Studying and Automating Issue Resolution for Software Quality



Antu Saha
William & Mary, USA
asaha02@wm.edu

Research Overview

Developers face various challenges when resolving software issues:
- ⛔ Limited understanding of real-world workflows
- ⛔ Low-quality issue reports
- ⛔ Lack of automated support

We aim to enhance the issue resolution lifecycle by:
- 🎯 Understanding practical issue resolution workflows
- 🎯 Proposing techniques to improve the issue report quality
- 🎯 Automating issue resolution tasks, e.g., localizing buggy UIs

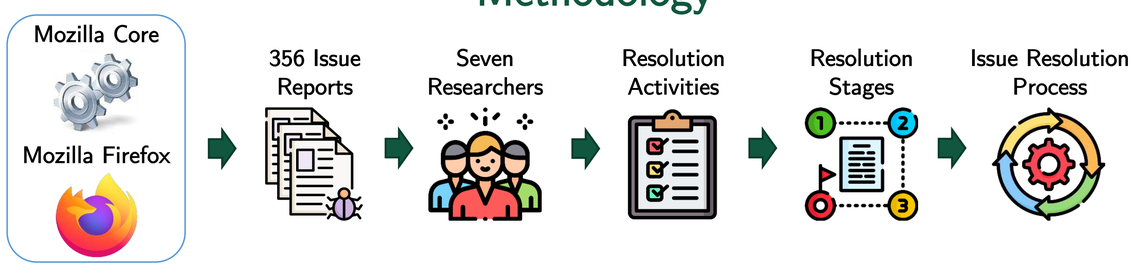
Collectively, our research introduces new techniques, empirical insights, and practical tools to improve the effectiveness of issue resolution, fostering a more reliable and maintainable ecosystem

Understanding the Issue Resolution Process

ICSE'25

- Qualitatively analyze developers' discussions in issue reports to understand the practical issue-resolution process

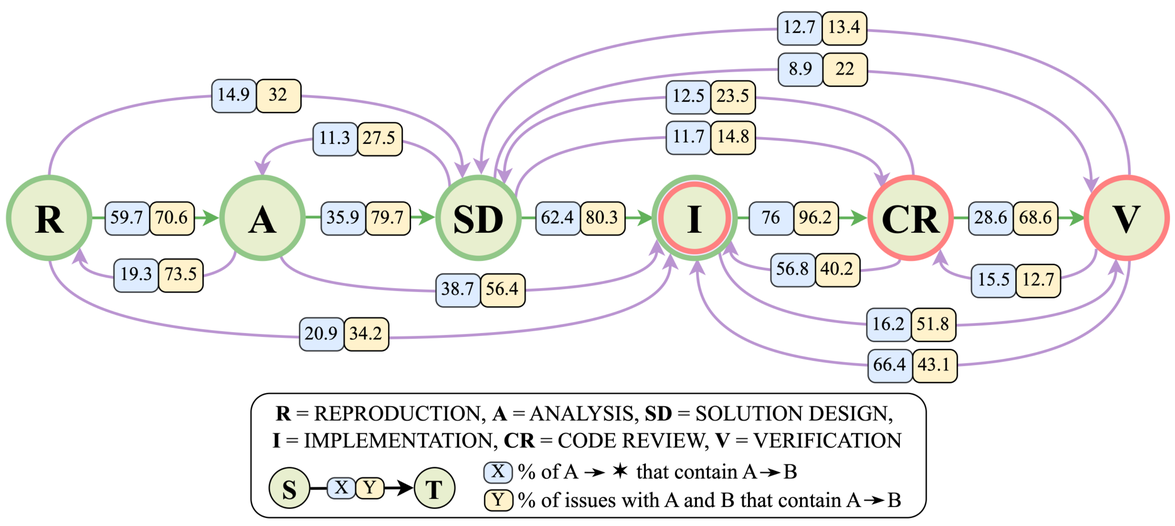
Methodology



Issue Resolution Stages

Reproduction (R)	Analysis (A)	Solution Design (SD)
47 (13.2%)	134 (37.6%)	150 (42.1%)
Implementation (I)	**Code Review (CR)**	**Verification (V)**
328 (92.1%)	264 (74.2%)	146 (41.0%)

Issue Resolution Process



- ✅ Firefox's issue resolution follows an iterative process that deviates from the (theoretical) linear process
- ✅ Developers go back and forth from one stage to another as needed to solve the issues
- ✅ We identified 47 issue resolution patterns, which indicate that solving issues at Firefox is done in a wide variety of ways
- ✅ 18 patterns are recurrently found in 80.6% of Firefox issues

Future Work

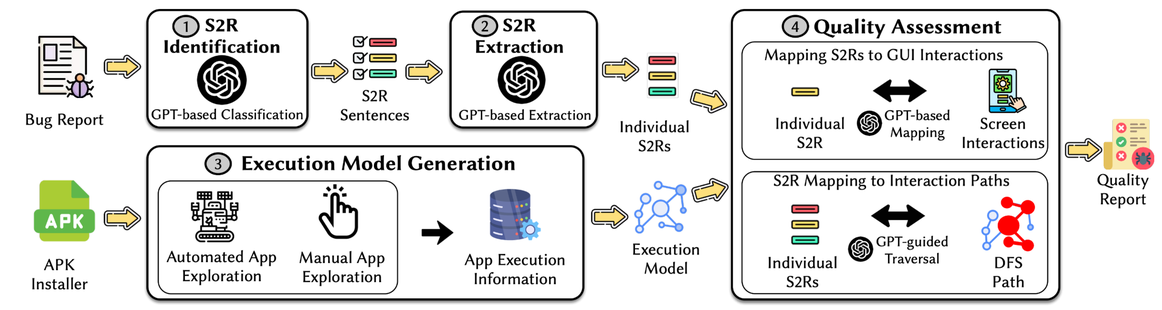
- 🚀 Investigate & characterize practical issue resolution process in AI/ML-integrated software systems via issue report analysis
- 🚀 Develop techniques to improve the entire issue report by leveraging LLMs and app-specific info. (e.g., GUI interactions)
- 🚀 Automatically identify solution-related content from developers' discussions by leveraging MLMs, PLMs, and LLMs

