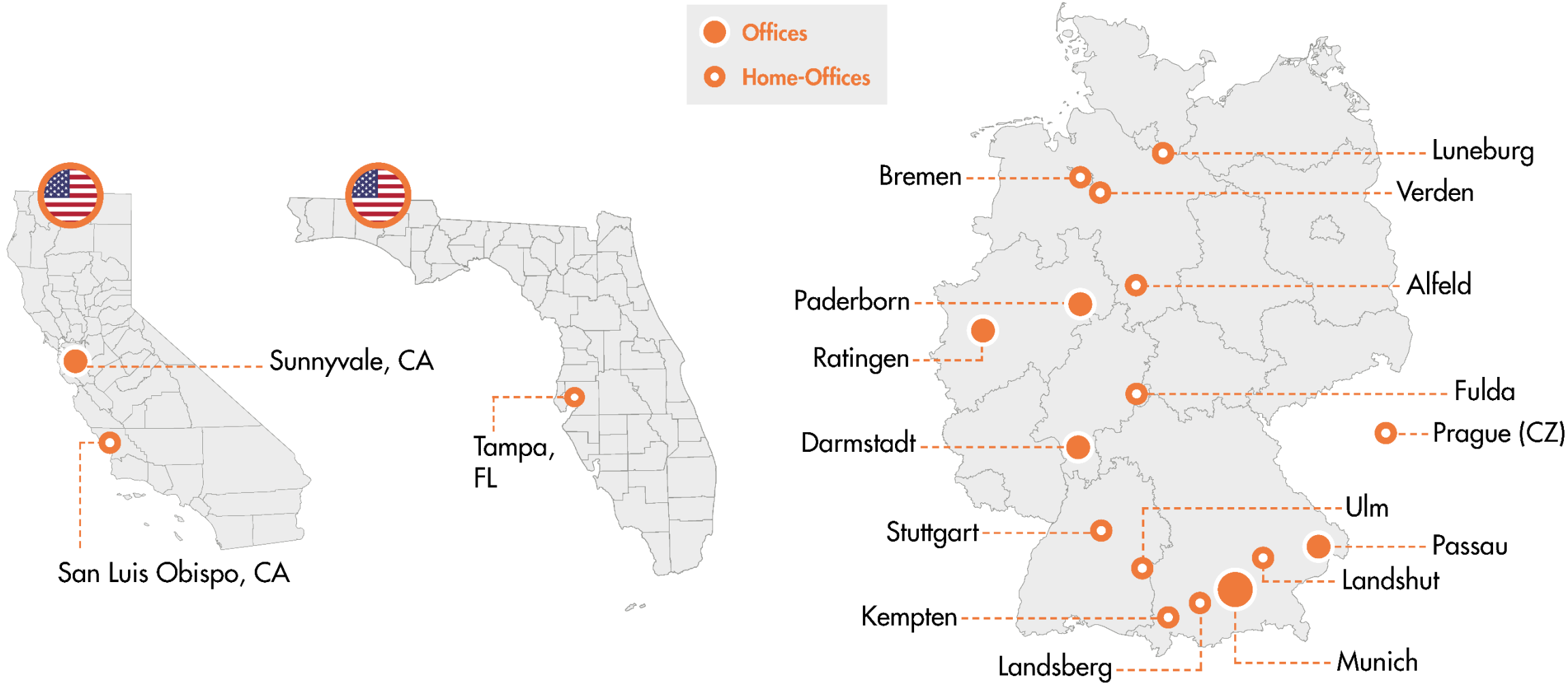
Teamscale by CQSE

Everyone wants to shift left, but our tests are just too slow!

