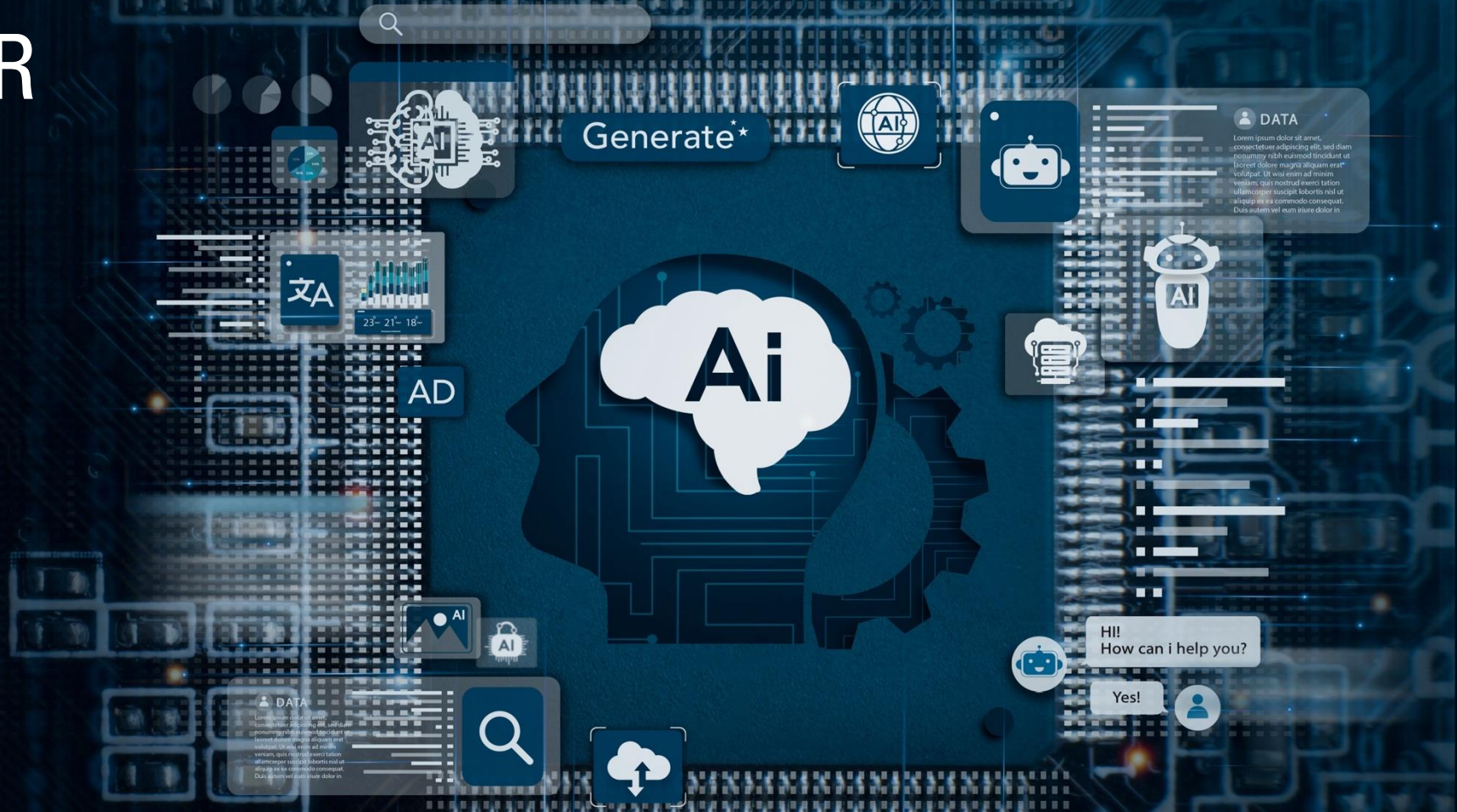
AGENTIC AI FOR SOFTWARE:
LESSONS IN TRUST

用于软件的智能体AI:
关于信任的经验



Abhik Roychoudhury
National University of Singapore
新加坡国立大学

LOVE FOR PROGRAMMING
对编程的热爱

(CLAUDE SONNET)
(由CLAUDE SONNET生成)

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
“HELLO WORLD”程序风格

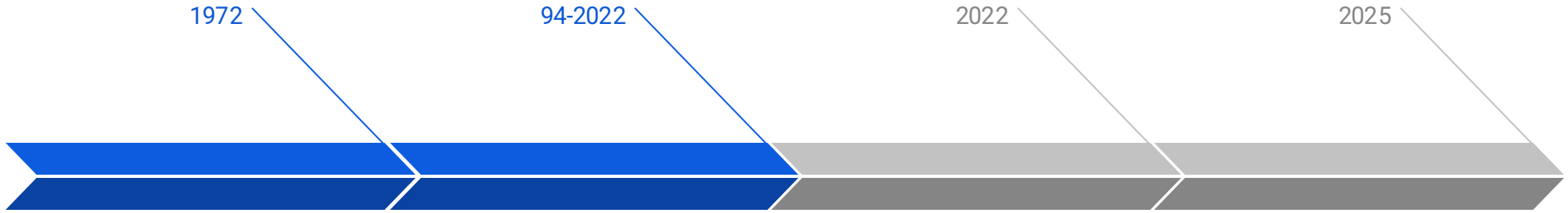
```
main( ) {
        extrn a, b, c;
        putchar(a); putchar(b); putchar(c); putchar('!*n');
}

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

B programming language

4-character limitation. ☺

B语言

4个字符的限制 ☺



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

BCPL/B 编写的Hello World 程序, 早于C语言

Brian Kerninghan

Windows / Linux

软件即服务

大型代码库 - 软件模型检查, Linux在2022年含有3千万行代码

互联网 -> 软件交付

GitHub Copilot

自动生成的代码被集成 – 程序验证的重要性上升.

大语言模型智能体的年份

自动生成的代码需要被验证从而集成

programming at scale 大规模编程

programming with trust 可信编程

SOFTWARE INDUSTRY OVER 50 YEARS
软件产业的50年历程

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

B和C语言编写的"Hello world"程序

~1975:
In-house
内部开发

~2000+:
SaaS
/Cloud
软件即服务/云端

Hosting of SW – Salesforce (CRM) Other app domains

软件托管 – Salesforce (客户关系管理) 其他业务领域

~2025:
Agentic
AI
智能体AI

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

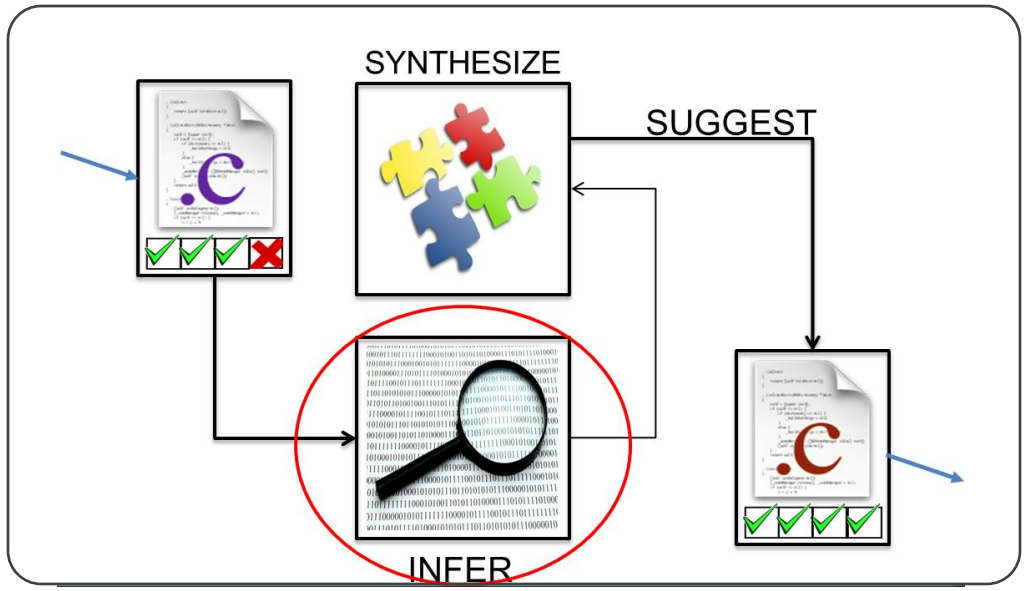
技术/水平方向: 软件本身的工程!
应用/垂直方向: ***下一个 SalesForce ?***

THE DAY OF A SOFTWARE ENGINEER
软件工程师的一天

- 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?

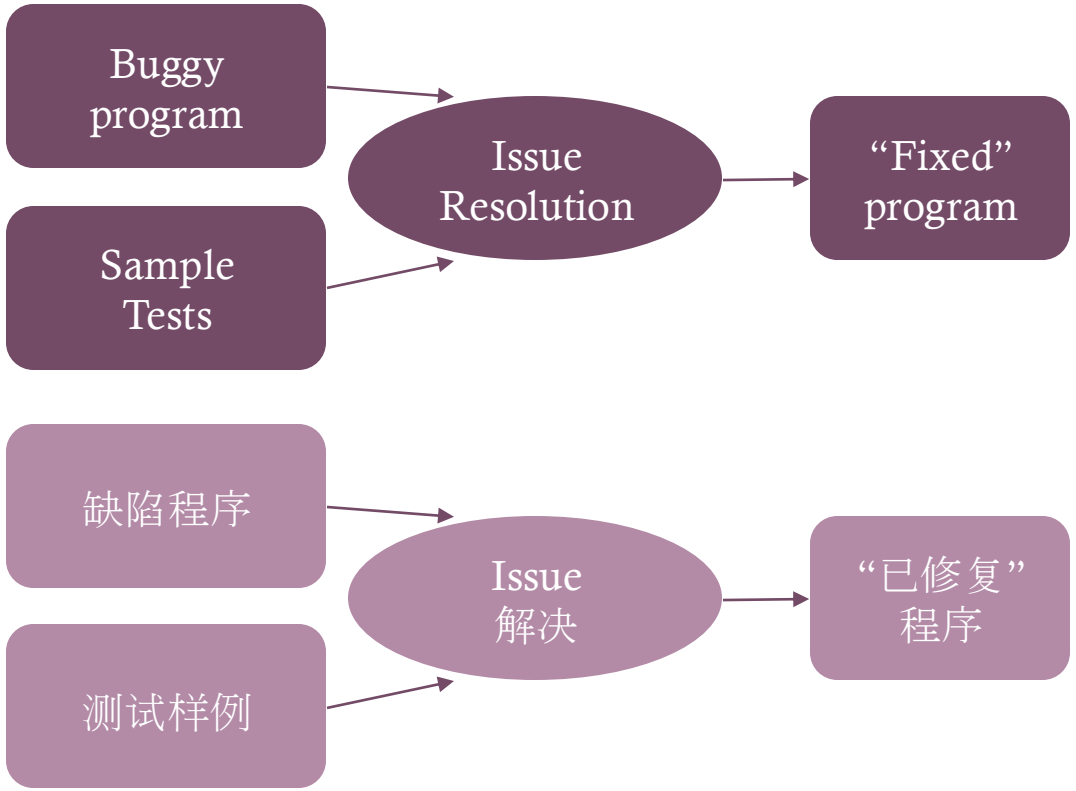
- 更多的是改进程序，而非从头编写代码
- 早晨的工作从许多“issue”开始
  - 一个issue可以是一个需要修复的缺陷报告
  - 或者新功能开发
  - 甚至是一部分代码的性能优化?

UNPACKING "ISSUES": INTENT
剖析"ISSUES": 程序意图



SemFix, ICSE 2013
Angelix, ICSE 2016

An issue can refer to a bug report and needed fix
一个issue可以是一个需要修复的缺陷报告



LEARNT AS A SCHOOL-CHILD ☺
小学知识

```
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?
可能至今还未掌握?

