Mark Harman
FACEBOOK
SOFTWARE ENGINEERING MANAGER

SAPIENZ TEAM; THIS TALK IS BASED ON WORK OF THE WHOLE TEAM

Mark Harman

FACEBOOK
SOFTWARE ENGINEERING MANAGER

SAPIENZ TEAM; THIS TALK IS BASED ON WORK OF THE WHOLE TEAM

WORK IN PROGRESS

UNFORTUNATELY, YOUR APP HAS STOPPED.

REPORT

OK

SLOW

PAINFUL

BORING

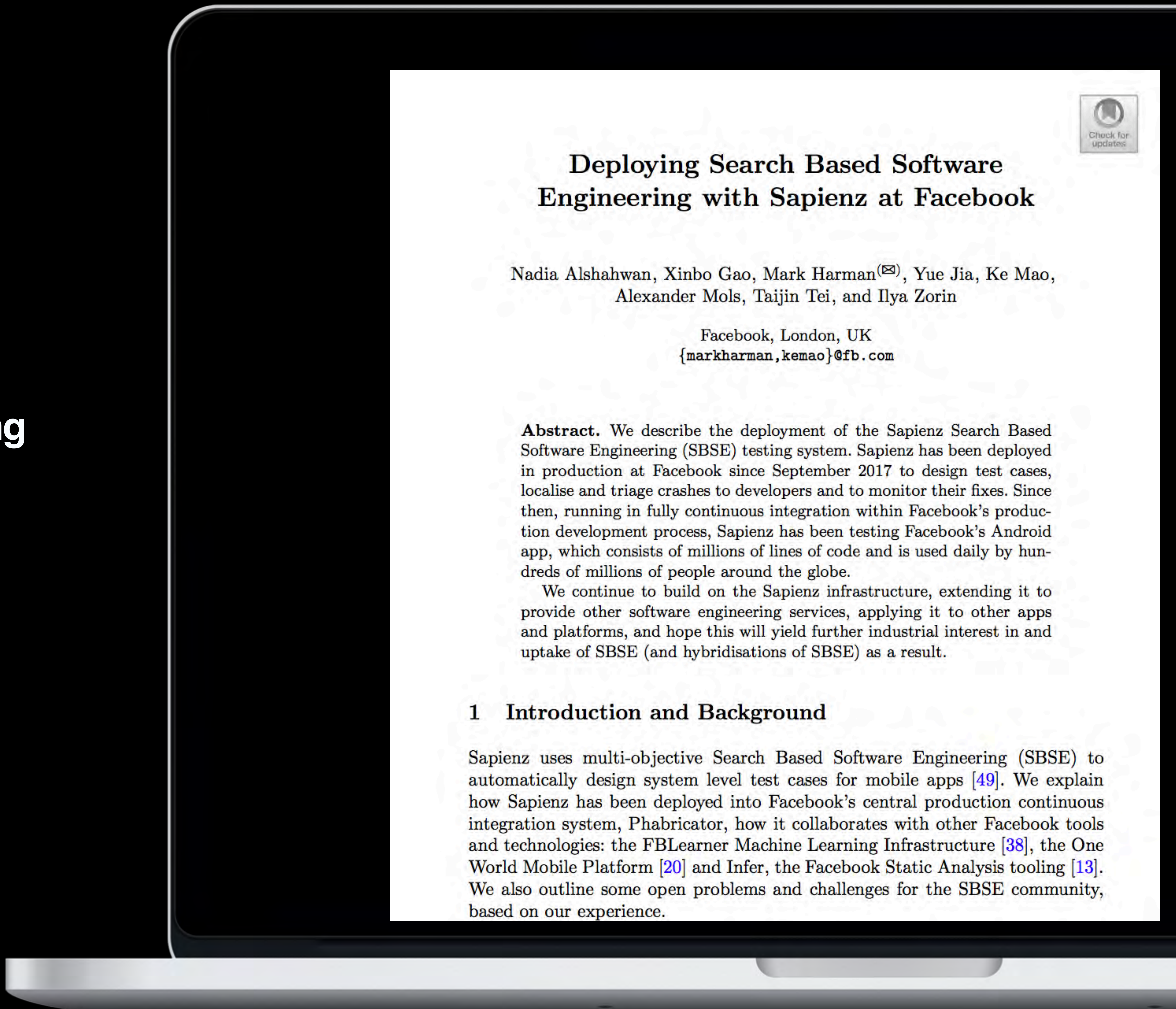
UNIMPORTANT

URGENT

Deploying Search Based Software Engineering with Sapienz at Facebook

NADIA ALSHAHWAN,
XINBO GAO,
MARK HARMAN,
YUE JIA,
KE MAO,
ALEXANDER MOLS,
TAIJIN TEI
AND
ILYA ZORIN

IN PROC. OF SSBSE, 2018.



Deploying Search Based Software Engineering with Sapienz at Facebook

Nadia Alshahwan, Xinbo Gao, Mark Harman(✉), Yue Jia, Ke Mao,
Alexander Mols, Taijin Tei, and Ilya Zorin

Facebook, London, UK
{markharman,kemao}@fb.com

Abstract. We describe the deployment of the Sapienz Search Based Software Engineering (SBSE) testing system. Sapienz has been deployed in production at Facebook since September 2017 to design test cases, localise and triage crashes to developers and to monitor their fixes. Since then, running in fully continuous integration within Facebook's production development process, Sapienz has been testing Facebook's Android app, which consists of millions of lines of code and is used daily by hundreds of millions of people around the globe.

We continue to build on the Sapienz infrastructure, extending it to provide other software engineering services, applying it to other apps and platforms, and hope this will yield further industrial interest in and uptake of SBSE (and hybridisations of SBSE) as a result.

1 Introduction and Background

Sapienz uses multi-objective Search Based Software Engineering (SBSE) to automatically design system level test cases for mobile apps [49]. We explain how Sapienz has been deployed into Facebook's central production continuous integration system, Phabricator, how it collaborates with other Facebook tools and technologies: the FBLearner Machine Learning Infrastructure [38], the One World Mobile Platform [20] and Infer, the Facebook Static Analysis tooling [13]. We also outline some open problems and challenges for the SBSE community, based on our experience.

6. logging of various internal soft error warnings,
7. numbers of replication of production failures,
8. the number and proportion of reproducibility (non flakiness) of test cases.

The DevOps reporting also includes a suite of data concerning the performance of the triage, and the response of developers to the signal provided to them.

As an illustration, consider Figure 9, which depicts a graph plot for six days in April 2018, showing activity coverage. The vertical axis is not shown and is not necessary for this illustration. As can be seen from the figure, activity coverage retains an apparently consistent level, which we regard as 'healthy', but on the 29th April a sudden drop is noticed. This points to potential problems in the deployment, occasioning further investigation, as necessary.

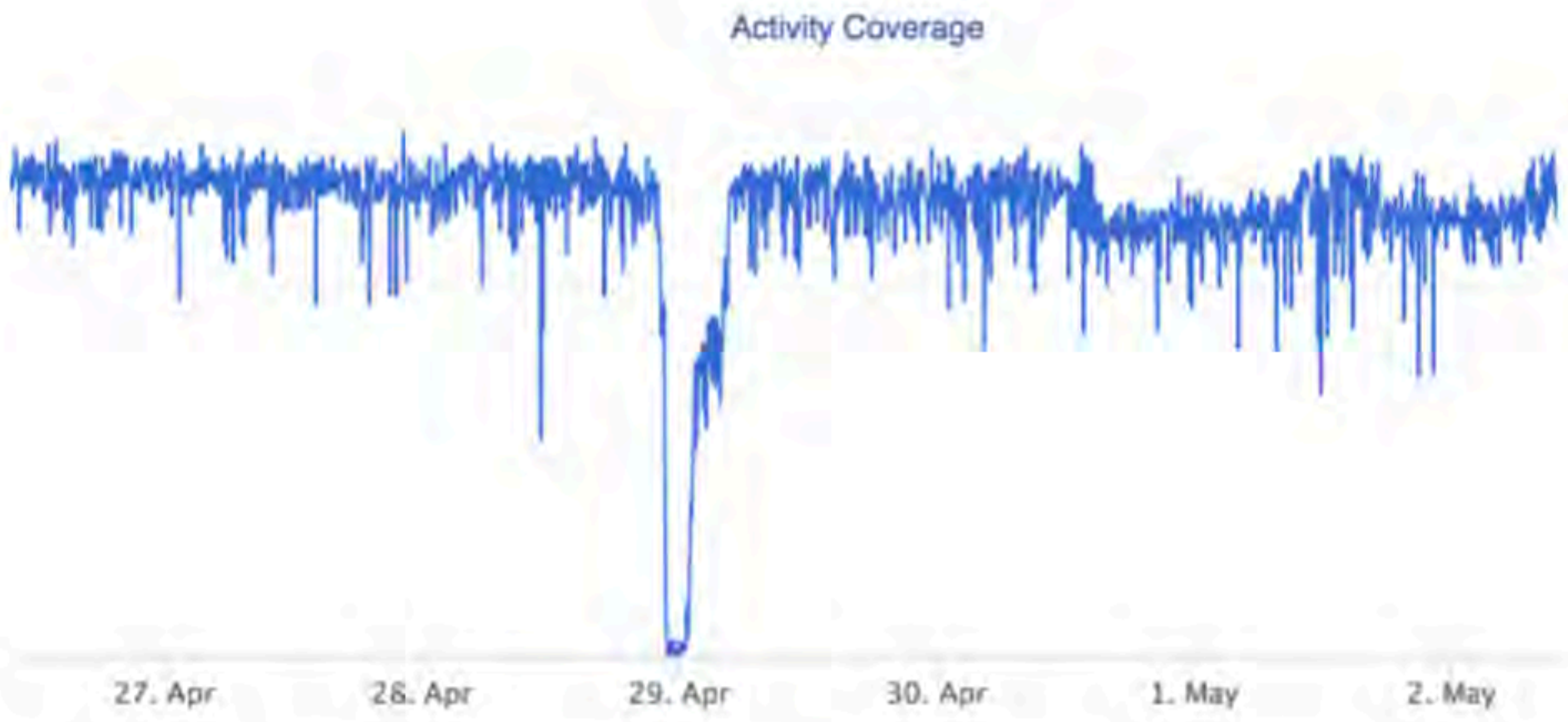


Fig. 9: Example DevOps monitoring: activity coverage over three days' worth of production runs in April 2018 (vertical axis deliberately occluded).

