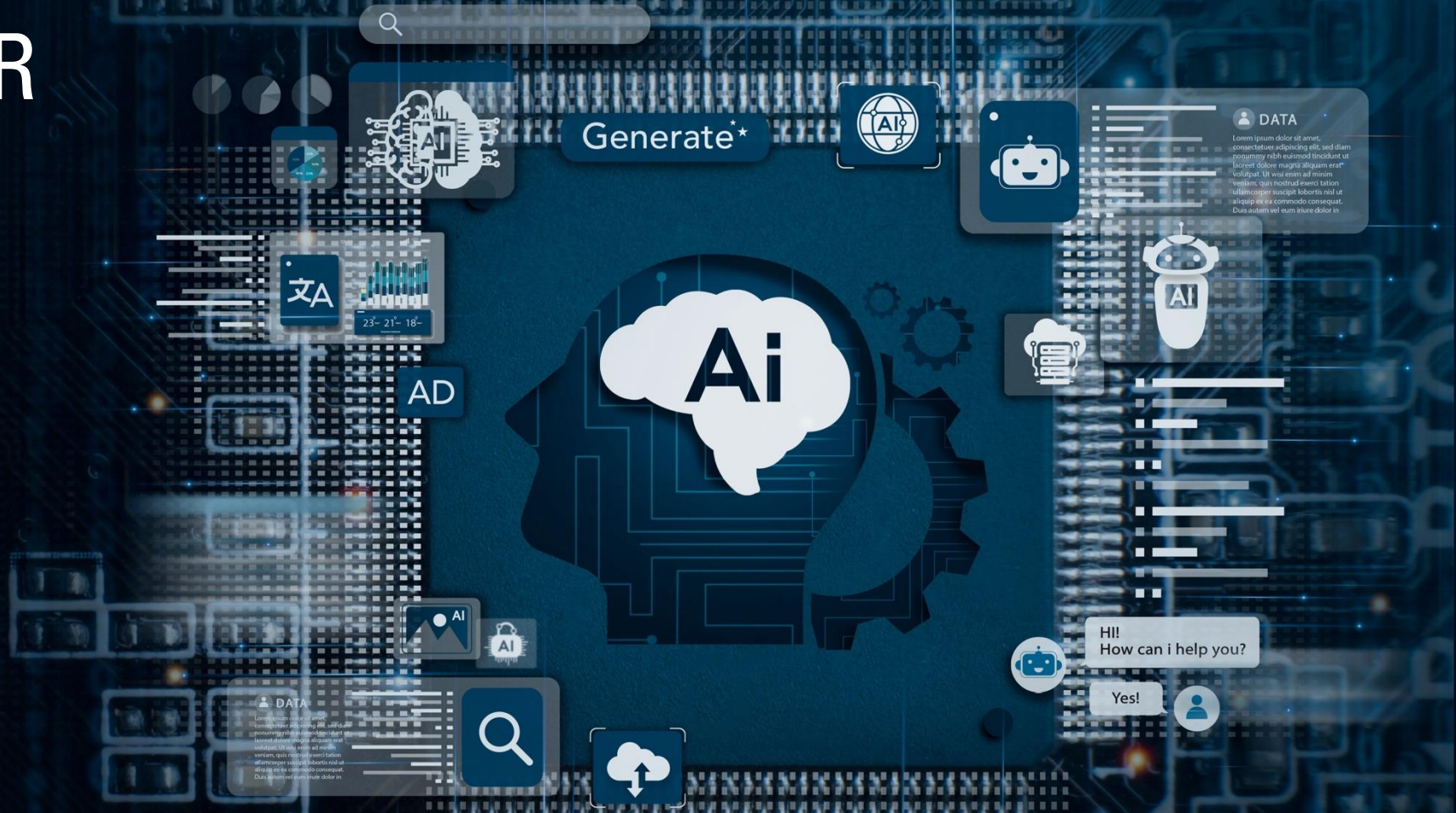
AGENTIC AI FOR SOFTWARE:

LESSONS IN TRUST



Abhik Roychoudhury
National University of Singapore

LOVE FOR PROGRAMMING

(CLAUDE SONNET)

Today

Still Gets Me

Even now, after thousands of lines of code, complex architectures, and sleepless debugging sessions, there's something beautifully pure about those two simple words. They remind us why we fell in love with programming in the first place.

```
> console.log("Hello, World!"); // Never gets old ✨
```

"No matter how complex your code becomes, you'll always remember the simple joy of your first 'Hello, World!'"

— Every Programmer, Ever

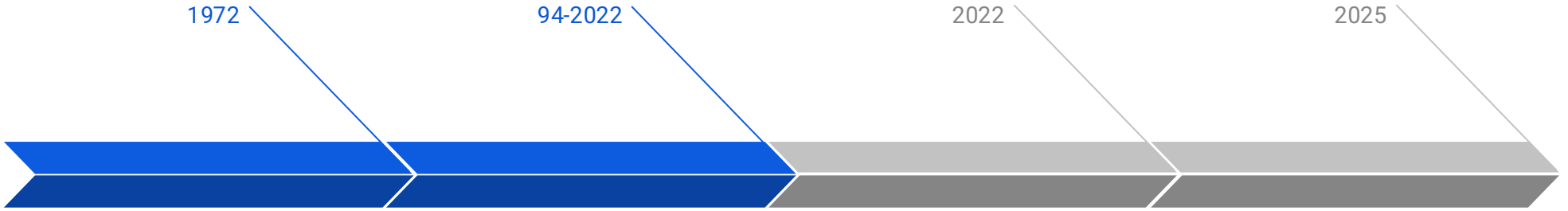
1972-3: “HELLO WORLD” STYLIZATION

```
main( ) {
        extrn a, b, c;
        putchar(a); putchar(b); putchar(c); putchar('!*n');
}

a 'hell';
b 'o, w';
c 'orld';
```

B programming language

4-character limitation. ☺



Hello World program in BCPL/B, before C

Brian Kerninghan

Windows / Linux

Software as a Service

Huge Code-base, - software model checking, Linux is 30M in 2022

Internet -> Software delivery

GitHub Copilot

Automatically generated code integrated - importance of verification rises.

Year of LLM Agents

Need for verification of auto generated code integration

programming at scale

programming with trust

SOFTWARE INDUSTRY OVER 50 YEARS

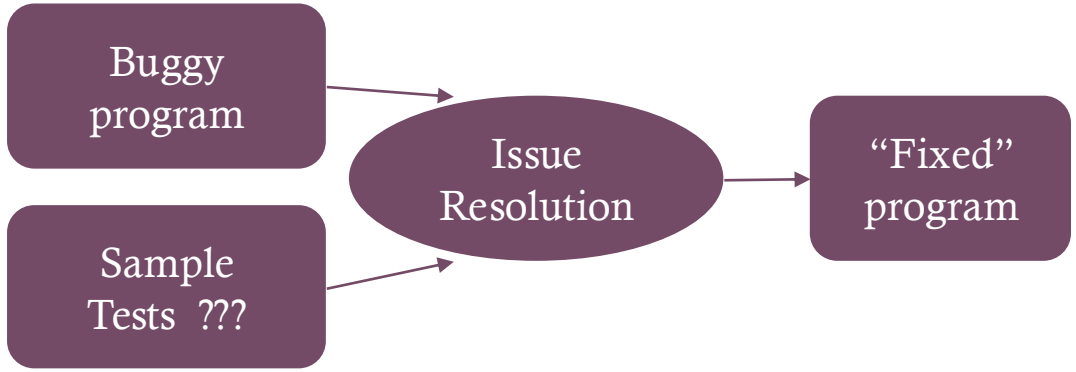
1972-3:
"Hello world"
program in B and C

Tech / Horizontal:
Engineering of SW itself!
App / Verticals: ***the next SalesForce?***

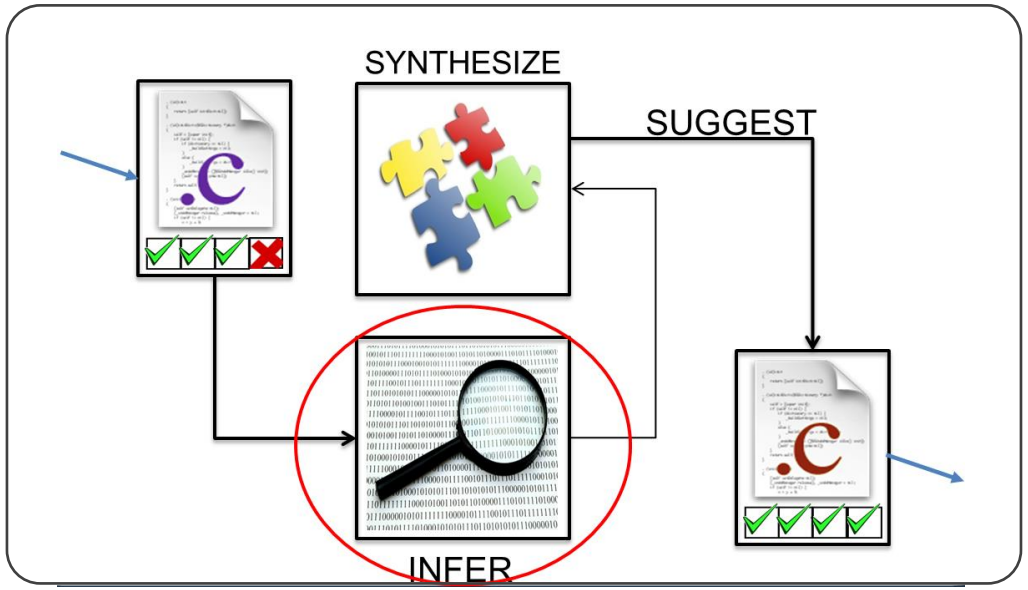
Hosting of SW –
Salesforce (CRM) Other app domains

THE DAY OF A SOFTWARE ENGINEER

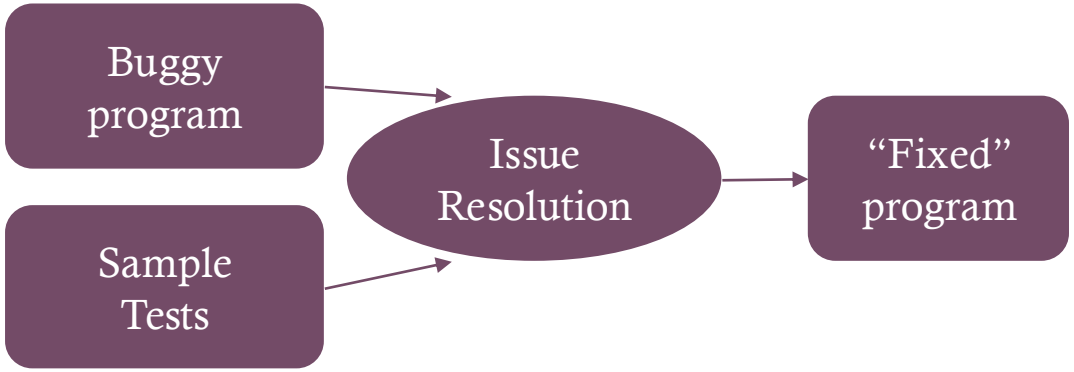
- More of program improvement, rather than coding
- Come in the morning, and see a host of "issues"
  - An issue can refer to a bug report and needed fix
  - Feature addition
  - Even efficiency improvement in a part of the code?