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
以测试用例形式给定的 "意图"

```
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 }
```

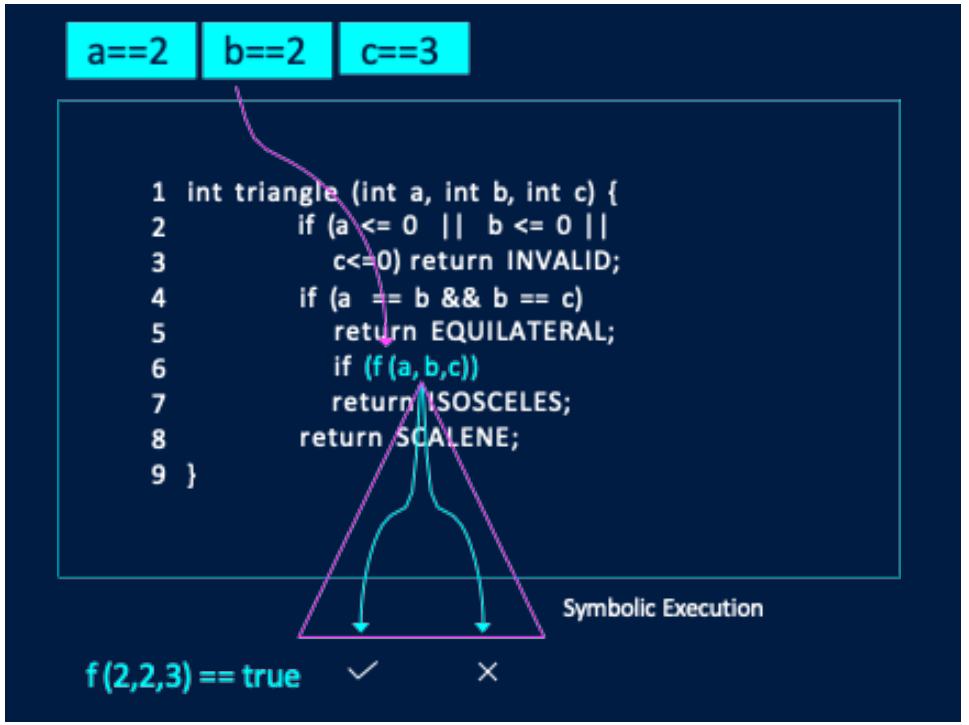
Buggy Program
缺陷程序

Automatically generate the fix 自动生成修复 `(a == b || b == c || c == a)`

FROM INTENT TO CODE – RELIABLY !
将意图转化为代码 – 以一种可靠的方式！

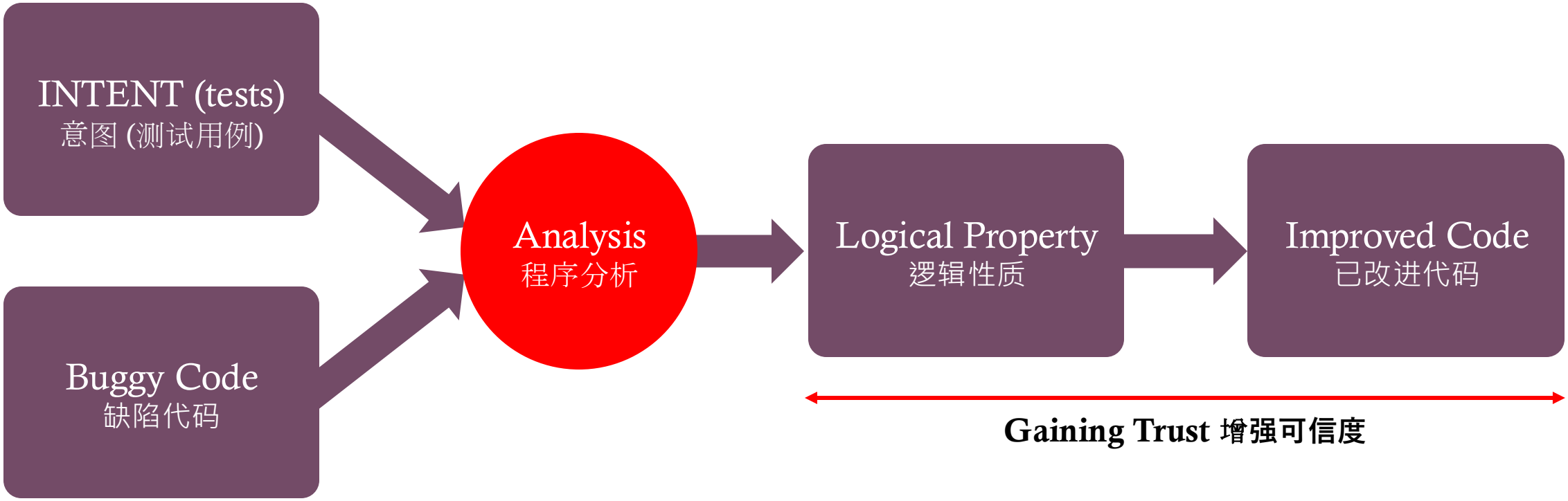
Test id	a	b	c	oracle	Pass
1	-1	-1	-1	INVALID	Yes
2	1	1	1	EQUILATERAL	Yes
3	2	2	3	ISOSCELES	Yes
4	2	3	2	ISOSCELES	Yes
5	3	2	2	ISOSCELES	NO
6	2	3	4	SCALANE	NO

Given ”intent” as tests
给定 “意图” 作为测试用例

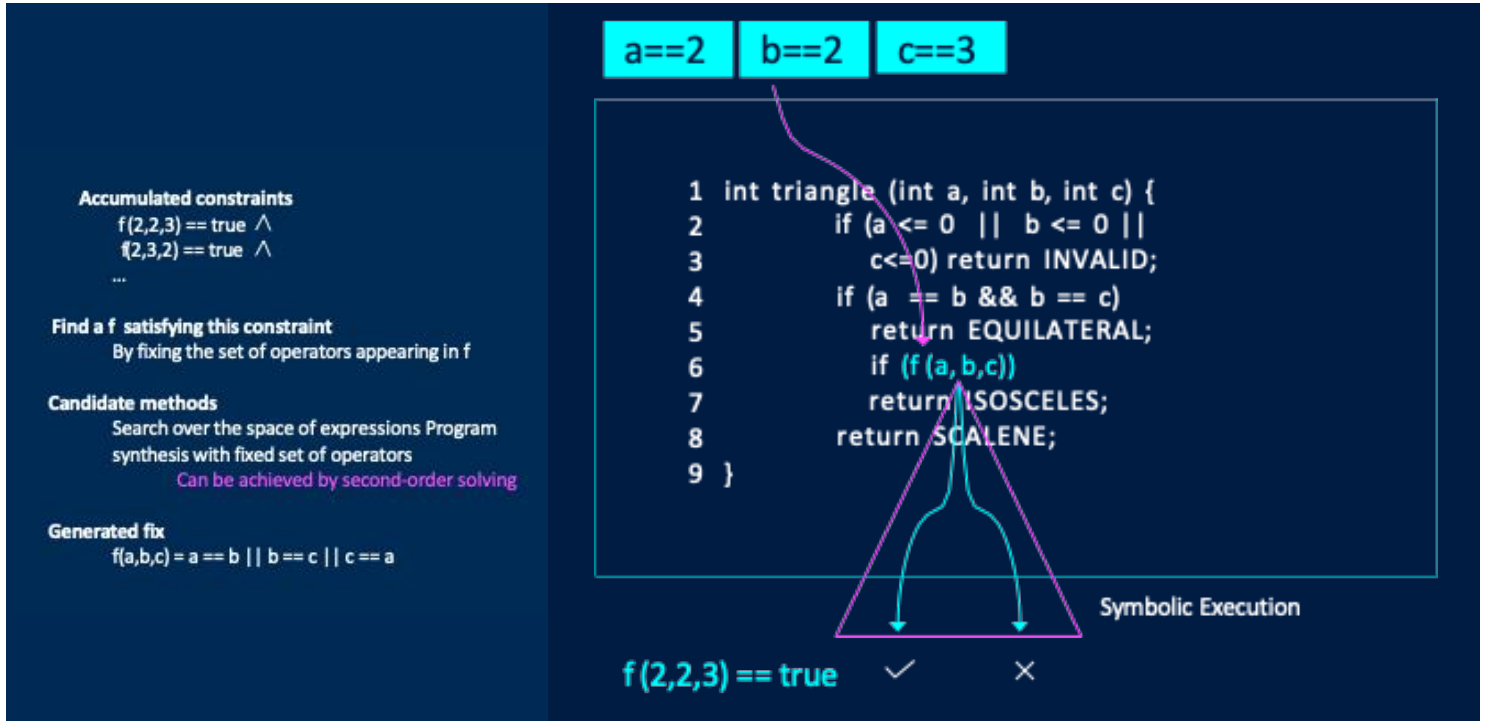


f(2,2,3) and f(2,3,2) and f(3,2,2) and not f(2,3,4) → `(a == b || b == c || c== a)`

TRUSTED AUTOMATIC PROGRAMMING
可信自动编程



INTENT FROM TESTS
从测试推断意图



Higher order logic inference from tests.
从测试推断高阶逻辑

Lot of machinery in achieving it efficiently in a first order logic framework.
通过许多手段在一阶逻辑框架下高效实现

