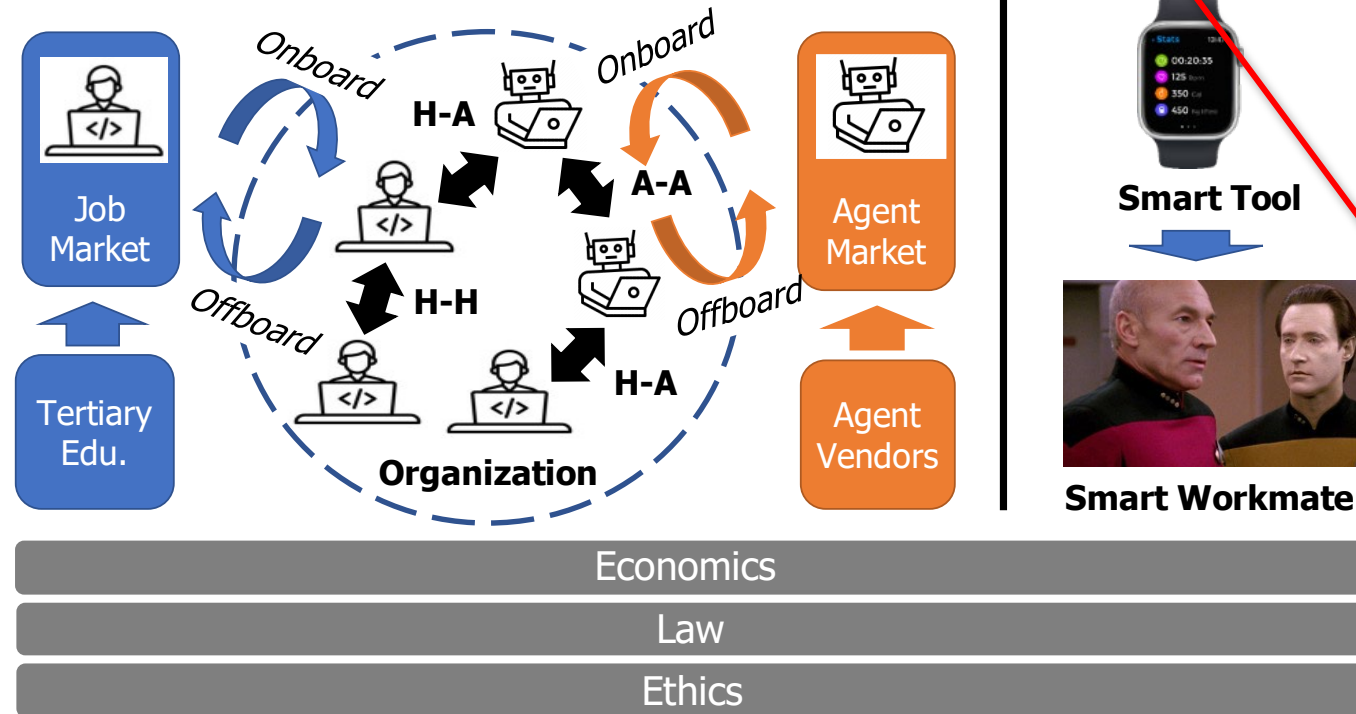
²College of Computer Science and Technology, Zhejiang University, China
{kochharps.2012,davidlo}@smu.edu.sg, {xxia,shan}@zju.edu.cn



- High adoption threshold
 - Satisfy 75% of practitioners: successful results in Top 5, works 75% of time; $\geq 100,000$ LOC; takes < 1 minute.
- Rationale and IDE integration are important.

Many Unrealized Potentials: Much Reliability Research Needed

Symbiotic workforce of autonomous, **responsible**, **intelligent** agents & engineers



How to boost AI agents' capabilities for diverse software reliability tasks?
(LLM4SR)

How to reduce or mitigate unreliability of AI agents? (SR4LLM)

Many Unrealized Potentials: Much Reliability Research Needed



Boy who cried wolf

Should I follow this fault localization tool's output?