UNPACKING "ISSUES": INTENT



SemFix, ICSE 2013
Angelix, ICSE 2016



An issue can refer to a bug report and needed fix

LEARNT AS A SCHOOL-CHILD ☺

```
1 int triangle(int a, int b, int c){
2     if (a <= 0 || b <= 0 || c <= 0)
3         return INVALID;
4     if (a == b && b == c)
5         return EQUILATERAL;
6     if (a == b || b != c) // bug !
7         return ISOSCELES;
8 return SCALENE;
9 }
```

MAY NOT HAVE LEARNT SO FAR?

Test id	a	b	c	oracle	Pass
1	-1	-1	-1	INVALID	Yes
2	1	1	1	EQUILATERAL	Yes
3	2	2	3	ISOSCELES	Yes
4	2	3	2	ISOSCELES	Yes
5	3	2	2	ISOSCELES	NO
6	2	3	4	SCALANE	NO

Given ”intent” as tests

```
1 int triangle(int a, int b, int c){
2    if (a <= 0 || b <= 0 || c <= 0)
3       return INVALID;
4    if (a == b && b == c)
5       return EQUILATERAL;
6    if (a == b || b != c) // bug !
7       return ISOSCELES;
8 return SCALENE;
9 }
```

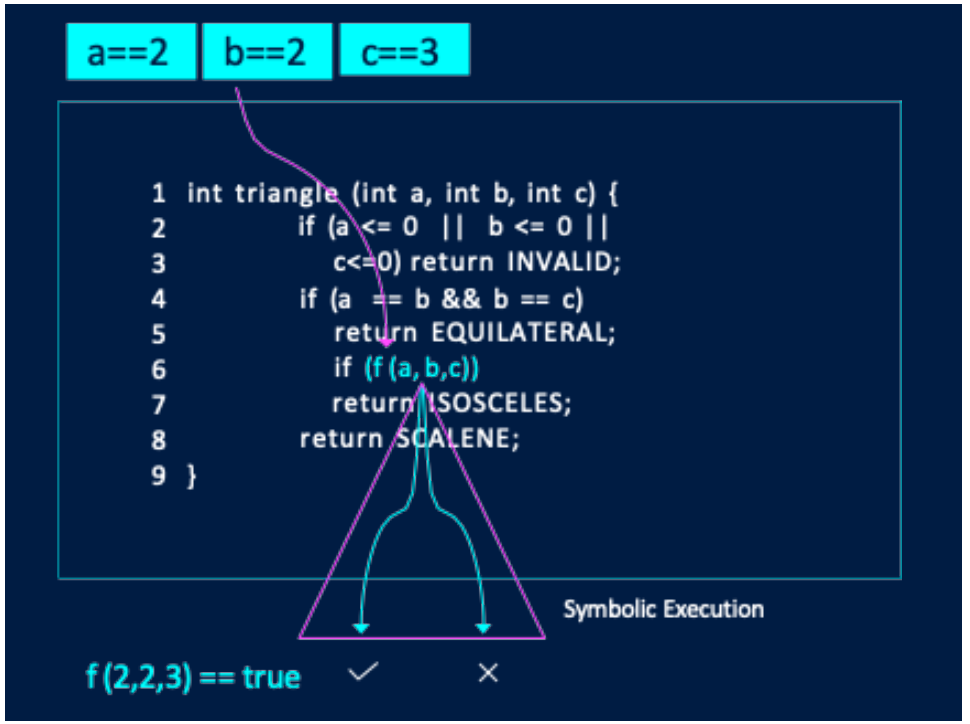
Buggy Program

Automatically generate the fix `(a == b || b == c || c == a)`

FROM INTENT TO CODE – RELIABLY !

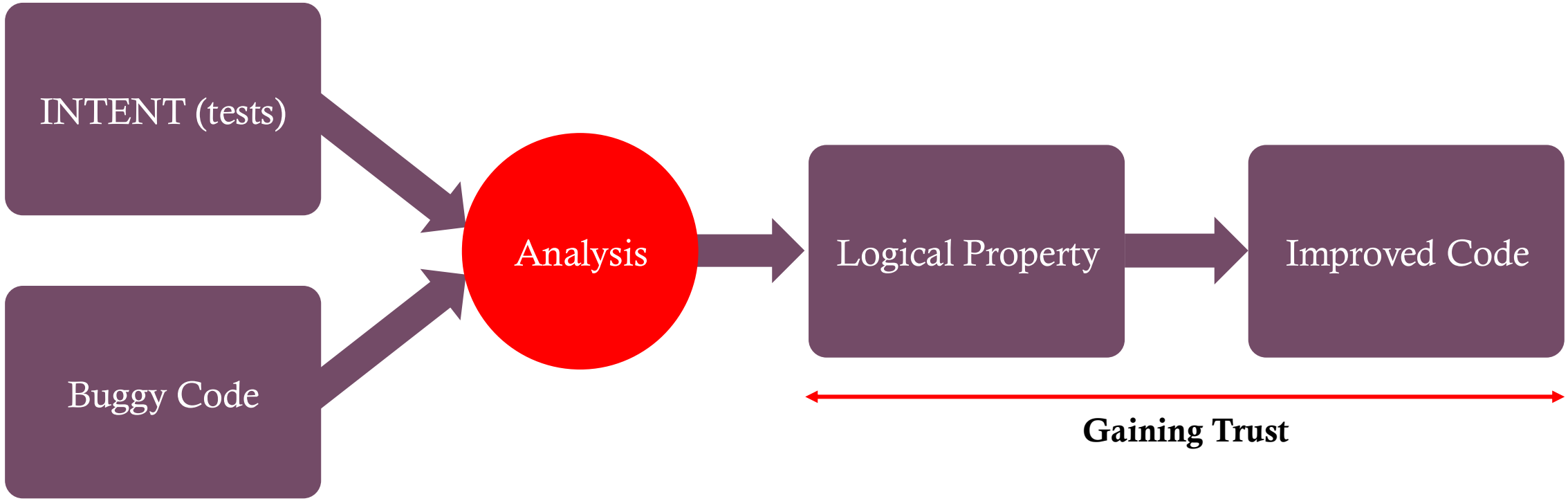
Test id	a	b	c	oracle	Pass
1	-1	-1	-1	INVALID	Yes
2	1	1	1	EQUILATERAL	Yes
3	2	2	3	ISOSCELES	Yes
4	2	3	2	ISOSCELES	Yes
5	3	2	2	ISOSCELES	NO
6	2	3	4	SCALANE	NO

Given ”intent” as tests



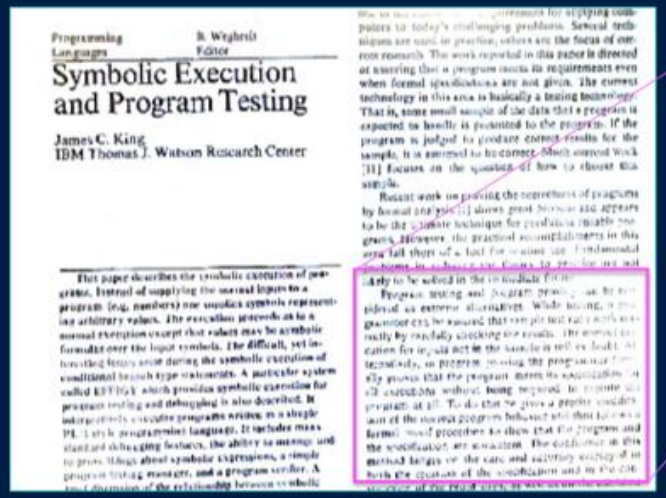
f(2,2,3) and f(2,3,2) and f(3,2,2) and not f(2,3,4) → `(a == b || b == c || c== a)`

TRUSTED AUTOMATIC PROGRAMMING



INTENT INFERENCE !

SYMBOLIC EXECUTION (1976)



Programming Languages — B. Wegbreit, Editor

Symbolic Execution and Program Testing

James C. King
IBM Thomas J. Watson Research Center

"Program testing and program proving can be considered as extreme alternatives.....

