On the Reproducibility of Software Defect Datasets

Hao-Nan Zhu and Cindy Rubio-González
University of California, Davis

45th International Conference on Software Engineering
May 14-20, 2023

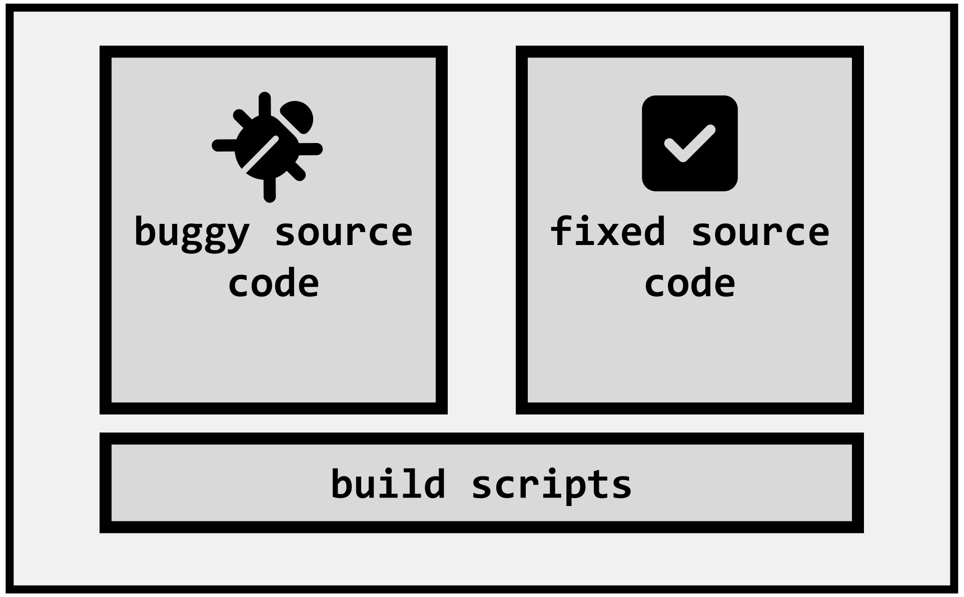
National Science Foundation Award
CNS-2016735

Reproducibility & Software Defect Datasets

Reproducibility

The same result of experiment can be repeated as long as following the same procedure.

Software Defect Artifact



Without Reproducibility…

- Wrong findings of downstream studies that facilitated from software defect datasets
- Unreproducible studies evaluated with software defect datasets
- Inability to perform future studies
- **Problematic for Software Engineering community**

Research Questions and Subjects

Questions

- What does reproducibility mean to specific software defect datasets?
- What is the reproducibility of each software defect datasets?

Study Subjects

Defects4J[1], GrowingBugs[2], Bugs.jar[3], BugSwarm[4], and Bears[5]

[1] Just et al., Defects4J: a Database of Existing Faults to Enable Controlled Testing Studies for Java Programs. *ISSTA 2014.*
[2] Jiang et al., Extracting Concise Bug-Fixing Patches from Human-Written Patches in Version Control Systems. *ICSE 2021.*
[3] Saha et al., Bugs.jar: a Large-scale, Diverse Dataset of Real-World Java Bugs. *MSR 2018.*
[4] Tomassi et al., BugSwarm: Mining and Continuously Growing a Dataset of Reproducible Failures and Fixes. *ICSE 2019.*
[5] Madeiral et al., BEARS: An Extensible Java Bug Benchmark for Automatic Program Repair Studies. *SANER 2019.*

Selection Criteria

- Software defect datasets for Java
- Artifact must consist of whole projects
- Datasets for functional bugs
- Publicly available at the time of study

Reproducibility is Defined Differently to Each Dataset

Source of Information for Reproducibility Criteria

- Dataset documentation
- Publications
- Dataset infrastructure source code

Dataset	**Definition of Reproducibility**			
	Existence	**Number**	**Name**	**Status**
Defects4J	✓	✓	✓	
GrowingBugs	✓	✓	✓	
Bugs.jar	✓			
BugSwarm	✓	✓	✓	✓
Bears	✓			

Reproducibility Criteria

Existence	as long as bug ***exists*** in buggy version
Number Match	bug exists in buggy version, with ***number*** of the failing tests matching the original reference
Name Match	bug exists in buggy version, with ***number*** and ***name*** of the failing tests matching the original reference
Status Match	only if ***number****,* ***name*** of failing tests and CI/CD ***build status*** match the original reference

