Charting New Gold Mines: Expanding the World of MSR

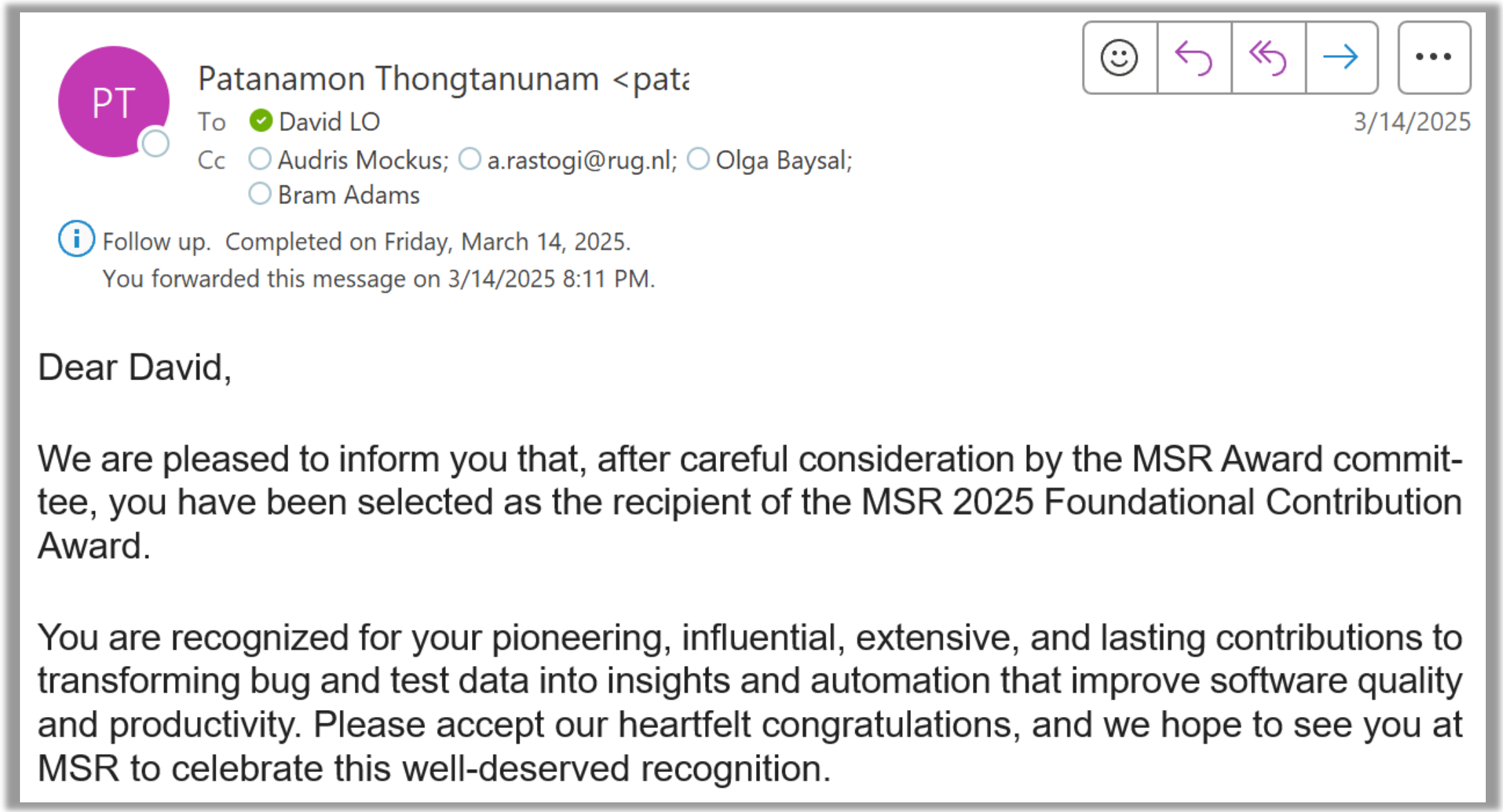
David Lo, FACM, FIEEE

MSR 2025, Ottawa, Canada, April 2025

SMU
SINGAPORE MANAGEMENT UNIVERSITY

School of
Computing and
Information Systems

Thank You Very Much!



PT

Patanamon Thongtanunam <pat

To David LO

Cc Audris Mockus; a.rastogi@rug.nl; Olga Baysal; Bram Adams

3/14/2025

Follow up. Completed on Friday, March 14, 2025.
You forwarded this message on 3/14/2025 8:11 PM.

Dear David,

We are pleased to inform you that, after careful consideration by the MSR Award committee, you have been selected as the recipient of the MSR 2025 Foundational Contribution Award.

You are recognized for your pioneering, influential, extensive, and lasting contributions to transforming bug and test data into insights and automation that improve software quality and productivity. Please accept our heartfelt congratulations, and we hope to see you at MSR to celebrate this well-deserved recognition.

Personal Journey: Programming Language (PL)

PROGRAMMING LANGUAGES AND SYSTEMS

School of Computing
National University of Singapore

My PhD Advisor:
Siau-Cheng Khoo

Mining Specifications

Glenn Ammons
Dept. of Computer Sciences
University of Wisconsin
Madison, Wisconsin, USA
ammons@cs.wisc.edu

Rastislav Bodík
Dept. of Computer Sciences
University of Wisconsin
Madison, Wisconsin, USA
bodik@cs.wisc.edu

James R. Larus
Microsoft Research
One Microsoft Way
Redmond, Washington, USA
larus@microsoft.com

MSR

POPL'02

Personal Journey: PL to Software Engineering (SE)

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

FSE'06

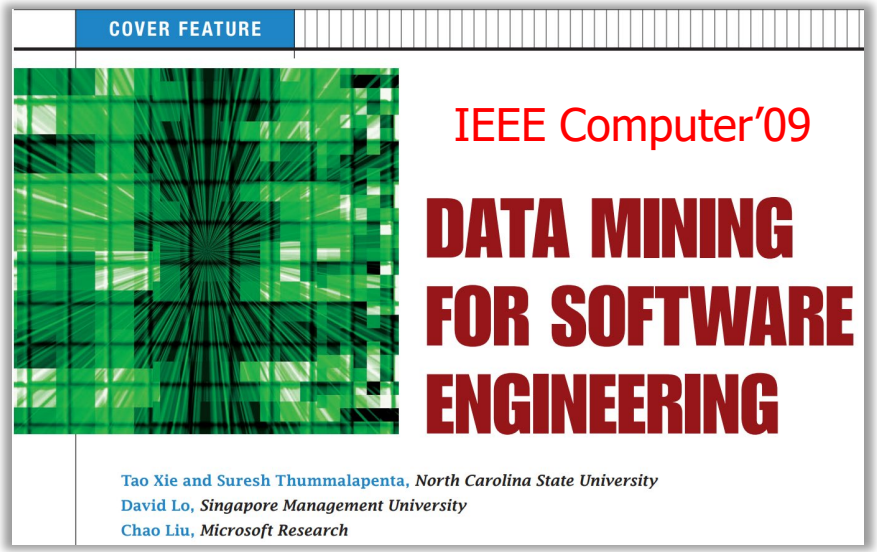
David Lo and Siau-Cheng Khoo
Department of Computer Science,National University of Singapore
{dlo,khoosc}@comp.nus.edu.sg

Efficient Mining of Iterative Patterns for Software Specification Discovery

KDD'07

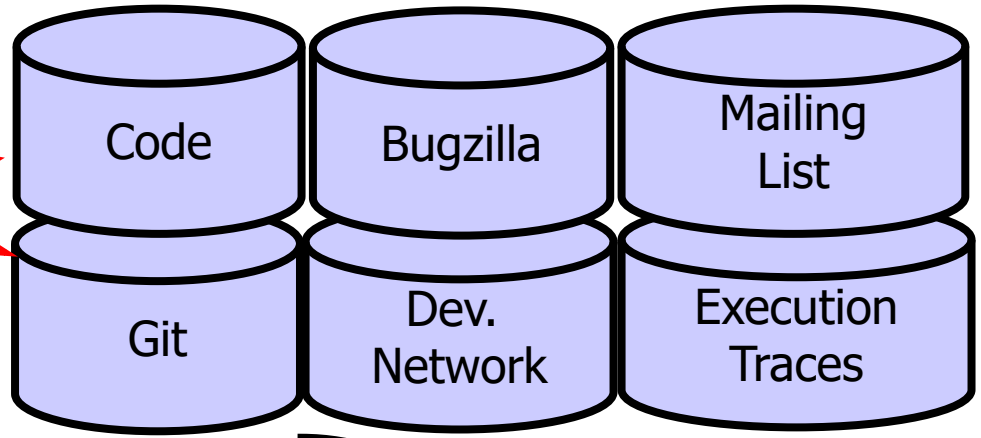
David Lo and Siau-Cheng Khoo
Department of Computer Science
National University of Singapore
{dlo,khoosc}@comp.nus.edu.sg

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



MSR

From SE to Mining Software Repositories (MSR)



MSR

Insight!

Automation!

Contributions to MSR

MSR 2012 June 2–3. Zurich, Switzerland

The 9th Working Conference on Mining Software Repositories

Home | Mining Challenge | Program | Venue | History

What Does Software Engineering Community Microblog About?

Yuan Tian, Palakorn Achananuparp, Ibrahim Nelman Lubis, David Lo, Ee-Peng Lim
Singapore Management University, Singapore
yuan.tian.2011@exchange.smu.edu.sg, palakorna@smu.edu.sg, lubisnelman@smu.edu.sg, davidlo@smu.edu.sg, eplim@smu.edu.sg

Are Faults Localizable?

Lucia, Ferdian Thung, David Lo, and Lingxiao Jiang
School of Information Systems, Singapore Management University
{lucia.2009,ferdianthung,davidlo,lxjiang}@smu.edu.sg

Contributions to MSR

Contributions to MSR

Dear David,

We are pleased to inform you that, after careful consideration by the MSR Award committee, you have been selected as the recipient of the MSR 2025 Foundational Contribution Award.