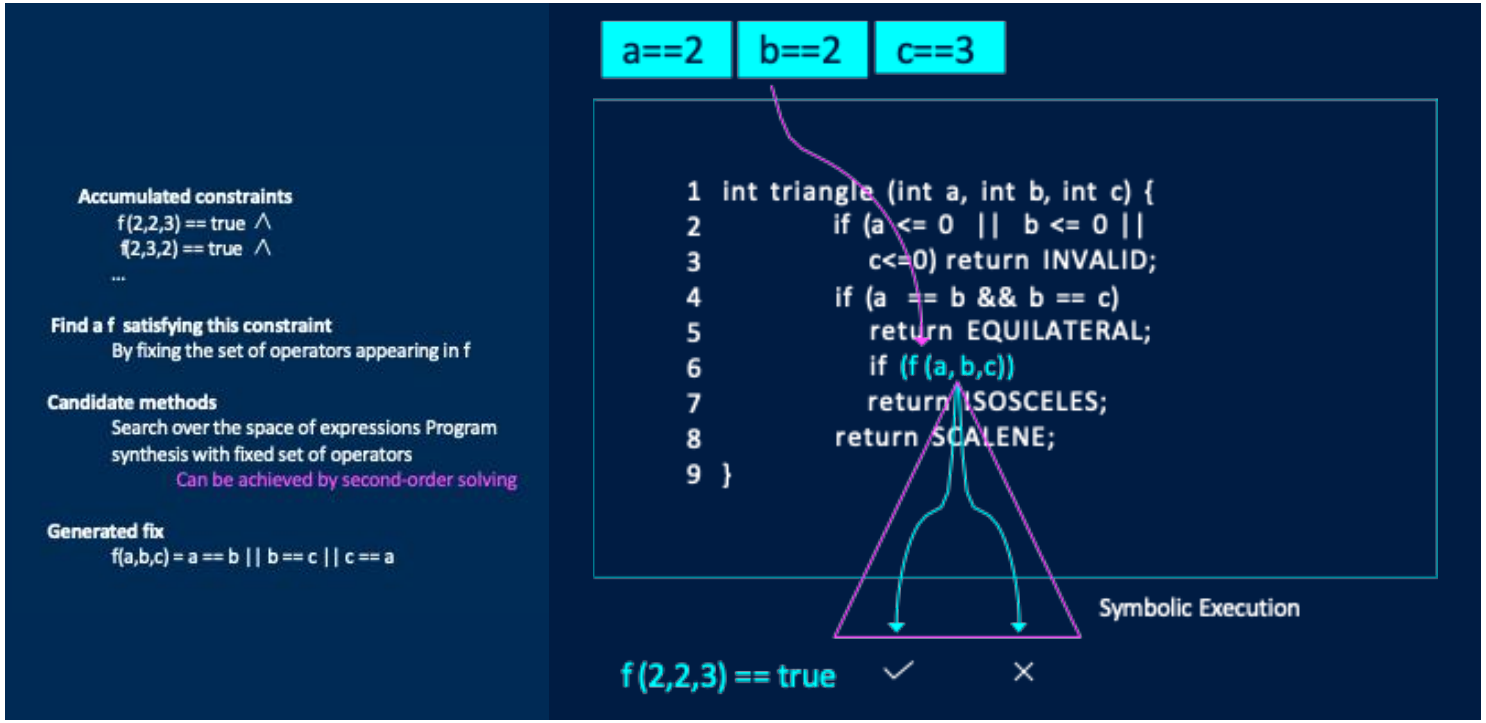
This paper describes a practical approach between these two extremes...

Each symbolic execution result may be equivalent to a large number of normal tests"

New usage In specification inference

54

INTENT FROM TESTS



Higher order logic inference from tests.

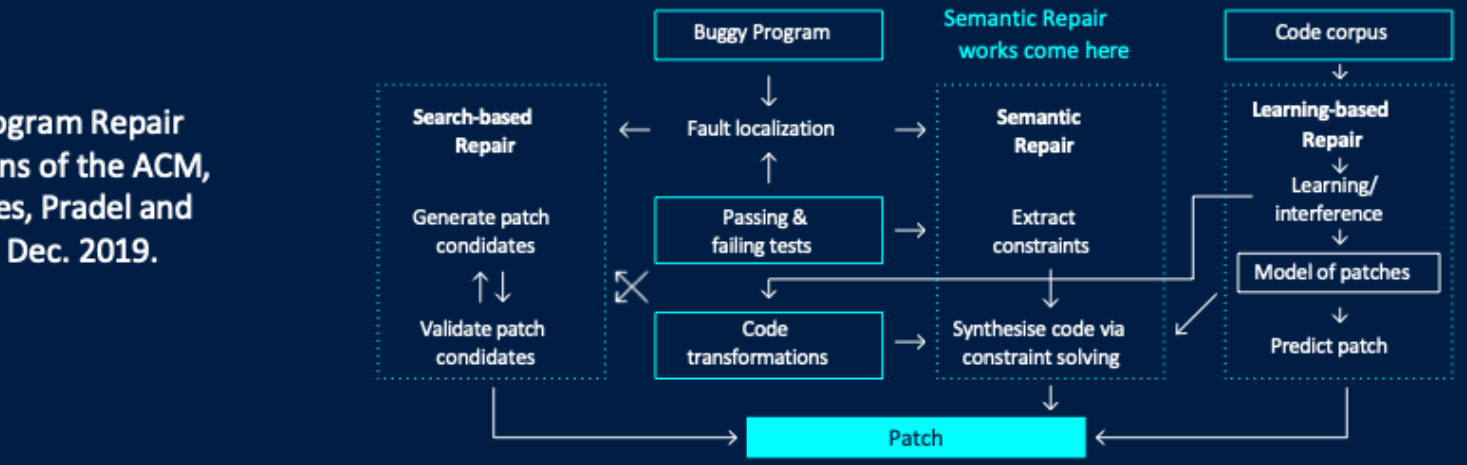
Lot of machinery in achieving it efficiently in a first order logic framework.

Need a mechanism for extracting intent when tests are absent.

PRE-LLM

SUMMARY

Automated Program Repair Communications of the ACM, 62(12), Le Goues, Pradel and Roychoudhury, Dec. 2019.




```
if (((1 > 0) && (1 > 0)))
if ((((! (image->res_unit == 3)) && (!image->res_unit == 3)))) if
( (! ((((! ((- 4) == 0))))) && ((! (0 == 0)) || (! (64 == 0))))))
if ( (! ((log_level && (! ((- 4) == 0))) && log_level)))
```

Can we generate less number of patches? Can we generate high quality patches?

```
Input-output examples:
   {(ip=2, op=4),(ip=3, op=9)}
buggy program:
   return ip+ip;
We would not want a repaired program like if
       ip=2  return 4;
else if  ip=3  return 9;
else           return -1;
```