Ramona Kühn

Raphael Nömmer



Teamscale

CQSE
Continuous Quality in Software Engineering

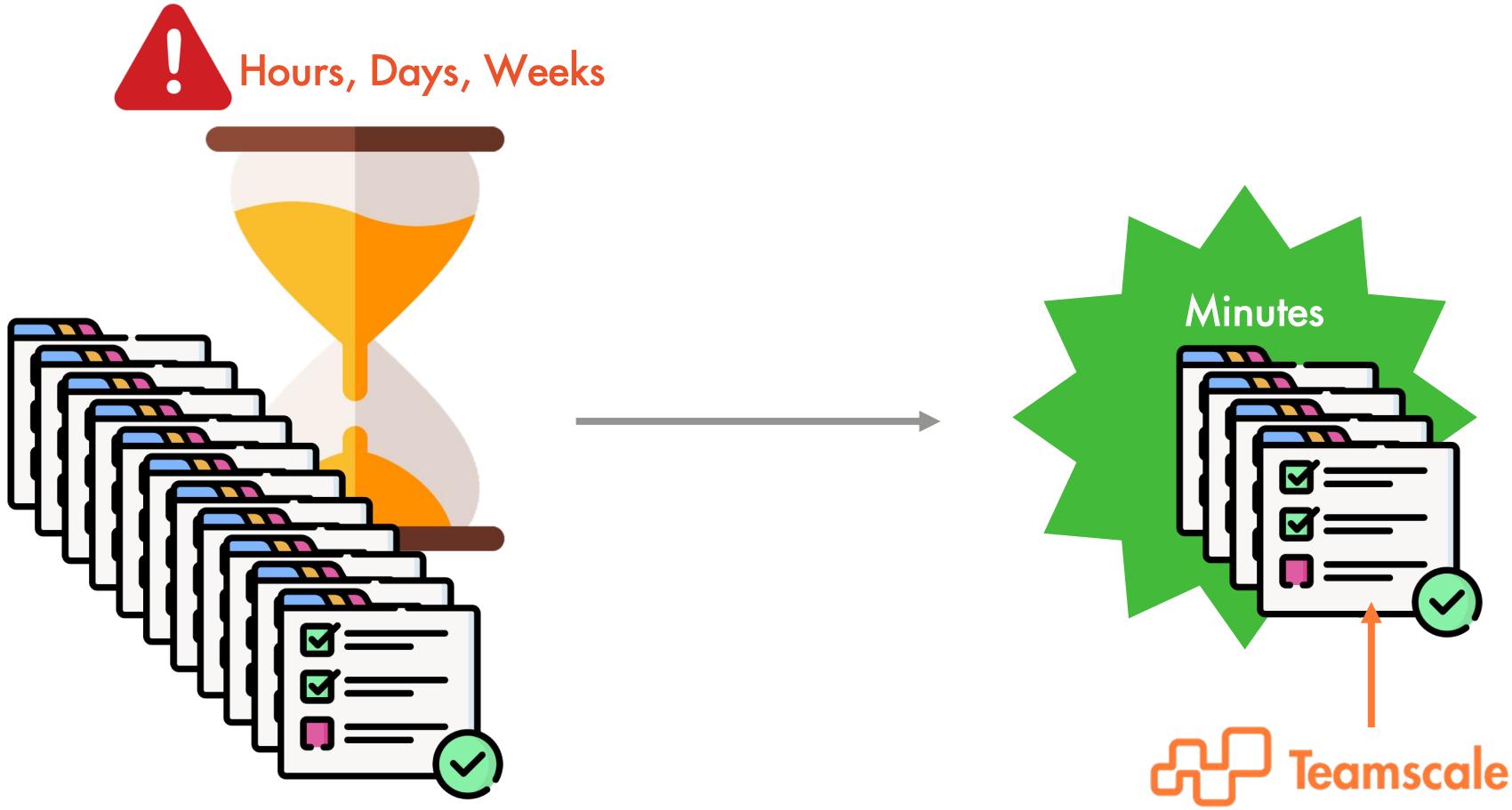
Raphael Nömmer

Test Intelligence Team

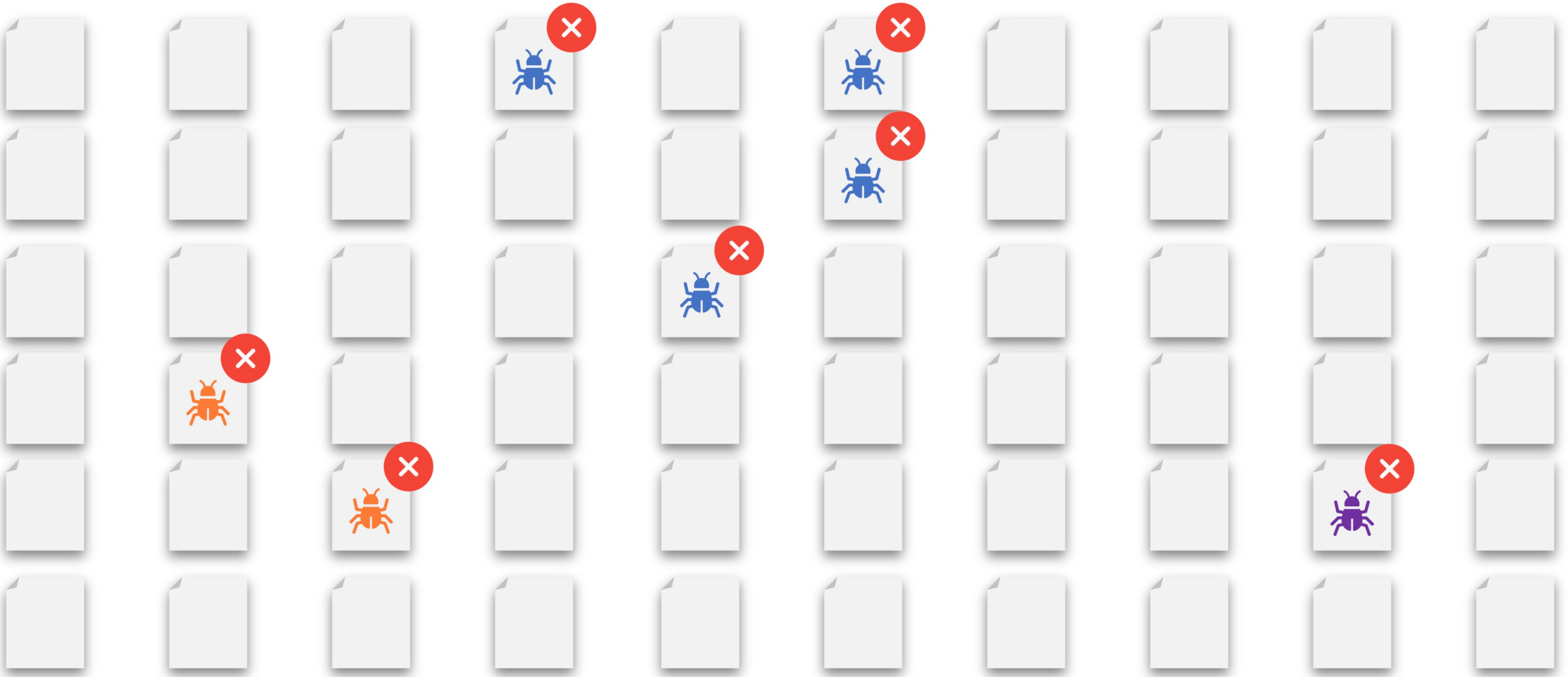
PhD Candidate

Onboarding new customers

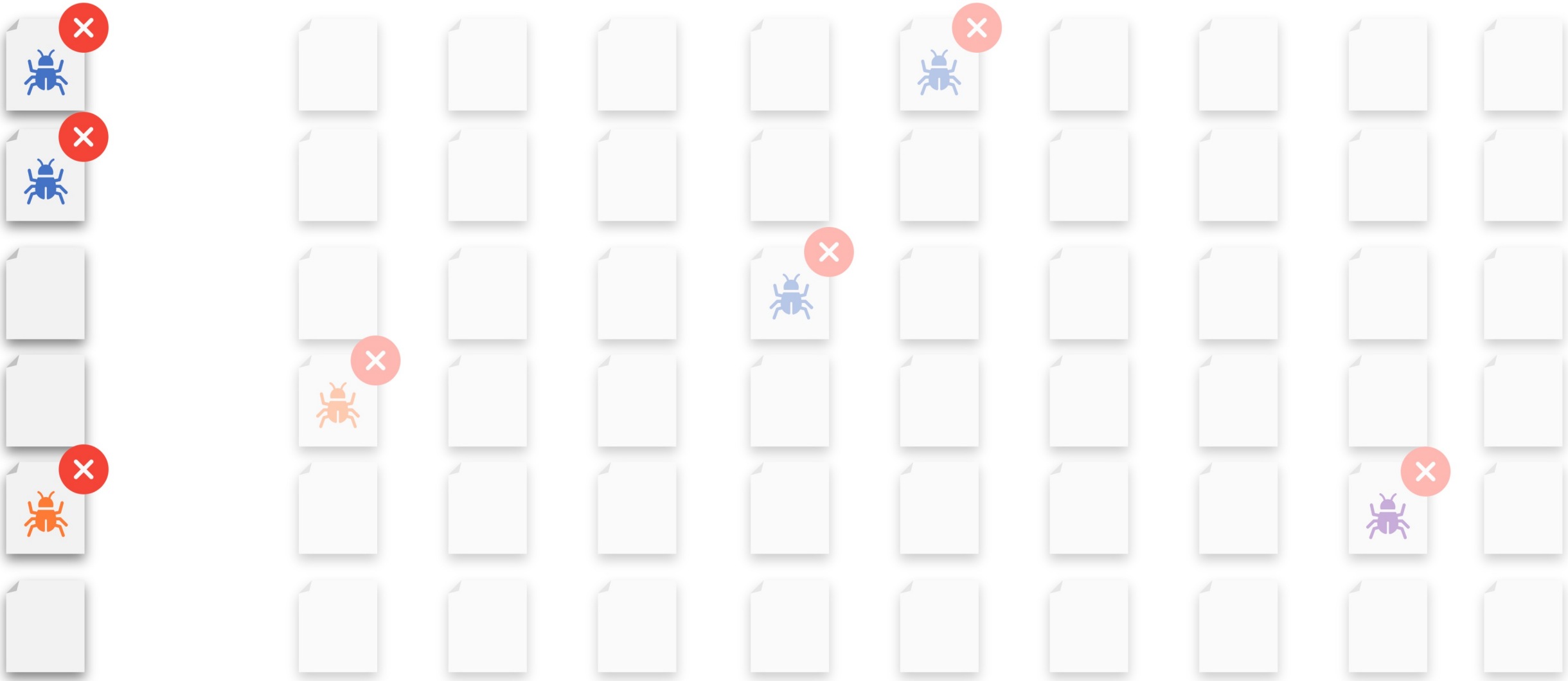
CQSE Teamscale

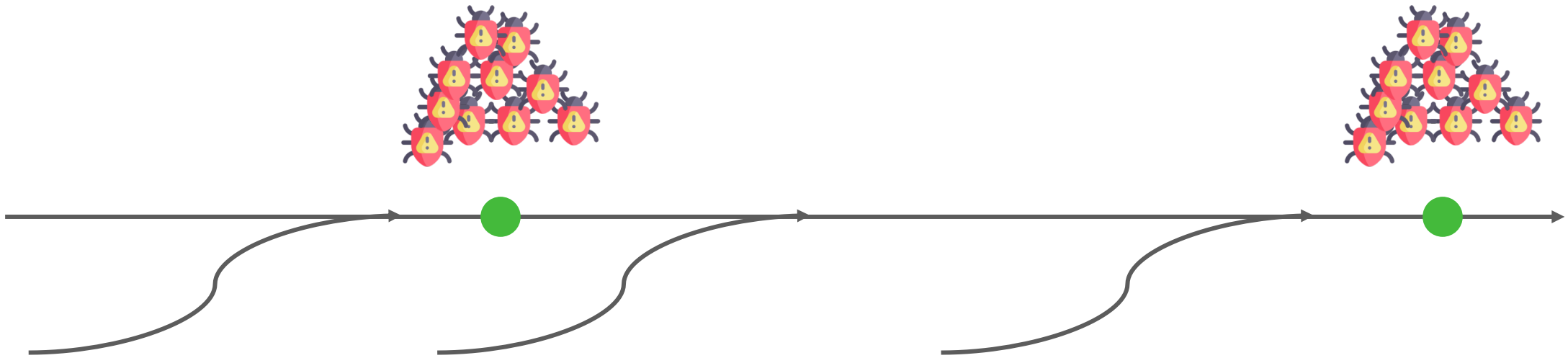


manual, automated,
E2E, …



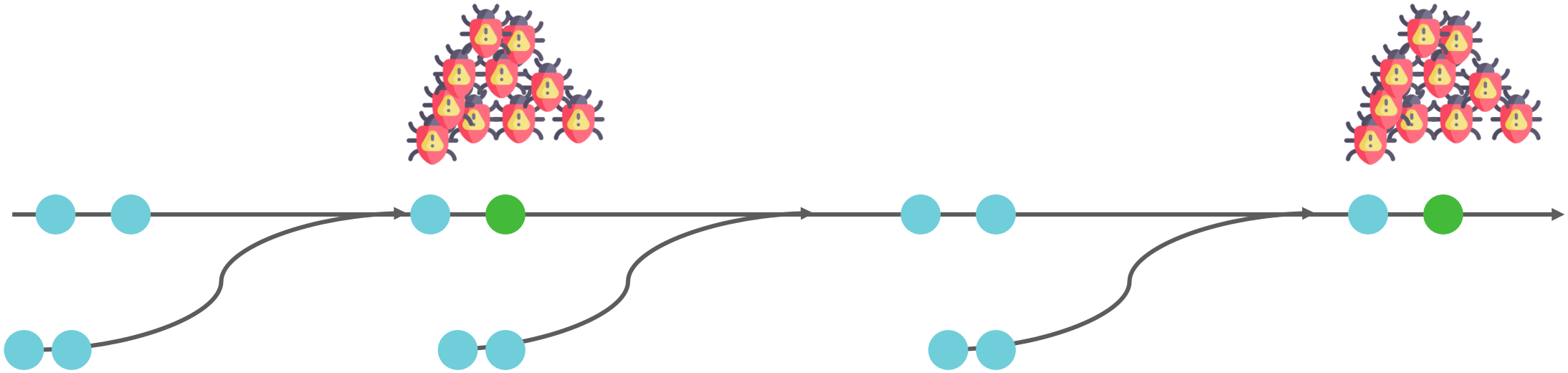
Test Selection





Complete Test Run

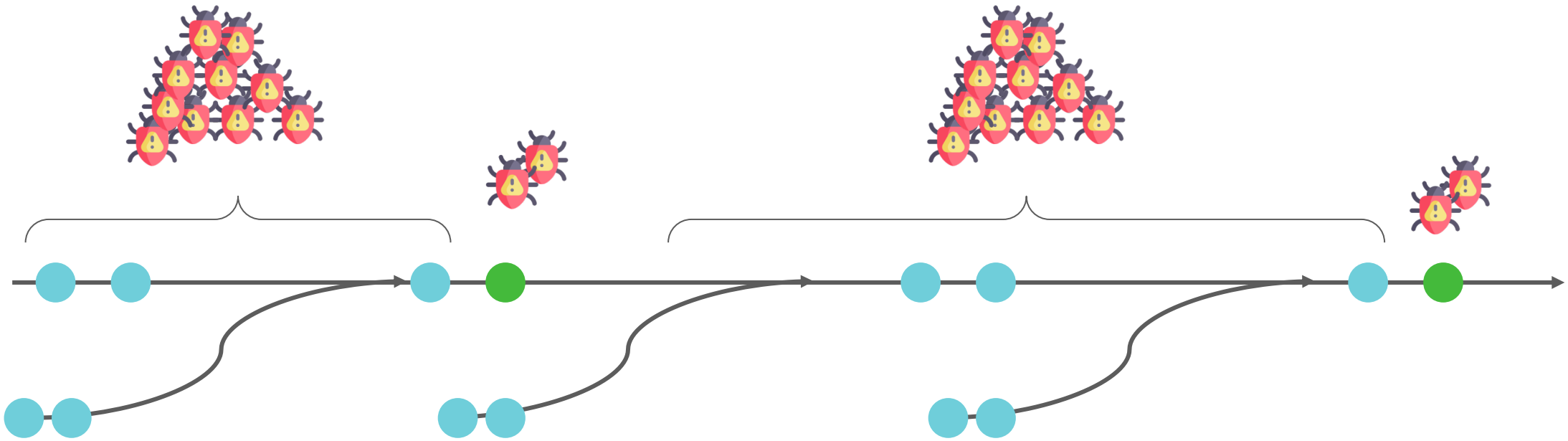
Shift Left



Complete Test Run

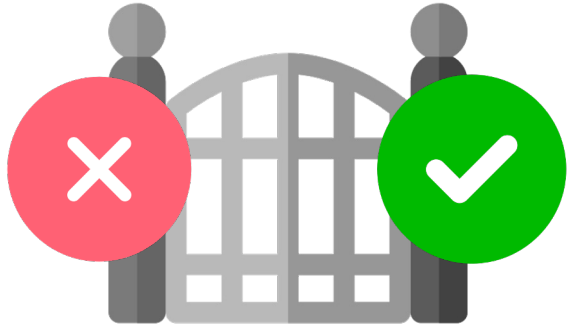
Smart Selection

Shift Left



Complete Test Run

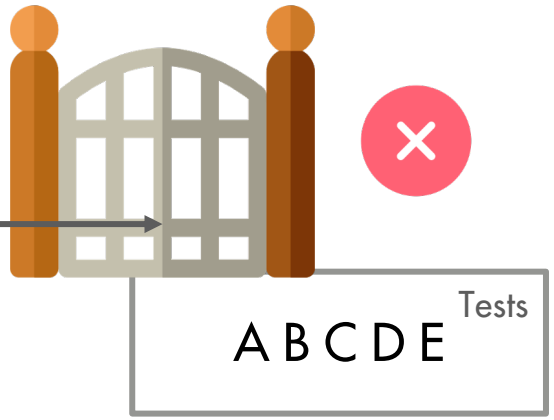
Smart Selection



Quality Gate

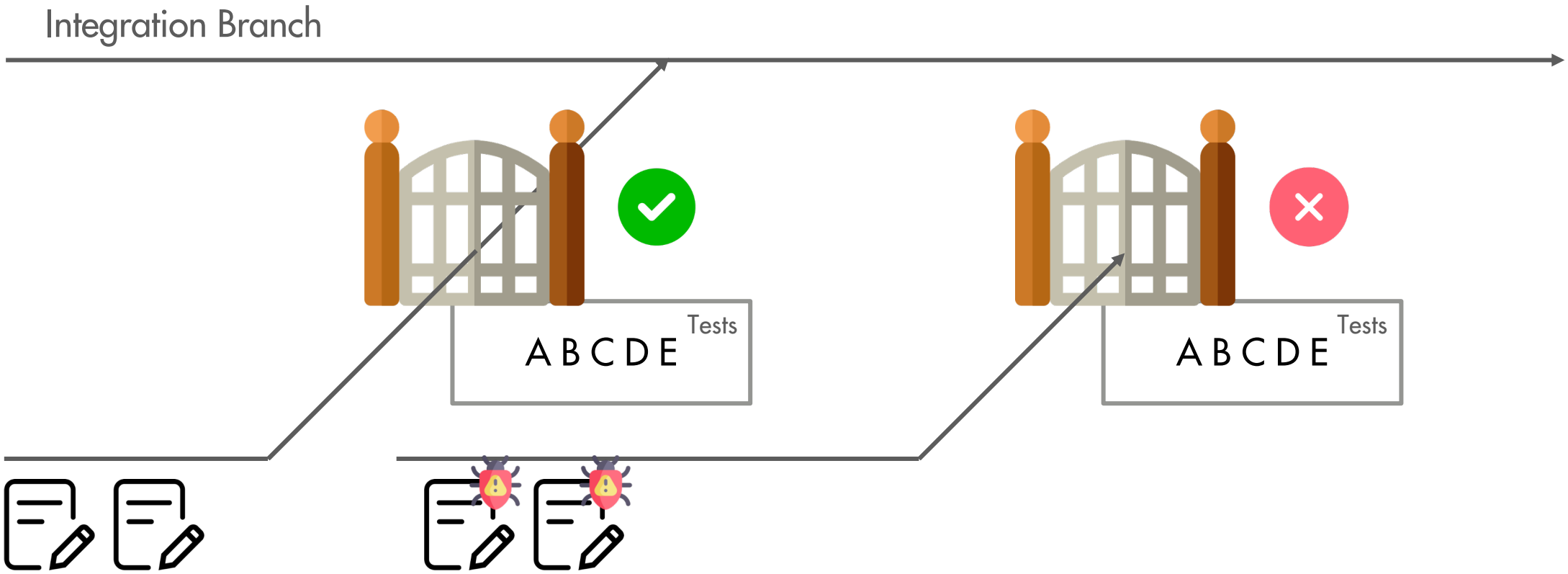
Change-Based Testing

Quality Gate



Expensive
Test Runs

Quality Gate



An Evaluation of Distance Based Test Suite Reduction Techniques

Alessandro Escher
Technical University of Munich
Munich, Germany
alessandro.escher@tum.de

***Abstract*—Efficient test suite selection is crucial in s**
testing due to the high cost of running extensive tests, parti
on large industry projects. Coverage-based techniques
maximize system execution within time constraints bu
suffer from costly and complex coverage recording pr
This study explores alternative selection methods usi
metadata and source code. Hierarchical Agglomerative
tering (HAC) and a greedy approach were evaluated al
distance measures based on package path distance and
representations of test code.

Evaluation on a variety of open-source projects and
industry project revealed that while the proposed m
maintained decent coverage, they did not significantly outp
a strictly time-based selection. We note that HAC lacks
time-budget stopping criterion and performs worse th
greedy approach and random selection. Furthermore, tec
that rely on execution times tend to neglect longer-runnin
which can have an impact on fault detection, particul
industry projects.

This study emphasizes the importance of effective test s
methods that balance coverage, cost, and fault detecti
suggest that a simple yet effective baseline such as
execution time first is a more robust baseline than a r
selection, especially for a cost based evaluation, and un
the need for more competitive baseline methods in te
optimization research.