Need a mechanism for extracting intent when tests are absent.
需要一种机制在测试缺失时提取意图

TRUSTED AUTOMATIC PROGRAMMING
可信自动编程

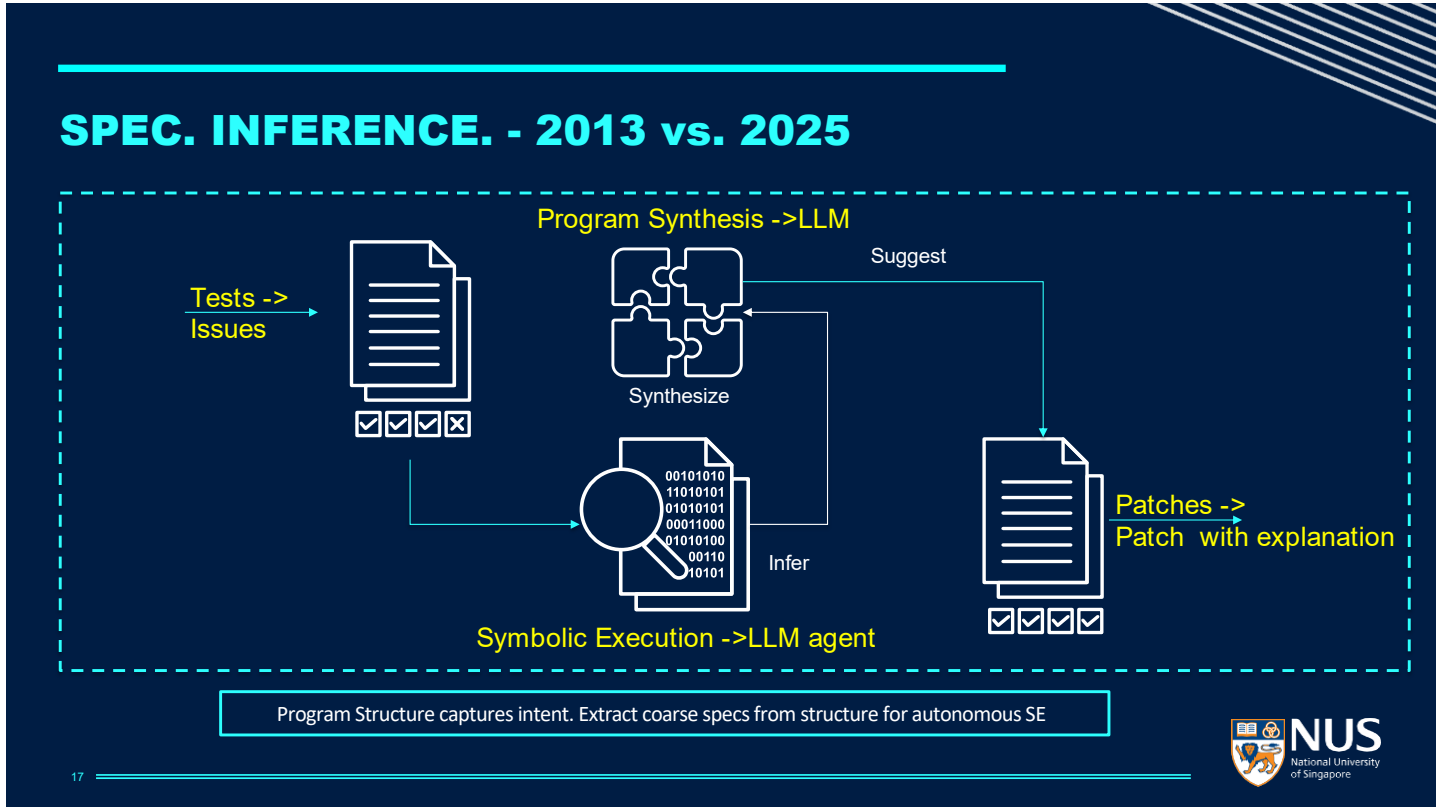
Buggy Code
缺陷代码

Analysis
程序分析

Improved Code
已改进代码

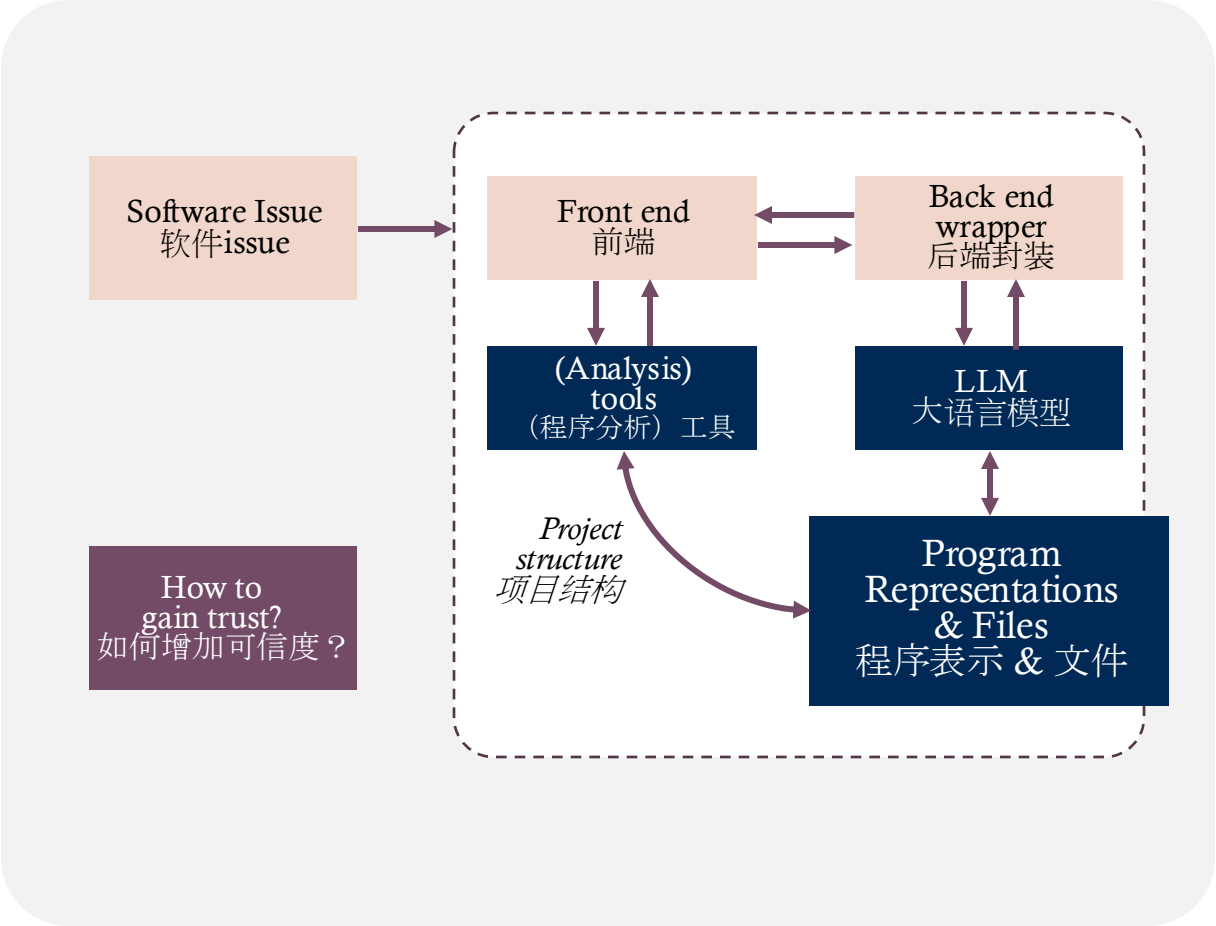
Gaining Trust 增强可信度

THEN AND NOW
从过去到现在，变与不变

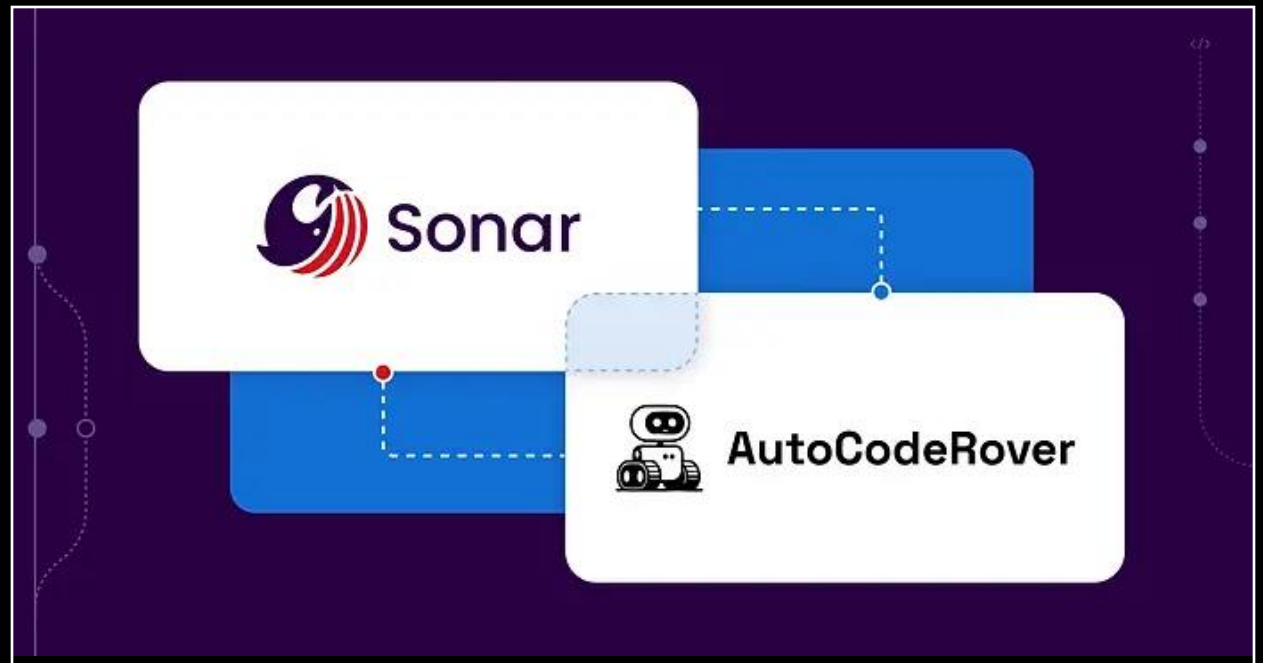


ISSUE RESOLUTION
解决 ISSUE

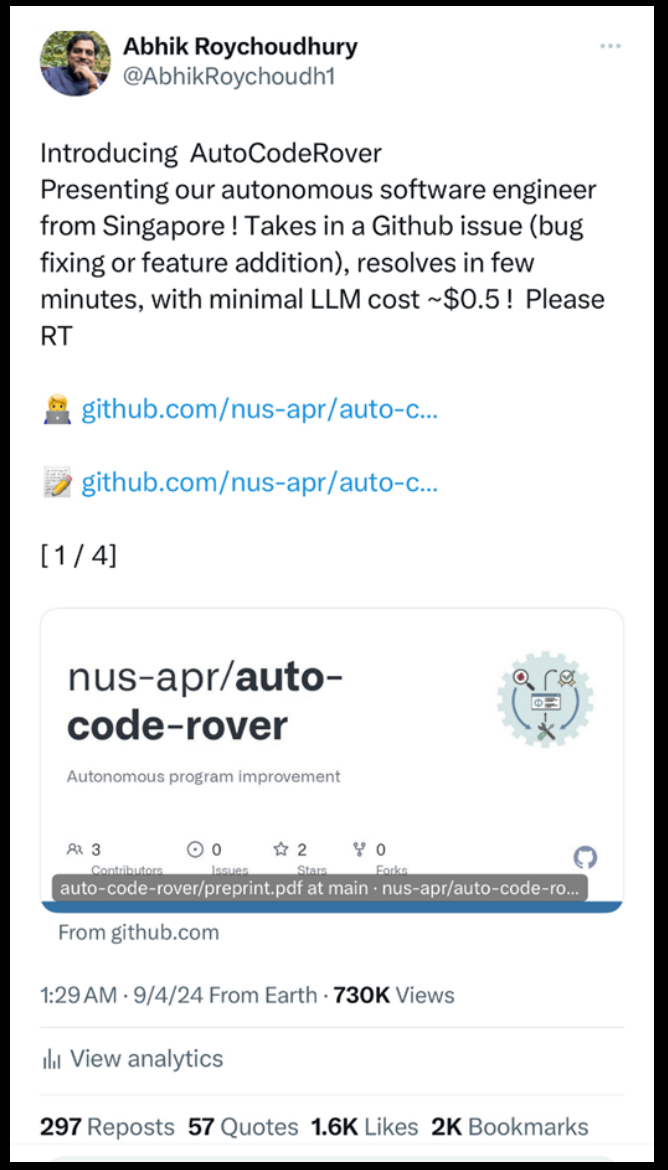
Do not see Code as text!
不要将代码看做纯文本！



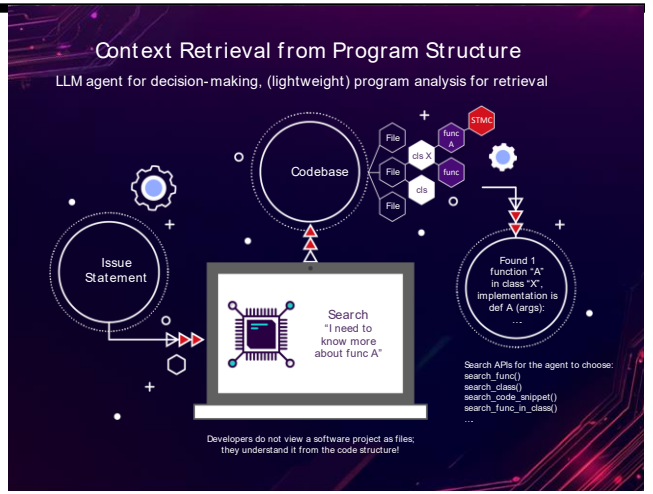
AUTOCODEROVER

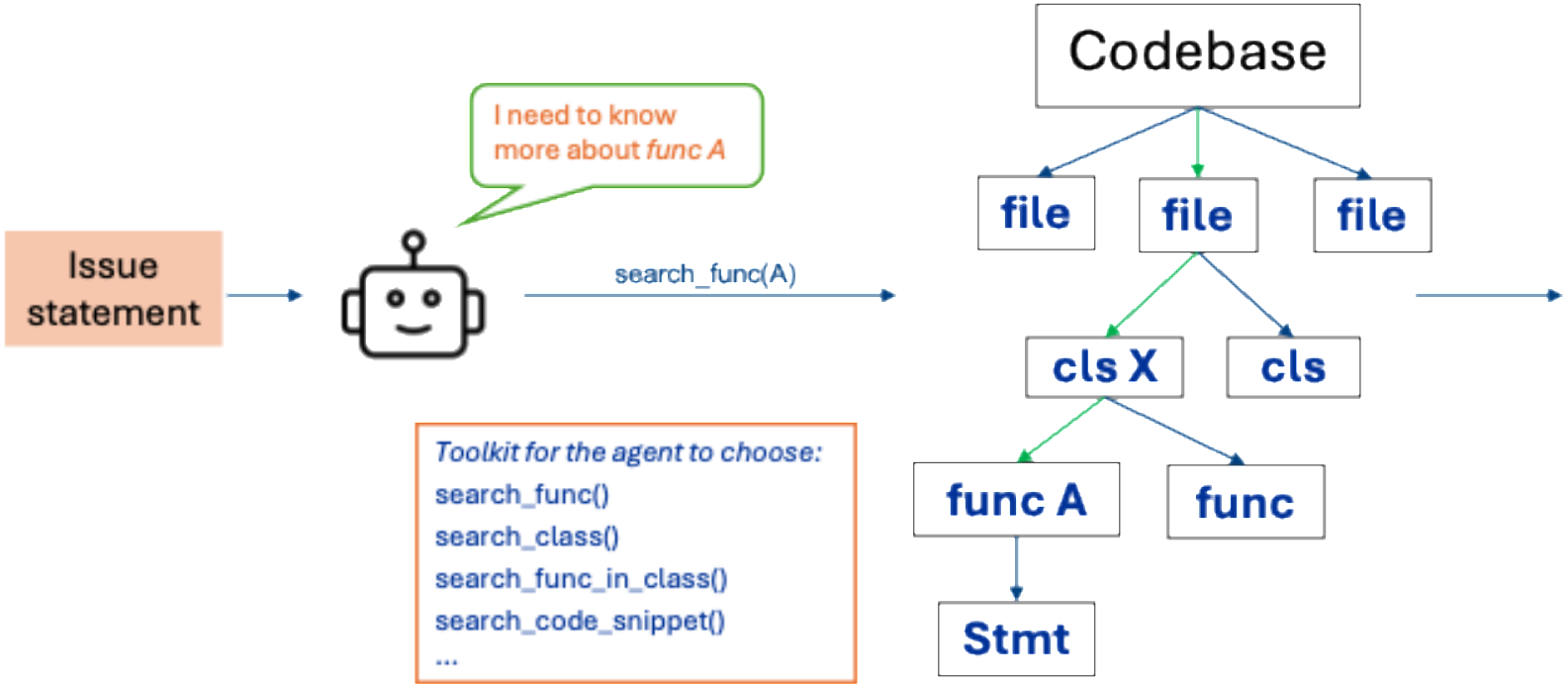


https://autocoderover.dev/



IMPLICIT INTENT
隐式意图





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

STRATIFIED SEARCH
分层搜索

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

Iteratively refine issue understanding based on latest context.

基于最新的上下文，迭代式细化issue理解

#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 …

$Context_1$ search_class("ModelChoiceField")