NUS National University of Singapore

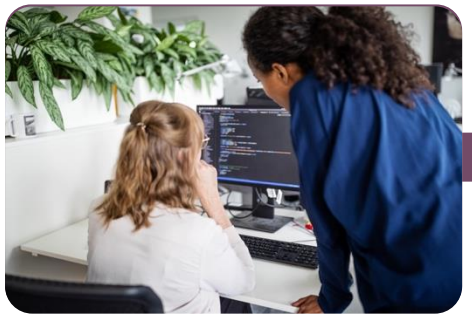
ICSE2023 MIP Award Talk

4

TRUSTED AUTOMATIC PROGRAMMING

Buggy Code

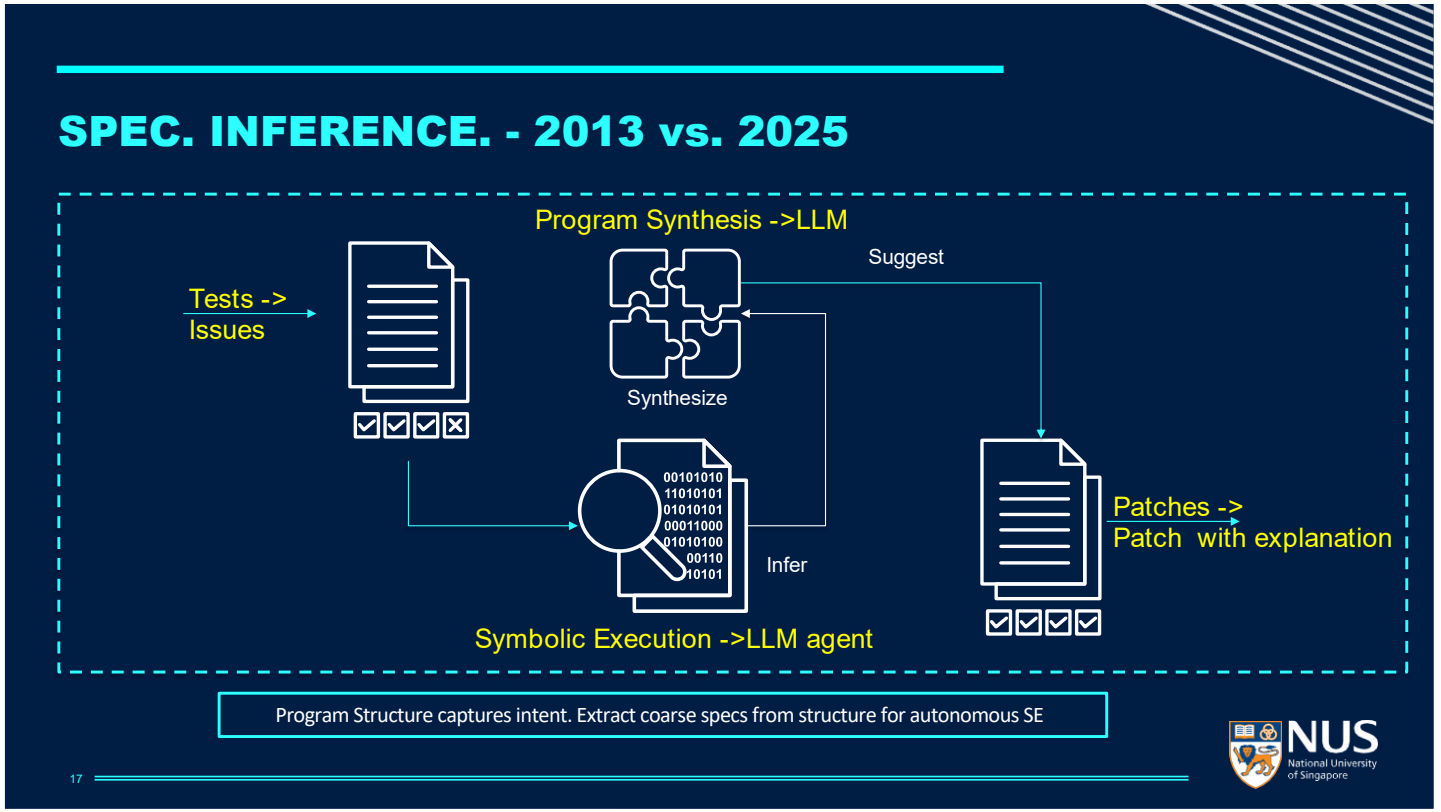
Analysis



Improved Code

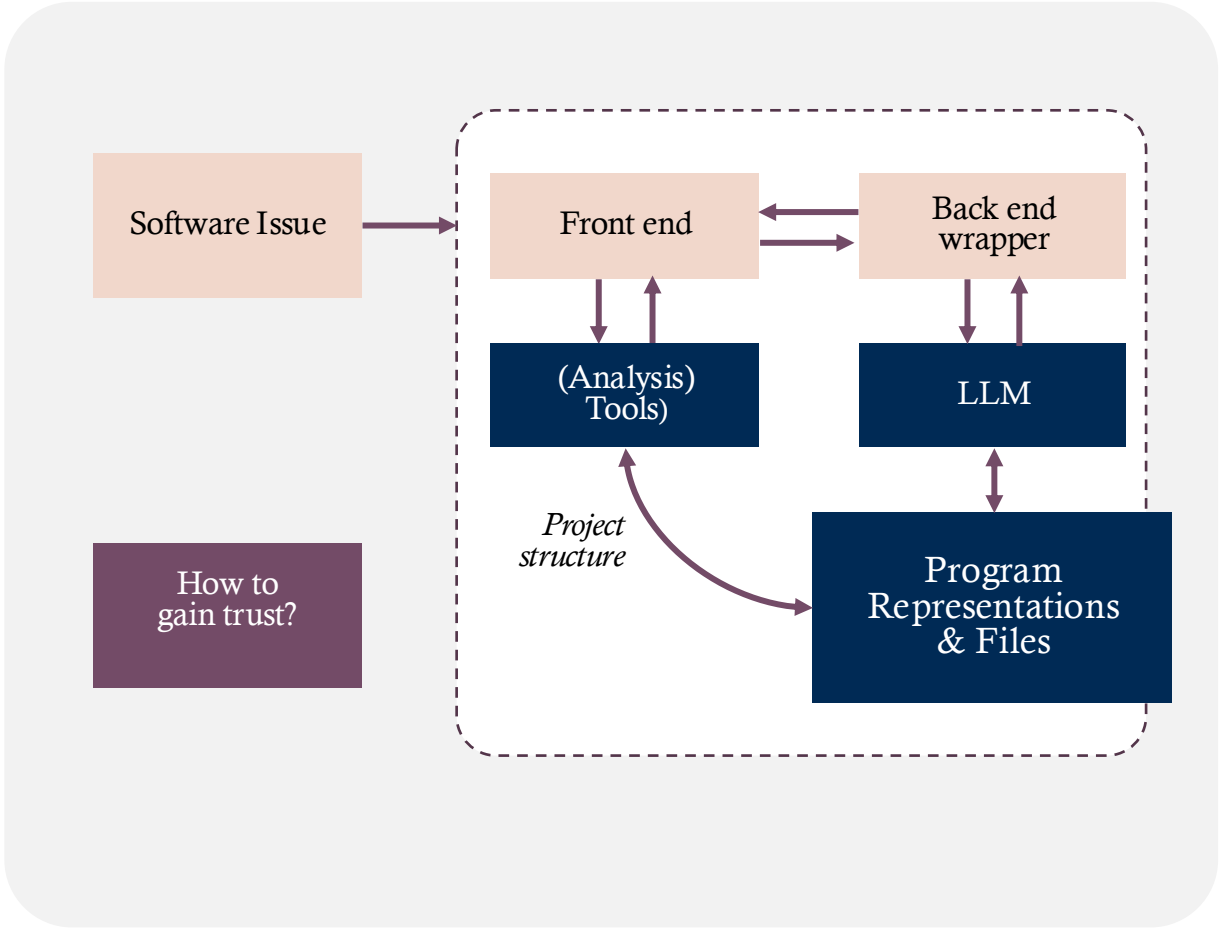
Gaining Trust

THEN AND NOW


SPEC. INFERENCE. - 2013 vs. 2025
Program Synthesis ->LLM
Tests ->
Issues
Suggest
Synthesize
00101010
11010101
01010101
00011000
01010100
00110
10101
Infer
Patches ->
Patch with explanation
Symbolic Execution ->LLM agent
Program Structure captures intent. Extract coarse specs from structure for autonomous SE
17
NUS
National University
of Singapore

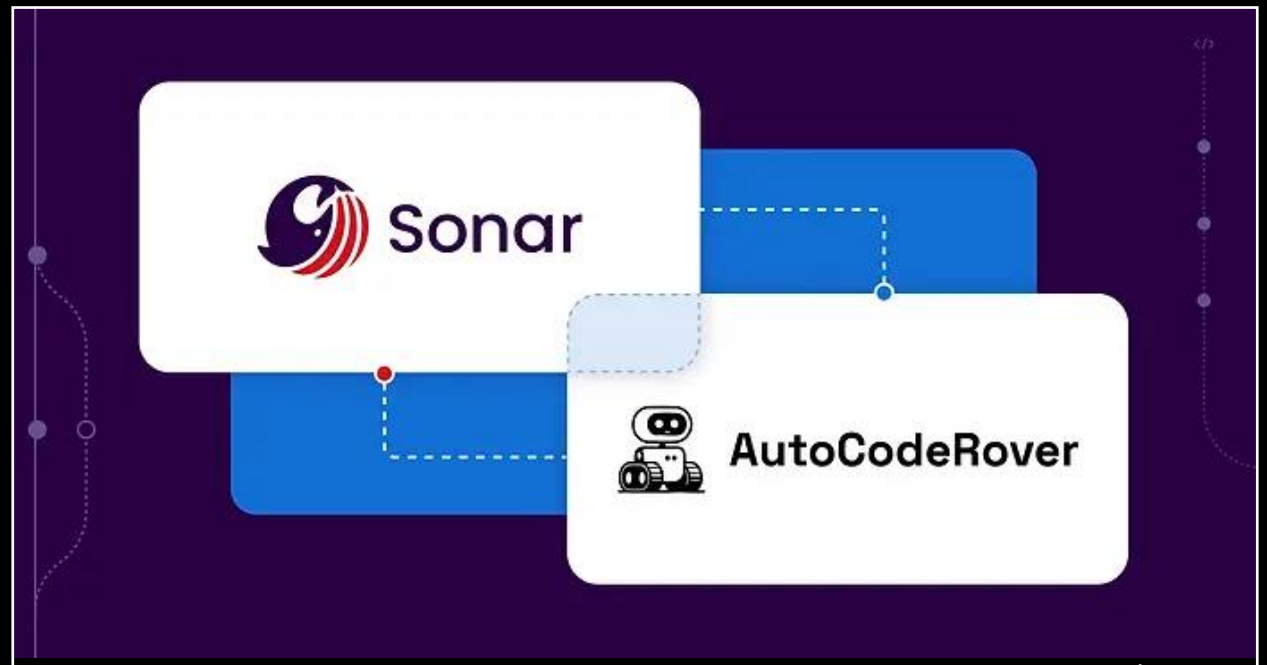
ISSUE RESOLUTION

Do not see Code as text!

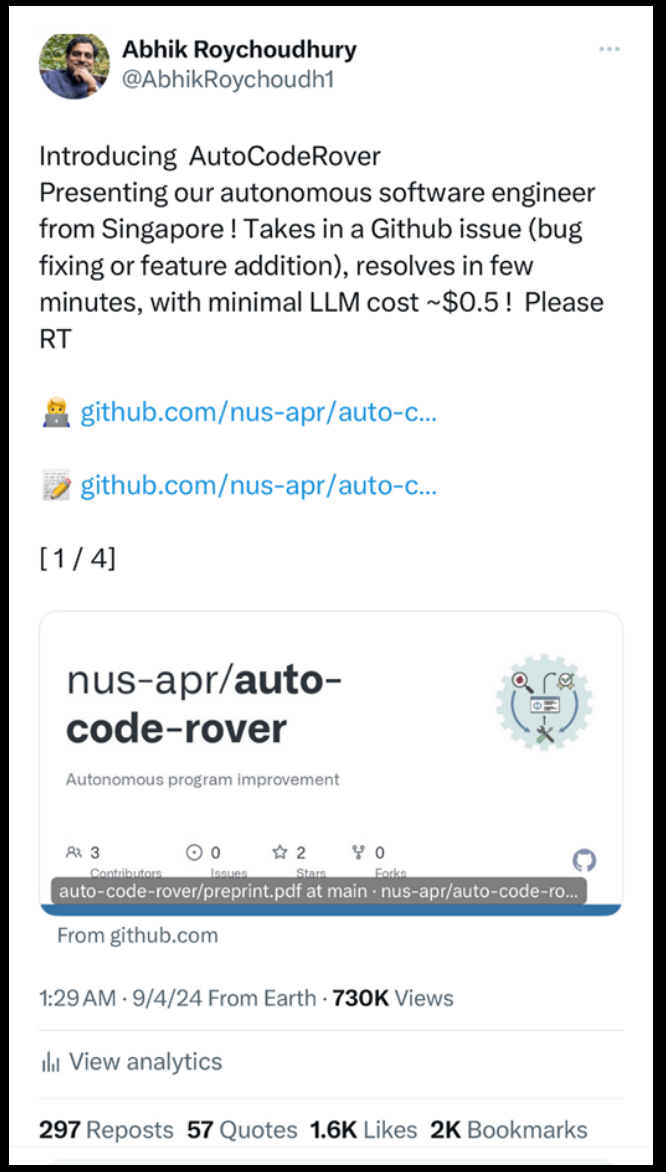


AutoCodeRover: ISSTA 2024, Zhang, Ruan, Fan, Roychoudhury.

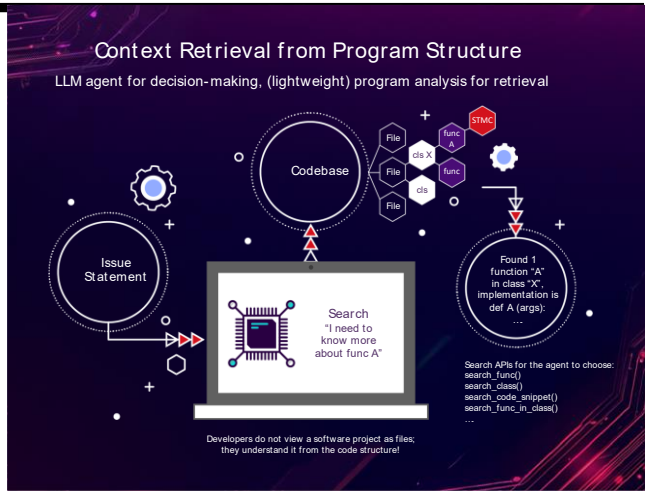
AUTOCODEROVER



https://autocoderover.dev/



IMPLICIT INTENT



Issue statement

I need to know more about *func A*

search_func(A)

Codebase

file | file | file

cls X | cls

func A | func

Stmt

Toolkit for the agent to choose:
search_func()
search_class()
search_func_in_class()
search_code_snippet()
...

Found 1 function "A" in class "X", implementation is
def A (args):
....

STRATIFIED SEARCH

- $Context_i = Search\_API(U_{i-1})$
- $U_i = Analysis(U_{i-1}, Context_i)$

Iteratively refine issue understanding based on latest context.