***Index Terms*—test selection, test suite reduction, clu**
code embeddings, topic model

I. Introduction

Software testing is an integral part of the softwa
velopment lifecycle of any application. In order to v
that the program works as intended and provides the r
functionality, a suite of tests is run—each focusing on di
components of the system and at differing granularit
various points in time before the software is released. F
sion testing is a popular approach for this. The test suite

SCHOOL OF COMPUTATION, INFORMAT
TECHNOLOGY — INFORMATIC

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics

he Us
gs in
nd Se
Esche

SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY —
INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Comparative Analysis of Different Approaches for Test Impact Analyses for Real World Test Suites

Malek Ben Slimane

Evaluating Information Retrieval
the use in Regression Test Selecti

Case Study

Author: Majd Akleh
Supervisors: **Prof. Dr. Ben Hermann**
TU Dortmund
Raphael Nömmer
CQSE GmbH
Date: September 2023

Change-Driven Testing

Sven Amann and Elmar Jürgens

Abstract Today, testers have to test ever larger amounts of software in ever small
periods of time. This makes it infeasible to simply execute entire test suites for eve
change. Also it has become impractical—if it ever was—to manually ensure that th
tests cover all changes. In response to this challenge, we propose Change-Driv
Testing. Change-Driven Testing uses Test-Impact Analysis to automatically find th
relevant tests for any given code change and sort them in a way that increases th
chance of catching mistakes early on. This makes testing more efficient, catchir
over 90% of mistakes in only 2% testing time. Furthermore, Change-Driven Testir
uses Test-Gap Analysis to automatically identify test gaps, i.e., code changes th
lack tests. This enables us to make conscious decisions about where to direct o
limited testing resource to improve our testing effectiveness and notifies us abo
where we are missing regression tests.

Keywords Software testing · Test automation · Test intelligence ·
Regression-test selection · Test prioritization · Test-resource management · Risk
management

1 A Vicious Circle

Today, testers have to test ever larger amounts of software in ever smaller periods
time. This is not only because software systems grow ever larger and more comple
but also because development processes changed fundamentally. Ten years ag
software was commonly released at most once or twice a year and only after
extensive test period of the overall system. Today, we see consecutive feature-driv
releases within a few months, weeks, or even days. To enable this, developme
happens on parallel feature branches, and testing, consequently, needs to happen
each such branch as well as on the overall system.

S. Amann · E. Jürgens
CQSE GmbH, München, Germany

Optimization of Automated and Manual Software Tests in Industrial Practice: A Survey and Historical Analysis

Roman Haas, Raphael Nömmer, Elmar Juergens and Sven Apel

software testing
ntial for project
ource-intensive,
est optimization
est case selection,
pplied to *manual*
manual testing
ifferent domains
ocessed the costs
dustrial practice.
analysis of two
impact analysis
this paper) on
ories.
fault detection
it runtime. For
n average, 80%
as compared to
and an average
dup of the time
m test ordering.
tion techniques
anual testing in
ution time and
process-related
implementation.
st optimization
es the practical
est optimization
systems under
corresponding
ites, no matter

industrial software projects, Haas et al. found that a single execution of a manual test suite takes, on average, 1.5 person months, in extreme cases even up to six person months [6]. The test suites of our industry research partners (see Section III-B) run, on average, a whole work week for automated tests, and, on average, 5 person months for manual tests. They suffer from late feedback and lack of resources to run all tests in reasonable time.

In the literature, there are several techniques for improving test feedback times for long-running test suites. Two common techniques are *test case selection* [7] and *prioritization* [8], on which we focus in this work. While test case selection and prioritization are well-understood for automated tests [9]–[18], the transferability to manual testing is challenging. Inherently, manual tests are conducted less frequently. This leads to substantial code changes between test cycles, and functionality usually tested on the system level, resulting in each test covering a significant portion of code. Beside their different nature, manual tests fail to fulfill prerequisites of existing optimization techniques [6], for example, manual tests may be under-specified, resulting in different execution traces for multiple runs of the same test case. Additionally, execution traces may not be easily separable for different test runs if manual test environments are not isolated. Finally, different test processes, software development and test environments, and test suite run times dictate how aggressive an optimization technique needs to be to provide fast feedback while still keeping the fault detection rate as high as possible.

Research Gap: There is a lack of evidence of the extent to which test optimization techniques for automated testing can be applied to manual testing processes in industrial practice, and what limitations need to be accepted.

Solution: To address this research gap, we, first, analyze

technische universität
dortmund

Master Thesis

Optimization and Evaluation of an Information Retrieval Based Test Selection Approach

Majd Akleh

June 3, 2024

Reviewer:
JProf. Dr.-Ing. Ben Hermann
Dr. Elmar Jürgens

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Obtaining Coverage per Test Case

Florian Dreier

Test Impact Analysis: Detecting Errors Early Despite Large, Long-Running Test Suites

Elmar Jürgens
CQSE GmbH
juergens@cqse.eu

Dennis Pagano
CQSE GmbH
pagano@cqse.eu

Andreas Göb
CQSE GmbH
goeb@cqse.eu

—Large test suites often take a long time to execute.
, in practice, they are usually not executed as part of
s integration (CI), but only less frequently and in later
s. As a result, many errors remain unrecognized during
re found late, which causes high costs.
pact Analysis allows us to run only those tests affected
hanges since the last test run. As a result, it is possible
xecute that section of a large test suite during CI that
ikely to find new errors. In our studies, we were able
0% of failed builds in 2% of the test execution time.
bles fast CI feedback with high error detection rates,
s of the size and duration of the entire test suite.

I. Motivation

tware grows, its test suite must also grow in order
y detect bugs. But this also increases the total test
time. In practice, we are increasingly seeing test
t run for several hours or even days.
experience, however, long-running test suites are
excluded from continuous integration (CI). Often,
executed only in subsequent test phases that take
g., before each release. Due to the less frequent
of the test suite, however, the period between error
ion and error detection grows. This has extensive
consequences:
rs can overlap each other and can only be detected
other errors have been corrected.
gging regression errors becomes more complex, as
ncreasing number of code changes may have caused
error.
fixes take longer because developers have to invest
e time understanding their own code if an error is
overed weeks after it has been introduced.
effects increase as systems and test suites grow.
, this jeopardizes the effectiveness of continuous
n, especially for large systems. These often have
arly high strategic value, and hence short feedback
ould be particularly important. How can we prevent
quickly find new bugs despite longer execution times
st suites in continuous integration?
est Impact Analysis (TIA) approach addresses this
by selecting and prioritizing test cases based on code
This is done by analyzing the code changes that have
de since the last successful test run. Then, only those
s are selected that run through changed code. These

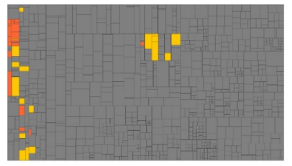


Figure 1. Changes for implementing a feature: Modified code is shown in yellow, new code is red, and unchanged code is gray.

