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

MILESTONE 1

Teammates:
Harshil Bhandari, Amritanand Sudheerkumar

GitHub Repo: https://www.github.com/Harshil7875/ECS289C-WQ2025-Team-Project

---

1. Introduction

Continuous integration (CI) has become ubiquitous in modern open-source development, generating vast amounts of build and test data. Despite this, developers and researchers often lack a unified tool to quickly understand key aspects of software defects—such as the types of failures that occur, how long they take to fix, and which kinds of failures are particularly challenging. Traditional bug databases are typically small or manually curated, limiting large-scale empirical studies.

BugSwarm addresses this limitation by providing a large, continuously growing repository of reproducible fail-pass build pairs from real-world CI workflows. Each BugSwarm artifact includes metadata on the failing and passing commits, build logs, and Docker images. In our project, Fail-Fix Explorer, we leverage these artifacts to build an analytics dashboard that visualizes metrics such as time-to-fix and failure types. The insights derived from this analysis can help developers identify recurring issues and researchers study the nature of bug fixes in large-scale software systems.

Note: Our initial analysis has been performed on a subset (138 artifacts) of the full 3,000-artifact dataset. This early exploration provides valuable insights that will guide the processing and analysis of the complete dataset.

---

2. Motivating Example

Consider the artifact identified by the image tag ocpsoft-rewrite-64363050:

- **Language:** Java
- **Failed Commit:** 2015-05-28T04:55:42Z
- **Passed Commit:** 2015-05-28T06:12:18Z
- **Time-to-Fix:** ~1.28 hours
- **Exception:** *AssertionError*

This single example illustrates the type of information available in each artifact. By aggregating similar data across thousands of artifacts, our dashboard will enable users to quickly determine which exceptions occur most frequently and whether certain failure types are associated with longer fix durations.

---

3. High-Level Technical Approach

Our technical approach is divided into three main components:

3.1 Data Ingestion and Preprocessing

We developed a shell script (`fetch_artifact_metadata.sh`) that automates the retrieval of artifact metadata from BugSwarm. The script performs the following steps:

- **Input Extraction:** It reads artifact image tags from an Export.json file (downloaded from BugSwarm's dataset website) using jq.
- **Metadata Retrieval:** For each image tag, the script calls the BugSwarm CLI to fetch detailed metadata.
- **Output Management:** The metadata for each artifact is stored as a separate JSON file in a dedicated "metadata" directory.
- **Error Handling:** The script includes a retry mechanism. If a rate-limiting error (e.g., "You are being rate limited") is detected, it waits for a specified delay (60 seconds) and retries up to five times before skipping the artifact.

This automated process ensures that we systematically collect metadata for thousands of artifacts while handling network and API limitations gracefully.

Data Tranformation:

Once the metadata is collected, we flatten the nested JSON data into a tabular format. Critical fields include commit timestamps, programming language, exception details, and classification flags (e.g., code, test, build). We compute the time-to-fix by subtracting the failing commit timestamp from the subsequent passing commit timestamp. Our preliminary EDA on a subset of artifacts has already revealed anomalies—such as negative time-to-fix values—that we plan to address as we process the full dataset.

3.2 Metrics Computation and Transformation

- **Time-to-Fix Analysis:**
  We compute statistical measures (mean, standard deviation) for time-to-fix and identify outliers (e.g., fix durations that exceed mean + 3×std). Preliminary results indicate that while most fixes occur quickly, a few extreme outliers (up to ~800 hours) skew the average.
- **Failure and Exception Grouping:**
  Artifacts are grouped by the type of exception encountered (e.g., *AssertionError*, *TypeError*, etc.) and by classification flags (e.g., whether the failure originated from code, tests, or the build process). Correlations between failure types and fix duration are then calculated, and differences across programming languages (notably Java vs. Python) are explored.

3.3 Dashboard Construction

- **Visualization:**
  The dashboard will feature a variety of visualizations:
  - Histograms and box plots to display the distribution of time-to-fix values.
  - Bar charts for exception frequency.
  - Comparative views that allow filtering by language, classification type, or repository.
  - An Outlier Explorer to identify and investigate artifacts with unusually long fix durations.

- **Interactive Interface:**
  We will use visualization tools (such as Tableau or Power BI) to create an interactive, user-friendly dashboard that allows stakeholders to drill down into the data.

---

4. Related Work

Our work builds upon several key research efforts:

- **BugSwarm:**
  Tomassi et al. introduced BugSwarm in "BugSwarm: Mining and Continuously Growing a Dataset of Reproducible Failures and Fixes" (ICSE 2019). BugSwarm provides the foundational dataset that our project leverages.
- **Defects4J:**
  Just et al. describe Defects4J in "Defects4J: A Database of Existing Faults to Enable Controlled Testing Studies for Java Programs" (ISSTA 2014). This work has influenced empirical bug analysis studies and offers insights into fault localization.
- **Automated Program Repair Benchmarks:**
  Le Goues et al. discuss benchmarks such as ManyBugs and IntroClass in "The ManyBugs and IntroClass Benchmarks for Automated Repair of C Programs" (TSE 2015). These benchmarks highlight the importance of diverse datasets for understanding bug fixes, a goal shared by our project.

In addition, our dashboard concept aligns with broader research in software analytics and empirical software engineering that examines CI data to inform defect resolution strategies.

---

5. Evaluation Methodology

Our evaluation strategy comprises the following components:

5.1 Data Coverage & Consistency

- **Completeness Check:**
  We will track the number of BugSwarm artifacts that include all necessary fields (commit timestamps, exception data, etc.) to ensure that our dataset is representative.
- **Data Quality Assessment:**
  Anomalies detected in our initial subset—such as negative time-to-fix values—will be investigated and corrected in subsequent preprocessing steps.

5.2 Accuracy & Validation

- **Validation of Transformations:**
  The computed time-to-fix values will be cross-validated against known commit logs to ensure that our calculations are correct.
- **Manual Inspection:**
  A subset of artifacts will be manually reviewed to confirm that the classification and exception extraction align with the original logs.

5.3 Usability & Insights

- **Dashboard Feedback:**
  We plan to solicit informal feedback from developers and classmates on the usability and intuitiveness of our dashboard visualizations and filters.
- **Insightfulness:**
  The dashboard will be evaluated based on its ability to clearly communicate which failure types are most common and which correlate with longer fix durations.

---

6. Tasks for Milestone 2

In Milestone 2, we plan to extend and refine our work based on the insights and challenges identified during Milestone 1. Key tasks include:

- **Complete Data Processing:**
  Process the full dataset of approximately 3,000 BugSwarm artifacts and resolve any data quality issues encountered in the initial subset.
- **Enhanced Feature Engineering:**
  Develop additional features (e.g., more granular exception categorization) and improve the transformation pipeline.
- **Refined Dashboard Development:**
  Build and iterate on an interactive dashboard prototype using tools like Tableau or Power BI, incorporating user feedback.
- **Preliminary Experimental Evaluation:**
  Begin a more rigorous evaluation of the technical approach, including correlation analyses and potential predictive modeling for artifact classification.

---

7. Conclusion

This milestone report presents our initial exploratory analysis and outlines our high-level technical approach for the Fail-Fix Explorer project. The preliminary EDA on 138 artifacts has revealed promising trends and several data quality issues that will be addressed as we scale up to the full dataset. Our next steps involve comprehensive data processing, enhanced dashboard construction, and an in-depth evaluation of our methods.