You are recognized for your pioneering, influential, extensive, and lasting contributions to transforming bug and test data into insights and automation that improve software quality and productivity. Please accept our heartfelt congratulations, and we hope to see you at MSR to celebrate this well-deserved recognition.

Reflection of the Past

Charting the Future

Reflection of the Past

Charting the Future

SE for Trustworthy AI (SE4TAI)

A Decade Ago

Few studies on Trustworthy AI (TAI) in the SE Community

Now

Yuriy Brun
Area Co-Chair for Software Engineering for AI
University of Massachusetts
United States

Wei Le
Area Co-Chair for Software Engineering for AI
Iowa State University
United States

Six Sessions on SE4AI

11:00 - 12:30 **SE for AI with Quality** 1
Chair(s): Jun Sun Singapore

3rd International Workshop on Responsible AI Engineering (RAIE 2025)

Can MSR Shed Light to Trustworthy AI?

MSR for Trustworthy AI (MSR4TAI)

An Empirical Study of Bugs in Machine Learning Systems

Ferdian Thung, Shaowei Wang, David Lo, and Lingxiao Jiang
School of Information Systems
Singapore Management University, Singapore
{*ferdianthung,shaoweiwang.2010,davidlo,lxjiang*}*@smu.edu.sg*

ISSRE'12

First work employing MSR to gain insight into ML bugs

MSR for Trustworthy AI (MSR4TAI)

Fabrizio Pastore @FabrizioPastore · Nov 2, 2022

David Lo receiving the Test of Time award @ISSREConf "An empirical Study of Bugs in Machine Learning System". A pioneering work that influenced many follow-on papers.

Aitor Arrieta
@aitorarrieta

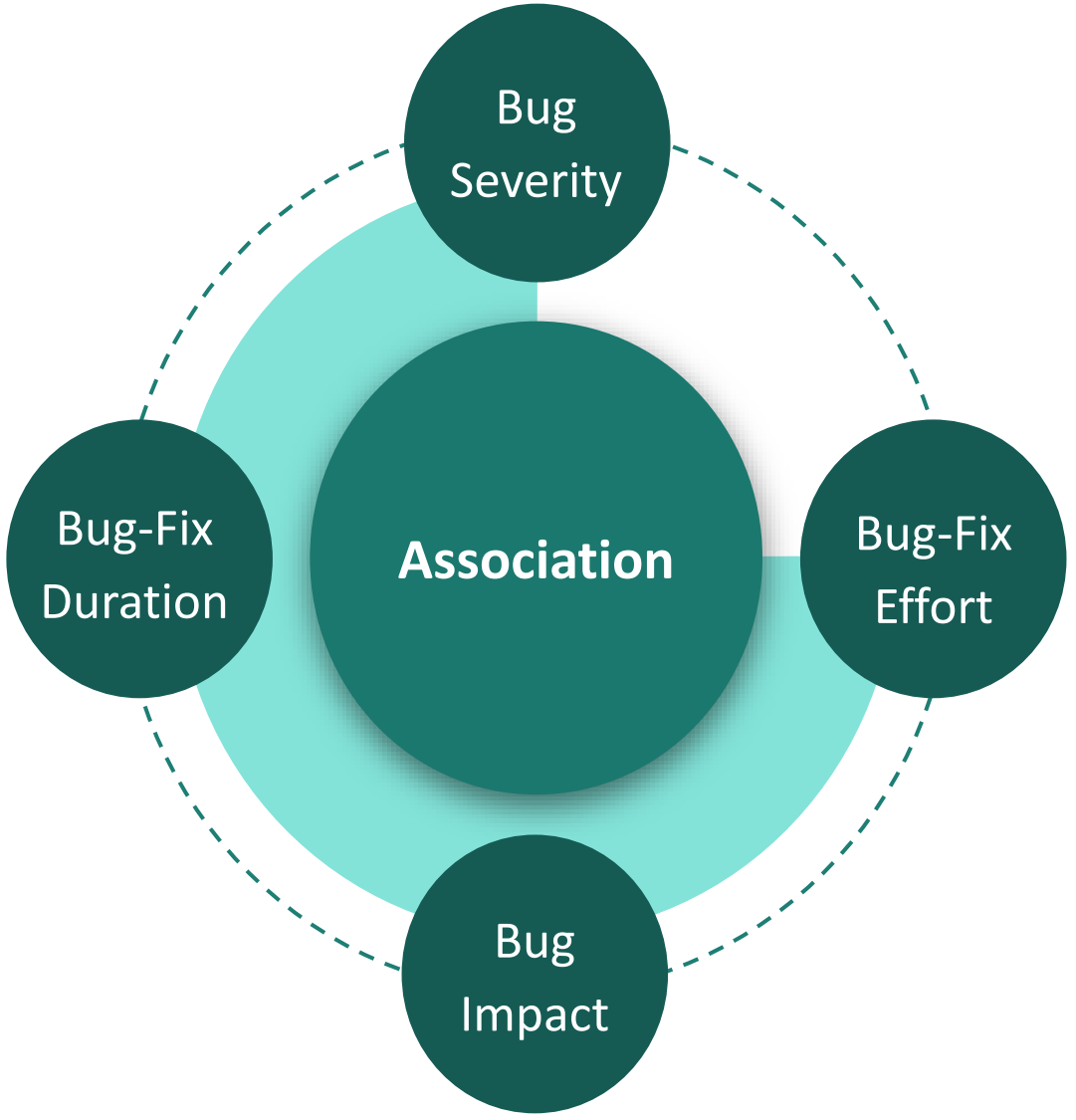
Test of time award for the paper “An empirical study of bugs in machine learning systems”, presented by @davidlo2015 at @ISSREConf. Interesting nowadays, but really visionary one of these kind of studies 10 years ago where ML wasn’t that important...

ISSRE’22 Most Influential Paper Award

Post-Mortem Bug Analysis for Machine Learning Systems

Bug Frequency & Density

Bug Type



Empirical Study Process

- Collect bug reports and fixes of popular open-source ML systems

Library	Size (LOC)	Bug Count	Active Dev.
MAHOUT	175,295	314	4.28 years
APACHE LUCENE	554,036	1,533	10.59 years
OPEN NLP	78,224	113	1.39 years

- Analyze the reports and fixes (200, 200, and 100 from Mahout, Lucene, and OpenNLP):
  - Manual categorization
  - Statistics computation

Empirical Study Process

Seaman et al. *ESEM 2008*

Chillarege et al. *TSE 1992*

Category	Definition
Algorithm/Method	The implementation of an algorithm/method does not follow the expected behavior.
Assignment/Init.	Error in assigning variable values.
Checking	Missing necessary checks that lead to an error or a wrong error message.
Data	Wrong use of data structure.
External Interface	Error in interfacing with other systems or users, such as using deprecated methods from other systems, requiring updates to own external interfaces for ease of usage, etc.
Internal Interface	Error in interfacing with another component of the same system, such as violating the contract of inheritance, wrong use of operations from other classes, etc.

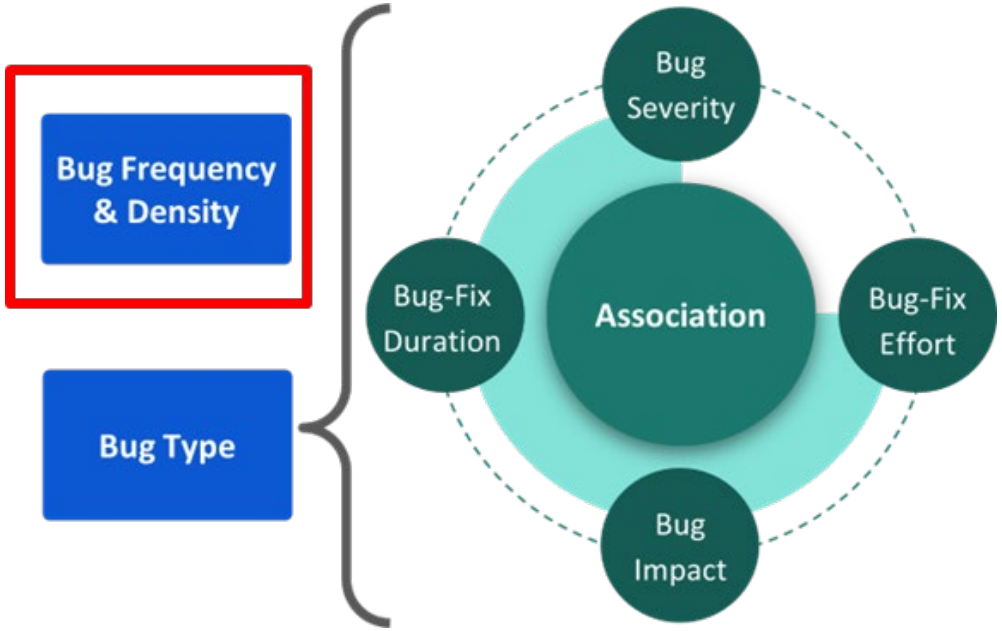
Empirical Study Process

Seaman et al. *ESEM 2008*

Chillarege et al. *TSE 1992*

Category	Definition
Logic	Incorrect expressions in conditional statements (e.g., if, while, etc.)
Non-functional	Violations in non-functional requirements such as improper variable or method names, wrong documentation to the implementation of a method, etc.
Timing/Optimization	Error that causes concurrency or performance issues, such as deadlock, high memory usage, etc.
Configuration	Error in non-code (e.g., configuration files) that affects functionality
Others	Other bugs that do not fall into one of the above categories.