Fortunately, in this case, the sudden drop proved to be merely the result of a temporary quota limit on emulators being reached and within a few minutes, normal behaviour resumed. This example is included as an illustration of the way in which a DevOps approach is used to tackle availability and resilience of the Sapienz infrastructure. Naturally, an interesting challenge is to automate, to the greatest extent possible, this resilience, so that little human intervention is required to maintain healthy operation.

6.2 Key Performance Indicators

Figure 10 shows the two key performance indicators of crashes triaged to developers by Sapienz, and fixes detected by the automated fix detection protocol

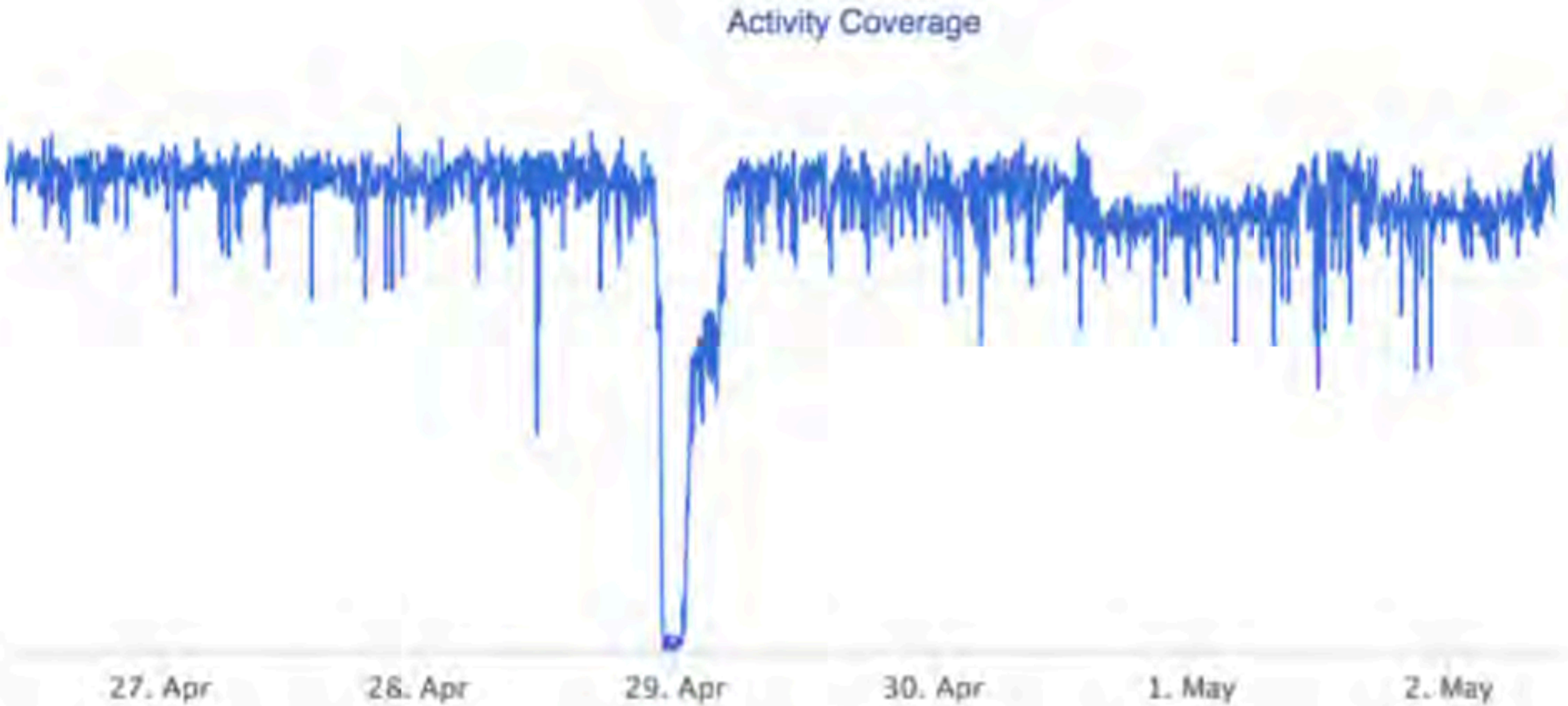
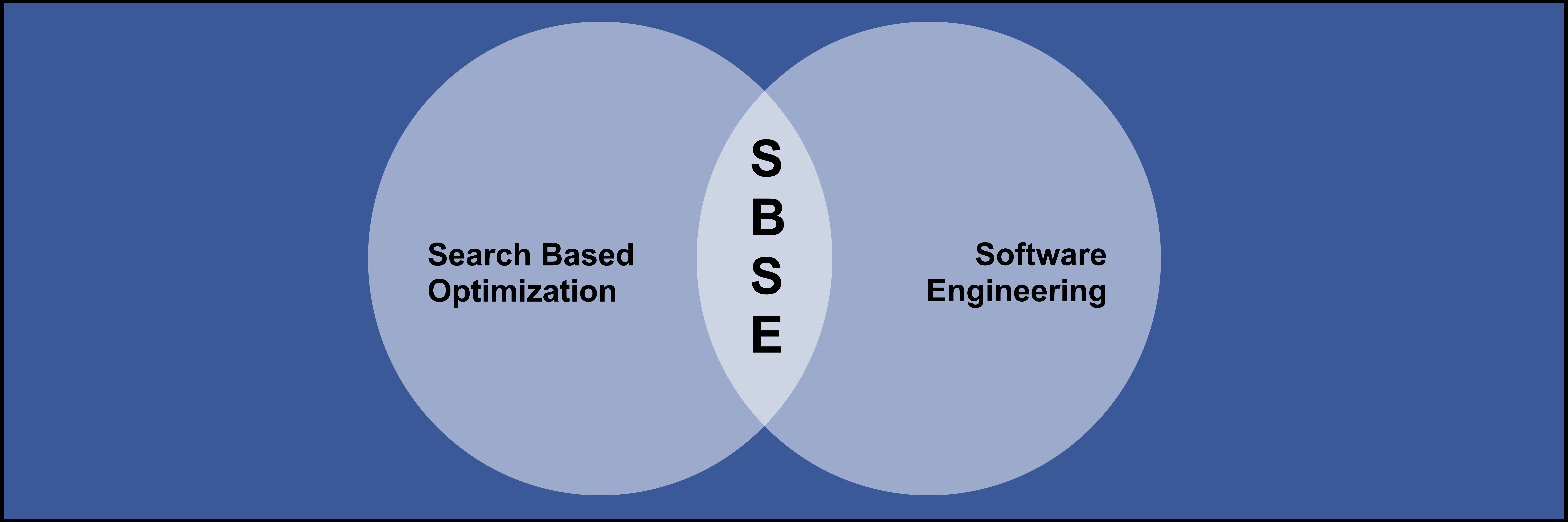