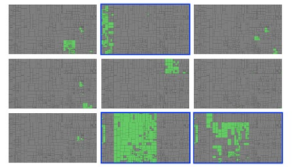
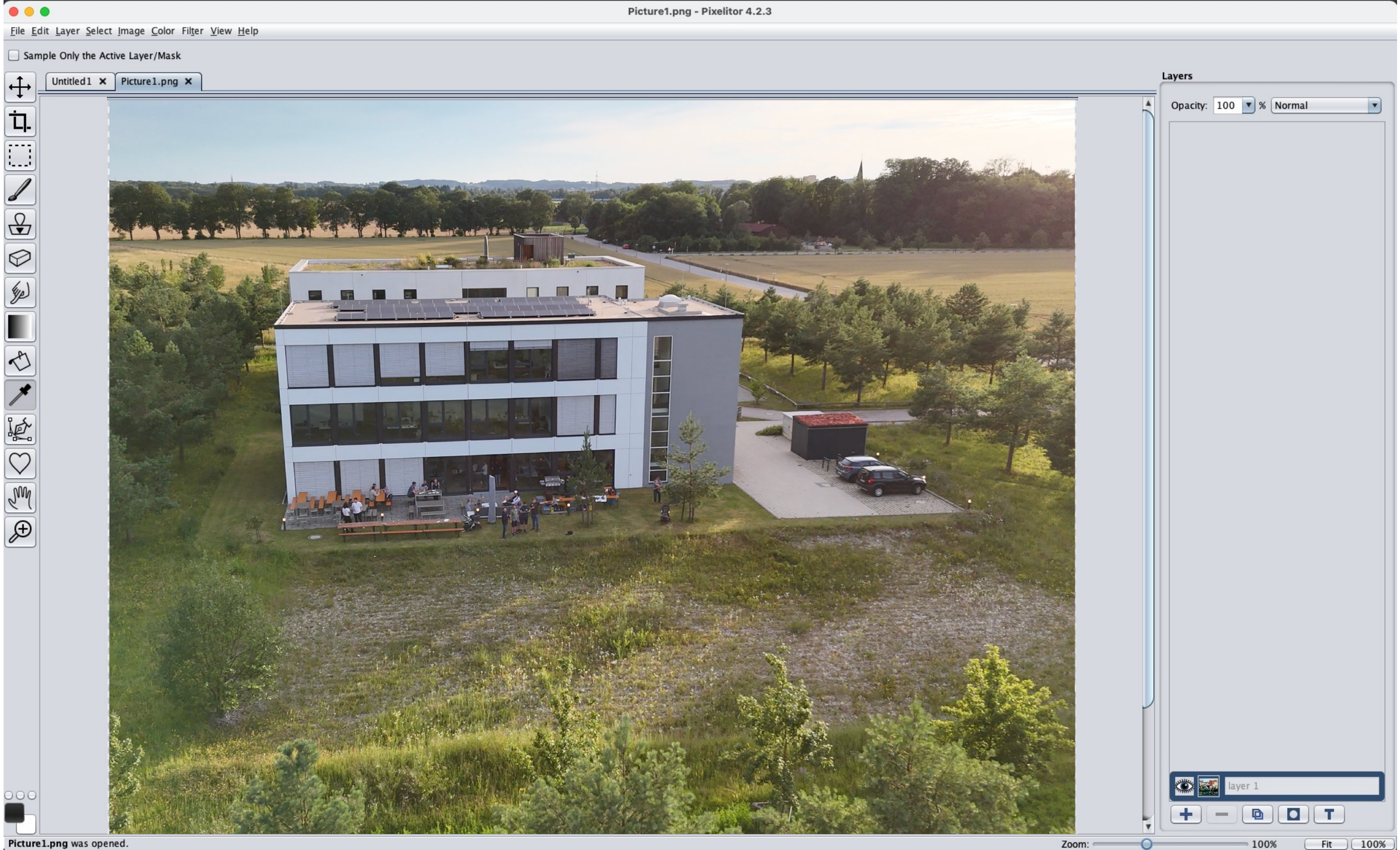


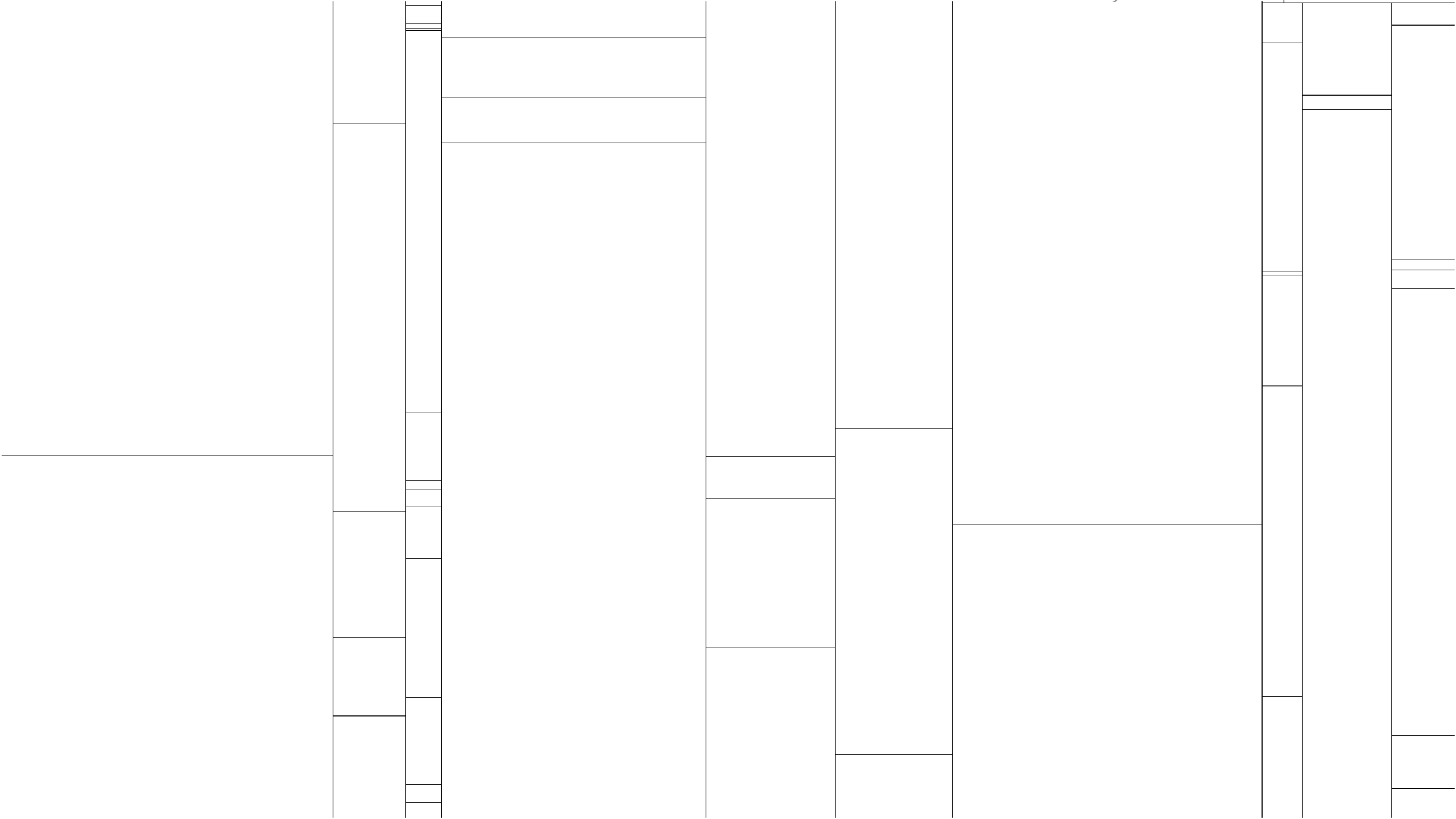
Figure 2. Test coverage of nine automated tests. If a test executes a method, it is shown in green, otherwise in gray.