I. Bug Frequency & Density

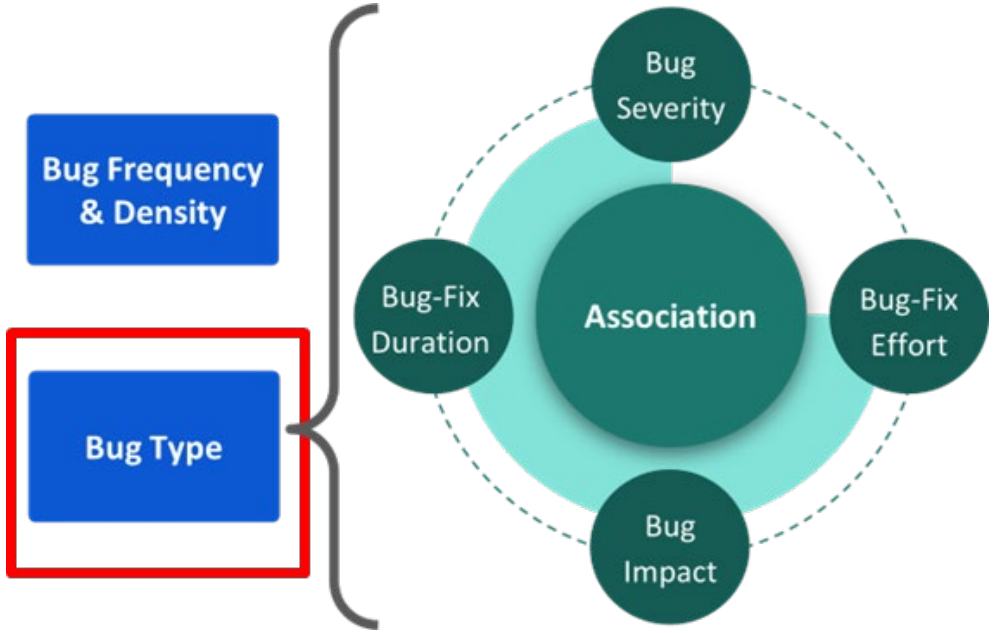


Libraries	#Bug per Year	#Bugs per kLOC
Mahout	73.36 bugs/year	1.79 bugs/kLOC
Lucene	144.76 bugs/year	2.77 bugs/kLOC
OpenNLP	95.68 bugs/year	1.45 bugs/kLOC

All three libraries have much higher bug densities than various operating systems

[Chou et al. SOSP 2001, Maji et al. ISSRE 2010, Palix et al. ASPLOS 2011]

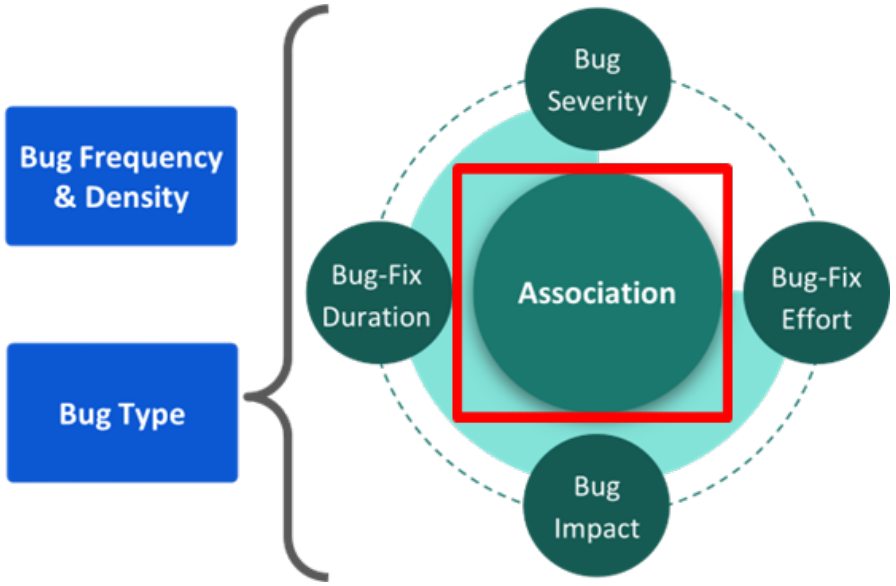
II. Distribution of Bug Types



Type	Count	Percentage
algorithm/method	113	22.60%
non-functional	78	15.60%
assignment/initialization	65	13.00%
checking	57	11.40%
external interface	38	7.60%
internal interface	38	7.60%
data	28	5.60%
logic	27	5.40%
configuration	27	5.40%
timing/optimization	24	4.80%
others	5	1%

The most common categories of bugs are: algorithm/method, non-functional, and assignment/initialization

IIIA. Bug Severity



Severity	Count	Proportion
Blocker	7	1.4%
Critical	7	1.4%
Major	288	57.6%
Minor	157	31.4%
Trivial	41	8.2%

Default severity label in Apache JIRA

Severe bugs (blocker and critical) are mostly categorized as algorithm/method (4/14), assignment/initialization (3/14), and checking (3/14)

IIIB. Bug-Fixing Duration

Type	Min	Max	Mean	Median
algorithm/method	0.0022	2433.7033	91.7238	3.8740
assignment/initialization	0.0003	160.9271	9.9160	0.5000
checking	0.0017	195.7766	17.1335	1.1175
configuration	0.0016	195.9011	22.3583	2.8032
data	0.0014	676.3825	40.8279	2.2666
external interface	0.0006	1700.5871	69.2463	0.4275
internal interface	0.0029	1688.5275	93.3543	2.4852
logic	0.0016	59.8305	6.8892	1.2537
non-functional	0.0006	1330.4142	47.8057	0.6949
timing/optimization	0.0017	569.7309	71.6649	3.3596
others	0.0005	1.3128	0.3344	0.0594

Bugs related to internal interface, algorithm/method, and timing/optimization take the longest to be fixed

IIIC. Bug-Fixing Effort

The number of revisions committed into the version control system to fix the bug to measure the difficulty of fixing a bug

Type	Min	Max	Mean	Median
algorithm/method	1	9	1.9646	1
assignment/initialization	1	4	1.2923	1
checking	1	11	1.7544	1
configuation	1	5	1.7037	1
data	1	5	1.6429	1
external interface	1	6	1.7368	1
internal interface	1	8	1.8947	1
logic	1	4	1.2593	1
non-functional	1	6	1.5641	1
timing/optimization	1	7	2.4167	2
others	1	1	1.0000	1

Timing/optimization, algorithm/method, and internal interface bugs require more revisions to fix them

IIID. Bug Impact

The number of files changed to fix the bug to measure the bug impact

Type	Min	Max	Mean	Median
algorithm/method	1	117	11.7434	3
assignment/initialization	1	74	5.2459	2
checking	1	115	7.2982	2
configuration	1	40	5.2692	1
data	1	124	12.3571	3
external interface	1	92	13.3514	3
internal interface	1	52	7.6842	4.5
logic	1	15	2.7778	2
non-functional	1	676	16.7143	2
others	1	47	10.2000	1
timing/optimization	1	69	11.5833	5

Non-functional, external interface, and data bugs impact the greatest number of files that require fixing

Take-home Messages on Bugs in ML Systems

- **Algorithm/method bugs**
  - Has the greatest number of bugs
  - Has many severe bugs
  - Has long bug-fixing durations
  - Requires much effort to fix
- **Timing/optimization bugs**
  - Has the least number of bugs, but
  - Has long bug-fixing durations
  - Requires the most effort to fix
- **Non-functional, external interface, and data bugs**
  - Affect many files
- **Configuration bugs (new category to Seaman et al.'s)**
  - Appear many times and are non-trivial to fix
    (harder to fix than many other categories)

Follow Up Work

1. Motivate future work analyzing bugs in machine learning systems

An empirical study on TensorFlow program bugs
Y Zhang, Y Chen, SC Cheung, Y Xiong… - Proceedings of the 27th …, 2018 - dl.acm.org
Deep learning applications become increasingly popular in important domains such as self-driving

A comprehensive study on deep learning bug characteristics
MJ Islam, G Nguyen, R Pan, H Rajan - … of the 2019 27th ACM Joint …, 2019 - dl.acm.org
Deep learning has gained substantial popularity in recent years. Developers mainly rely on librar

Taxonomy of real faults in deep learning systems
N Humbatova, G Jahangirova, G Bavota… - Proceedings of the …, 2020 - dl.acm.org
The growing application of deep neural networks in safety-critical domains makes the analys

An empirical study on program failures of deep learning jobs
R Zhang, W Xiao, H Zhang, Y Liu… - 2020 IEEE/ACM 42nd …, 2020 - ieeexplore.ieee.org
Deep learning has made significant achievements in many application areas. To train and …

An empirical study on real bugs for machine learning programs
X Sun, T Zhou, G Li, J Hu, H Yang… - 2017 24th Asia-Pacific …, 2017 - ieeexplore.ieee.org
Due to the availability of various open source Machine Learning (ML) tools and libraries, e …

An empirical study on bugs inside tensorflow
L Jia, H Zhong, X Wang, L Huang, X Lu - International Conference on …, 2020 - Springer
In recent years, deep learning has become a hot research topic. Although it achieves incredible positive results in some scenarios, bugs inside deep learning software can …

Follow Up Work

An exploratory characterization of bugs in covid-19 software projects
A Rahman, E Farhana - arXiv preprint arXiv:2006.00586, 2020 - arxiv.org
Context: The dire consequences of the COVID-19 pandemic has influenced development of
COV

Toward understanding compiler bugs in GCC and LLVM
C Sun, V Le, Q Zhang, Z Su - … of the 25th International Symposium on …, 2016 - dl.acm.org
Compilers are critical, widely-used complex software. Bugs in them have significant impact,
and c

Bug characteristics in blockchain systems: a large-scale empirical study
Z Wan, D Lo, X Xia, L Cai - 2017 IEEE/ACM 14th International …, 2017 - ieeexplore.ieee.org
Bugs severely hurt blockchain system dependability. A thorough understanding of
blockch

An empirical study of bugs in industrial financial systems
X Xuan, X Zhao, Y Wang, S Li - IEICE TRANSACTIONS on …, 2015 - search.ieice.org
Bugs in industrial financial systems have not been extensively studied. To address this gap,
rPro