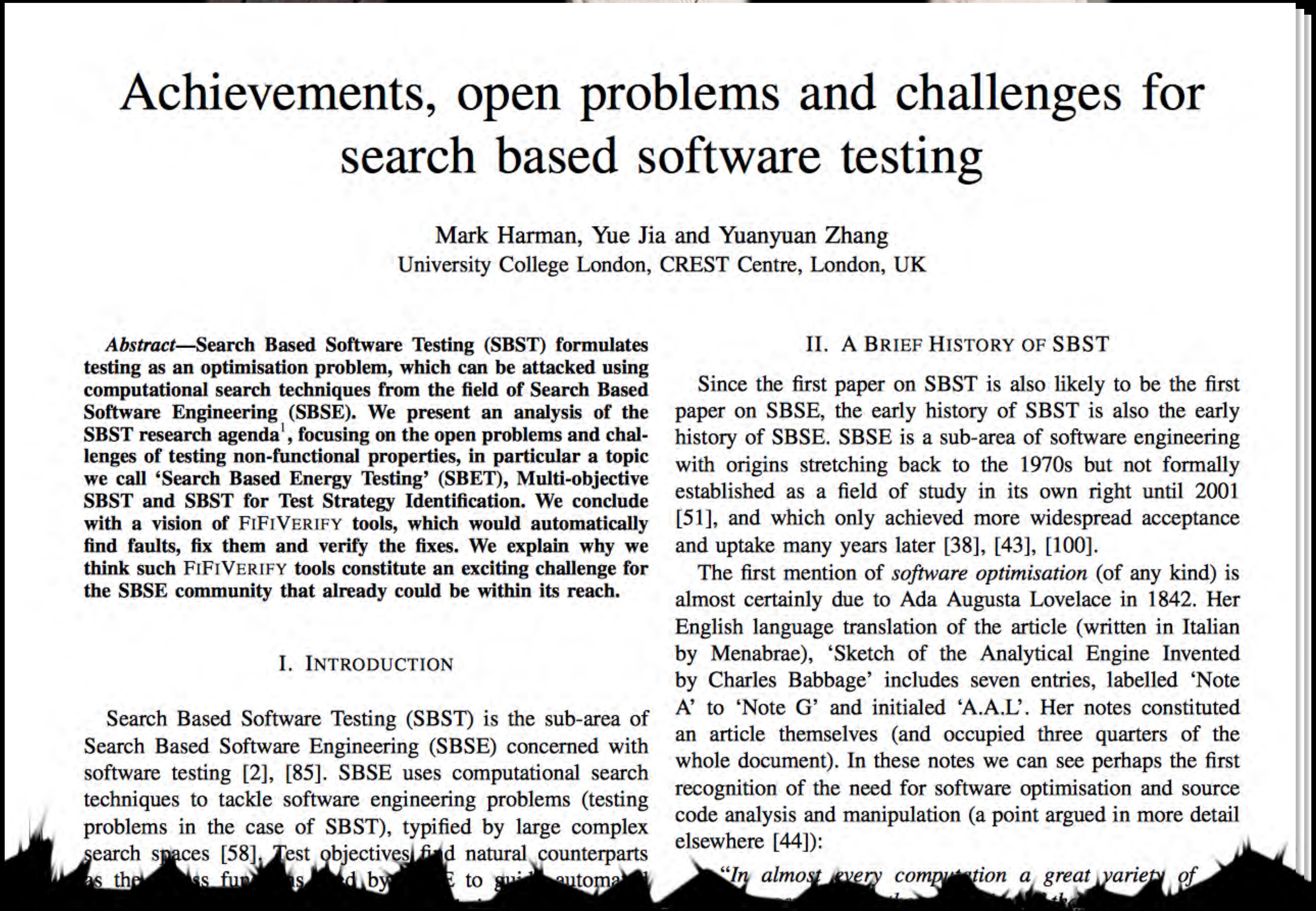
Fig. 9: Example DevOps monitoring: activity coverage over three days' worth of production runs in April 2018 (vertical axis deliberately occluded).

We use Search Based Software Engineering



widely studied in academia; now starting to reach into industry

... covers history of SBST up to 2015



Achievements, open problems and challenges for search based software testing

Mark Harman, Yue Jia and Yuanyuan Zhang
University College London, CREST Centre, London, UK

Abstract—**Search Based Software Testing (SBST) formulates testing as an optimisation problem, which can be attacked using computational search techniques from the field of Search Based Software Engineering (SBSE). We present an analysis of the SBST research agenda[1], focusing on the open problems and challenges of testing non-functional properties, in particular a topic we call 'Search Based Energy Testing' (SBET), Multi-objective SBST and SBST for Test Strategy Identification. We conclude with a vision of FIFIVERIFY tools, which would automatically find faults, fix them and verify the fixes. We explain why we think such FIFIVERIFY tools constitute an exciting challenge for the SBSE community that already could be within its reach.**

I. Introduction

Search Based Software Testing (SBST) is the sub-area of Search Based Software Engineering (SBSE) concerned with software testing [2], [85]. SBSE uses computational search techniques to tackle software engineering problems (testing problems in the case of SBST), typified by large complex search spaces [58]. Test objectives find natural counterparts as the fitness functions used by SBSE to guide automated

II. A Brief History of SBST

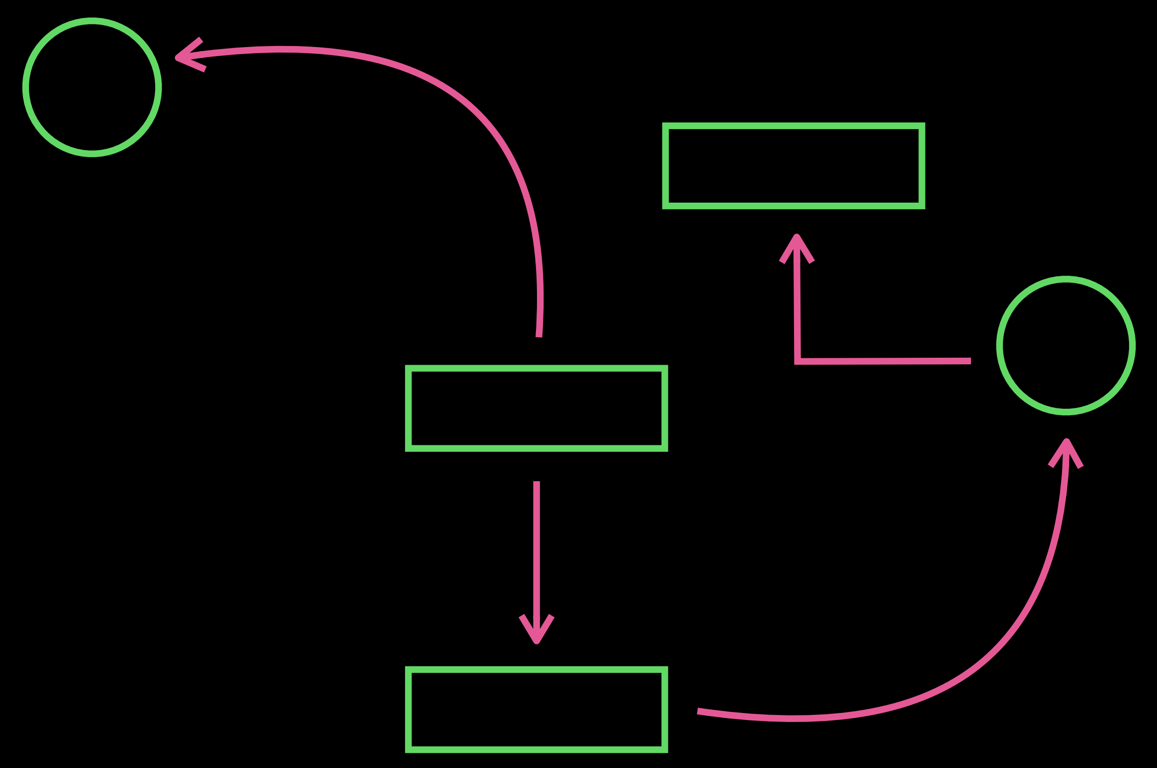
Since the first paper on SBST is also likely to be the first paper on SBSE, the early history of SBST is also the early history of SBSE. SBSE is a sub-area of software engineering with origins stretching back to the 1970s but not formally established as a field of study in its own right until 2001 [51], and which only achieved more widespread acceptance and uptake many years later [38], [43], [100].

The first mention of *software optimisation* (of any kind) is almost certainly due to Ada Augusta Lovelace in 1842. Her English language translation of the article (written in Italian by Menabrae), 'Sketch of the Analytical Engine Invented by Charles Babbage' includes seven entries, labelled 'Note A' to 'Note G' and initialed 'A.A.L'. Her notes constituted an article themselves (and occupied three quarters of the whole document). In these notes we can see perhaps the first recognition of the need for software optimisation and source code analysis and manipulation (a point argued in more detail elsewhere [44]):

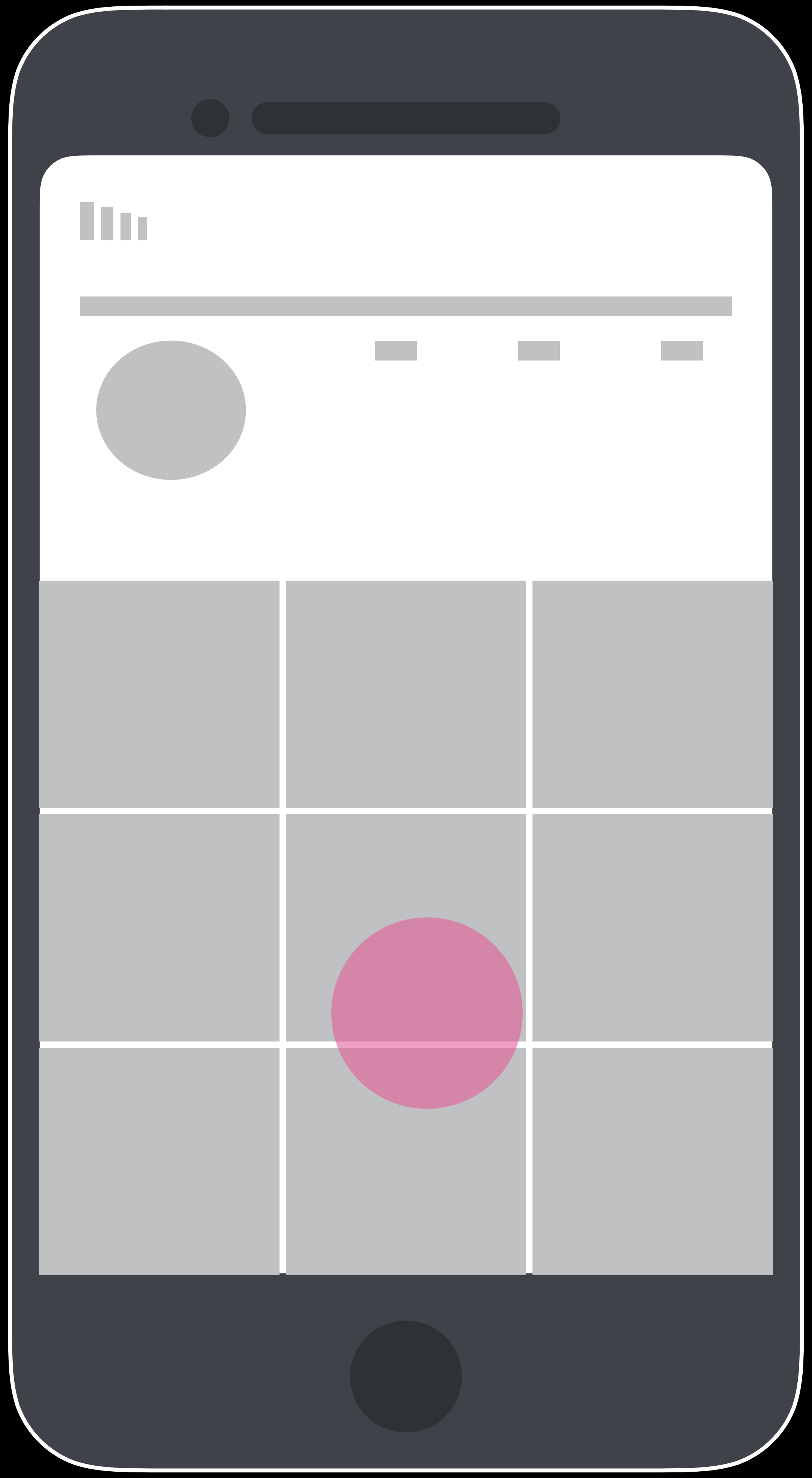
> "*In almost every computation a great variety of*