#32347 closed New feature (fixed)

ModelChoiceField does not provide value of invalid choice when raising ValidationError

U_0

We need to understand how the `ModelChoiceField` class handle validation and error messages …

search_class("ModelChoiceField")

$Context_1$

Signature of the `ModelChoiceField` class:

```
def __init__(…):
def get_limit_choices_to(self):
def __deepcopy__(self, memo):
def label_from_instance(self, obj):
… …
def prepare_value(self, value):
def to_python(self, value):
def validate(self, value):
def has_changed(self, initial, data):
```

U_1

`ModelChoiceField` does not have a `clean` method. It likely uses another method like `validate` or `to_python` to validate inputs. Next step: Examine the `validate` and `to_python` methods of `ModelChoiceField`.

search_method_in_class("validate", "ModelChoiceField")
search_method_in_class("to_python", "ModelChoiceField")

$Context_2$

Implementation of `validate`:
…
Implementation of `to_python`:
…

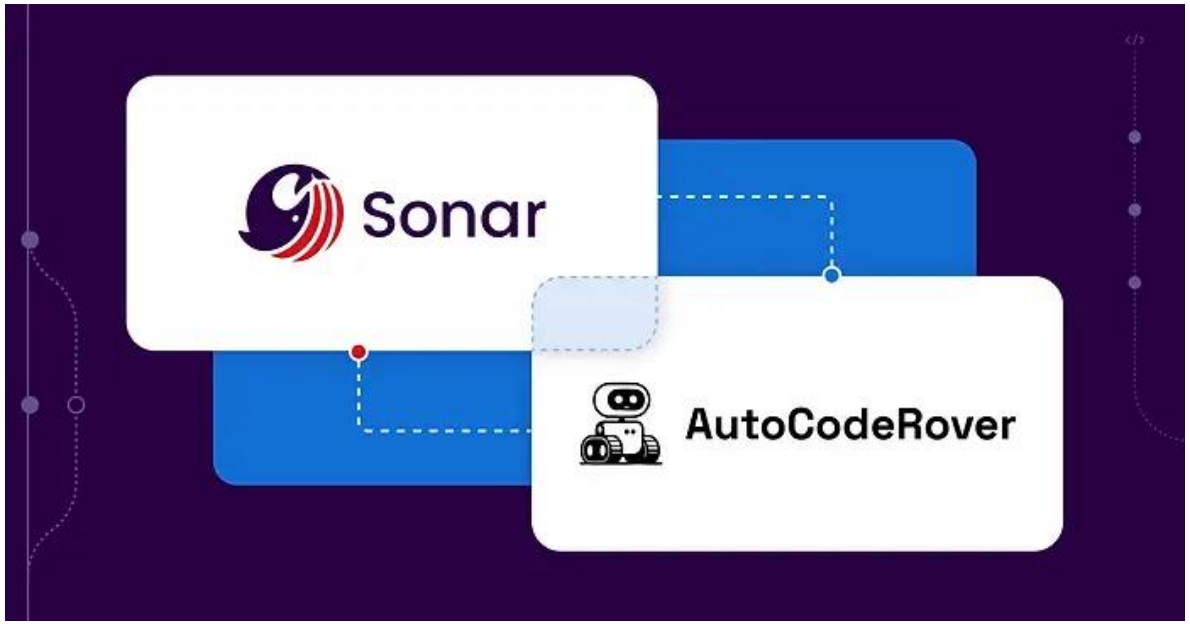
U_2

Method `validate` simply calls the `validate` method of its parent class.

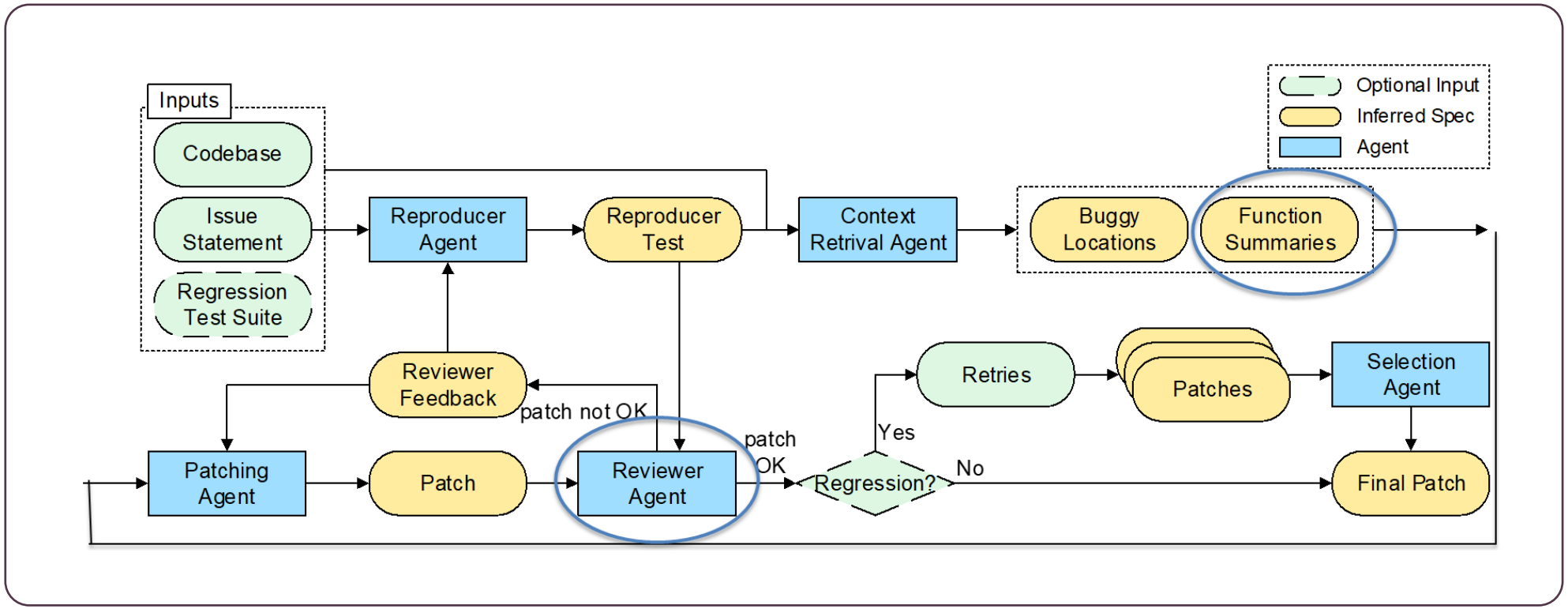
`to_python` method is responsible …
Should adjust the error message …

AGENTIC DESIGN

- Analysis embedded inside the agent
  - Could invoke tools as part of the analysis
- **Cannot be accomplished simply by mathematical analysis of code**
- **Cannot be accomplished simply by natural language analysis of text**
- In this example used only program structure for analysis. More involved analysis is possible!

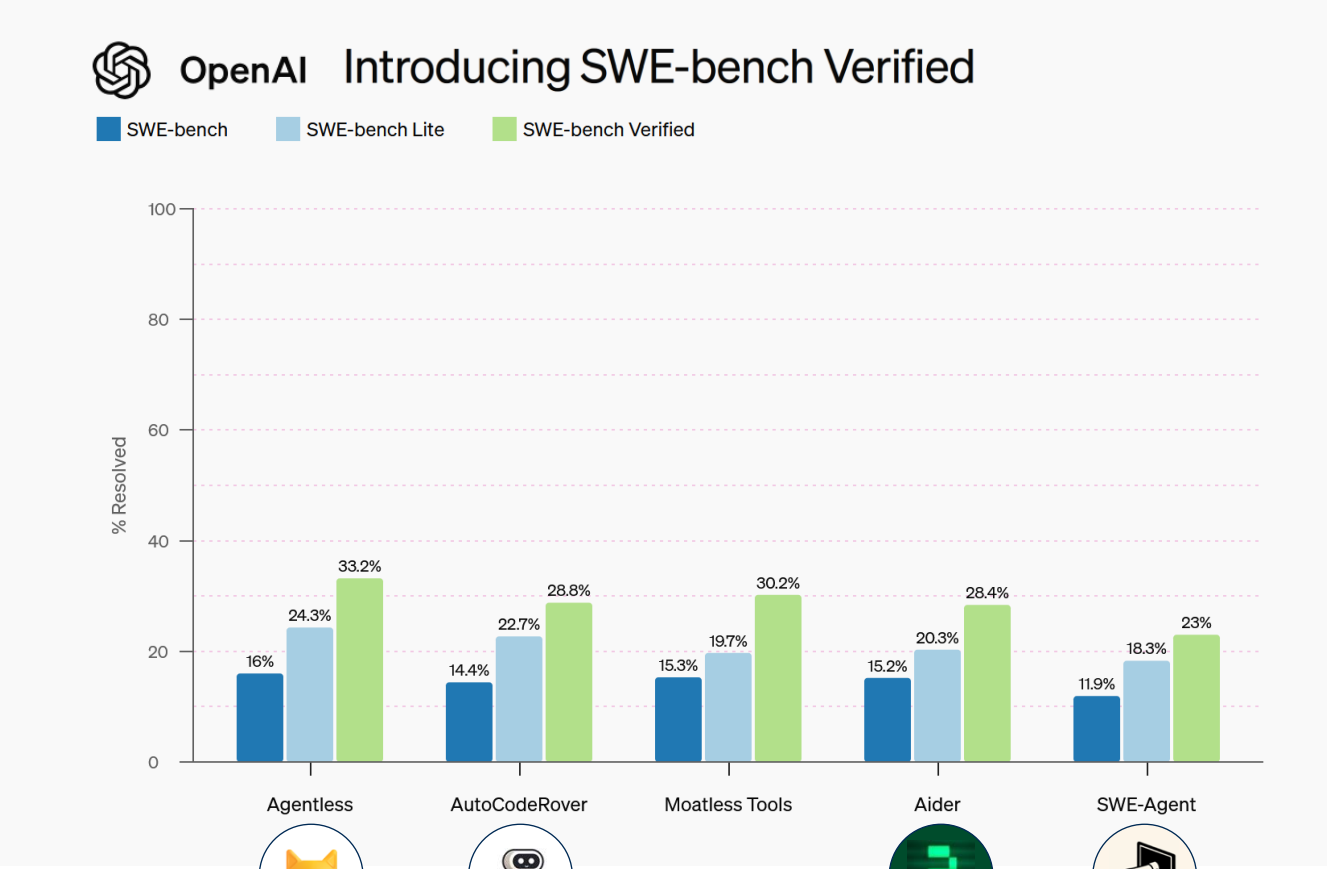


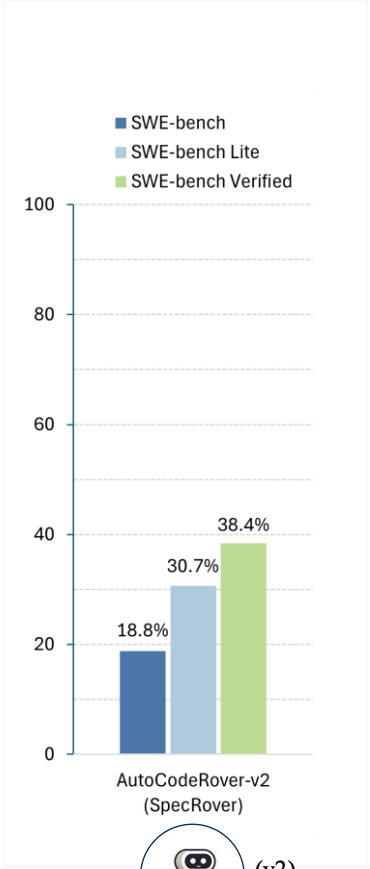
EXPLICIT INTENT



SpecRover: ICSE 2025, Ruan, Zhang, Roychoudhury.