A comprehensive study of deep learning compiler bugs
Q Shen, H Ma, J Chen, Y Tian, SC Cheung… - Proceedings of the 29th …, 2021 - dl.acm.org
There are increasing uses of deep learning (DL) compilers to generate optimized code,
nal …

Gang of eight: A defect taxonomy for infrastructure as code scripts
A Rahman, E Farhana, C Parnin… - 2020 IEEE/ACM 42nd …, 2020 - ieeexplore.ieee.org
Defects in infrastructure as code (IaC) scripts can have serious consequences, for example,
creating large-scale system outages. A taxonomy of IaC defects can be useful for …

2. Motivate studies on bugs in other emerging domain-specific software systems

Follow Up Work

Multiple-implementation testing of supervised learning software
O Alebiosu - 2017 - ideals.illinois.edu

Mac
man

Testing machine learning code using polyhedral region
MS Ahmed, F Ishikawa, M Sugiyama - … of the 28th ACM Joint Meeting on …, 2020 - dl.acm.org

To da
gene

Deriving and evaluating a fault model for testing data science applications
A Aftab Jilani, S Sherin, S Ijaz… - Journal of Software …, 2022 - Wiley Online Library

Data science (DS) applications not only suffer from traditional software faults but may also
suffe

Testing of Autonomous Driving Systems: where are we and where should we go?
G Lou, Y Deng, X Zheng, T Zhang, M Zhang - arXiv preprint arXiv …, 2021 - arxiv.org

Autonomous driving shows great potential to reform modern transportation. However, its
itional …

Test-Case Generation for Finding Neural Network Bugs
M Rezaalipour, CA Furia - arXiv preprint arXiv:2112.05567, 2021 - arxiv.org

As neural networks are increasingly included as core components of safety-critical systems,

Deep Learning Framework Fuzzing Based on Model Mutation
X Shen, J Zhang, X Wang, H Yu… - 2021 IEEE Sixth …, 2021 - ieeexplore.ieee.org

Deep learning (DL) frameworks are widely used for neural network model training and prediction in a lot of areas such as computer vision, natural language processing, medical …

3. Motivate work on testing machine learning systems

MSR Can Shed Light to New Domains (e.g., Trustworthy AI)

SE for Trustworthy AI (SE4TAI)

A Decade Ago

Few studies on Trustworthy AI (TAI) in the SE Community

Now

Yuriy Brun
Area Co-Chair for Software Engineering for AI
University of Massachusetts
United States

Wei Le
Area Co-Chair for Software Engineering for AI
Iowa State University
United States

Six Sessions on SE4AI

11:00 - 12:30 **SE for AI with Quality** 1
Chair(s): Jun Sun Singapore I

3rd International Workshop on Responsible AI Engineering (RAIE 2025)

Can MSR Shed Light to Trustworthy AI?

YES!

School of Computing and Information Systems

SMU Singapore Management University

11

Automated Program Repair (APR)

Automatically Finding Patches Using Genetic Programming *

Westley Weimer
University of Virginia
weimer@virginia.edu

ThanhVu Nguyen
University of New Mexico
tnguyen@cs.unm.edu

Claire Le Goues
University of Virginia
legoues@virginia.edu

Stephanie Forrest
University of New Mexico
forrest@cs.unm.edu

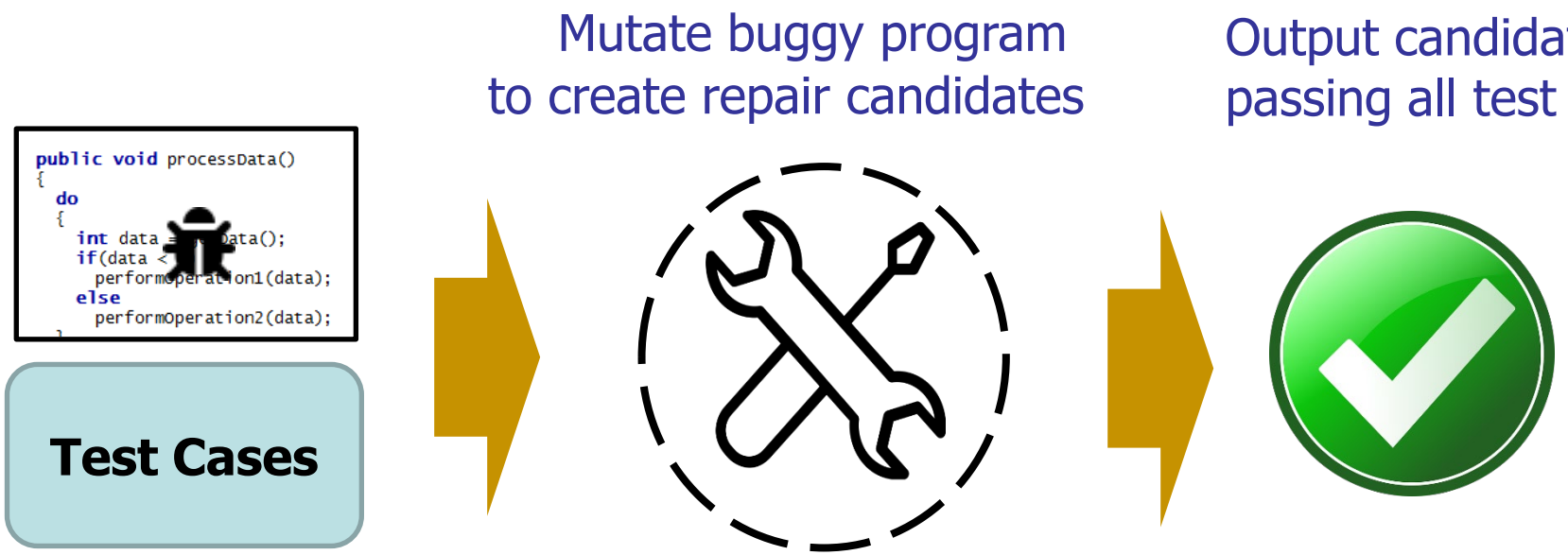
Abstract

Automatic program repair has been a longstanding goal in software engineering, yet debugging remains a largely manual process. We introduce a fully automated method for locating and repairing bugs in software. The approach works on off-the-shelf legacy applications and does not require formal specifications, program annotations or special

To alleviate this burden, we propose an automatic technique for repairing program defects. Our approach does not require difficult formal specifications, program annotations or special coding practices. Instead, it works on off-the-shelf legacy applications and readily-available test-cases. We use genetic programming to evolve program variants until one is found that both retains required functionality and also avoids the defect in question. Our technique

ICSE 2009
MIP ICSE 2019

Search-Based APR



E.g., GenProg (ICSE 2009, TSE 2011, ICSE 2012), etc.

Issues of Search-Based APR (in 2016)

- Overfitting, nonsensical patches

```
// Human fix: fa * fb > 0
if (fa * fb >= 0){
  throw new ConvergenceException("..");
}
```

- Long computation time to produce patches
- Lack of knowledge on bug fix history

Can MSR Help Automated Program Repair?

MSR for Automated Program Repair (MSR4APR)

History Driven Program Repair

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

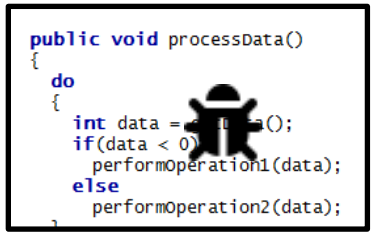
Claire Le Goues
School of Computer Science
Carnegie Mellon University
clegoues@cs.cmu.edu

SANER'16

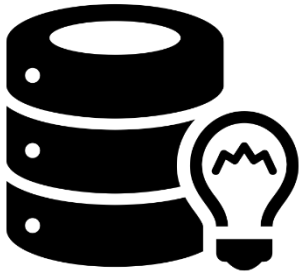
First work that automatically analyze hundreds of repositories for more effective automated program repair

Most cited SANER'16 paper

History-Driven Program Repair



Test Cases



Knowledge base: Learned from bug fixes in dev. history

Mutate buggy program to create repair candidates

Output candidates:
- frequently occur in the knowledge base
- make the failed tests pass

Faster

Avoid nonsensical patches

Our Framework (HDRepair)