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):
```

STRATIFIED SEARCH
分层搜索

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

Iteratively refine issue understanding based on latest context.
基于最新的上下文，迭代式细化issue理解

#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 …

$Context_1$ search_class("ModelChoiceField")

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`.

$Context_2$ search_method_in_class("validate", "ModelChoiceField")
search_method_in_class("to_python", "ModelChoiceField")

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

STRATIFIED SEARCH
分层搜索

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

Iteratively refine issue understanding based on latest context.
基于最新上下文，迭代式细化issue理解

#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 …

$Context_1$ search_class("ModelChoiceField")

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`.

$Context_2$ search_method_in_class("validate", "ModelChoiceField")
search_method_in_class("to_python", "ModelChoiceField")

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

U_2

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

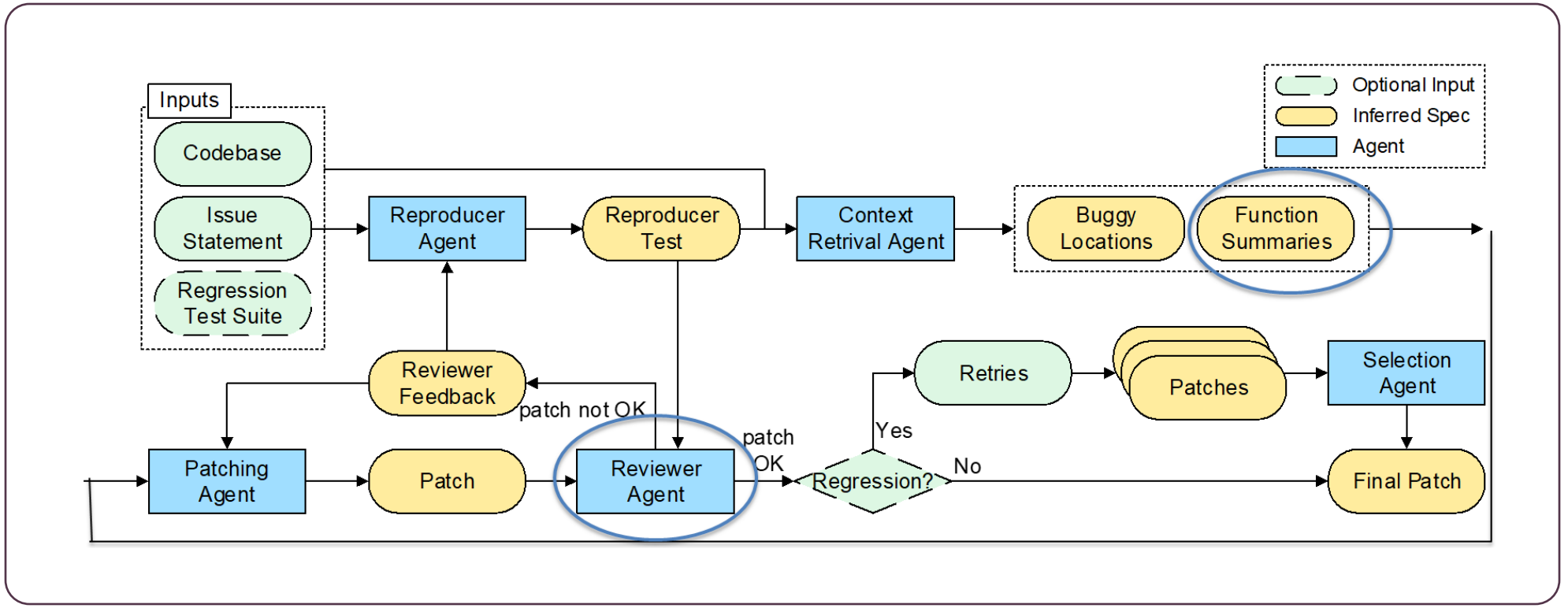
`to_python` method is responsible …
Should adjust the error message …

AGENTIC DESIGN
智能体设计

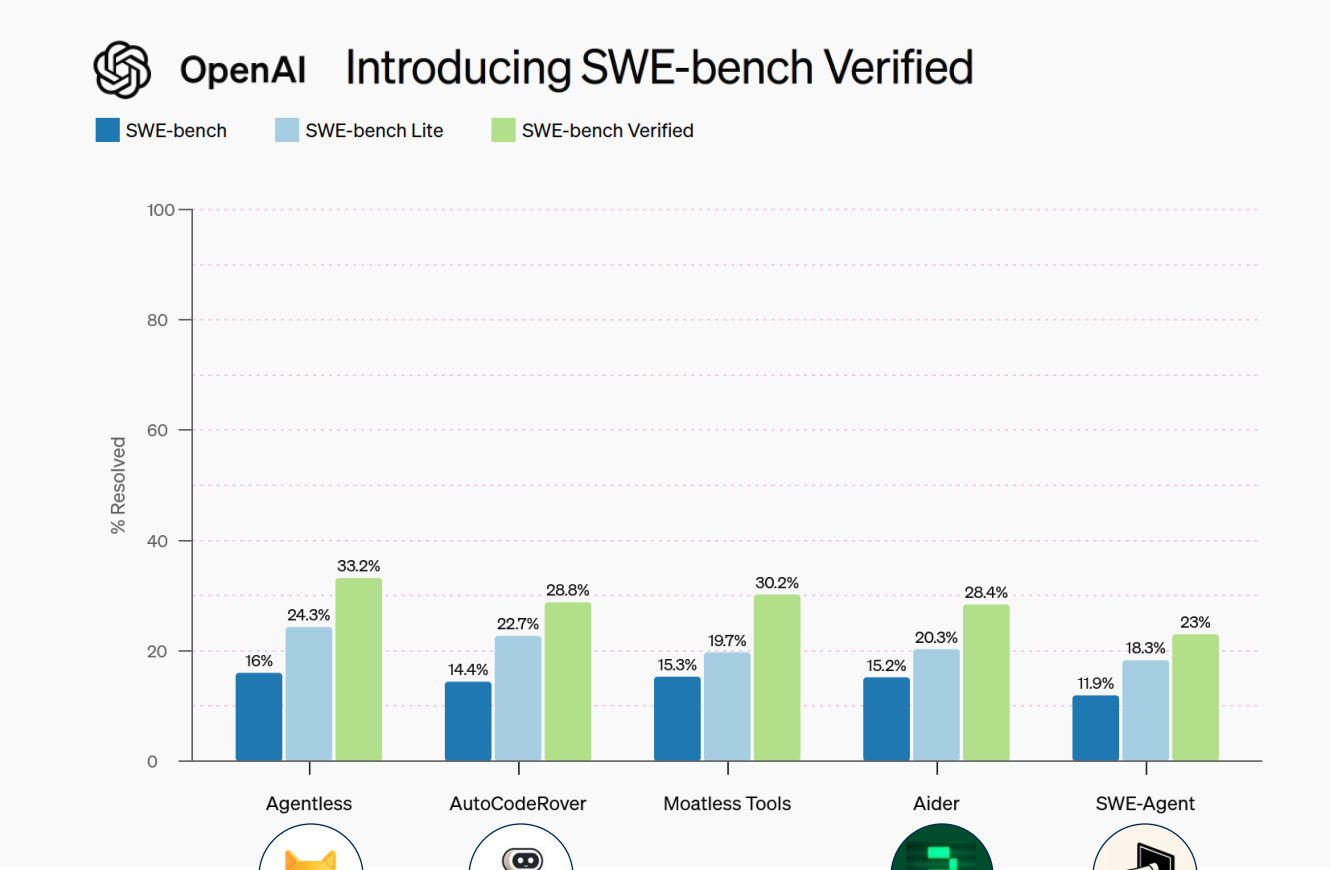
- Analysis embedded inside the agent
  - Could invoke tools as part of the analysis
- **Cannot be accomplished simply by mathematical analysis of code**
- **Cannot be accomplished simply by natural language analysis of text**