AstroBR: Evaluating & Improving S2Rs

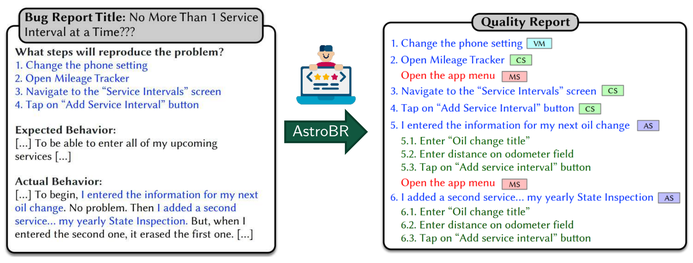
ICPC'25

- Combines LLMs & dynamic application analysis to automatically evaluate and enhance steps-to-reproduce (S2Rs) in issue reports
- Identifies S2R sentences from an issue report, extracts the atomic S2Rs, and maps them with the application interactions from the app execution model constructed via dynamic analysis
- Generates a quality report by detecting correct, ambiguous, and/or missing steps and suggesting missing steps

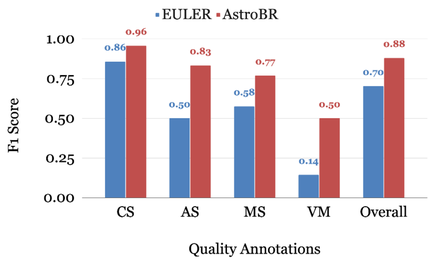
Approach



Quality Report



Evaluation



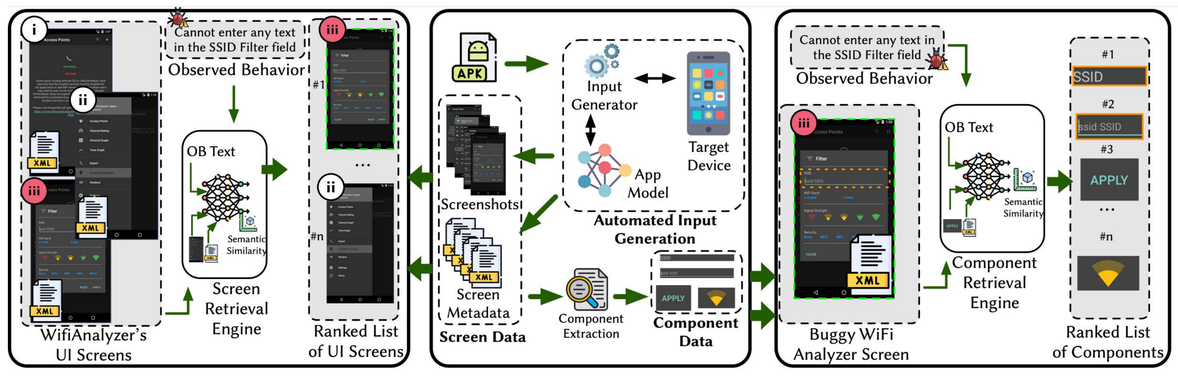
- ✅ Outperforms the baseline in all quality labels and achieves 25.2% overall performance improvement
- ✅ Suggests more accurate missing S2Rs than the baseline by 71.4% in terms of F1 score

Buggy UI Localization

ISSTA'24

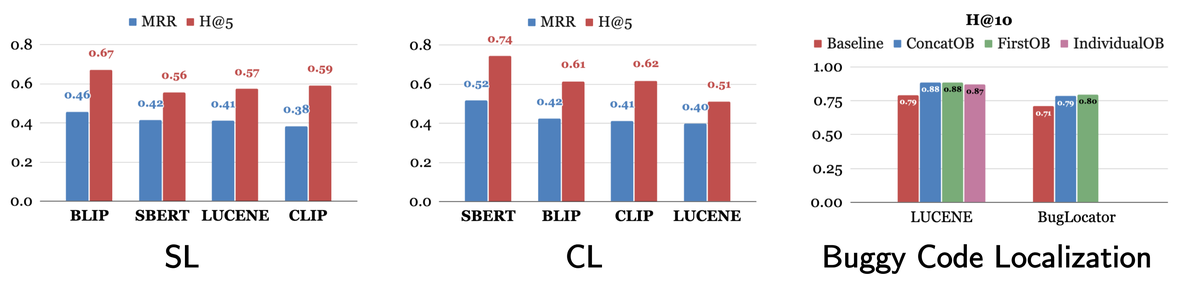
- Automatically identify buggy UI screens (SL) and components (CL) for the given bug description by leveraging traditional/deep learning models, i.e., LUCENE, SBERT, CLIP, and BLIP
- Automate & improve buggy code localization using SL

SL & CL Demonstration



Screen Localization (SL) | GUI Info. Extraction | Component Localization (CL)

Results



- ✅ Existing traditional/DL approaches can be used for Buggy UI Localization tasks (SL & CL)
- ✅ Both visual & textual sources of information are effective for Buggy UI Localization
- ✅ Buggy UI Localization can automate & improve Bug Localization