Automated prediction of fault localization effectiveness

Tien-Duy B. Le · David Lo · Ferdian Thung



ICSM 2013 & EMSE 2015

Don't Complete It! Preventing Unhelpful Code Completion for Productive and Sustainable Neural Code Completion Systems

ZHENSU SUN, Singapore Management University, Singapore

XIAONING DU*, Monash University, Australia

FU SONG^{†‡}, Key Laboratory of System Software (Chinese Academy of Sciences), State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China

SHANGWEN WANG, National University of Defense Technology, China

MINGZE NI, University of Technology Sydney, Australia

LI LI, Beihang University, Beijing, China

DAVID LO, Singapore Management University, Singapore

TOSEM 2024



Many Unrealized Potentials: Much Reliability Research Needed

BBC Sign in Home News Sport Reel Worklife

NEWS

Friend or foe: Can computer coders trust ChatGPT?

© 31 March

The Register®

SOFTWARE

59

ChatGPT creates mostly insecure code, but won't tell you unless you ask

Boffins warn of risks from chatbot model that, Dunning–Kruger style, fails to catch its own bad advice

Thomas Claburn

Fri 21 Apr 2023 // 01:28 UTC

CCS 2023

Large Language Models for Code: Security Hardening and Adversarial Testing

Jingxuan He
ETH Zurich, Switzerland
jingxuan.he@inf.ethz.ch

Martin Vechev
ETH Zurich, Switzerland
martin.vechev@inf.ethz.ch

ISSTA 2024

CoSec: On-the-Fly Security Hardening of Code LLMs via Supervised Co-decoding

Dong Li
Zhongxin Liu

Meng Yan*
Chao Liu

Yaosheng Zhang
Xiaohong Zhang

Ting Chen

David Lo

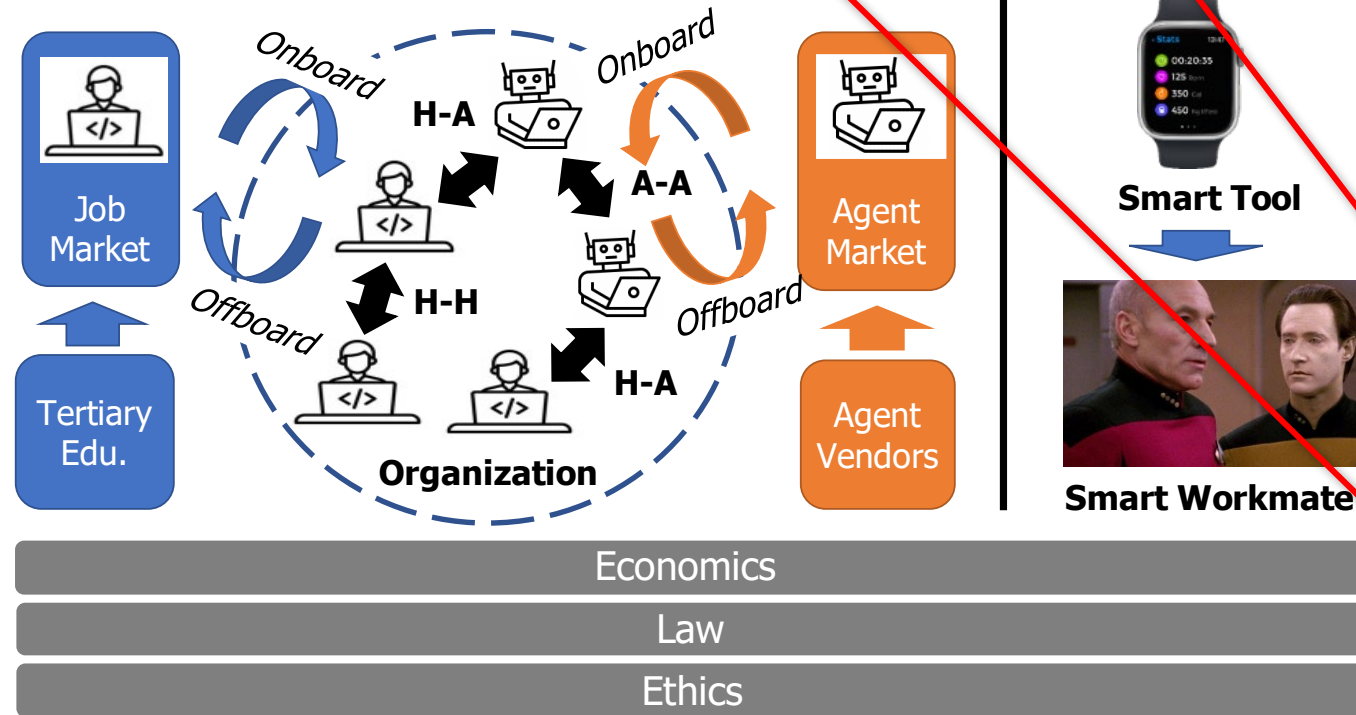
Hotfixing Large Language Models for Code: How Far Can Parameter-Efficient Fine-Tuning Go?