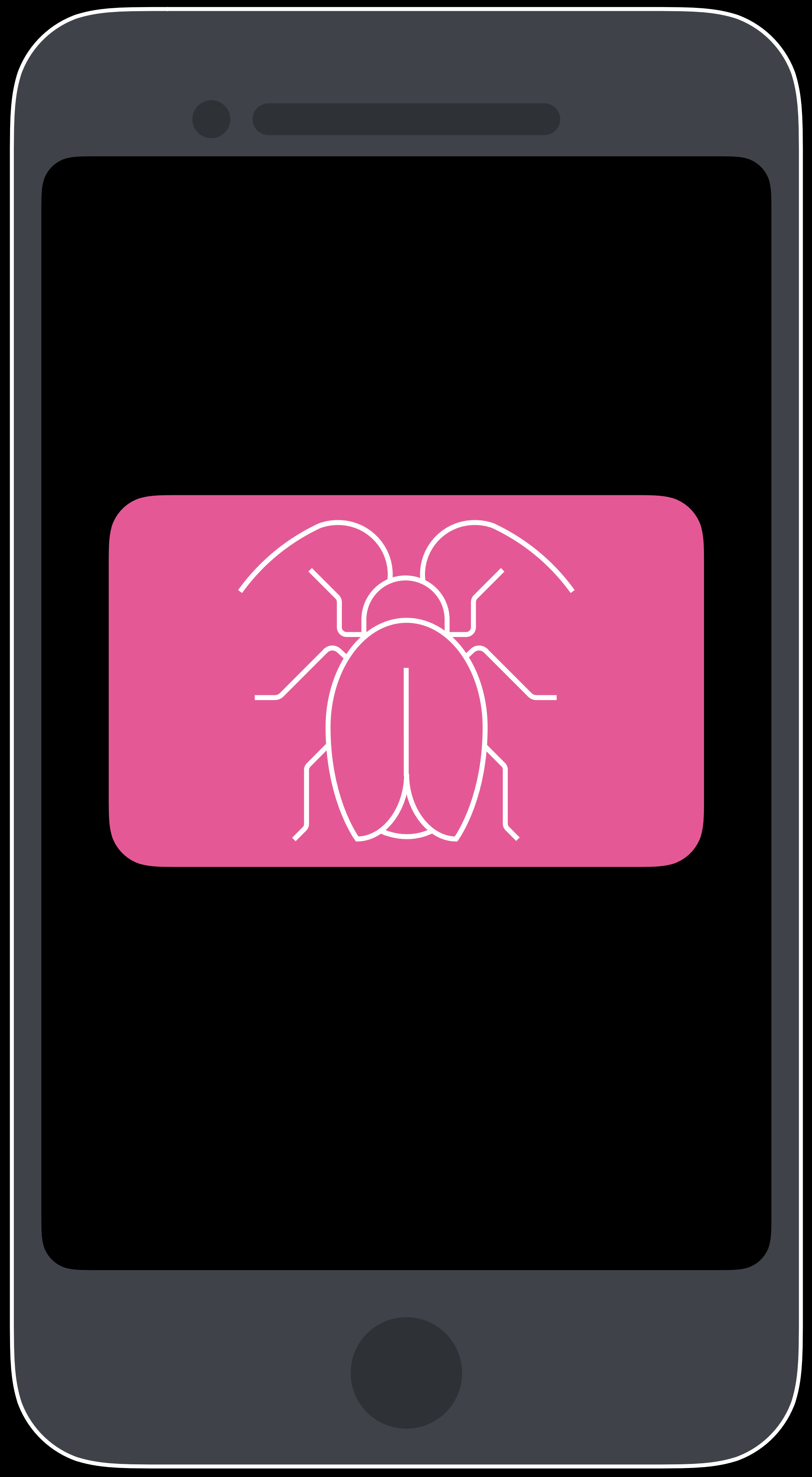
Test live in a search space

Engineers Design



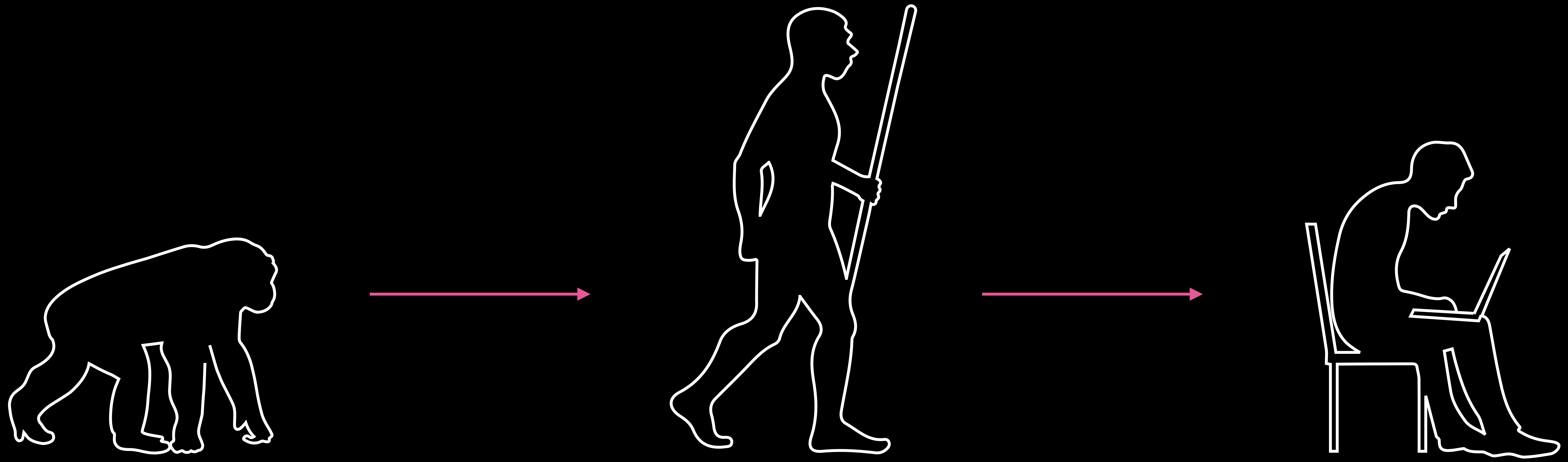
Machines Execute





Unfortunately,
your app has stopped.