“UPS” AND “DOWNS” IN INNOVATION





All agents using gpt-4o as the backend LLM.

SIGNAL TO NOISE RATIO !

If an automated tool has efficacy of 20%, does it mean the user needs to manually examine and reject the wrong patch in the remaining 80% of the cases?

- Signal-to-noise ratio is important!
- Does the reviewer agent improve signal-to-noise ratio?

"patch is accepted" => when reviewer agent decides both test and patch are correct.

Four categories:
- True positive: accepted and correct.
- True negative: rejected and incorrect.
- False positive: accepted but incorrect.
- False negative: rejected but correct.

$Tot = TP+FP+TN=FN$

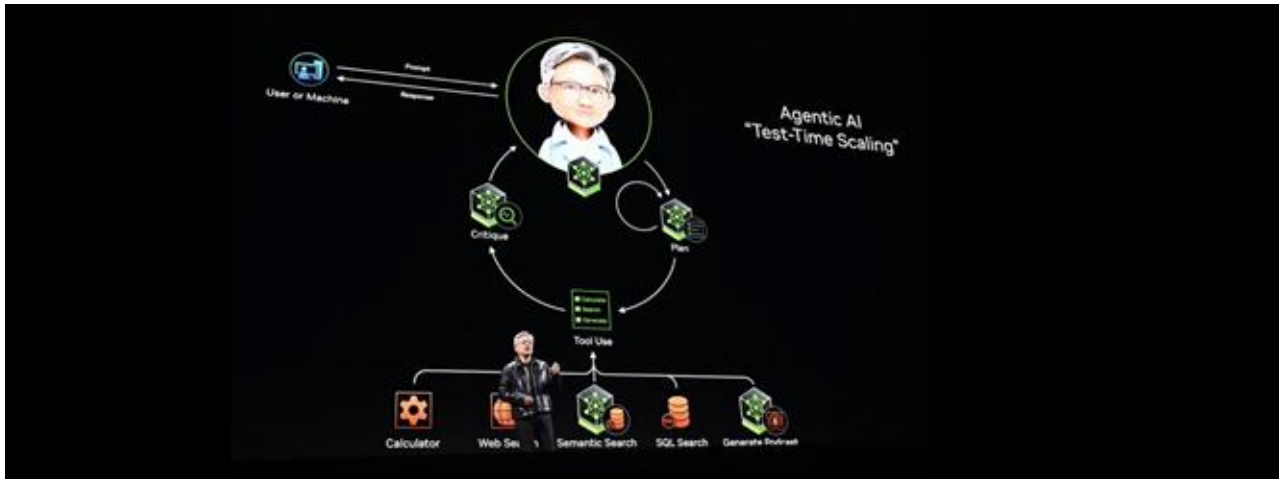
$Acc. = TP+TN \ / \ Total$
$Prec. = TP \ / \ (TP + FP)$
$Rec. = TP \ / \ (TP + FN)$

In practical deployment, only send a patch if it is accepted by reviewer.
⇒ Higher signal-to-noise ratio
⇒ Greater trust !

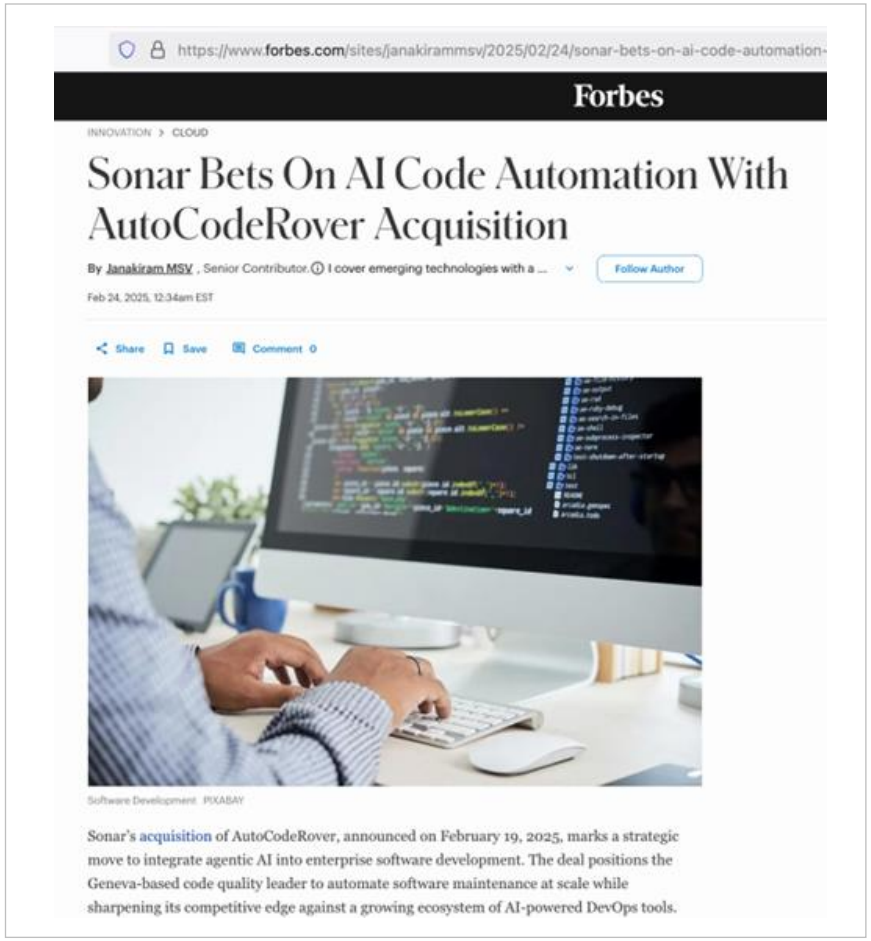
AGENT: BEYOND PROMPTS: AUTOCODEROVER

Nvidia CEO Jensen Huang Consumer Electronics Show (CES) 2025 unveiled advanced AI for training agents, robots and cars.
Photo by：Artur Widak/Anadolu via Getty Images）

2025: "AI agents represent a multi-trillion $ opportunity"



Integrated inside SonarQube Code Analysis tool SonarQube, which is in use by >100,000 enterprise customers for enhancing code quality and security. Continuing work.



https://www.forbes.com/sites/janakirammsv/2025/02/24/sonar-bets-on-ai-code-automation-

Forbes

INNOVATION > CLOUD

Sonar Bets On AI Code Automation With AutoCodeRover Acquisition

By Janakiram MSV , Senior Contributor. I cover emerging technologies with a ... Follow Author

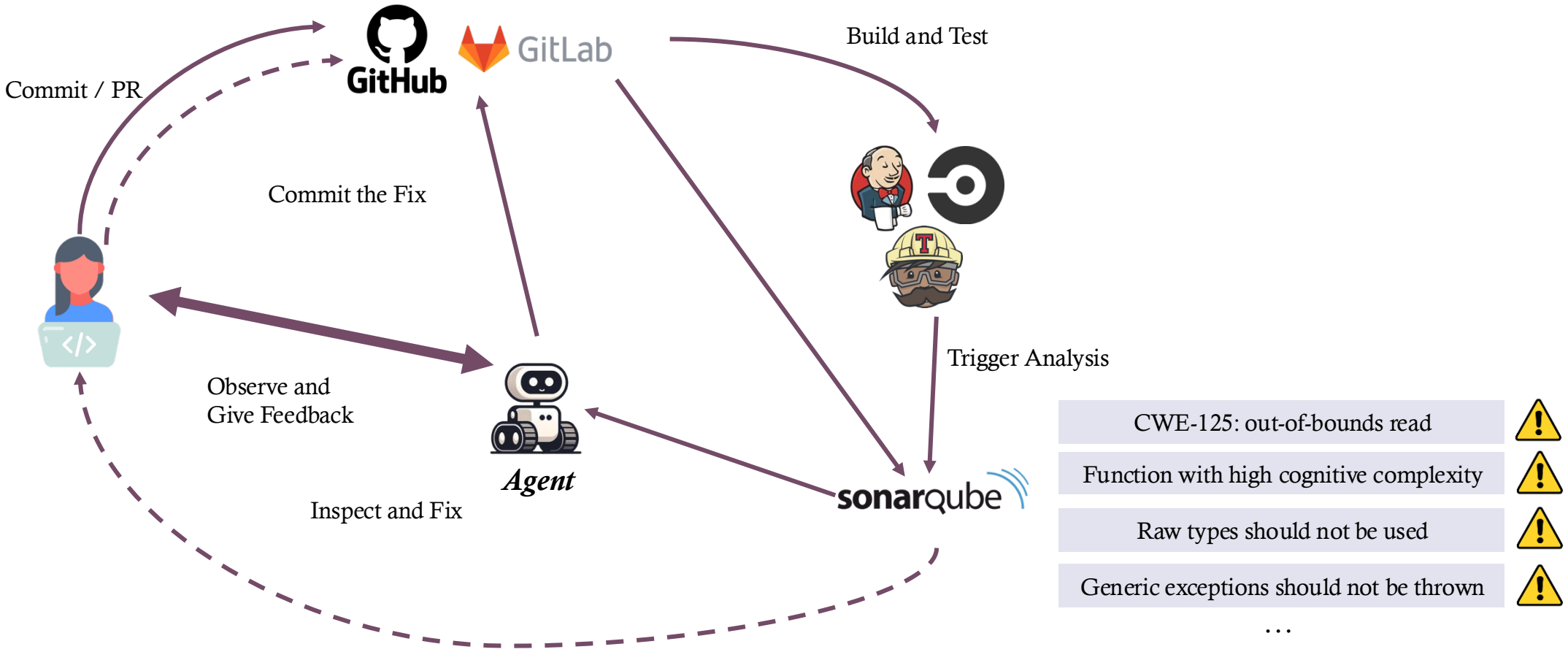
Feb 24, 2025, 12:34am EST

Share Save Comment 0

Software Development PIXABAY

Sonar's acquisition of AutoCodeRover, announced on February 19, 2025, marks a strategic move to integrate agentic AI into enterprise software development. The deal positions the Geneva-based code quality leader to automate software maintenance at scale while sharpening its competitive edge against a growing ecosystem of AI-powered DevOps tools.