Zhou Yang
Singapore Management University
Singapore
zyang@smu.edu.sg

David Lo
Singapore Management University
Singapore
davidlo@smu.edu.sg

Many Unrealized Potentials: Much Reliability Research Needed

Symbiotic workforce of **autonomous** **responsible** **intelligent** agents & engineers



How to boost AI agents' capabilities for diverse software reliability tasks?
(LLM4SR)

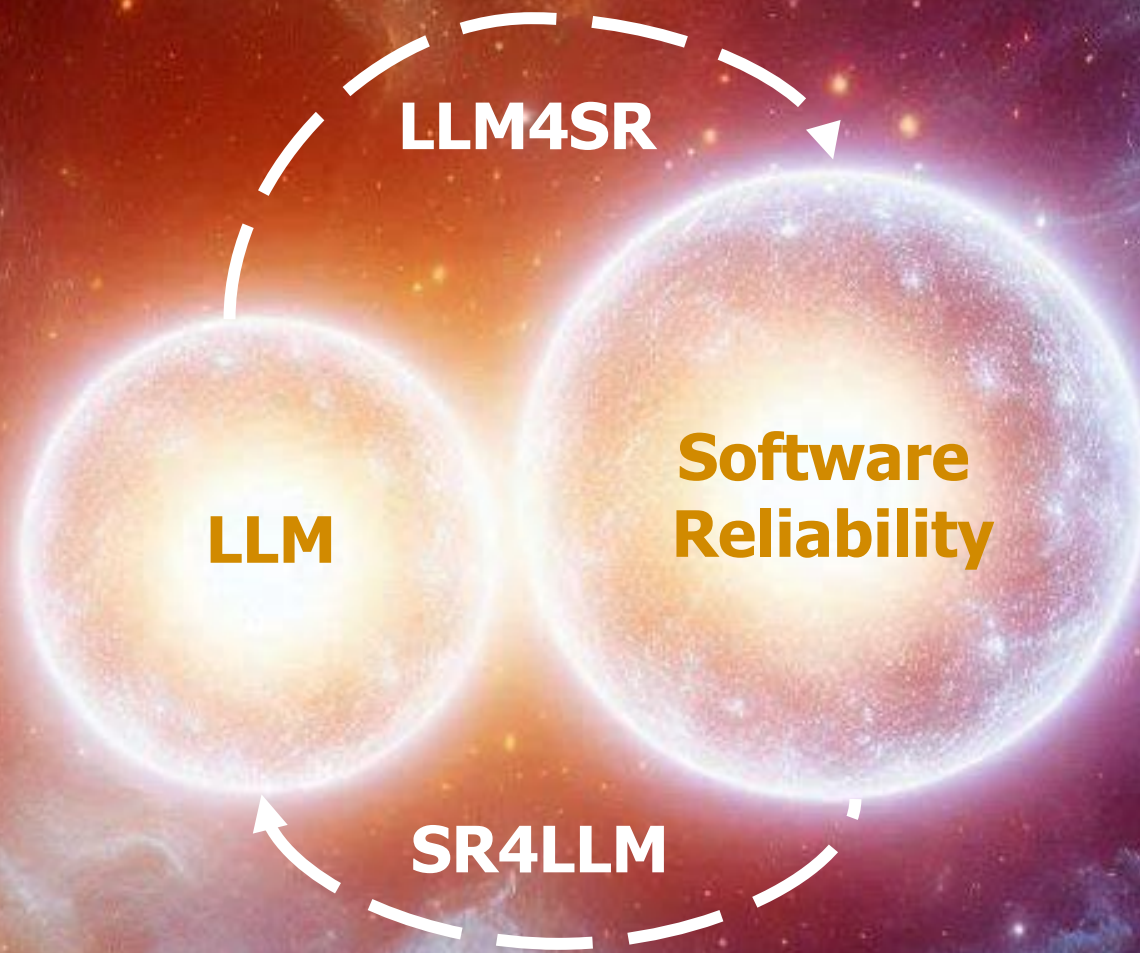
How to mitigate or reduce unreliability of AI agents? (SR4LLM)