- In this example used only program structure for analysis. More involved analysis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

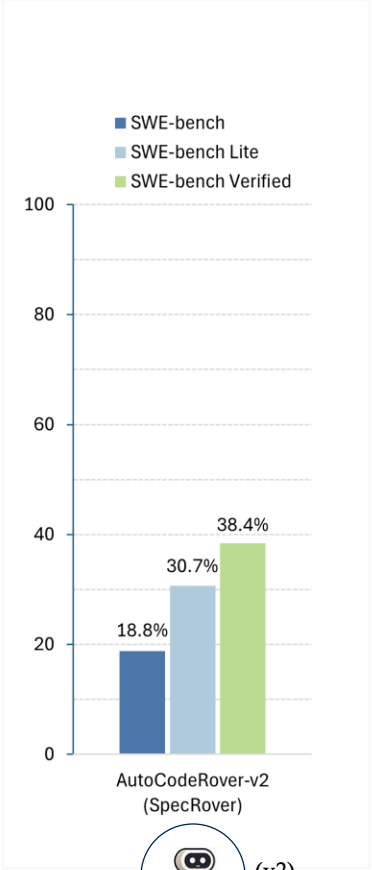
- 将分析嵌入智能体中
  - 分析过程可以调用工具
- **无法仅靠对代码的数学分析达成**
- **无法仅靠对文本的自然语言分析达成**
- 这个例子仅仅分析了程序结构。更复杂的分析完全有可能!

EXPLICIT INTENT
显式意图



“UPS” AND “DOWNS” IN INNOVATION
创新中的“起”与“落”





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

所有智能体都使用gpt-4o作为后端大语言模型。

USER ACCEPTABILITY IN AUTONOMOUS PROGRAM IMPROVEMENT
自主程序改进中的用户可接受性

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?
假如一个自动化工具的有效性为20%, 这是否意味着其余80%的情况中，用户需要手动检查并拒绝错误的补丁？

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

- 信噪比至关重要！
- Reviewer智能体能否提高信噪比?

"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.

"补丁被接受" => 当reviewer智能体断定测试用例和补丁均正确时

四个类别:
- 真正例: 被接受，正确
- 真负例: 被拒绝，错误
- 假正例: 被接受，错误
- 假负例: 被拒绝，正确

$Tot = TP+FP+TN=FN$

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

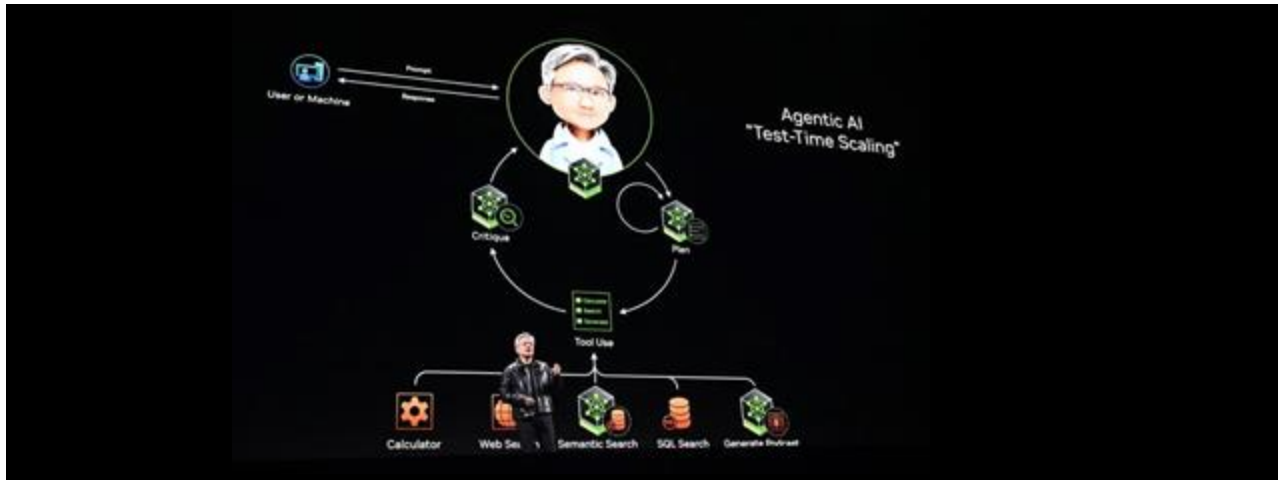
In practical deployment, only send a patch if it is accepted by reviewer.
⇒ Higher signal-to-noise ratio
⇒ Greater trust !

在实际部署中，只提交被reviewer智能体接受的补丁
⇒ 更高的信噪比
⇒ 更高的可信度!

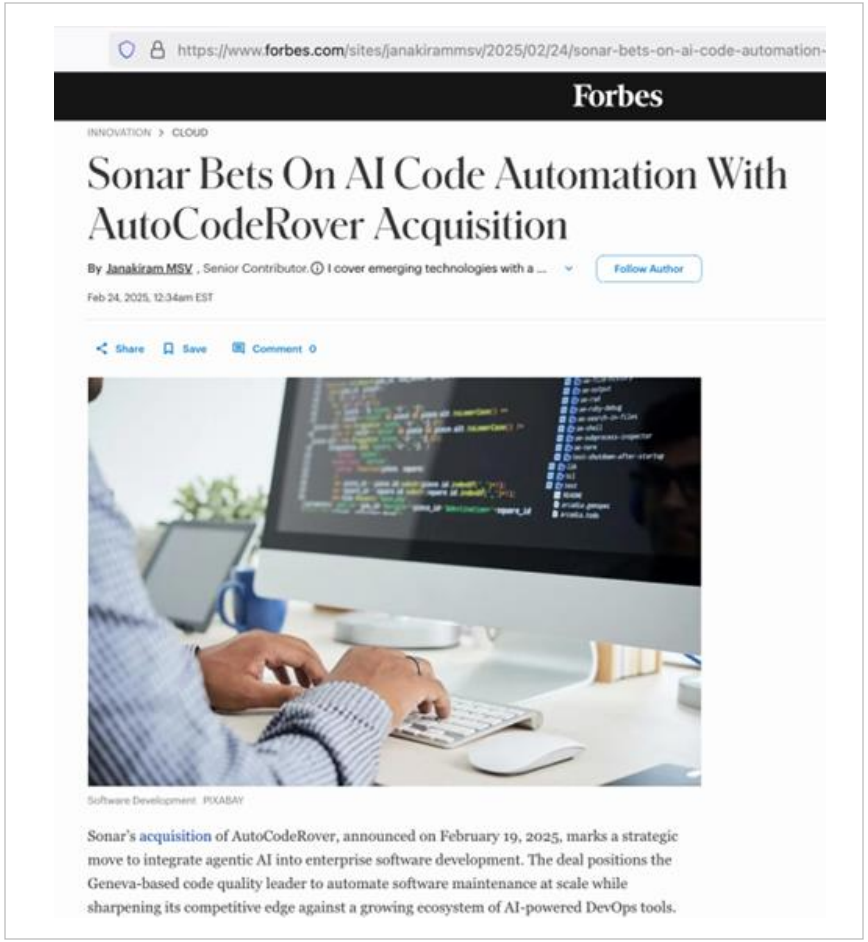
AGENT: BEYOND PROMPTS: AUTOCODEROVER
智能体: 超越提示词: AUTOCODEROVER

Nvidia CEO Jensen Huang Consumer Electronics Show (CES) 2025 unveiled advanced AI for training agents, robots and cars.
在2025年国际消费电子展上，英伟达首席执行官黄仁勋发布了用于训练智能体、机器人及汽车的先进人工智能技术。（ Photo by 图片来源：Artur Widak/Anadolu via Getty Images）

2025: "AI agents represent a multi-trillion $ opportunity"
2025: "AI智能体代表着一次价值数万亿美元的机遇"