Phase I: Bug Fix History Extraction

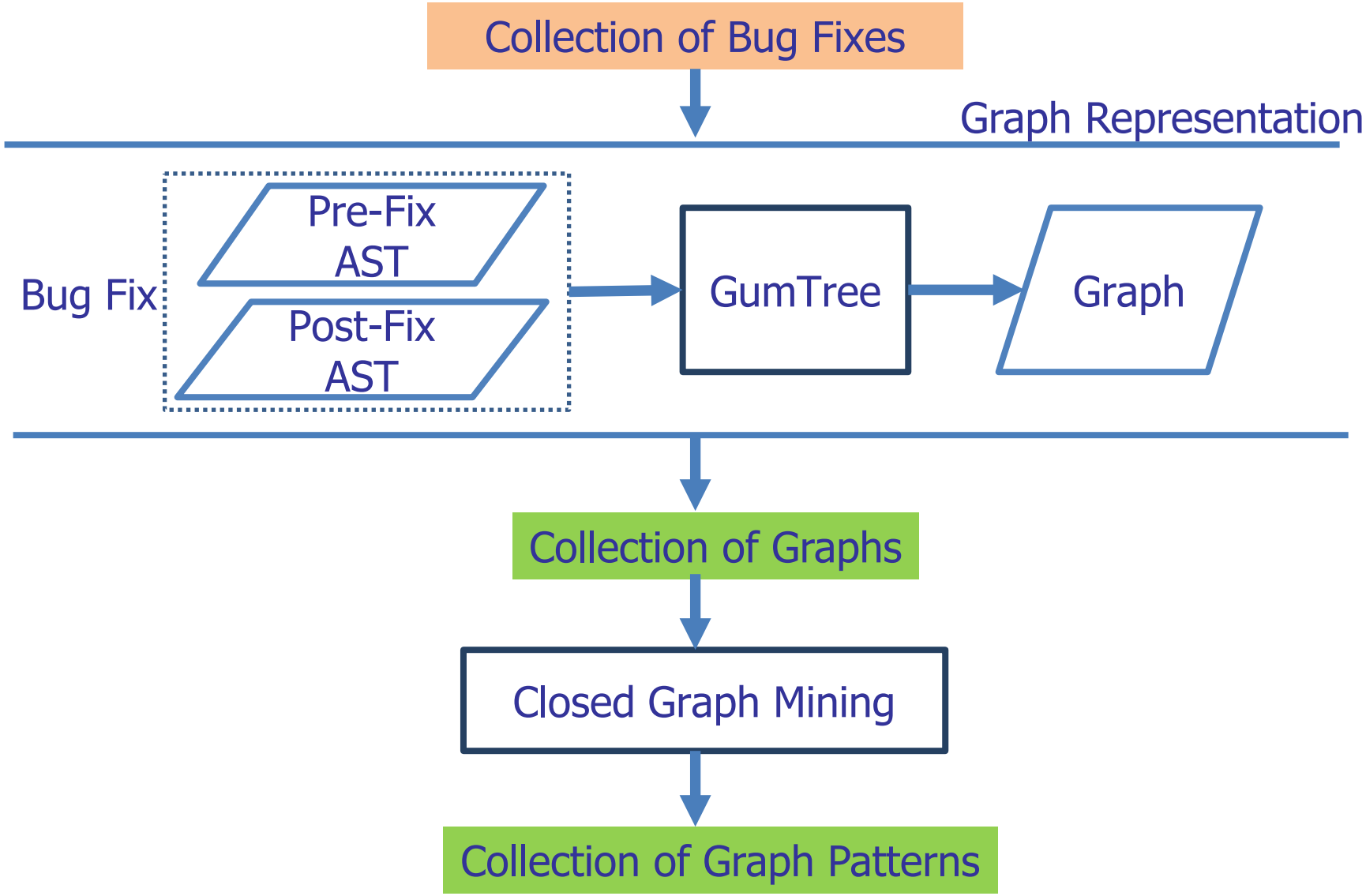
Phase II: Bug Fix History Mining

Phase III: Bug Fix Candidate Generation

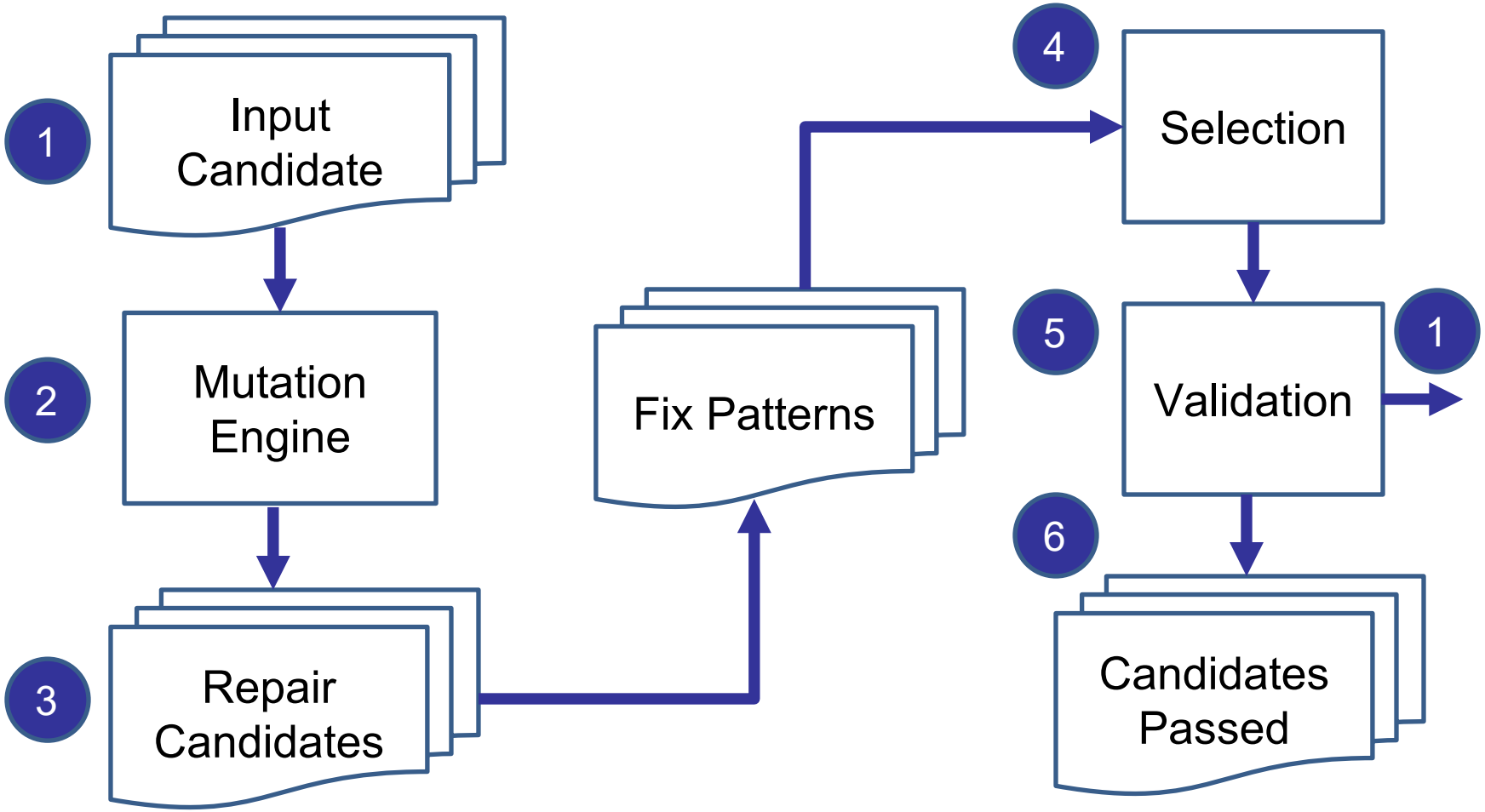
Phase I – Bug Fix History Extraction

- Active, large and popular Java projects
  - Updated until 2014, >= 5 stars, >= 100MBs
- *Likely* bug-fix commits
  - Commit message: *fix, bug fix, ~~fix typo~~, ~~fix build~~, ~~non-fix~~*
  - Submission of at least one test case
  - Change no more than two source code lines
- Result: 3,000 bug fixes from 700+ projects

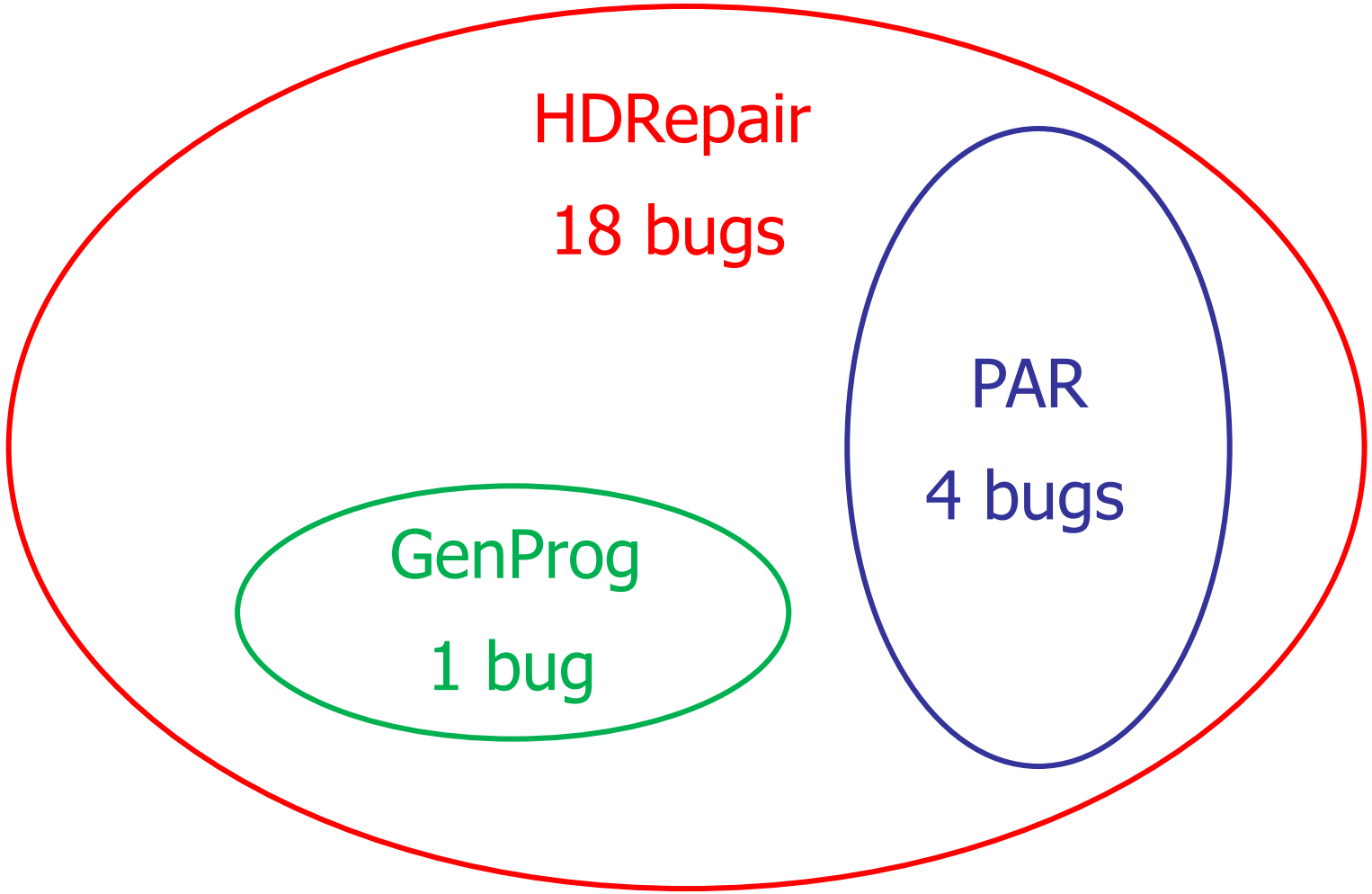
Phase II – Bug Fix History Mining



Phase III – Bug Fix Candidate Generation



Number of Bugs Correctly Fixed



Follow Up Work

Getafix

Learning to fix bugs automatically

Andrew Scott
Facebook
andrewscott@fb.edu

Johannes Bader
Facebook
jobader@fb.com

Satish Chandra
Facebook
satch@fb.com

"***History driven program repair*** *as investigated by Le et al.* ***influenced*** *our work, the overall diffing/mining pipeline is similar."* – Facebook Engineer

