Developing Technique for Anomaly Detection in Software Behaviors

Yutaro Kashiwa, Nara Institute of Science and Technology, Japan

1. Project Challenge

Internal developers are no longer able to be trusted

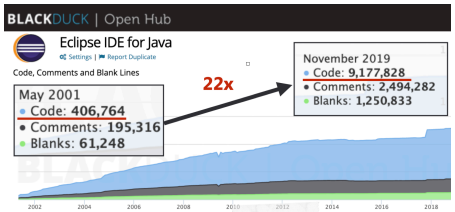
SOPHOS NEWS

Linux team in public bust-up over fake "patches" to introduce bugs

Embarrassed overreaction or righteous indignation? An academic research group has provoked the Linux crew to ban their whole university!

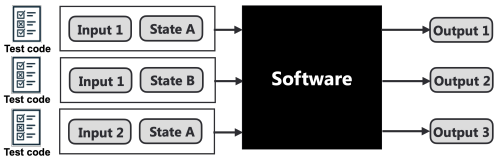
Patches with defects passed Linux's quality assurance process known for its strictness

Why these defects cannot be detected?



Software becomes large-scale and complex but time and human-resources for testing are limited

Tests verify only the inputs and outputs

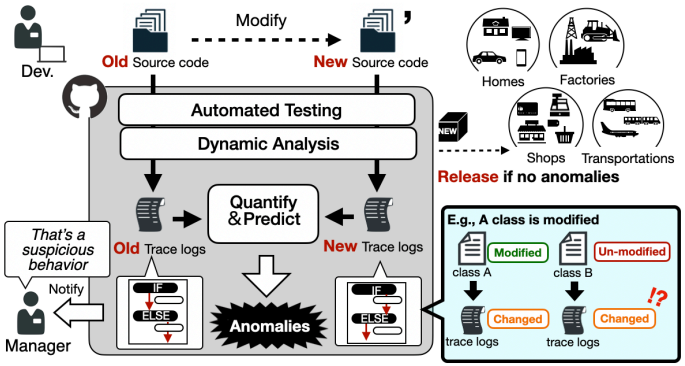


- ✓ **Easy for humans to make tests**
- ✗ **Does not verify which lines of source code were exercised**
  - →Likely to miss defects **if source code is not well-covered by test suites**

It is challenging to create test-suites covering all possible inputs and states because of limited time and complex societies

2. Intellectual Merit

GOAL: Find anomalies without human-created tests



Monitors behaviors before and after changes

Exploit trace logs generated during test exercises with dynamic analysis tools

3. Broader Impact

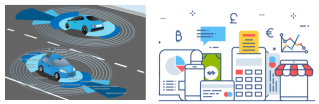
Quality Assurance as a Service (QAaaS)

The proposed approach does not require knowing the specifications of products
→Enables the **outsourcing of quality assurance**

Dramatically reduce software development effort

Realtime Anomaly Detection in Software Behavior

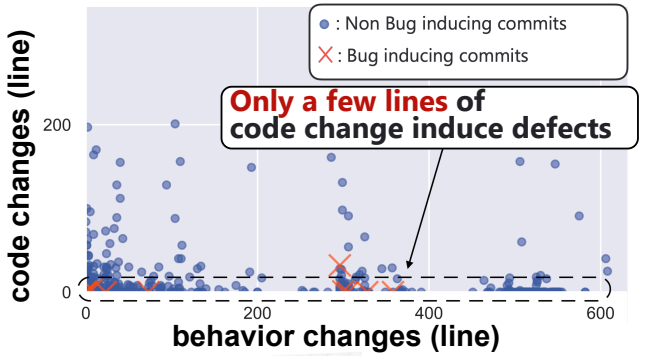
Developing light-weight dynamic software analysis tools enable the monitoring of running software



Detect software anomalies before physical anomalies happen

4. Major Outcomes

Previous studies assume that larger changes are more likely to induce bugs but in reality…



Behavior metrics improves up to 45% of the precision to predict defects in changes

Accepted in the 31st IEEE Intl. Conf. on Software Analysis, Evolution and Reengineering (SANER'24) [CORE A]
Title: "TraceJIT: Evaluating the Impact of Behavioral Code Change on Just-In-Time Defect Prediction"

5. Future Goals

This research plan tackles three challenges