Integrated inside SonarQube Code Analysis tool SonarQube, which is in use by >100,000 enterprise customers for enhancing code quality and security. Continuing work.
集成到SonarQube代码分析工具 SonarQube， 被超过100,000个企业客户用于提升代码质量和安全性


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.

REAL INCIDENT, ACTUAL TIMING
真实事件回放

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

Crucial time in the innovation cycle

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

2023年5月18日： 因2013年发表于国际软件工程大会（ICSE）的论文荣获“最具影响力论文奖”，并做主题演讲。

创新流程中的关键时刻

2023年10月24日： 开始开发用于软件工程的大型语言模型智能体解决方案。
“设想所有程序分析都能被自动调用”
2024年4月8日： 在X发布公告，AutoCodeRover引发广泛关注。
2025年2月19日： 宣布被SonarSource收购，美东时间上午9点，新加坡时间晚上10点。
2025年2月20日： 收到团体照拍摄通知，才发现根本没有合影！
2025年2月21日： 团队首次在工作以外聚会

REFLECTIONS
反思

"Hello World"
1972

~50 years 年

Programming at Scale
大规模编程

Linux Kernel in 2024
~30M LoC
Linux 内核 2024年约3千万行代码

Automatically generated code
自动生成的代码

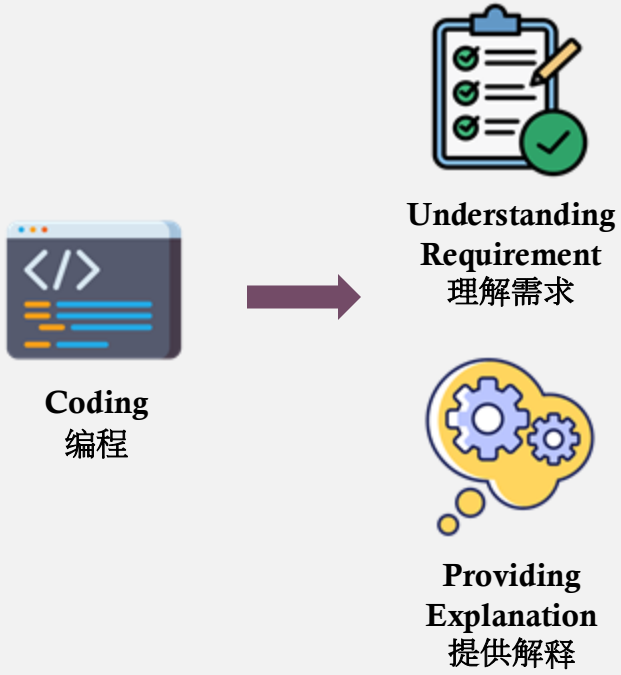
~X years 年

Programming with Trust
Role for Verification
可信编程
程序验证将发挥重要作用

Cooperative Intelligence
协作式智能

When to trust the agent?
什么时候可以信任智能体？

FROM CODING TO COMPLIANCE
从编程到合规

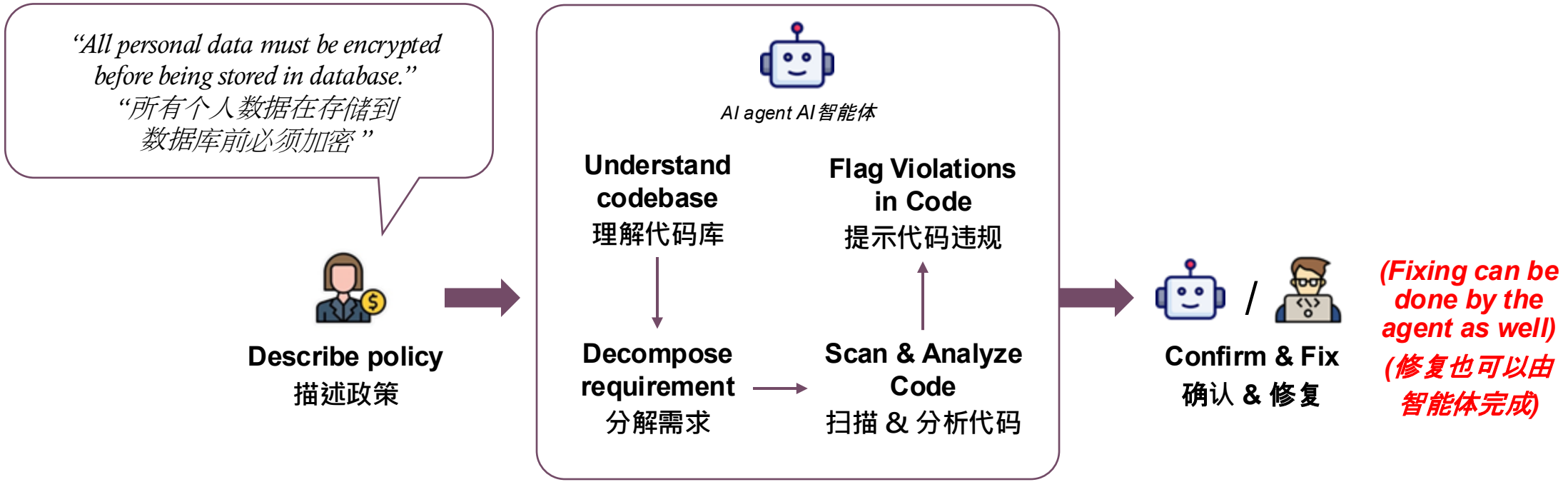


- Clarifying requirements stated at high level
  (*not* at the issue/code level)
- Enforce those Requirements
- Show that the requirements are enforced at code level
- Provide Evidence or explanations of meeting requirements
- Security audit - beyond manual audits - related to explanations

---

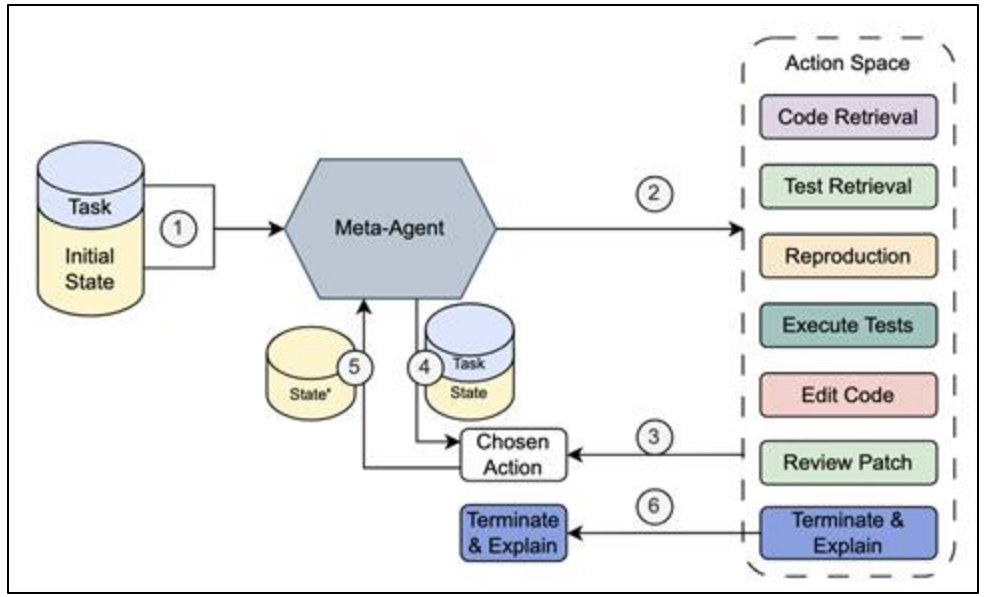
- 澄清高层次的要求
  (*高于* issue / 代码层次)
- 强制执行这些要求
- 表明这些需求在代码层面得到执行
- 提供要求被满足的证据或解释
- 安全审计 – 超越人工审计 – 与解释相关

REGULATORY COMPLIANCE
合规检查