Follow Up Work

History driven program repair

XBD Le, D Lo, C Le Goues - 2016 IEEE 23r

… 1) We propose a generic and efficient **his**

that uses information stored in revision contr

☆ Save 99 Cite Cited by 457 Related a

Agentless: Demystifying llm-based software engineering agents

CS Xia, Y Deng, S Dunn, L Zhang - arXiv preprint arXiv:2407.01489, 2024 - arxiv.org

Recent advancements in large language models (LLMs) have significantly advanced the automation of software development tasks, including code synthesis, program repair, and …

☆ Save 99 Cite Cited by 86 Related articles All 3 versions »

Repairagent: An autonomous, llm-based agent for program repair

I Bouzenia, P Devanbu, M Pradel - arXiv preprint arXiv:2403.17134, 2024 - arxiv.org

Automated program repair has emerged as a powerful technique to mitigate the impact of software bugs on system reliability and user experience. This paper introduces RepairAgent …

☆ Save 99 Cite Cited by 74 Related articles All 5 versions »

Automated program repair via conversation: Fixing 162 out of 337 bugs for $0.42 each using ChatGPT

CS Xia, L Zhang - Proceedings of the 33rd ACM SIGSOFT International …, 2024 - dl.acm.org

Automated Program Repair (APR) aims to automatically generate patches for buggy programs. Traditional APR techniques suffer from a lack of patch variety as they rely heavily …

☆ Save 99 Cite Cited by 25 Related articles All 2 versions

MSR Can Effectively Power Automation (e.g., APR)

Issues of Search-Based Repair Tools (in 2016)

- Overfitting, nonsensical patches

```
// Human fix: fa * fb > 0
if (fa * fb >= 0){
   throw new ConvergenceException(“..”);
}
```

- Long computation time to produce patches
- Lack of knowledge on bug fix history

YES!

Can MSR Help Automated Program Repair?

School of Computing and Information Systems
SMU SINGAPORE MANAGEMENT UNIVERSITY

Reflections

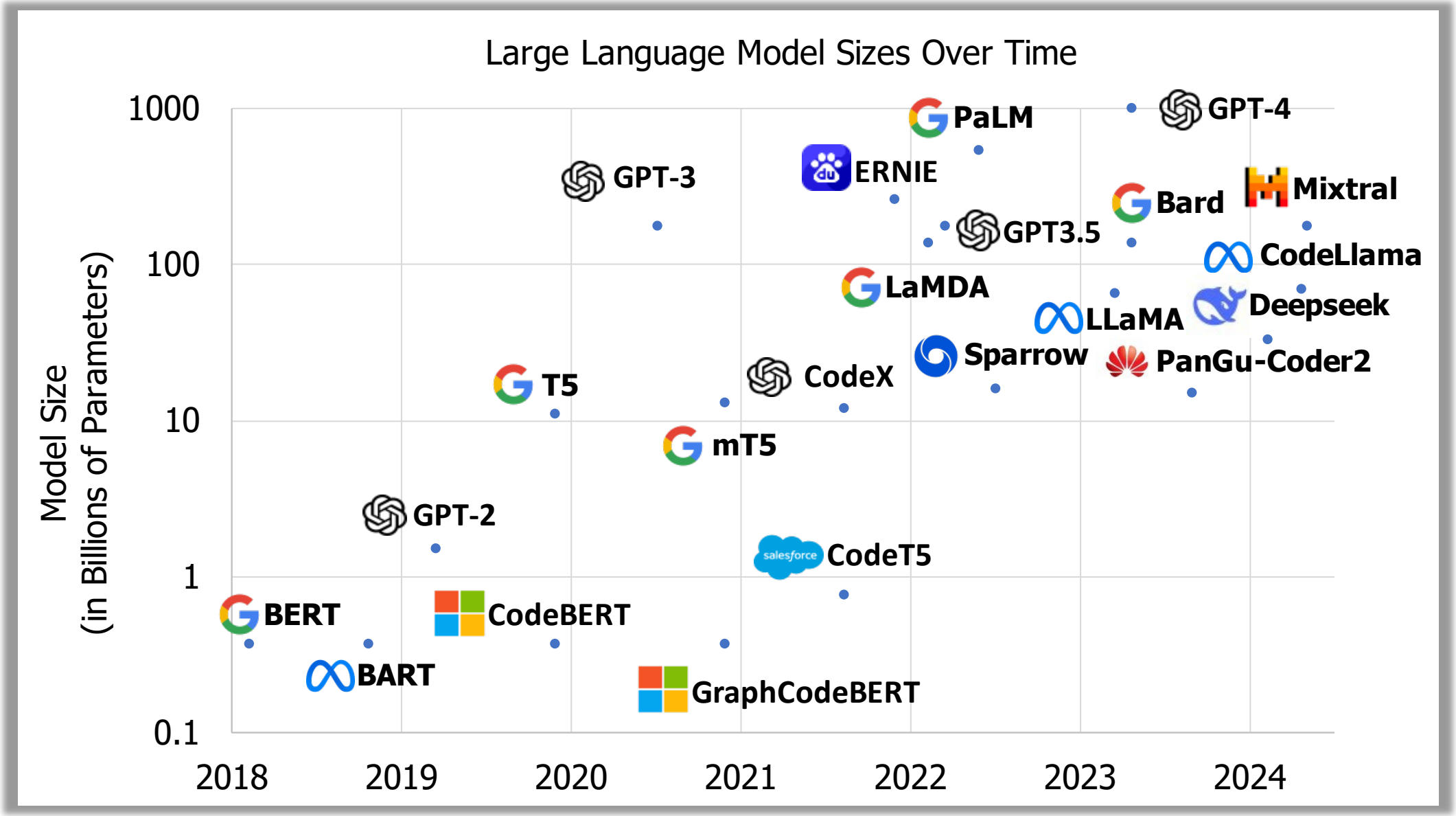
- MSR can shed insights to new domains and power more automation
- **As miners we want to expand the "world" of MSR**
- Find opportunities to demonstrate the power of MSR to various SE problems, including:
  - ✓ MSR4TAI
  - ✓ MSR4APR

Reflection of the Past

Charting the Future

New Era of LLM

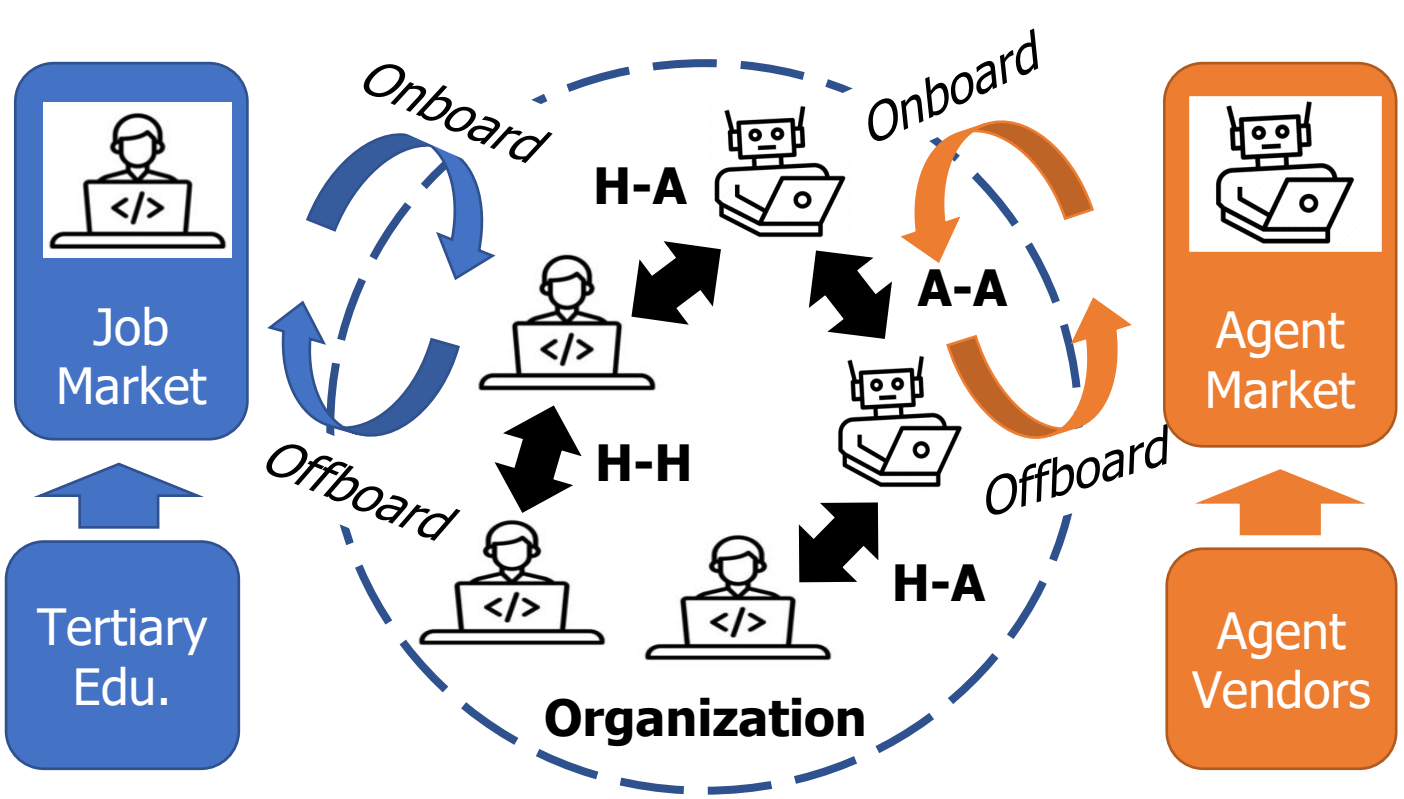
"Hic sunt dracones"



New Era of Software Engineering 2.0

"Hic sunt dracones"

Symbiotic workforce of autonomous, responsible, intelligent agents & engineers



Smart Tool

Smart Workmate

ICSE'23

Economics

Law

Ethics

MSR4TAI: Are There New Gold Mines?