All Datasets Experience Breakages

Experimental Setup

- Check out source code
- Build project & run tests
- Collect & analyze logs
- Compare with original reference
- Repeat for both buggy & fixed versions

Dataset	#Artifacts	Reproducibility for Different Criteria			
		Existence	Number	Name	Status
Defects4J	864	837 (96.9%)	837 (96.9%)	837 (96.9%)	N/A
GrowingBugs	570	175(30.7%)	170 (29.8%)	170 (29.8%)	N/A
Bugs.jar	1,158	308 (26.6%)	303 (26.2%)	303 (26.2%)	N/A
BugSwarm	1,795	1,392 (77.5%)	1,388 (77.3%)	1,387 (77.3%)	1,239 (69%)
Bears	251	137 (54.6%)	134 (53.4%)	134 (53.4%)	N/A

Results

- All datasets experience breakages, especially those created automatically
- Defects4J reaches the highest reproducibility of 96.9%
- Automatically created datasets have reproducibility ranging from 26.6% to 69%

Reproducibility of BugSwarm - a Case Study

BugSwarm

BugSwarm[1] is an automatically constructed software defect dataset with 1,795 non-flaky Java artifacts.

Research Questions

- How often do software breakages occur?
- What are root causes & fixes for software breakages?
- How can we prevent software breakages?

Experimental Setup

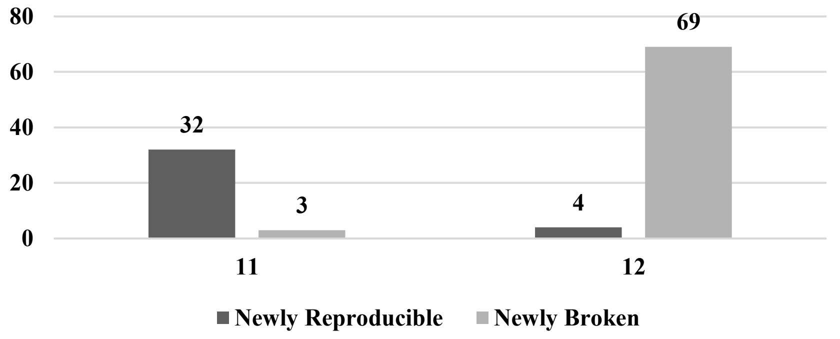
- Reproducibility tests on average every 11.7 days over a 13-month period
- Results collected from 36 test suites
- The strictest status match criterion used to determine reproducibility

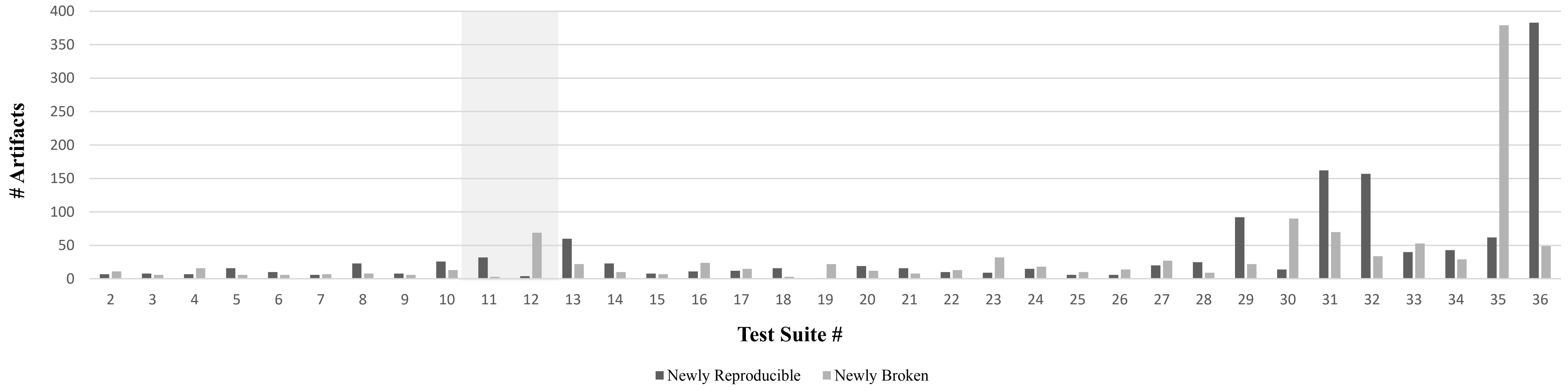
[1] Tomassi et al., BugSwarm: Mining and Continuously Growing a Dataset of Reproducible Failures and Fixes. *ICSE 2019.*