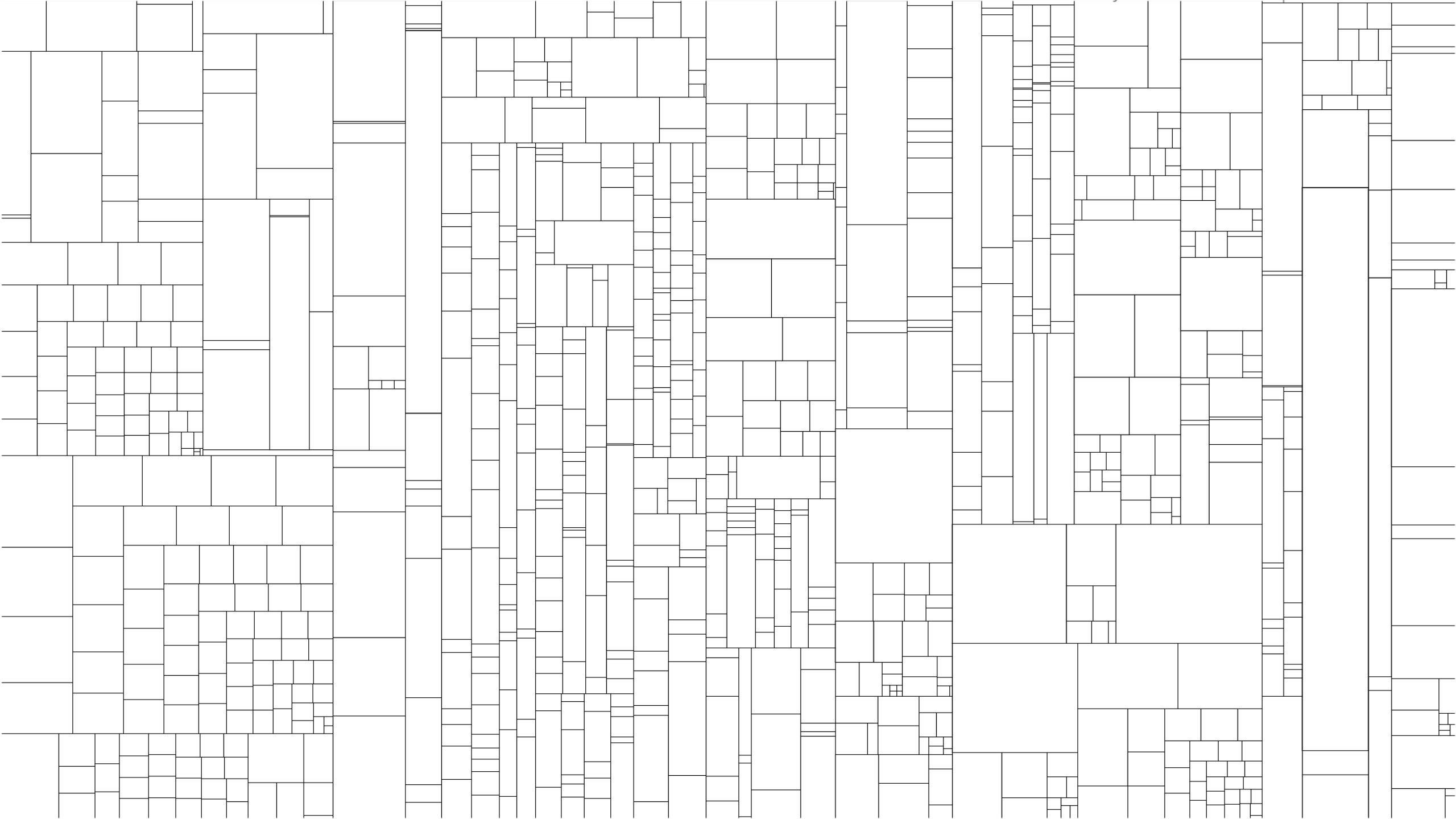
tests are then sorted so that they find errors as early as possible in the test run. The more frequently TIA is performed, the less changes are made between two test runs, and the greater the expected execution time savings are.