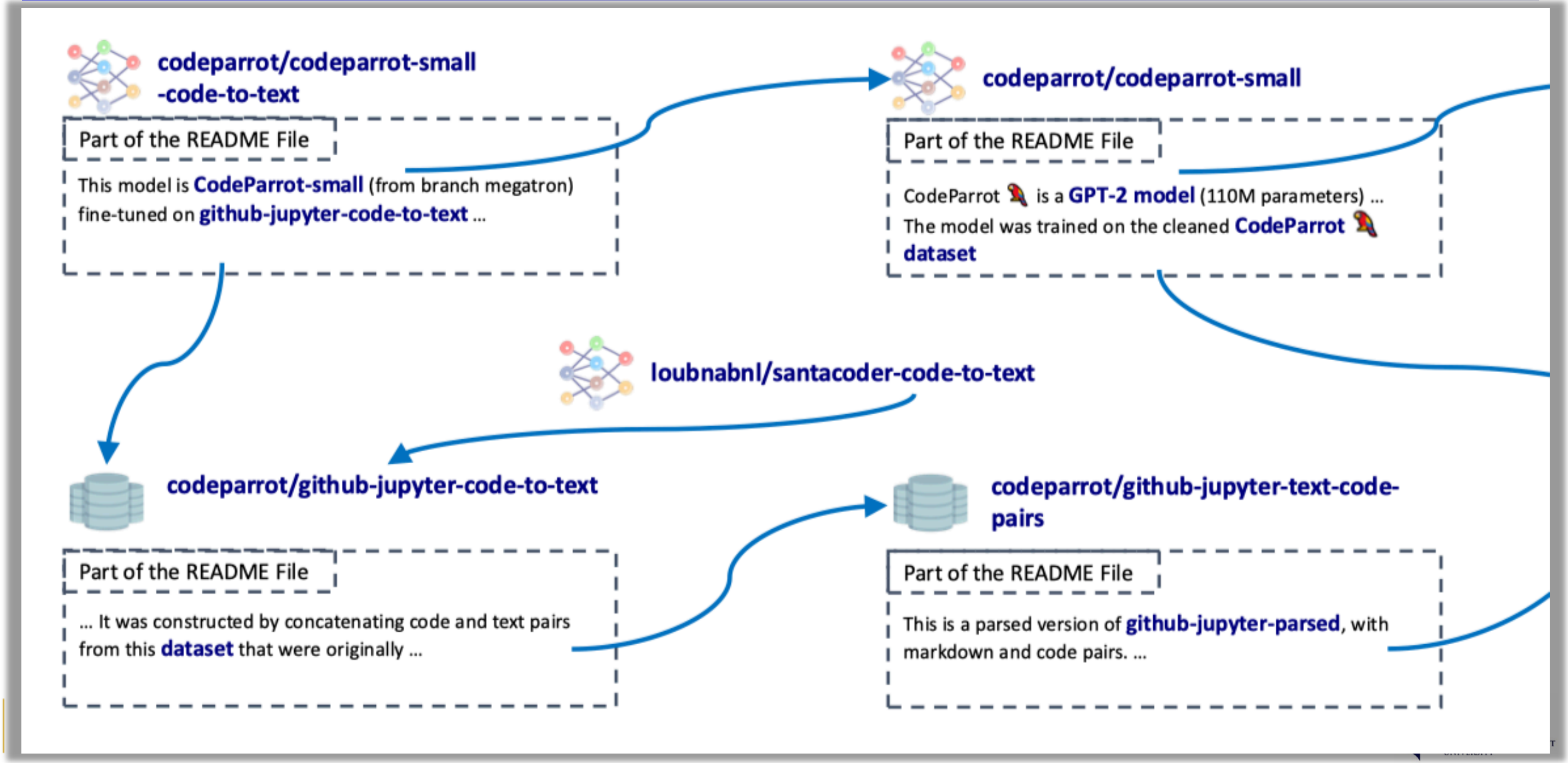
Ecosystem of Large Language Models for Code

ZHOU YANG, Singapore Management University, Singapore
JIEKE SHI, Singapore Management University, Singapore
PREMKUMAR DEVANBU, Department of Computer Science, UC Davis, USA
DAVID LO, Singapore Management University, Singapore