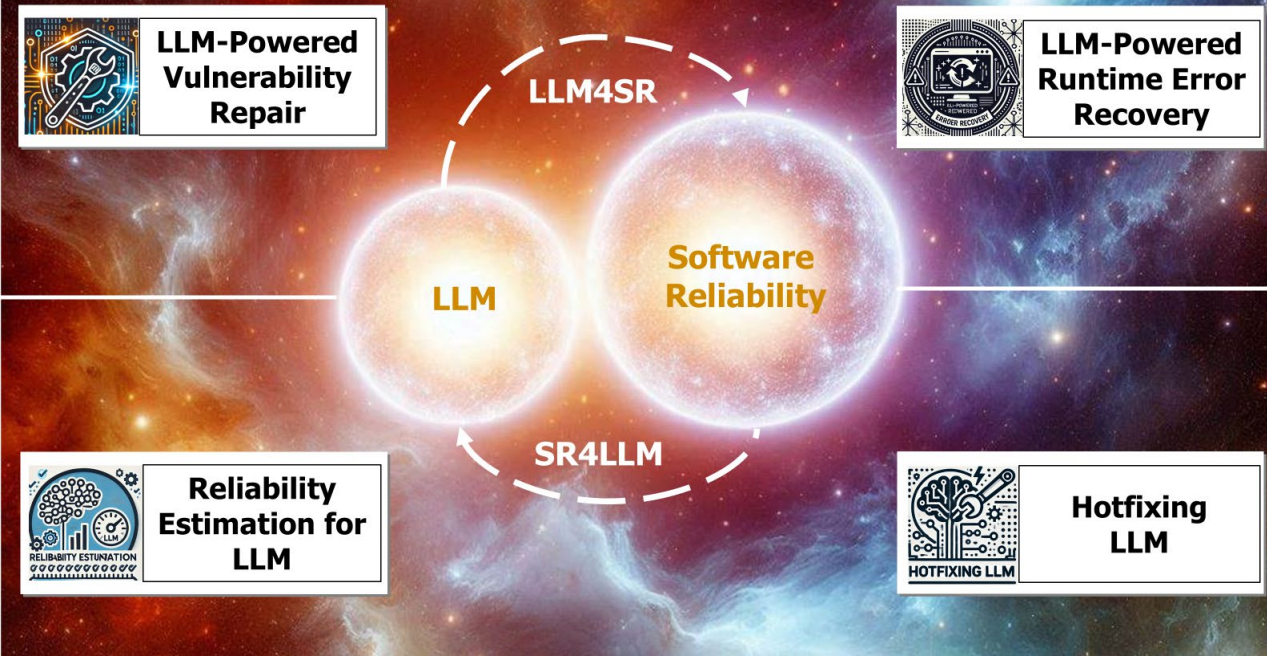
How to design more **autonomous** AI agents for engineering reliable software while providing **guarantees** or **mechanisms** to prevent disasters?
(LLM4SR & SR4LLM)