System Level Testing

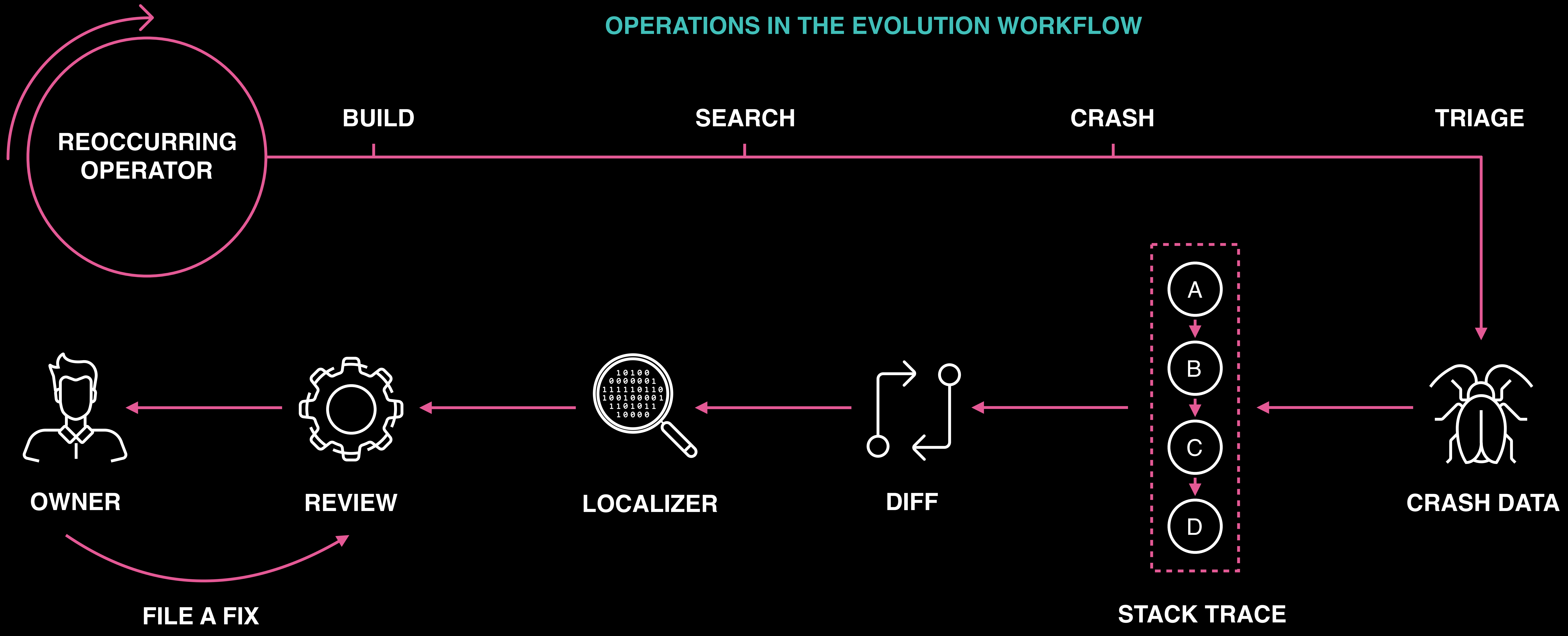


RANDOM FUZZER

sapienz

HUMAN TESTERS

Fault Triage Process

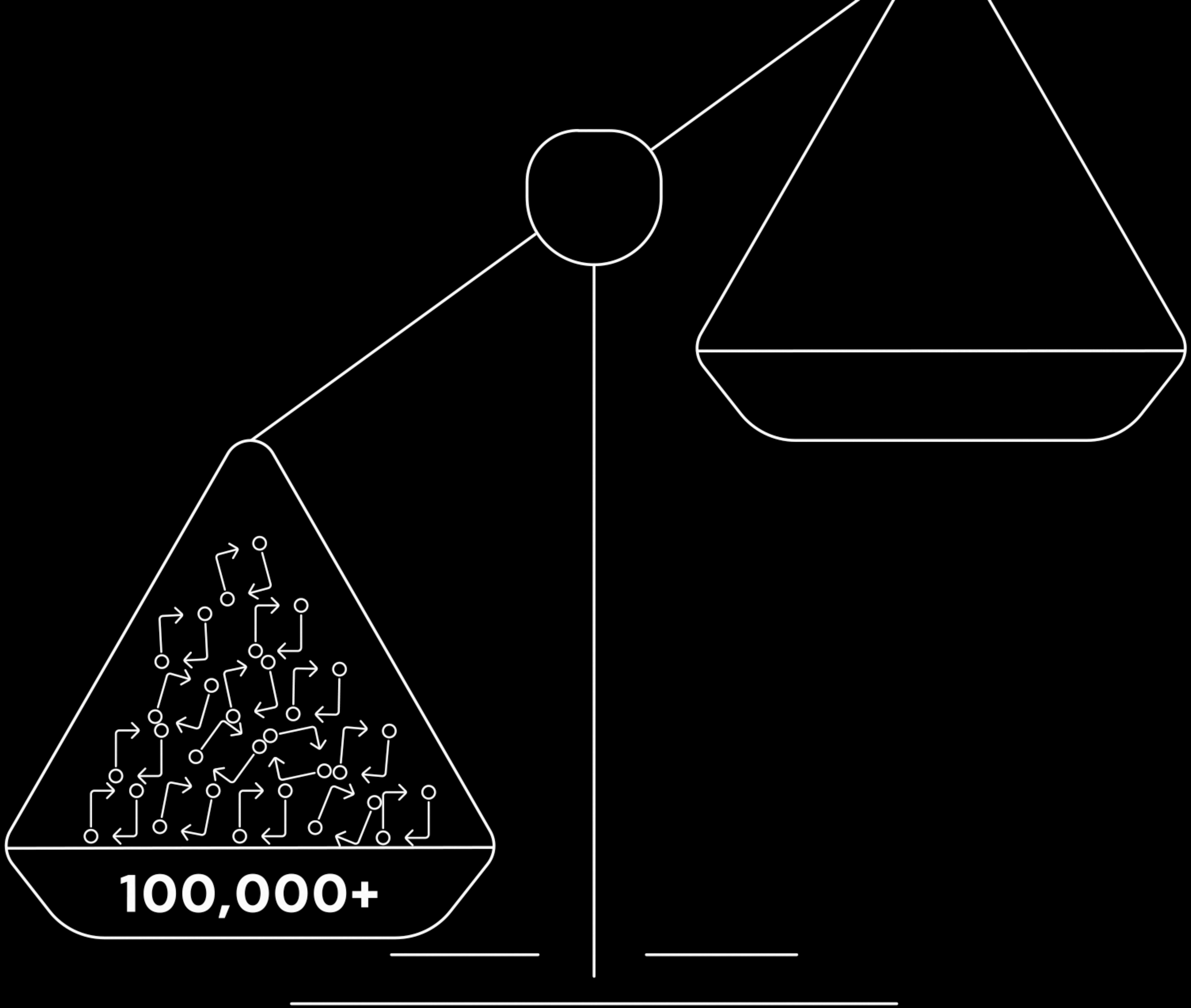


>1M

source control
commands run per day

>100K

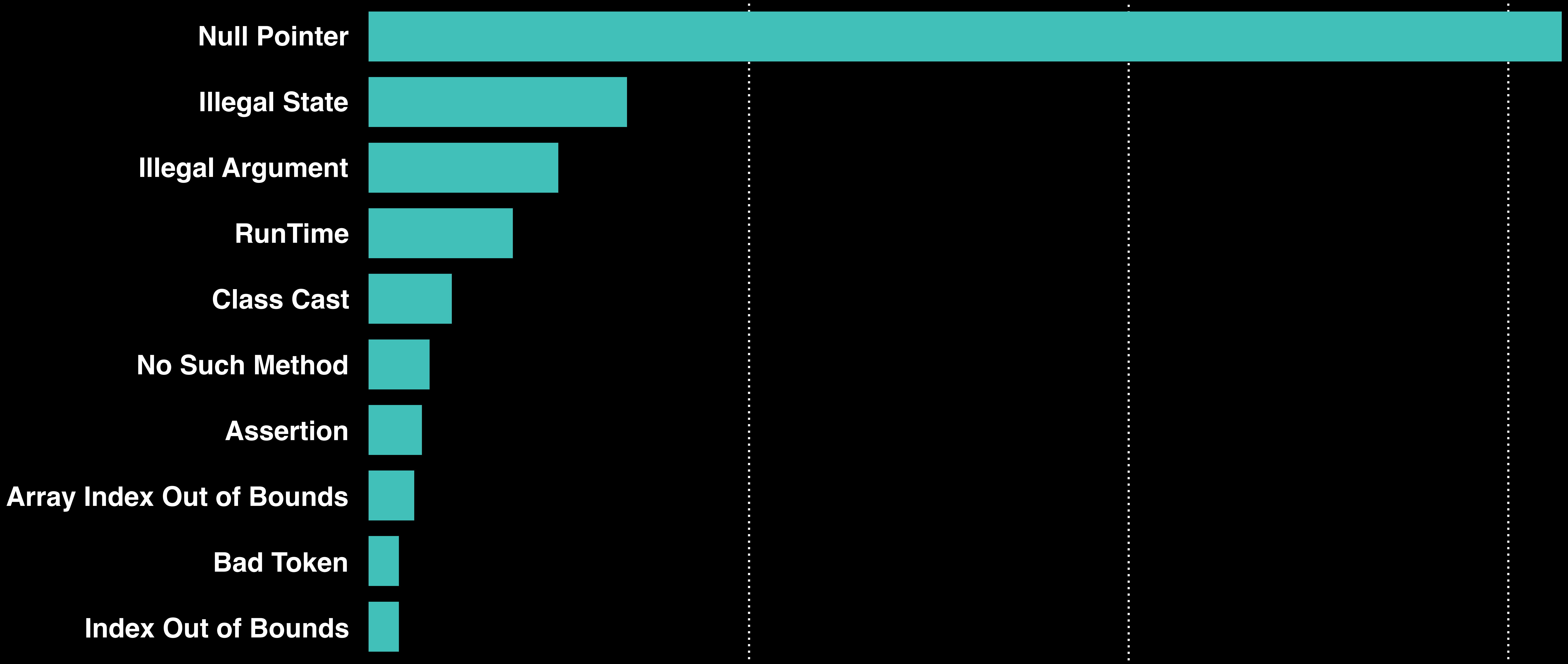
commits per week


100,000+

Facebook for iOS
Facebook for Android
Workplac e for Android
Messenge r for Android
Instagram for Android