Software Defect Artifacts Break Frequently

Breakage Frequency

- 1,124 out of 1,795 artifacts broke at least once
- 275 artifacts broken multiple times
- On average, we have 38 newly reproducible and 32 newly broken artifacts in each test suites





Root Causes & Fixes for Software Breakages

Breakage Root Causes & Fixes

- 1,606 individual instances of breakages impacted 1,124 artifacts
- 11 root causes identified
- 10 patches implemented
- 2,948 fixed performed on 1,055 artifacts

Insights

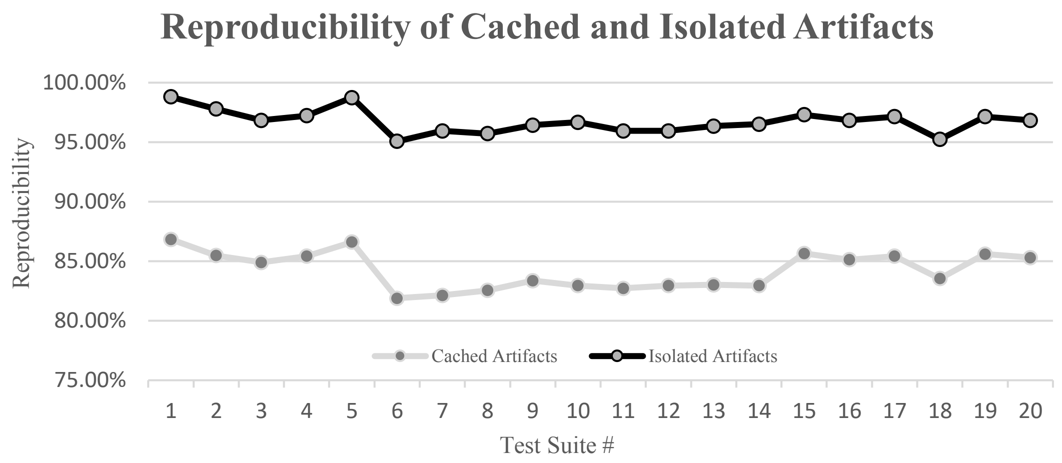
- Most of patches are related to system or project dependencies
- 44% of fixes are for project dependencies

Root Cause	Patch
Maven TLS Failure	Update TLSv1.0 to TLSv1.2
Unavailable PPAs	Remove PPAs no longer available
Unavailable Ubuntu Release	Change URLs for repository
Insecure Link	Change URLs using HTTP to HTTPS
Unavailable JDK Version	Retrieve JDK version from official repository
Unavailable Gradle Plugin	Update URL of specific Gradle Plugin
Unavailable NodeJS Installer	Change URL to retrieve NodeJS installer
Incompatible NPM Package	Pin NPM package version
Unavailable XML	Update URL to retrieve DTD files
Deprecated checkstyle Link	Replace deprecated checkstyle URL
Unexpected Test Failures	N/A

Breakage Prevention

Dependency Caching

- Leverage the offline mode of build system to download all required dependencies
- 1700 artifacts are successfully cached



Artifact Isolation

- Cached artifacts are expected to be reproducible without internet connection
- 920 of 1700 cached artifacts are initially isolated
- After further fix on 337 artifacts, 1257 artifacts are successfully isolated

Evaluation & Results

- 20 extra test suites in an 8-month period
- Reproducibility of cached artifacts: *81% - 85%*
- Reproducibility of isolated artifacts: *>95%*

Conclusions

Lessons Learned

- All software defect datasets suffer from software breakages, especially for automatically constructed ones
- Most of software breakages are involved with issues related to software dependencies
- Dependency caching and artifact isolation effectively prevent software breakages and ensure long-term reproducibility

Replication Package

Link: *https://github.com/ucd-plse/on-the-reproducibility*

DOI: *10.5281/zenodo.7577662*

Real-Time Reproducibility

http://www.bugswarm.org/statistics/