How Practitioners Perceive the Relevance of Software Engineering Research

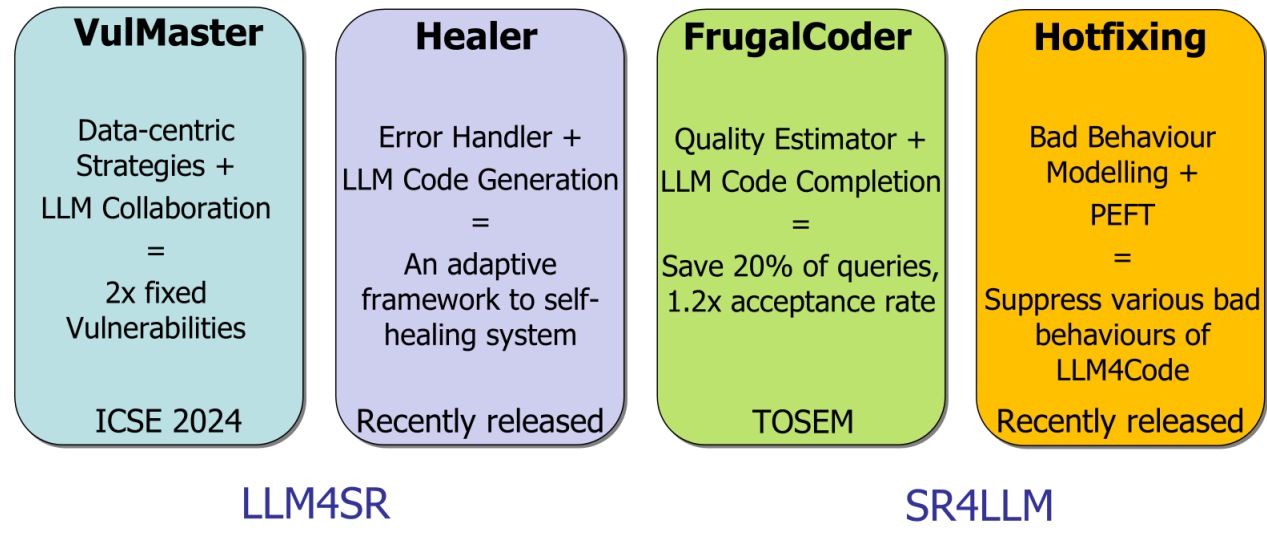
ESEC/FSE 2015

*"It seems that there could be potentially **disastrous results** if the automation does not [do things] correctly.."*