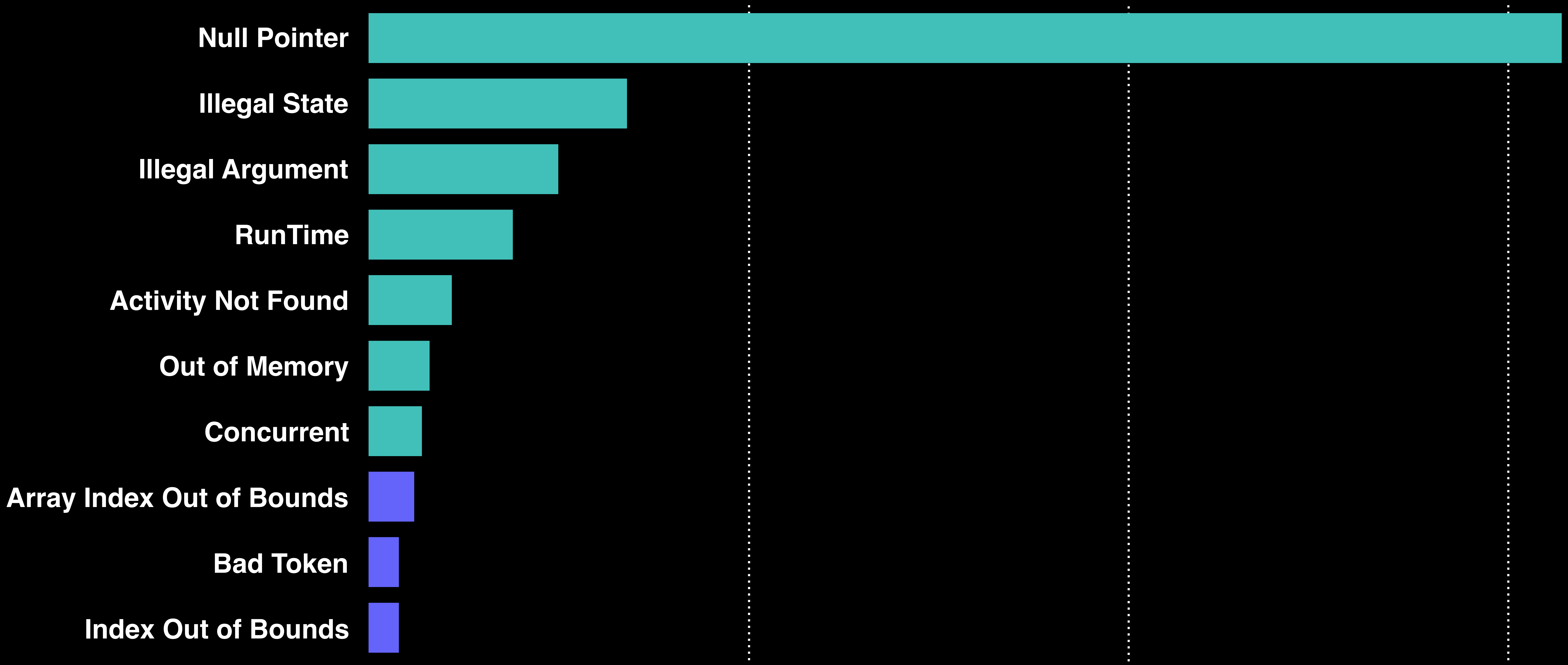
~75%

Distribution (FB)



TOP CRASHES TYPES ON FACEBOOK FOR ANDROID (BY SAPIENZ)

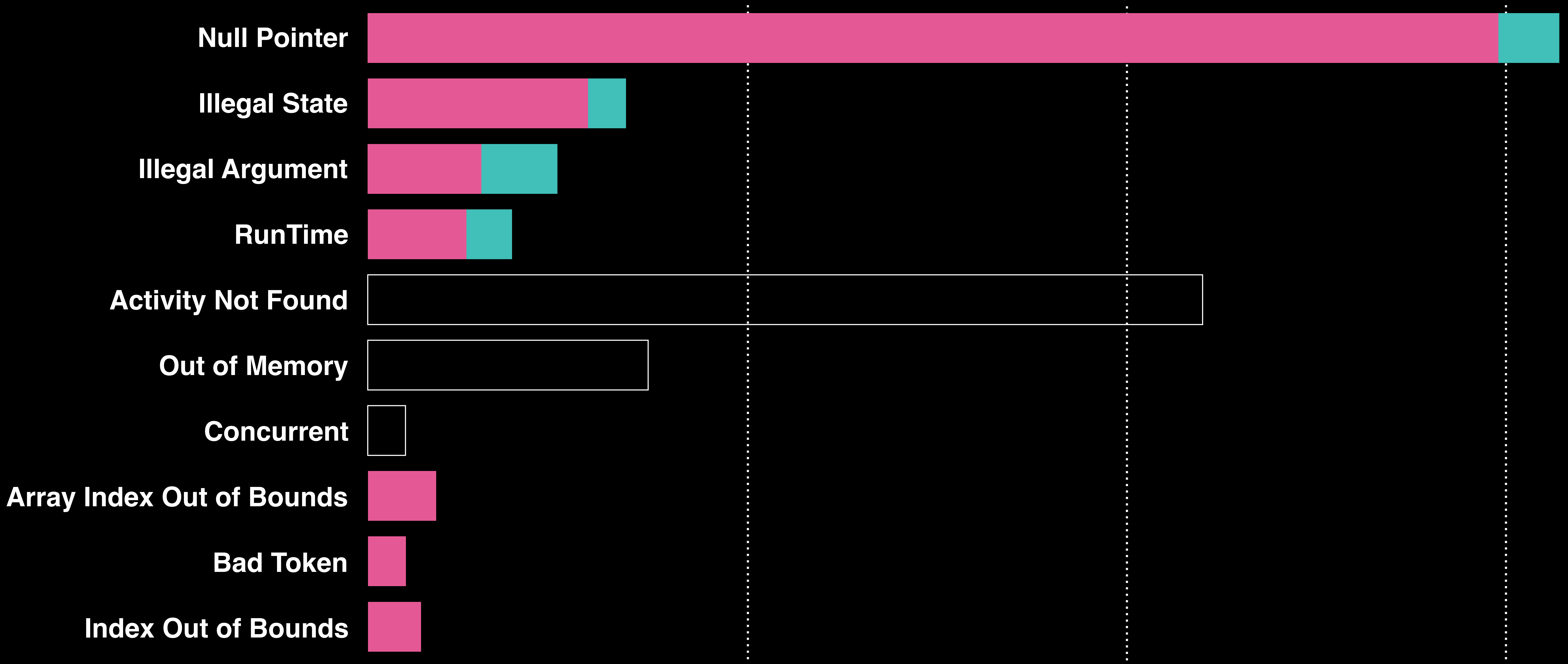
Distribution (Research)



TOP CRASHES TYPES ON 1000 ANDROID APPS (BY SAPIENZ[1])

[1] K. MAO, M. HARMAN, AND Y. JIA, “SAPIENZ: MULTI-OBJECTIVE AUTOMATED TESTING FOR ANDROID APPLICATIONS,” IN PROC. OF ISSTA’16, 2016

Distribution (Research)



TOP CRASHES TYPES ON 1000 ANDROID APPS (BY SAPIENZ[1])

[1] K. MAO, M. HARMAN, AND Y. JIA, “SAPIENZ: MULTI-OBJECTIVE AUTOMATED TESTING FOR ANDROID APPLICATIONS,” IN PROC. OF ISSTA’16, 2016

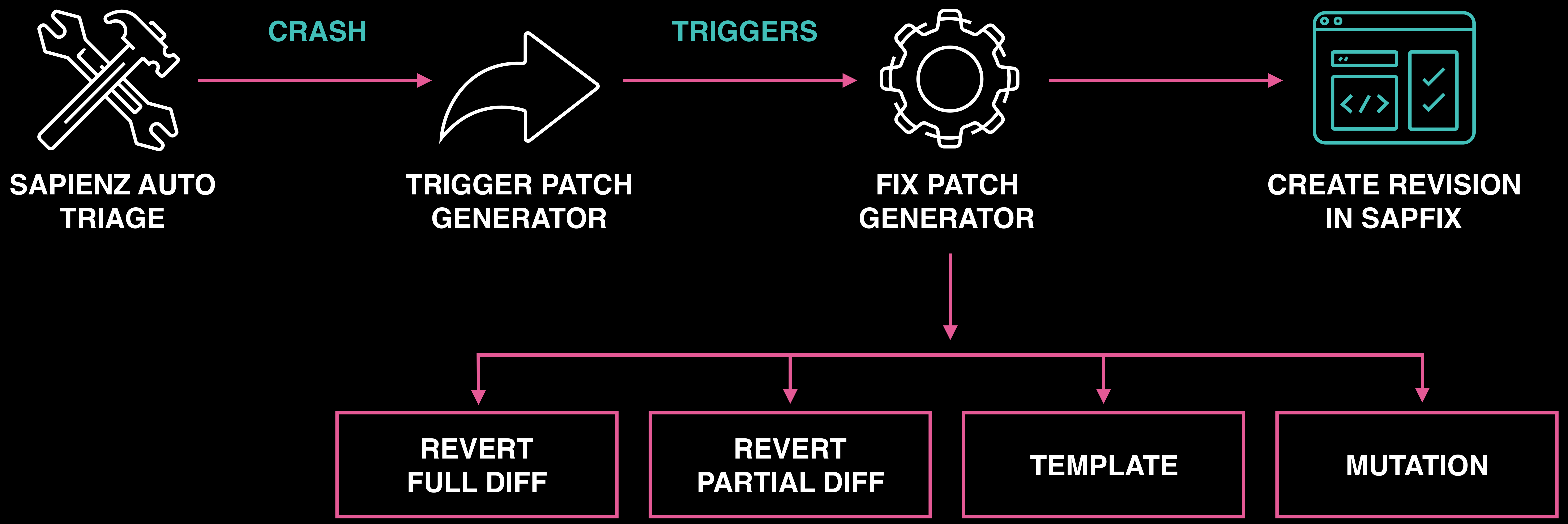
Auto Test Design

Auto Fix

Auto Boost

Auto Fix

Auto Fix Workflow (Generation)