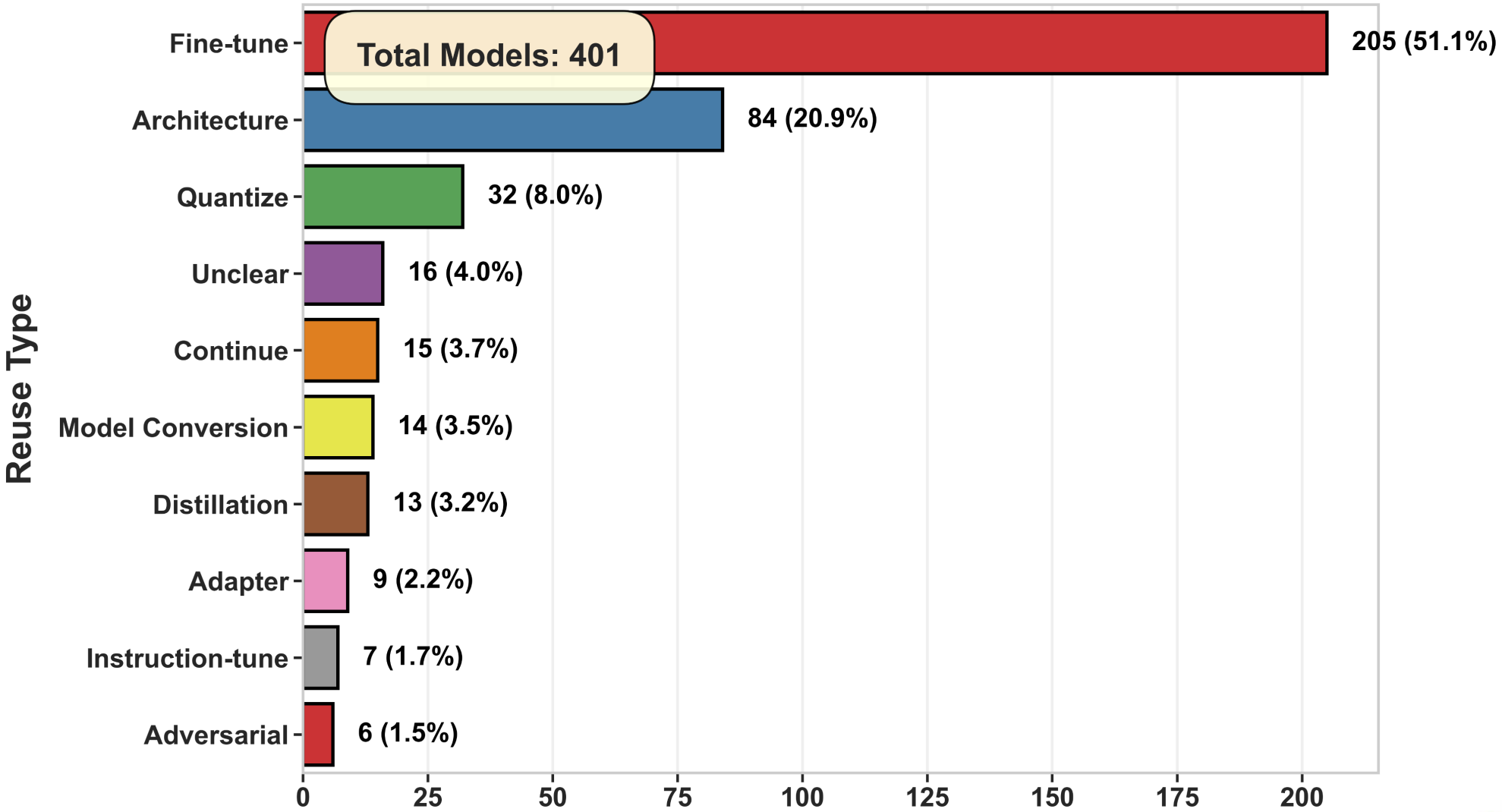
TOSEM'25

LLM Ecosystem: A Rich Repository to Mine

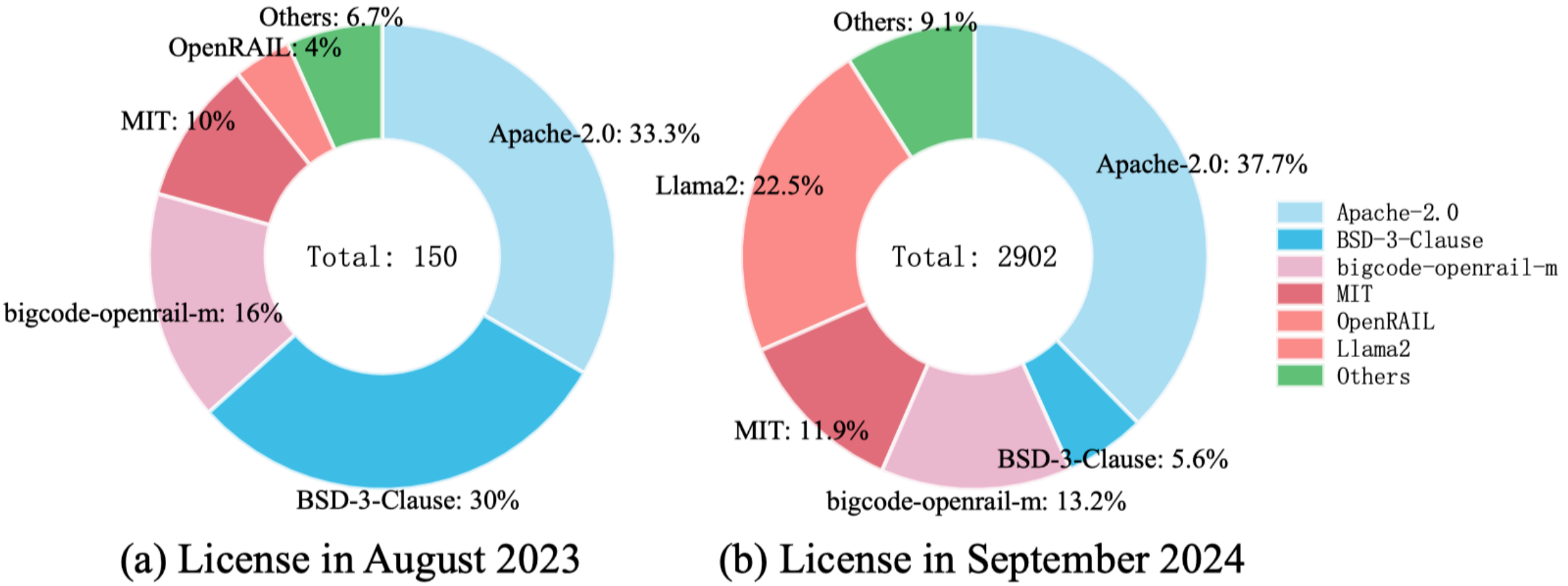
Hugging Face



Findings: Relationships between Models



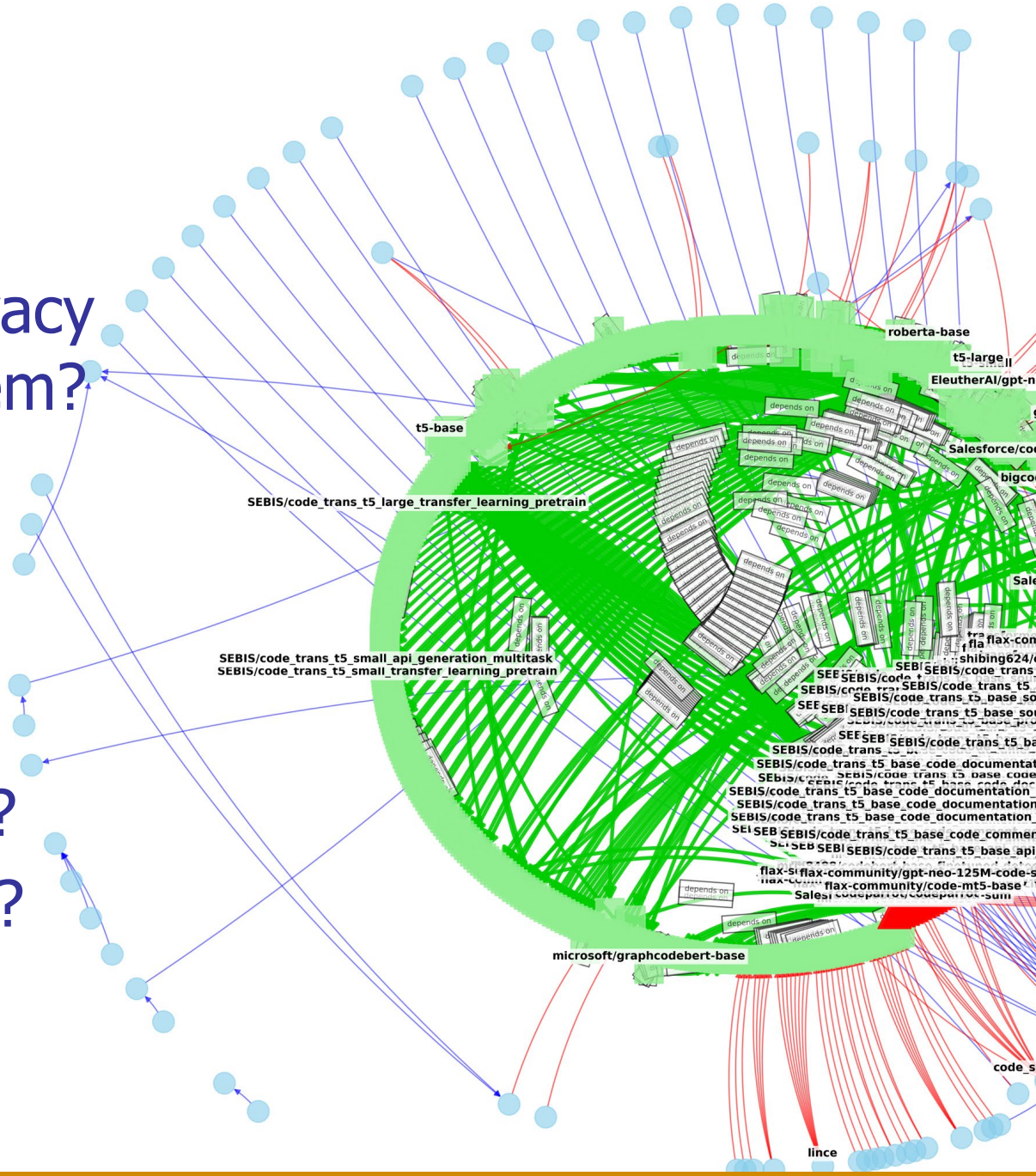
Findings: Licenses of LLMs



Many more findings in the paper

More to be Mined

- How does the ecosystem evolve?
- How risks (like data poisoning, privacy leakage) propagate in the ecosystem?
  - How to mitigate such risks?
- How the development of LLM4SE impact the society (e.g., energy consumption)?
- How contributors join & contribute?
- What make successful contributors?
- Many more !



MSR4APR: Are There New Gold Mines?

ICSE'24

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

Historical Data

- Historical Fixing Data
- Historical Vul-fixing Data
- CWE Knowledge

\+

Synthesized Data

- Synthesized Fixing Data from CWE Knowledge

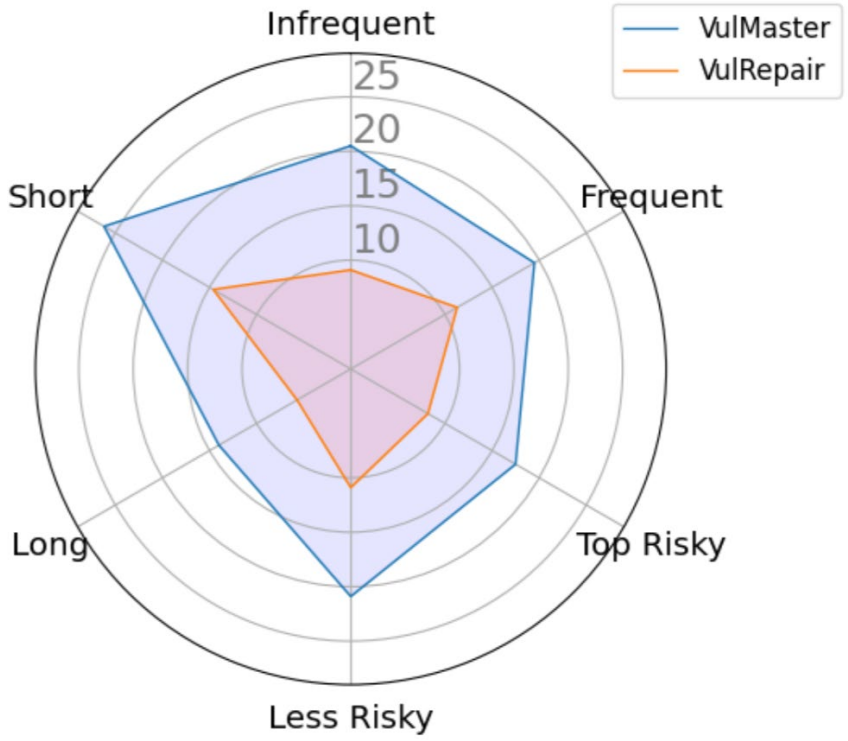
\+ **LLMs** = **2x Fixed Vulnerabilities**

MSR4APR: Are There New Gold Mines?

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 almost **doubles the Exact Match** (EM) score compared to the previous SOTA.
- VulMaster consistently outperforms the SOTA in diverse aspects.



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

Expanding M**SR**: What is a "Software Repository"?

Code vs. Models & Data Ecosystem

Code History vs. Model & Data Provenance

Historical vs. Synthesized

MSR4<?>: Plenty of **Open Explorations** in the World of MSR

Acknowledgements

SMU
SINGAPORE MANAGEMENT UNIVERSITY

OUB Chair
Professorship Fund

Thank you!

Questions? Comments? Advice?

davidlo@smu.edu.sg

MSR for Trustworthy AI (MSR4TAI)

An Empirical Study of Bugs in Machine Learning Systems

Ferdian Thung, Shaowei Wang, David Lo, and Lingxiao Jiang
School of Information Systems
Singapore Management University, Singapore
{ferdianthung,shaoweiwang.2010,davidlo,lxjiang}@smu.edu.sg

ISSRE'12

First work employing MSR to gain insight into ML bugs

MSR for Automated Program Repair (MSR4APR)

History Driven Program Repair

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

SANER'16

First work that automatically analyze hundreds of repositories for more effective automated program repair

Most cited SANER'16 paper

Reflection of the Past

Expanding M**SR**: What is a "Software Repository"?

Code vs. Models & Data Ecosystem

Code History vs. Model & Data Provenance

Historical vs. Synthesized

MSR4<?>: Plenty of **Open Explorations** in the World of MSR

Charting the Future