Fail-Fix Explorer: An Analytics-Based Dashboard for Time-to-Fix and Failure Types Using BugSwarm

ECS 289C – Winter 2025
Project Proposal

Teammates
Harshil Bhandari
Amritanand Sudheerkumar

Problem Description & Motivation

Continuous integration (CI) in contemporary open-source projects generates enormous volumes of build and test data. Developers and researchers usually lack a simplified method to comprehend which defects occur most frequently, how long it takes to cure them, and which fixes are more difficult, even though these statistics offer insightful information about how software bugs are introduced and subsequently fixed. Large-scale empirical analysis is limited by the tiny or manually curated size of traditional bug databases. BugSwarm fills this void by offering a sizable and steadily expanding collection of actual fail-pass build pairs from open-source continuous integration workflows. Each pair is packaged as an artifact that includes metadata explaining the failing and subsequent passing commits, logs, and Docker images. The goal of our project is to use BugSwarm artifacts to create an analytics dashboard called Fail-Fix Explorer, which integrates failure type and exception analysis with time-to-fix metrics. Developers, maintainers, and researchers can use this dashboard to rapidly determine which failures are most often and whether handling particular bug or exception kinds takes a disproportionately long time.

Proposed Technical Approach

1. **Data Ingestion:**
   With an emphasis on fields like commit timestamps, errors thrown, language, and classification (code, test, build), we will obtain artifact metadata from BugSwarm's MongoDB or REST API. The layered data will be flattened into a tabular format that can be used with dashboard applications such as Tableau or Power BI.

2. **Key Metrics & Transformations:**
   Time-to-Fix: Compute fix duration (in hours/days) by subtracting the failing commit's timestamp from the subsequent passing commit's timestamp.
   Failure & Exception Analysis: Group artifacts by exceptions (e.g., `NullPointerException`, `ImportError`) and classification flags (`code`, `test`, `build`).
   Correlation: Calculate average fix durations for each exception or classification category, potentially slicing by language or number of failing tests.

3. **Dashboard Construction:**
   We will create visualizations (bar charts, box plots, line charts) showing:
   - Time-to-Fix distributions (e.g., histograms or time-series of average fix duration).
   - Exception Frequency and correlation with fix duration.
   - Filters for language, classification, or repository.
   - An Outlier Explorer for identifying anomalies such as extremely long fix durations.

Proposed Evaluation Methodology

1. **Data Coverage & Consistency:**
   Ensure our final dataset includes a sufficiently large, representative sample of BugSwarm artifacts. Track how many artifacts have all necessary fields (timestamps, exceptions, etc.) to confirm the approach covers a broad spectrum of fail-pass cases.

2. **Accuracy & Validation:**
   Validate computed time-to-fix intervals against known commits or official logs to confirm correctness. Cross-check a subset of artifacts manually to ensure classification (code, test, build) and exceptions align with the original failing logs.

3. **Usability & Insights:**
   Assess whether the dashboard effectively communicates which failures are most frequent and which require longer fixes. Solicit informal feedback from potential users (classmates) to see if the visualizations and filters are intuitive.

Related Papers

- Tomassi et al. "BugSwarm: Mining and Continuously Growing a Dataset of Reproducible Failures and Fixes", ICSE 2019.
- Just et al. "Defects4J: A Database of Existing Faults to Enable Controlled Testing Studies for Java Programs", ISSTA 2014.
- Le Goues et al. "The ManyBugs and IntroClass Benchmarks for Automated Repair of C Programs", TSE 2015.