SAPIENZ AUTO TRIAGE

CRASH

TRIGGER PATCH GENERATOR

TRIGGERS

FIX PATCH GENERATOR

CREATE REVISION IN SAPFIX

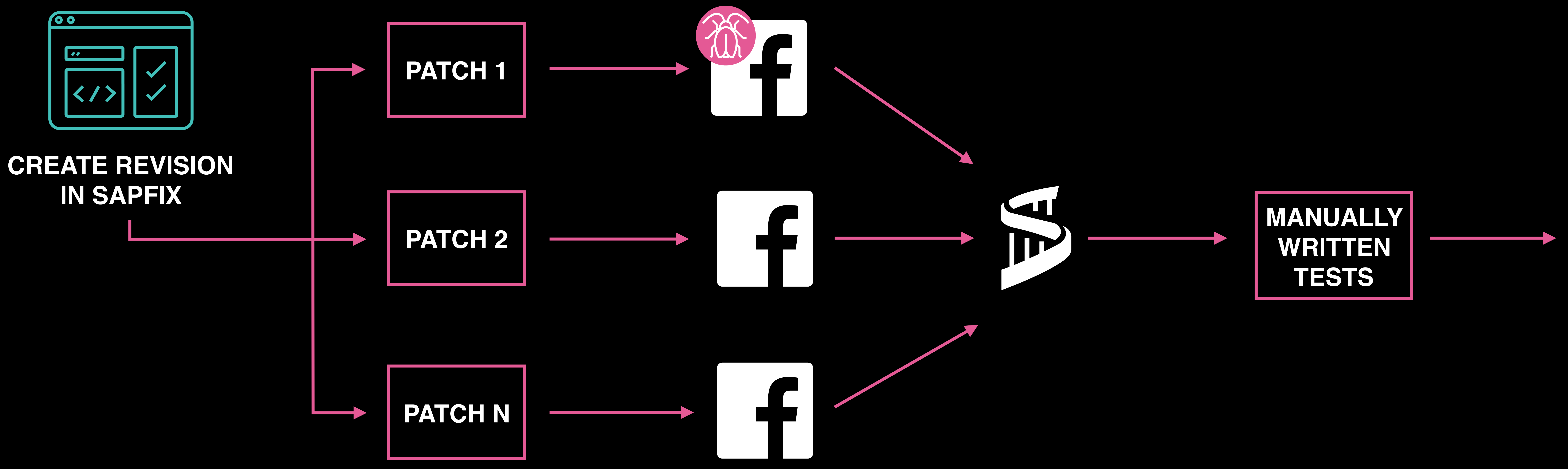
REVERT FULL DIFF

REVERT PARTIAL DIFF

TEMPLATE

MUTATION

Auto Fix Workflow (Validation)



CREATE REVISION IN SAPFIX

PATCH 1

PATCH 2

PATCH N

MANUALLY WRITTEN TESTS

BUILD

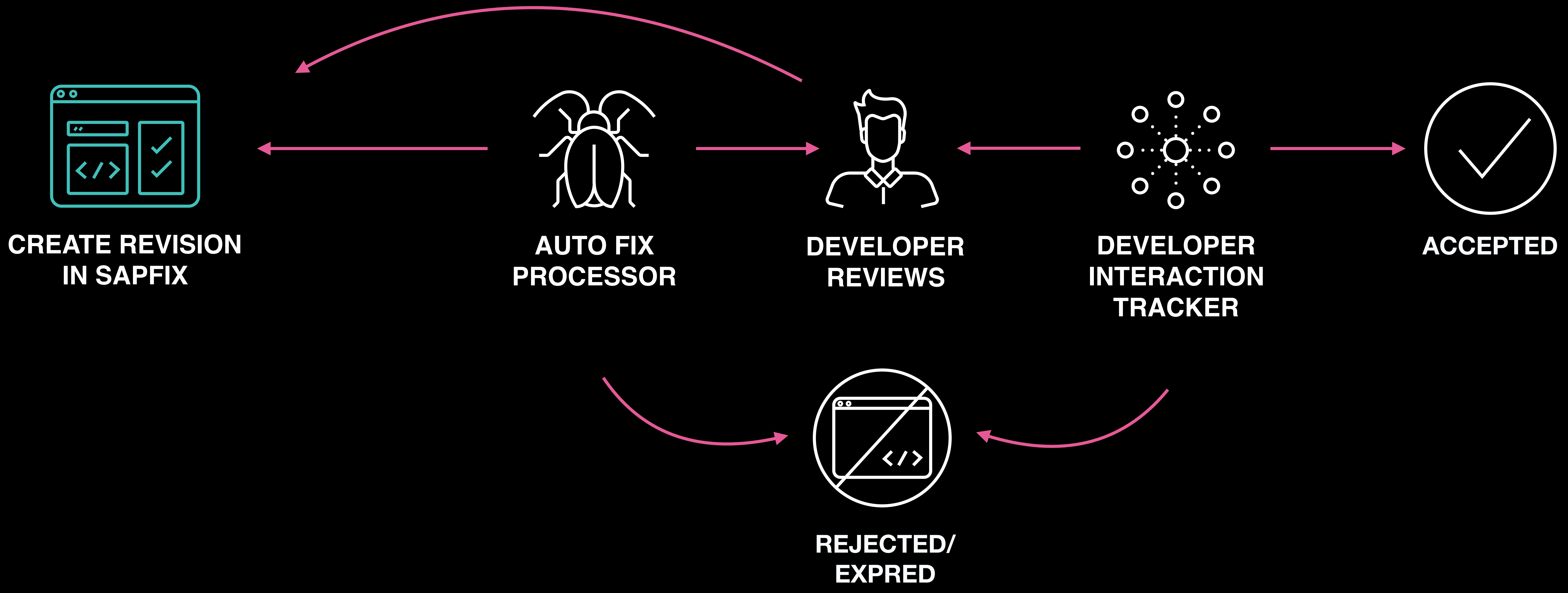
COMPILATION ERROR?

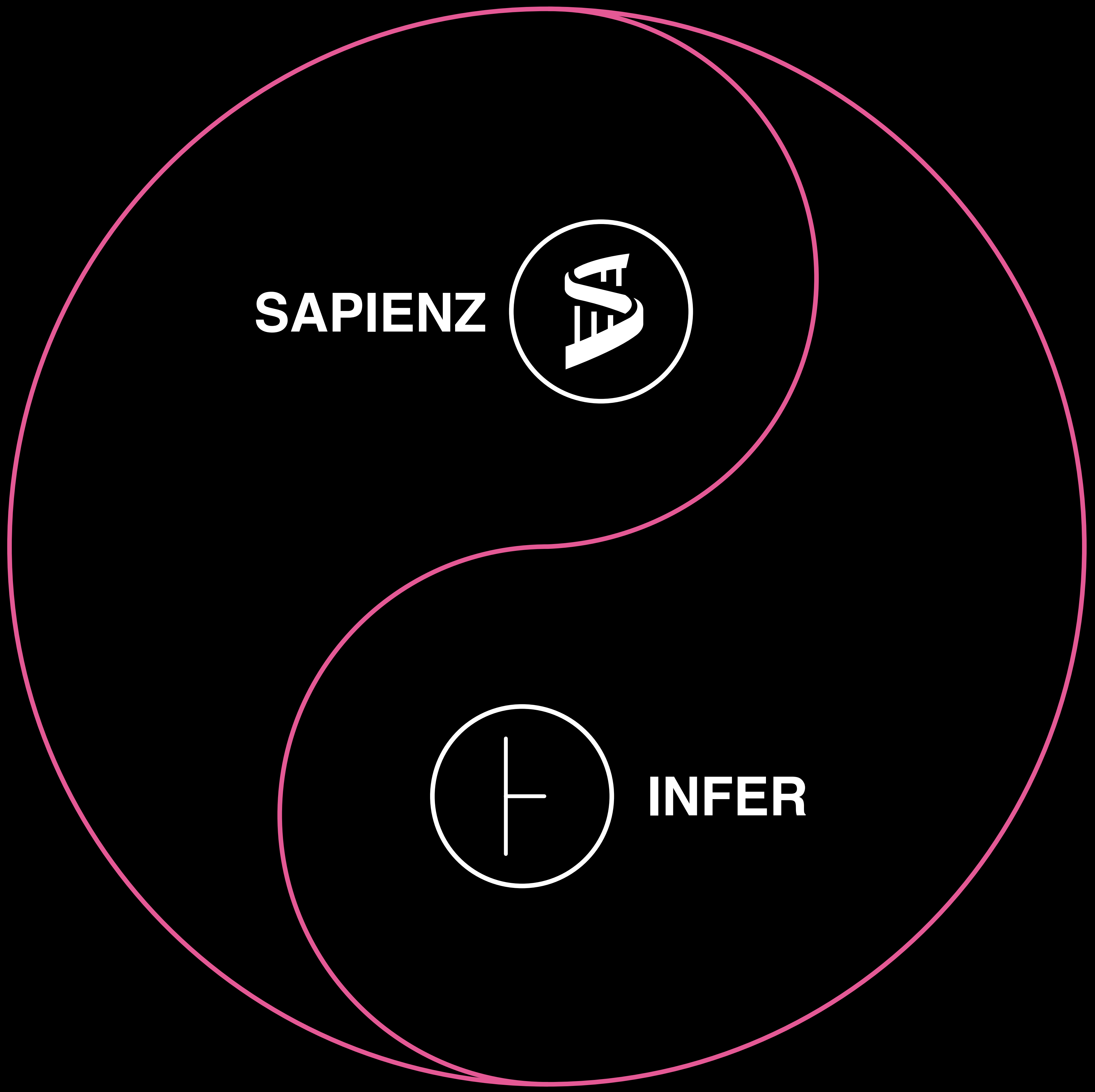
SAPIENZ

E2E

CRASH FIXED? NEW CRASH?