SONARQUBE REMEDIATION AGENT



sonarqube-agent bot commented 5 days ago

Agent Fix (Issue 1 of 25)

Quality	Issue
Maintainability 🟠 Medium	Unexpected lexical declaration in case block.

Location 1: **src/core/cff_parser.js**:914-914

```
--- a/src/core/cff_parser.js
+++ b/src/core/cff_parser.js
@@ -914,1 +914,1 @@ class CFFParser {
-        case 0:
+        case 0: {
```

▼ View suggestion details

Explanation 1:

This opens a block `{ ... }` for `case 0` in a switch statement. That allows the `const` and `let` declarations inside that case (such as glyphsCount) to be properly scoped within a block instead of being hoisted across the entire switch, addressing the warning about unexpected lexical declarations directly inside case clauses.

▼ Show 1 other location

Location 2: **src/core/cff_parser.js**:920-920

```
--- a/src/core/cff_parser.js
+++ b/src/core/cff_parser.js
@@ -919,0 +920,1 @@ class CFFParser {
+        }
```

▶ View suggestion details

sonarqube-agent bot commented 3 minutes ago

Agent Fix (Issue 3 of 7)

Quality	Issue
Reliability 🔴 High	Use an asynchronous file API instead of synchronous open() in this async function.

Location 1: **examples/advanced_startup.py**:60-61

```
--- a/examples/advanced_startup.py
+++ b/examples/advanced_startup.py
@@ -60,2 +60,2 @@ class CustomBot(commands.Bot):
-        f = open("bot_status.txt", "w")
-        f.write("Bot is starting up")
+        loop = asyncio.get_running_loop()
+        await loop.run_in_executor(None, self._write_startup_status)
```

▼ View suggestion details

Explanation 1:

Previously, an async method directly called open() and wrote to a file, which is a blocking file operation inside an async function. This hunk replaces that with getting the current event loop and awaiting loop.run_in_executor(None, self._write_startup_status). That offloads the blocking file I/O to a thread pool executor, so the event loop is no longer blocked by the synchronous open() call, addressing the warning about using synchronous file APIs inside async code.

▶ Show 1 other location

Review

☐ **Select fix**

Select one or more fixes above to enable this action.

🔒 **Commit changes (0 of 7 selected)**

Select this to commit all previously selected fixes. If it takes longer than a few seconds, try refreshing the page.

Go back to Summary ⬆️

Did this fix help?

☐ 👍 ☐ 👎

REAL INCIDENT, ACTUAL TIMING

May 18 2023: Most Influential Paper Award Talk for 2013 paper Intl. Conf on SW Engg (ICSE)

Oct 24, 2023: Started solution on Large Language Model agents for SW Engg.
"Imagine all of the program analysis can be invoked autonomously"
Apr 8, 2024: Public announcement in X, Excitement around AutoCodeRover.
Feb 19, 2025: Acquisition by SonarSource announced, 9 am EST, 10 pm SGT.
Feb 20, 2025: Contacted for a group photo, realized there are no photos at all!!
Feb 21, 2025: Met for the first time outside work as a group

Crucial time in the innovation cycle

REFLECTIONS

"Hello World"
1972

~50 years

Programming at Scale

Linux Kernel in 2024
~30M LoC

Automatically generated code

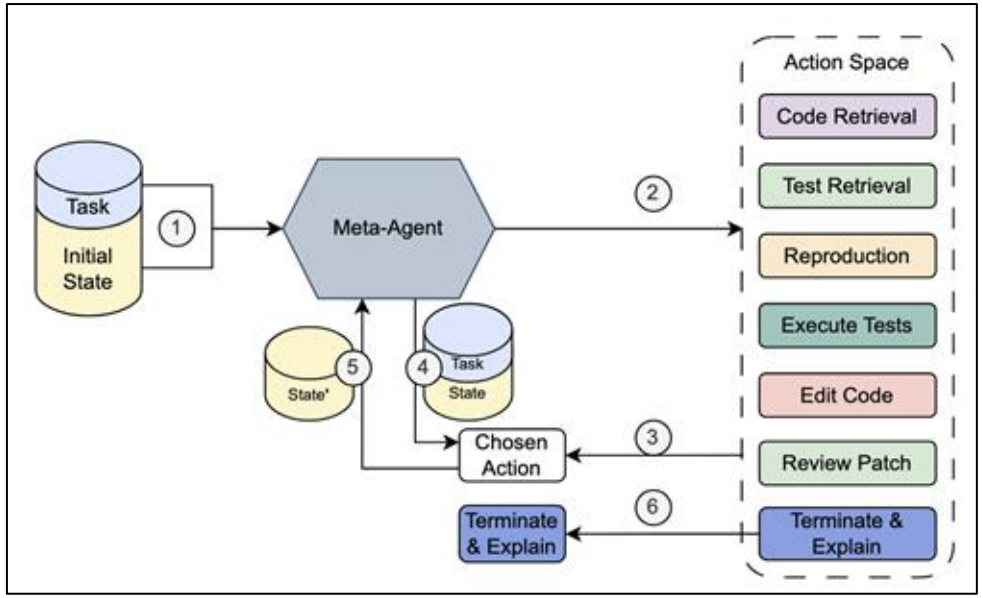
~X years

Programming with Trust
Role for Verification

Cooperative Intelligence

When to trust the agent?

BEYOND CODING, SE ?



Unified agent

Handles multiple task types without manual configuration

Dynamically deciding its next action like human SWE

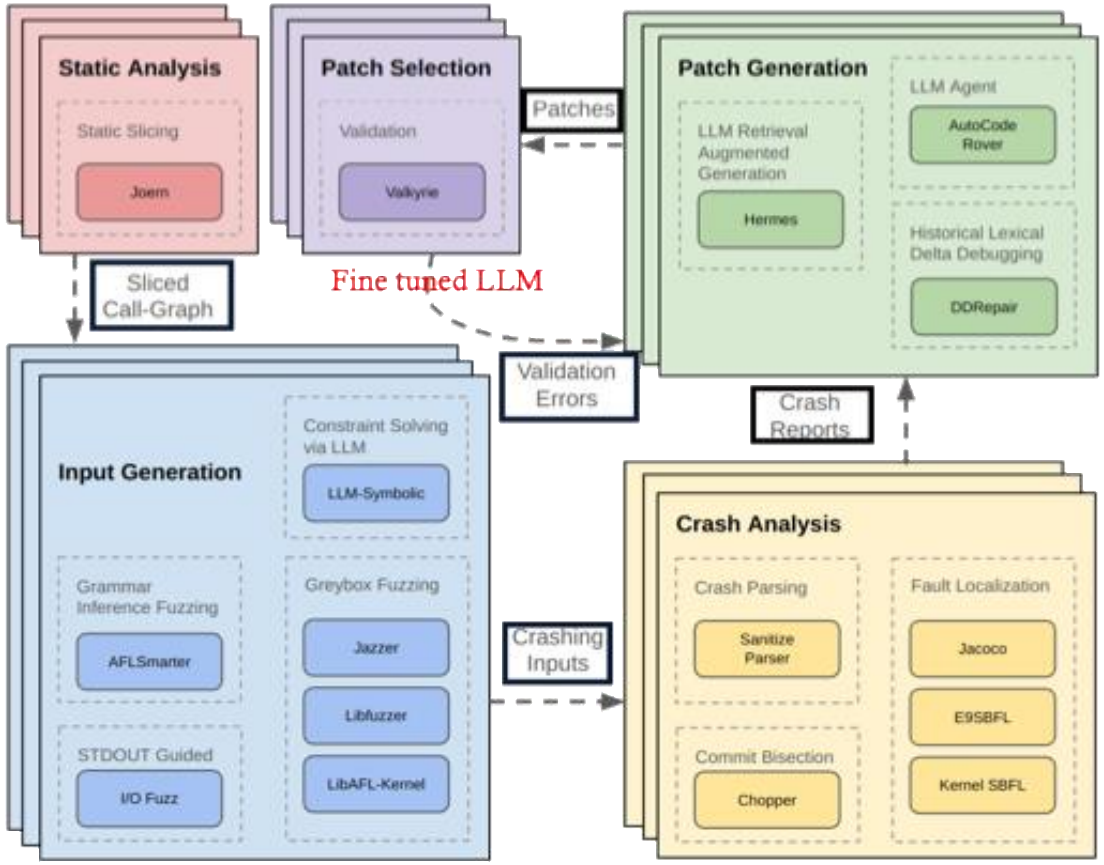
- Issue resolution
- Regression testing
- Code generation
- Test generation
- Partial fix improvement …

- Architecture exploration
- Requirements clarification

[Unified Software Engineering Agent ICSE 26]

SECURITY ?