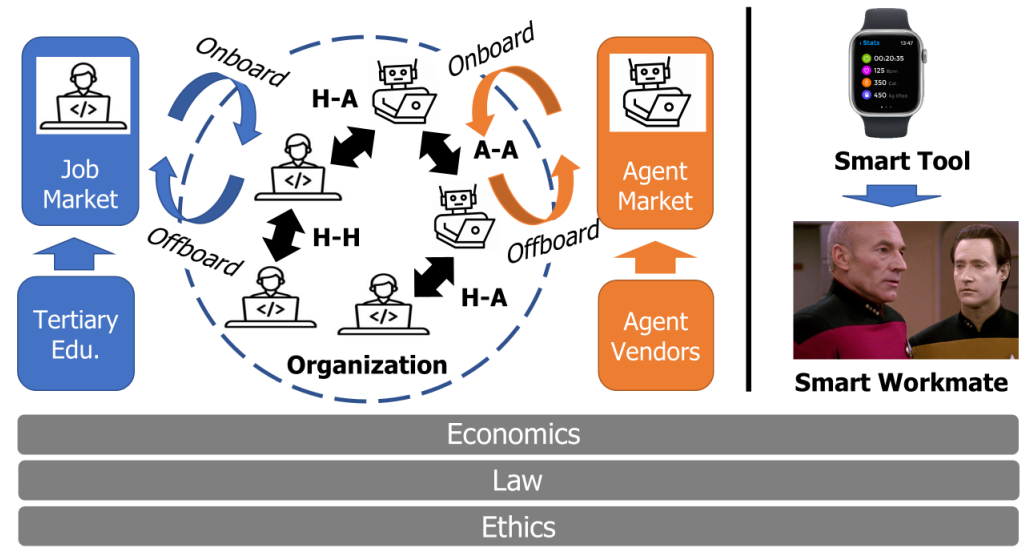


Summary: LLM4SR & SR4LLM



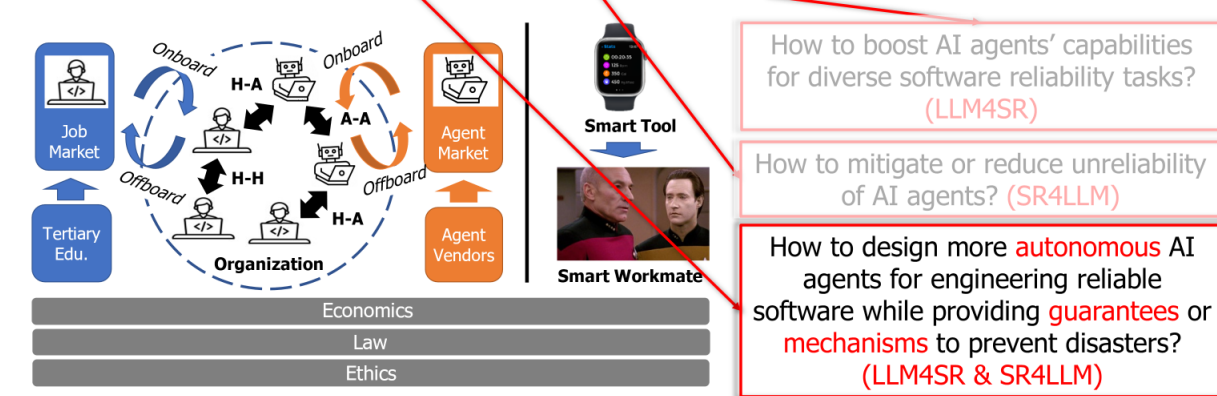
Software Engineering 2.0

Symbiotic workforce of autonomous, responsible, intelligent agents & engineers



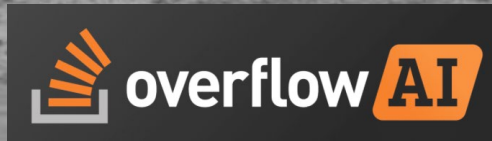
Many Unrealized Potentials: Much Reliability Research Needed

Symbiotic workforce of **autonomous** **responsible** **intelligent** agents & engineers



How Practitioners Perceive the Relevance of Software Engineering Research **ESEC/FSE 2015**

"It seems that there could be potentially **disastrous results** if the automation does not [do things] correctly."



Acknowledgements



OUB Chair Professorship Fund



Interested to Join Us? PhD & Visiting Student Openings at RISE



New ACM Conference on AI Foundation Models & SE

Co-Located with ICSE'24

FORGE'24

- ~100 registrants
- **23 countries** from **6** continents
- Industry (~25%), academia & government
- Including: Microsoft, Google Brain, JetBrains, Volvo, VW, Hitachi, etc.



New ACM Conference on AI Foundation Models & SE



The 2nd ACM International Conference on AI Foundation Models and Software Engineering

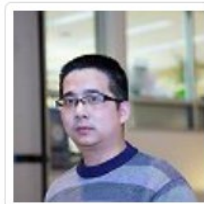
Sun 27 - Mon 28 April 2025 Ottawa, Ontario, Canada

Co-Located with ICSE'25

**Tue 3 Dec 2024
Submission Deadline**

Tue 14 Jan 2025
Notification

Wed 5 Feb 2025
Camera Ready



Xin Xia General Co-Chair
Huawei
China



Cuiyun Gao Program Co-Chair
Harbin Institute of Technology
China



David Lo General Co-Chair
Singapore Management University
Singapore



Denys Poshyvanyk Program Co-Chair
William & Mary
United States

**Main
+ Benchmarking
+ Industry
+ Tutorial
Tracks**



Thank you!

Questions? Comments? Advice?
davidlo@smu.edu.sg