Agent should have capabilities beyond coding!
智能体的能力应该不止于编写代码！

UNIFIED AGENT: BEYOND CODING
统一智能体: 不止于编程



Unified agent — 统一智能体

Handles multiple task types without manual configuration — 处理多种任务类型，无需手动配置

Dynamically deciding its next action like human SWE — 动态决定下一步行动，如同人类软件工程师

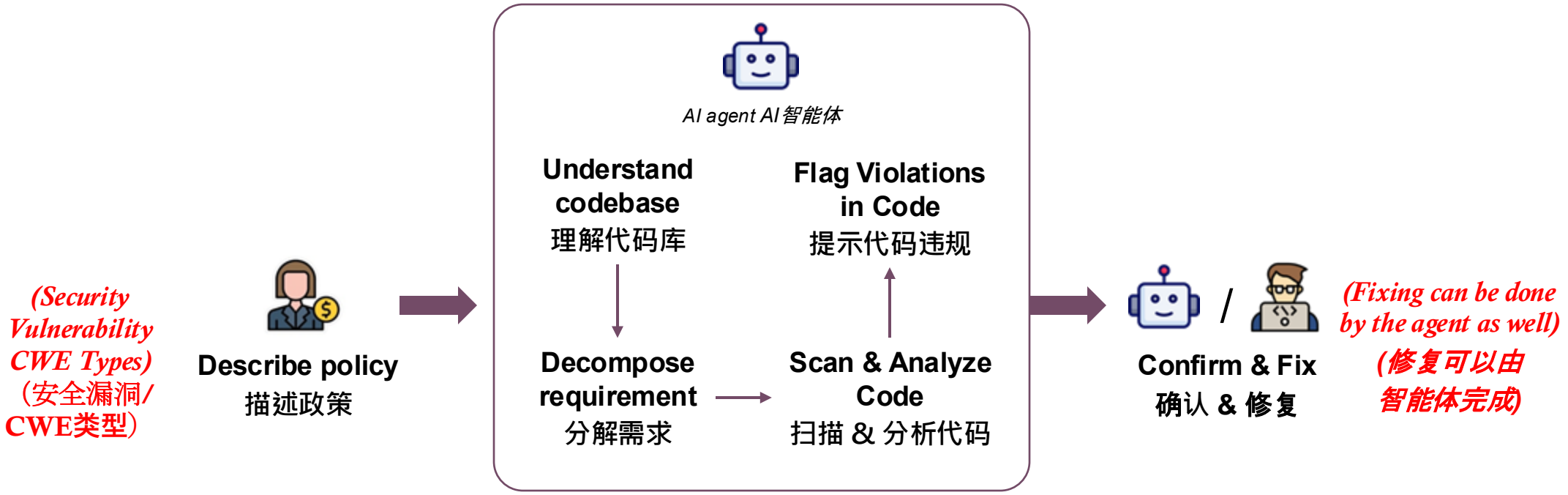
- Issue resolution
- Regression testing
- Code generation
- Test generation
- Partial fix improvement …

- Issue解决
- 回归测试
- 代码生成
- 测试生成
- 不完全修复的改进…

- Architecture exploration
- Requirements clarification

- 架构探索
- 需求澄清

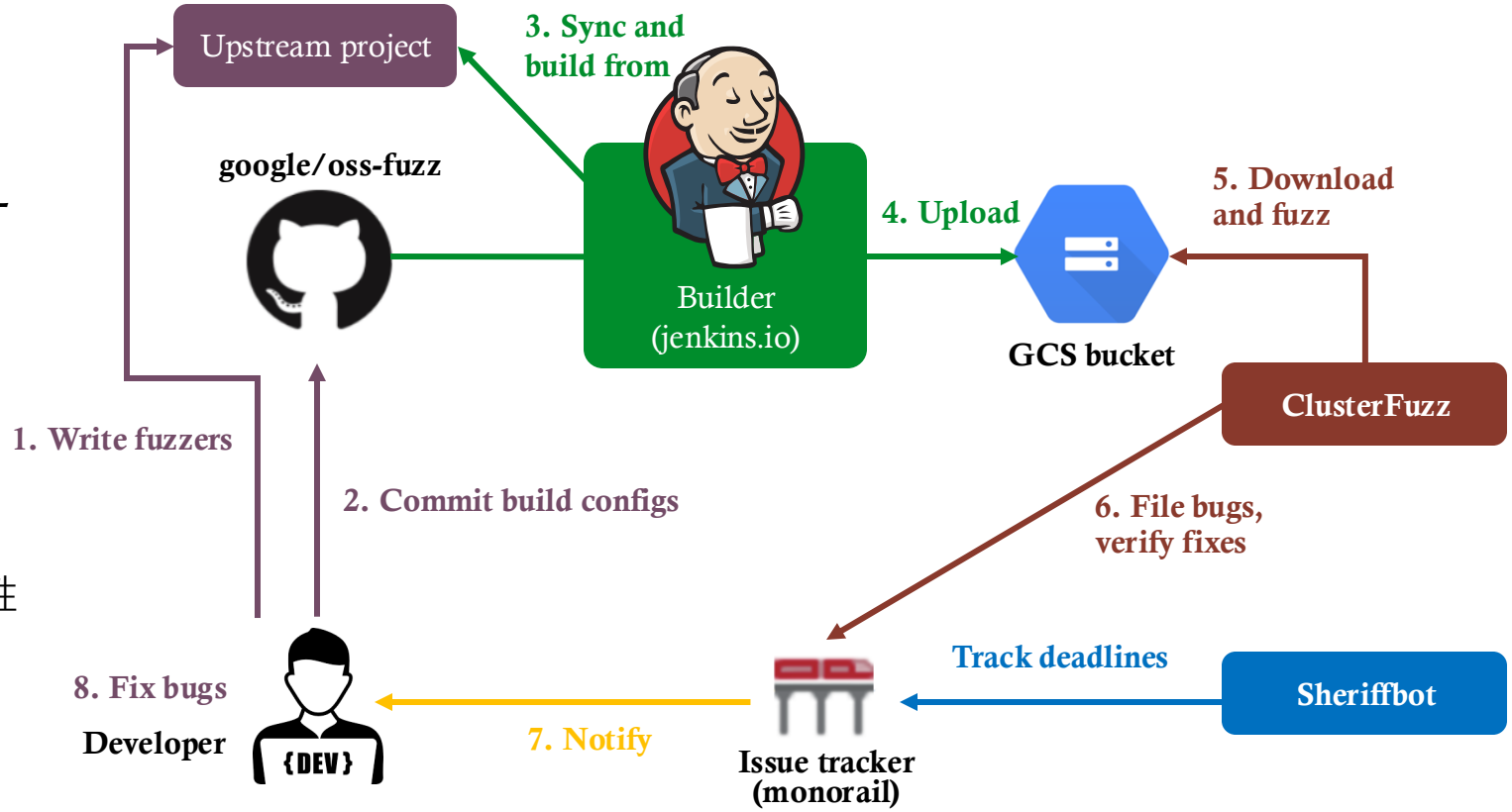
FROM COMPLIANCE TO SECURITY
从合规到安全



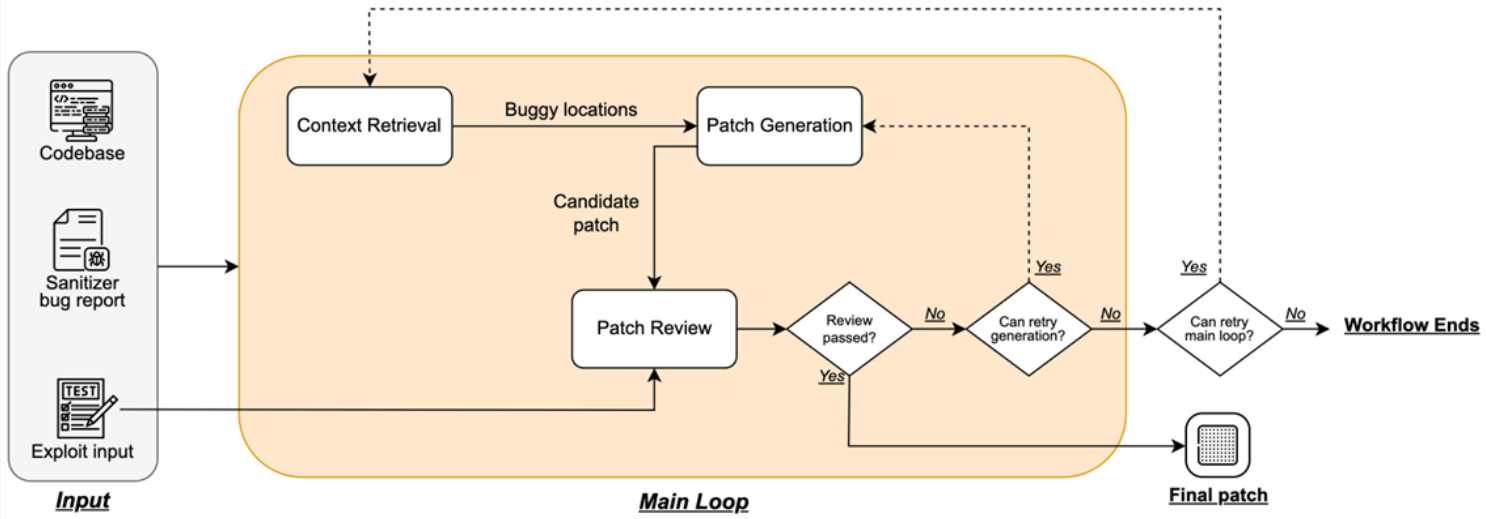
FINDING VULNERABILITIES AS IT IS DONE TODAY
现有漏洞发现方法

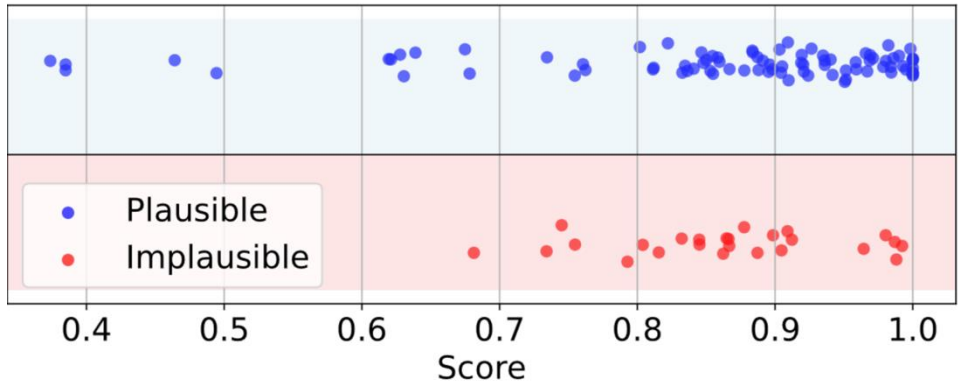
- Continuous Fuzzing Service: Initiated by Google to improve the security and stability of critical open-source software.
- Detected over 12,000 bugs in more than 1000+ open-source projects.

- 持续模糊测试服务: 由Google发起，旨在提升关键开源软件的安全性和稳定性
- 在超过1000个开源软件中发现超过12,000个缺陷



END-TO-END SOFTWARE SECURITY
端到端软件安全





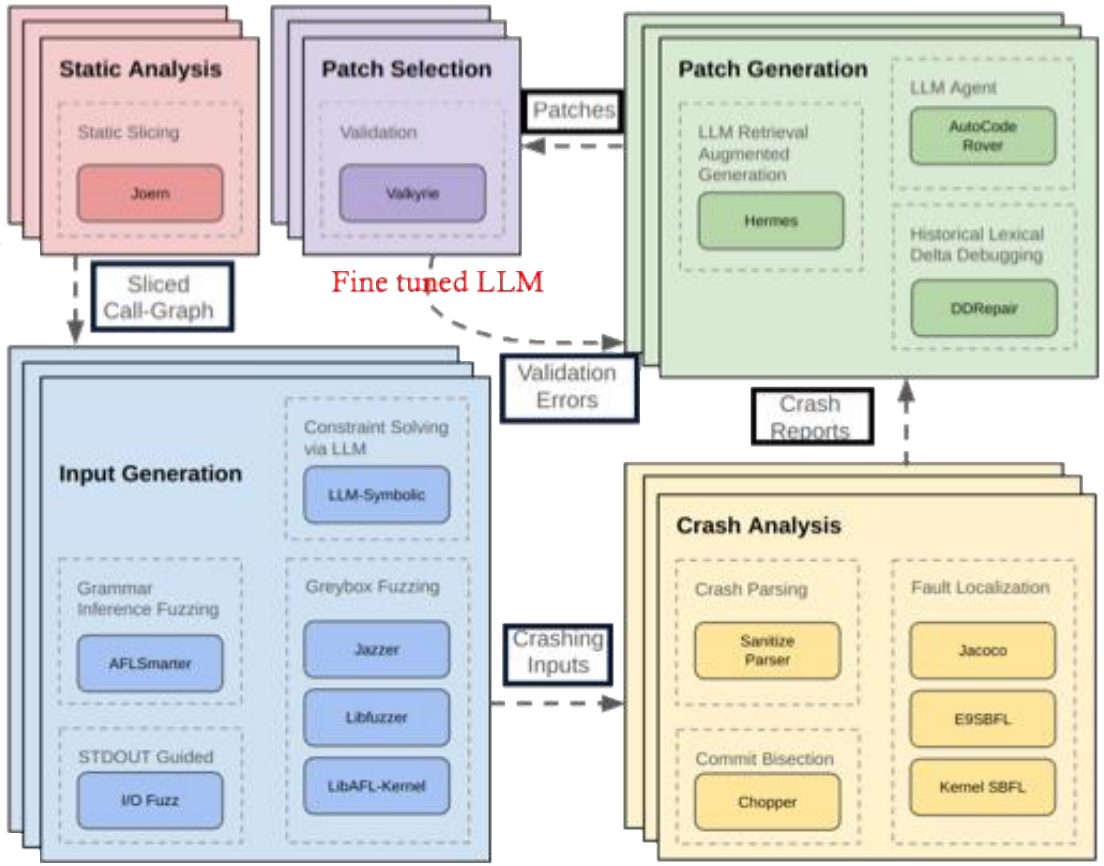
Cannot use AI techniques out of the box

无法直接使用AI技术

DIGITAL INFRA. PROTECTION
数字基础设施保护

Role of text-based analysis to be seen

基于文本的分析将发挥作用



LLM agent with significant analysis

进行深入程序分析的大语言模型智能体

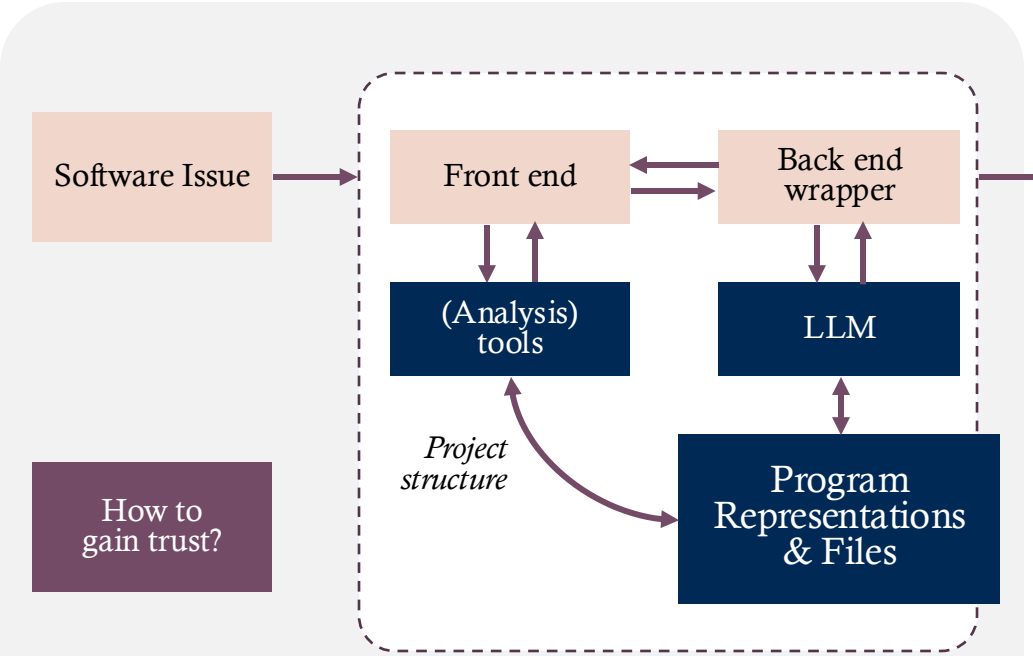
Fuzzing → LLM guided random/fuzzing search

模糊测试 → 大语言模型引导的随机/模糊测试

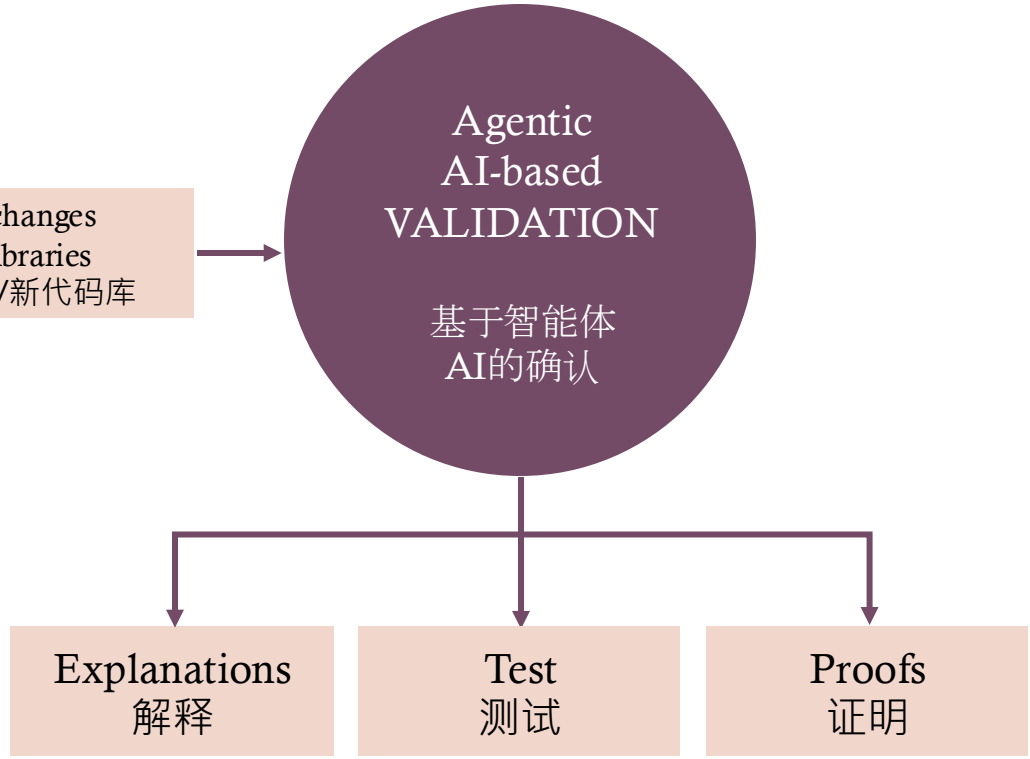
LLM agent

大语言模型智能体

FUTURE CODING
未来编程

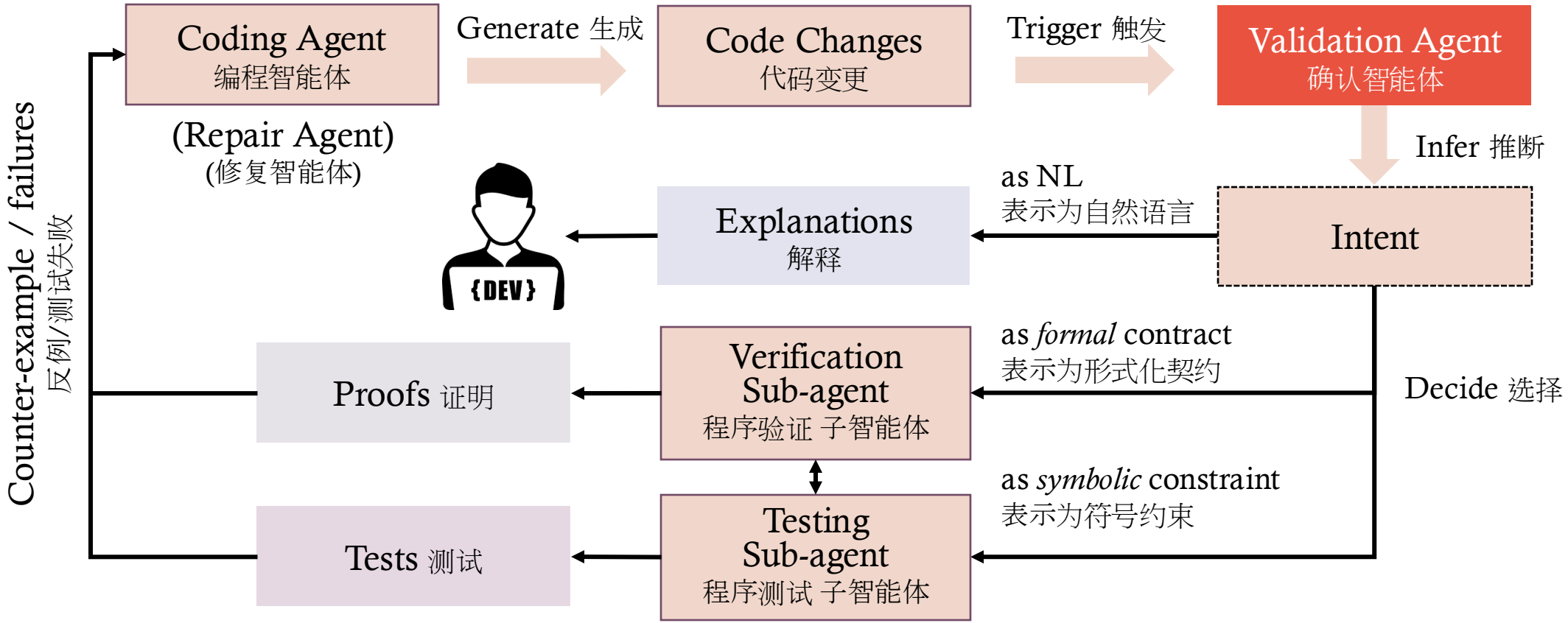


Sample Current Coding agent
示例编程智能体



AI based V&V of AI generated Code