Role of text-based analysis to be seen

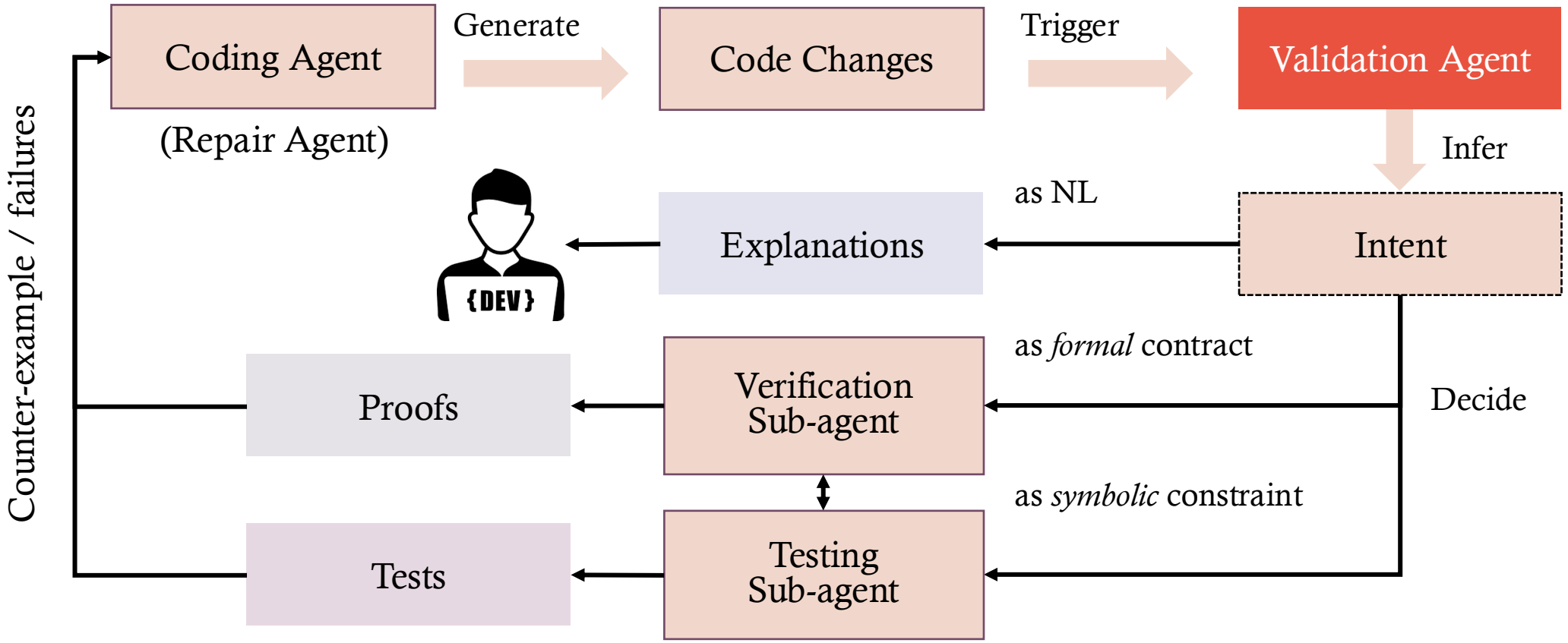


LLM agent with significant analysis

Fuzzing
→
LLM guided random/fuzzing search

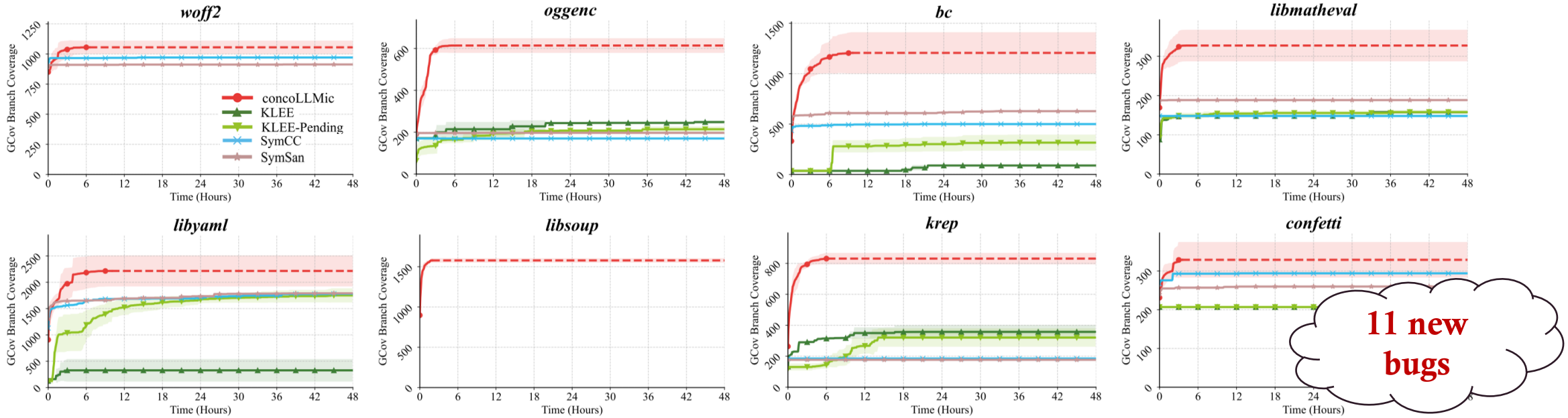
LLM agent

V & V ?

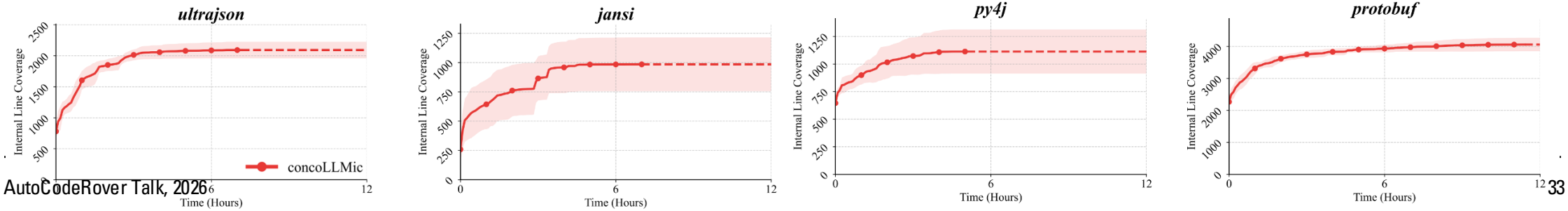

Coding Agent
(Repair Agent)
Generate
Code Changes
Trigger
Validation Agent
Infer
Intent
as NL
Explanations
{DEV}
as *formal* contract
Verification Sub-agent
Decide
as *symbolic* constraint
Testing Sub-agent
Proofs
Tests
Counter-example / failures

SYMBOLIC REASONING FOR TESTS ?

1. *Monolingual Projects:* KLEE: +234%, KLEE^{pending}: +135%, SymCC: +130%, SymSan: +115%



2. *Polyglot Systems*: Not supported by existing tools. Our tool still achieves consistent coverage growth

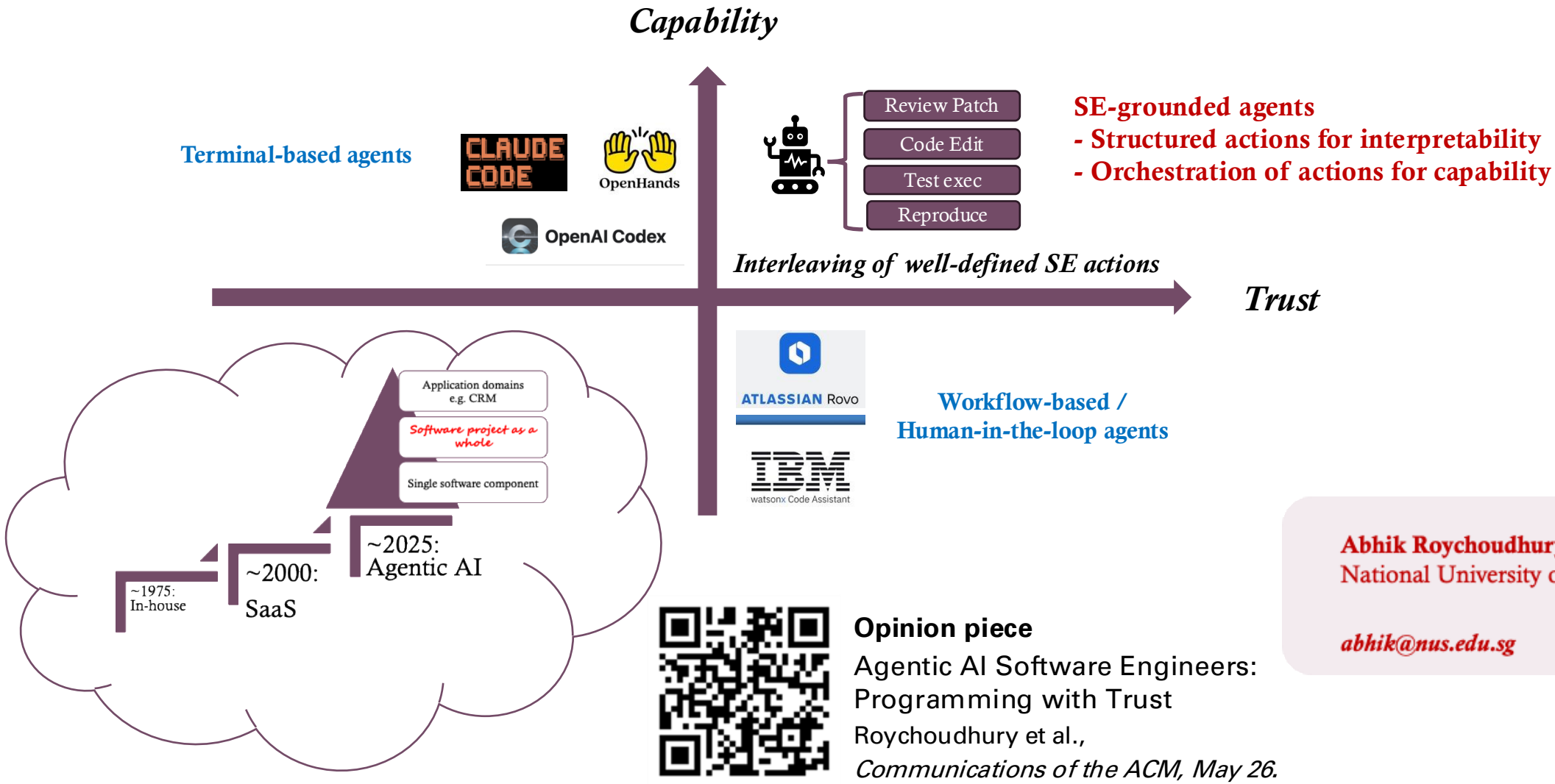


LOOKING FORWARD: PROOFS ?

- Automation in software development is overwhelming
- Put agentic AI to task in combating scale for proving program properties
- Automatic verification of automatically generated code!
  - Proof Search Automation exists
  - Lemma discovery is a research frontier

New Project 2026
AI for Program Reasoning
NUS & Imperial College London

WHERE IS SOFTWARE GOING ?



Opinion piece

Agentic AI Software Engineers: Programming with Trust

Roychoudhury et al.,

Communications of the ACM, May 26.

Abhik Roychoudhury
National University of Singapore

abhik@nus.edu.sg