Auto Fix Workflow (Signal)





SAPIENZ

INFER

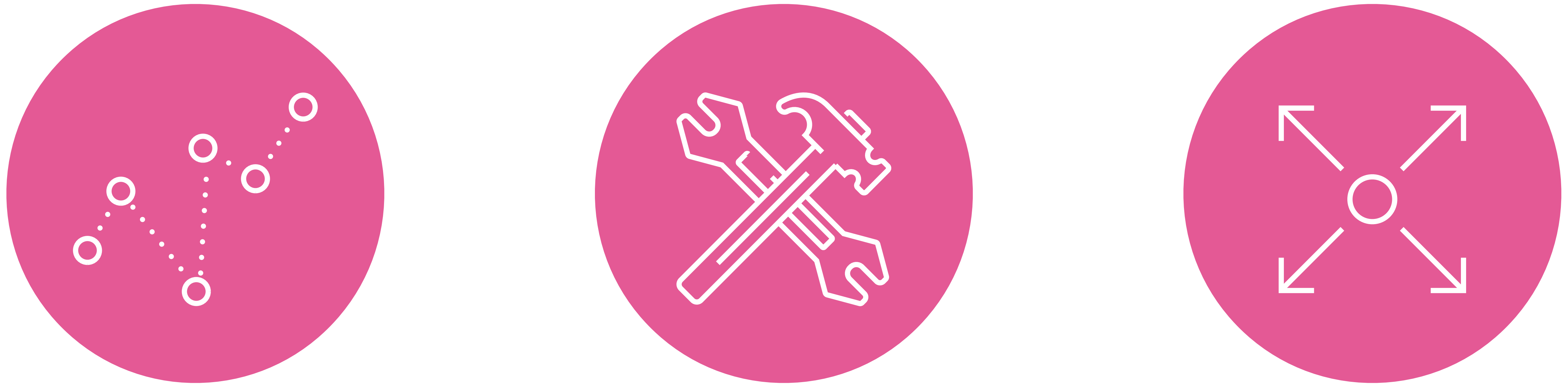
Localization: Sapienz + Infer

SAPIENZ

```
1559  // Sapienz Revealed NPE Crash
      contextDateMap.put("name",
      displayedUser.getName());
```

INFER

```
      // INFER Warning
1559  The value of
      'UserDetailFragment.displayedUser
      in the call to 'getName()'
      could be null.
```



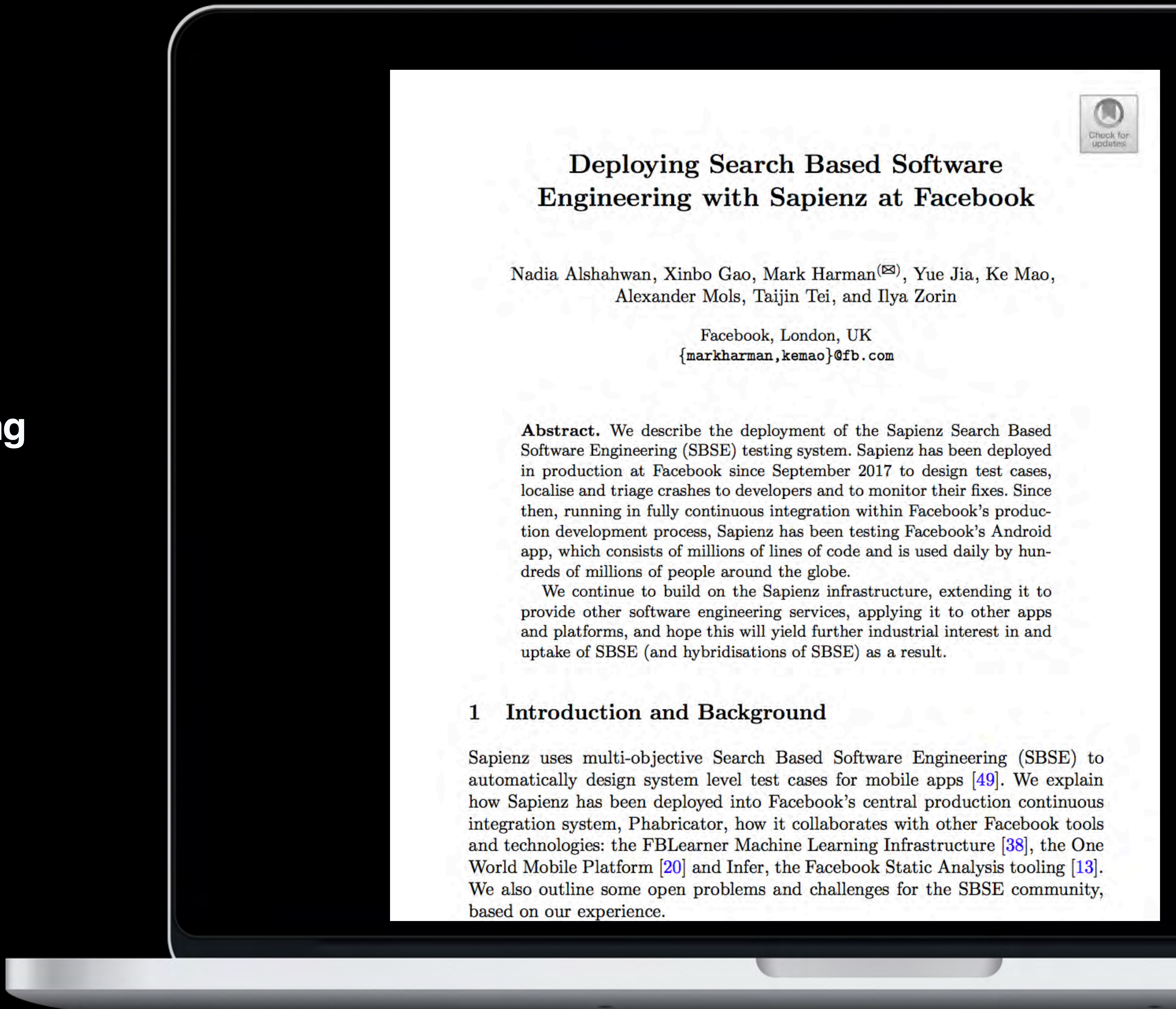
Flaky tests
Fix detection
Automated Oracles
Wider search spaces
Smarter white box coverage
Combining static and dynamic
Unit test from system tests
Human machine test hybrids
Fully parallel search algorithms
Auto Fix and perf improvement

Resources

Deploying
Search Based Software Engineering
with Sapienz
at Facebook

NADIA ALSHAHWAN,
XINBO GAO,
MARK HARMAN,
YUE JIA,
KE MAO,
ALEXANDER MOLS,
TAIJIN TEI
AND
ILYA ZORIN

IN PROC. OF SSBSE, 2018.



Deploying Search Based Software Engineering with Sapienz at Facebook

Nadia Alshahwan, Xinbo Gao, Mark Harman(✉), Yue Jia, Ke Mao,
Alexander Mols, Taijin Tei, and Ilya Zorin

Facebook, London, UK
{markharman,kemao}@fb.com

Abstract. We describe the deployment of the Sapienz Search Based Software Engineering (SBSE) testing system. Sapienz has been deployed in production at Facebook since September 2017 to design test cases, localise and triage crashes to developers and to monitor their fixes. Since then, running in fully continuous integration within Facebook's production development process, Sapienz has been testing Facebook's Android app, which consists of millions of lines of code and is used daily by hundreds of millions of people around the globe.

We continue to build on the Sapienz infrastructure, extending it to provide other software engineering services, applying it to other apps and platforms, and hope this will yield further industrial interest in and uptake of SBSE (and hybridisations of SBSE) as a result.

1 Introduction and Background

Sapienz uses multi-objective Search Based Software Engineering (SBSE) to automatically design system level test cases for mobile apps [49]. We explain how Sapienz has been deployed into Facebook's central production continuous integration system, Phabricator, how it collaborates with other Facebook tools and technologies: the FBLearner Machine Learning Infrastructure [38], the One World Mobile Platform [20] and Infer, the Facebook Static Analysis tooling [13]. We also outline some open problems and challenges for the SBSE community, based on our experience.