基于 AI 的 AI 生成代码验证与确认

AGENTIC AI-BASED VALIDATION
基于智能体AI的确认



AGENTIC VALIDATION VIA TESTS
基于智能体通过测试进行确认

Agentic Symbolic Execution

Input:

- Source Code (e.g. from coding agent)

Output:

- *Concrete* test cases (e.g. counter-examples)
- *Symbolic* constraints (i.e. partial intent)

基于智能体的符号执行

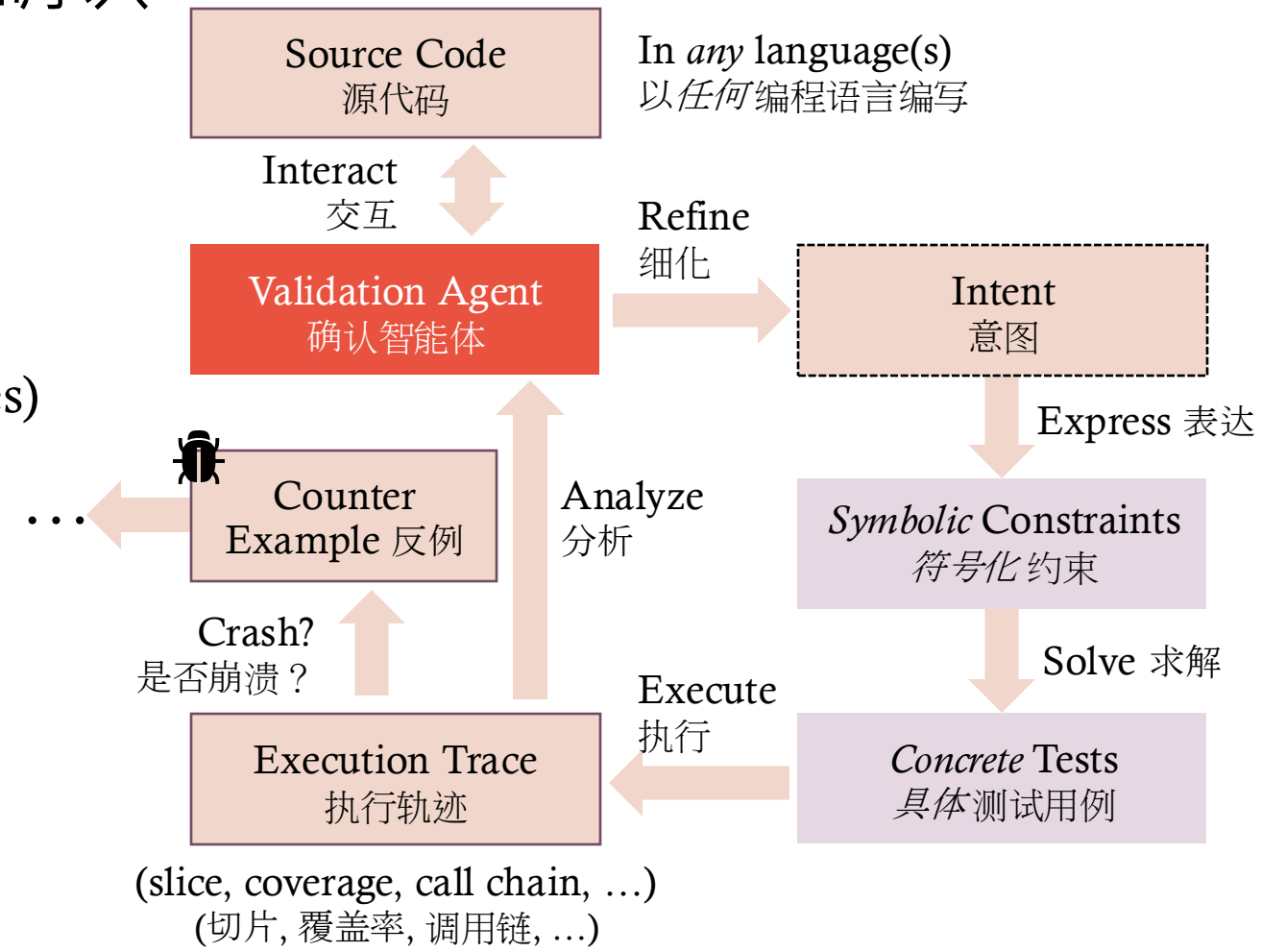
输入:

- 源代码 (可以来自编程智能体)

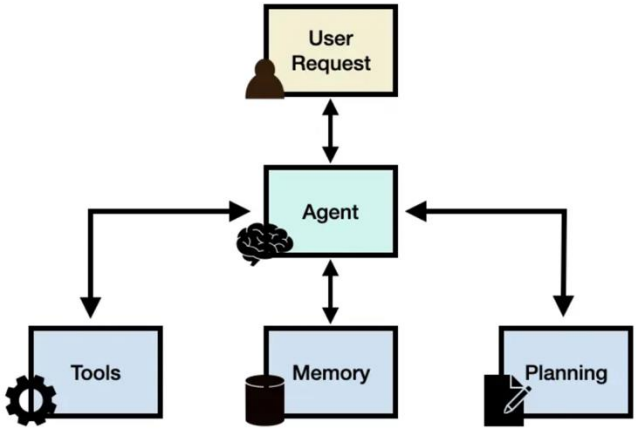
输出:

- *具体*测试用例 (违反安全要求的反例)
- *符号化*约束 (代表部分的意图)

[S & P 2026]

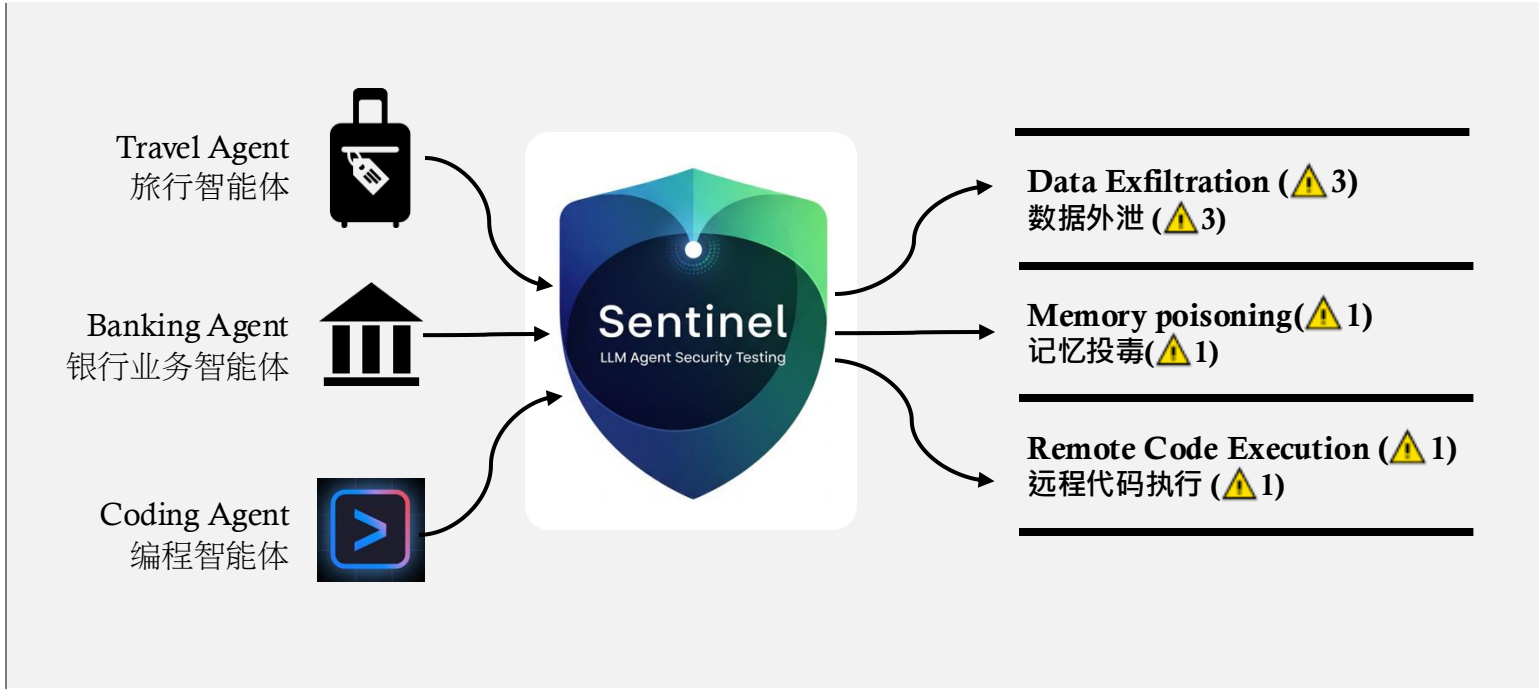


FROM SOFTWARE SECURITY TO AGENT SECURITY
从软件安全到智能体安全



Autonomy and environment interaction

Many security concerns!
许多安全隐患!

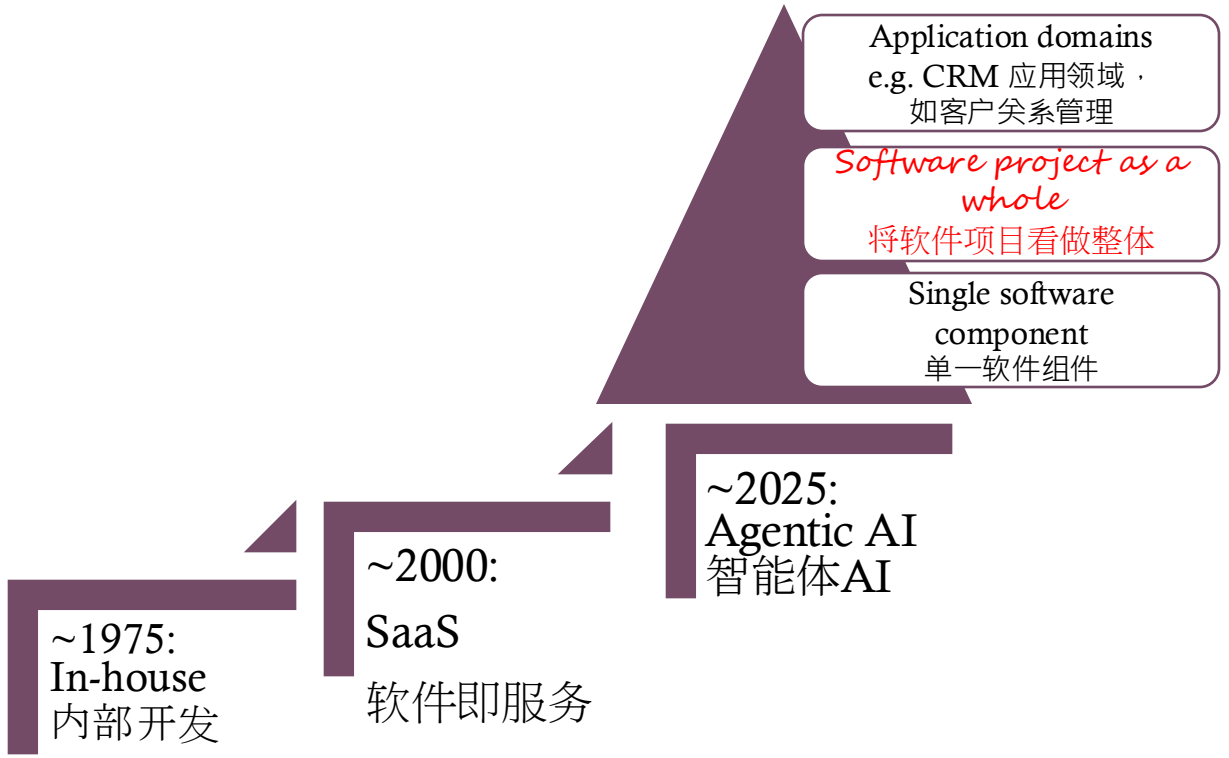


Differences between technology space and commercial space on this matter!
智能体安全在技术上已经有可能，但还未广泛商用

TRANSFORMING INDUSTRIES
革新软件产业

- Automated Program Repair ~ extracting specifications
- ***AGENTIC AI TECH***
  - ***Re-imagining software and workflows***
  - **Re-thinking software design, testing, coding tasks**
  - **Software as a field of study, and as an industry !!**
  - **Agents for trading, healthcare, CRM !**

- 自动程序修复 ~ 提取规约
- ***智能体AI技术***
  - ***重新构想软件和工作流***
  - 重新思考软件的设计、测试和编码
  - 软件作为一个研究领域，以及一个产业 !!
  - 用于交易、医疗保健、客户关系管理的智能体 !



POINTERS TO SHARE
更多相关信息

AI for Code Roundtable 19 Jan 2026, NUS
Scan the QR
AI for Code 圆桌论坛 2026.1.19, NUS
扫码了解

Opinion piece 评论文章
Agentic AI Software Engineers:
Programming with Trust
智能体AI软件工程师: 可信编程
Roychoudury et al. (2025), ***Communications of the ACM***

Abhik Roychoudhury
National University of Singapore
新加坡国立大学
abhik@nus.edu.sg