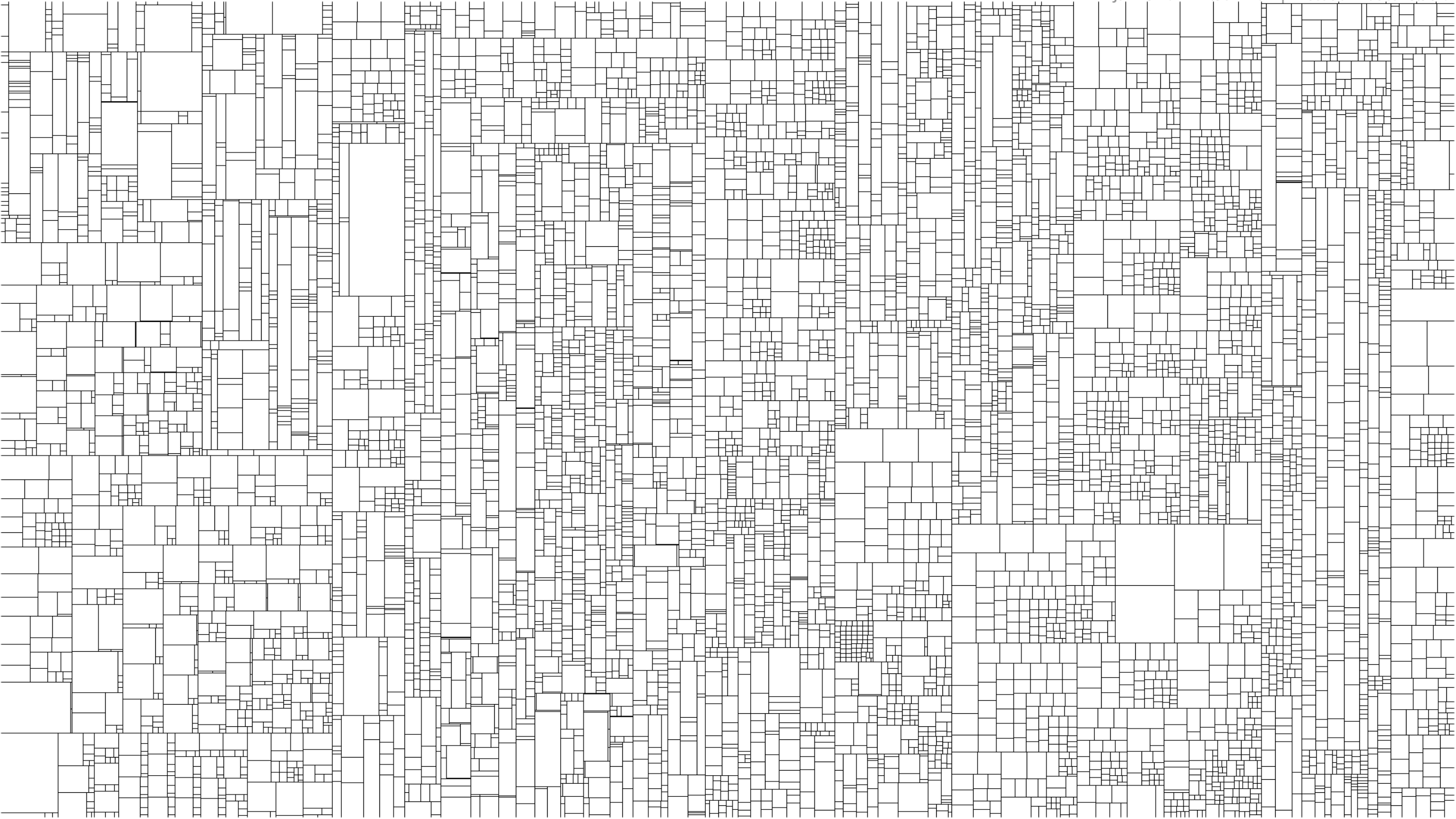
Figures 1 and 2 show an example from our own development. Each of the illustrations shows a treemap. Each small rectangle in the treemap represents a method in the source code of the *system under test*. Figure 1 shows the changes that were made as part of the development of a feature. Unchanged code is gray, modified code is orange, and new code is red. Figure 2 shows the code coverage of 9 automated test cases (unit tests and integration tests). If a test case executes a method, it is

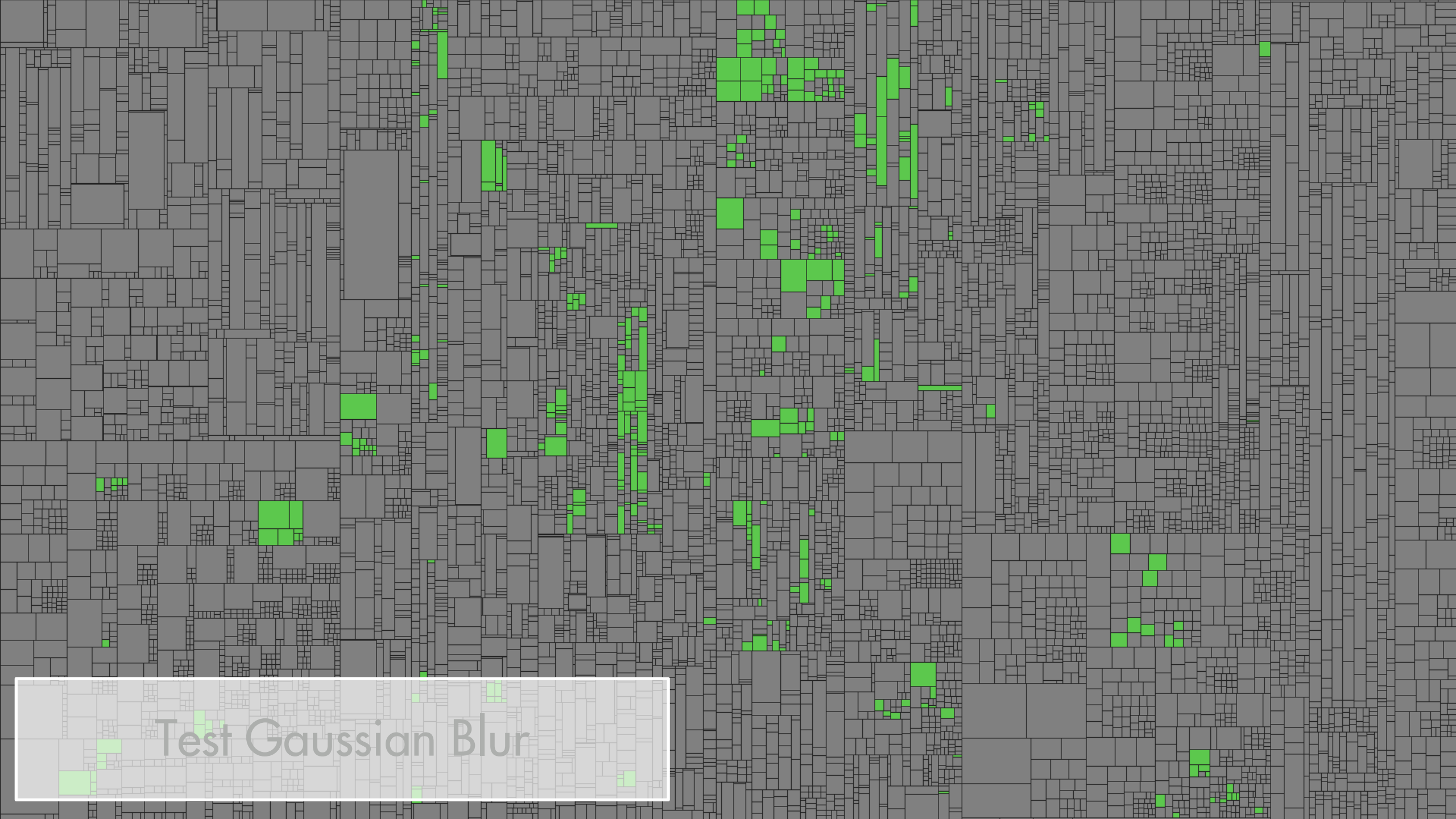
Highly Redundant

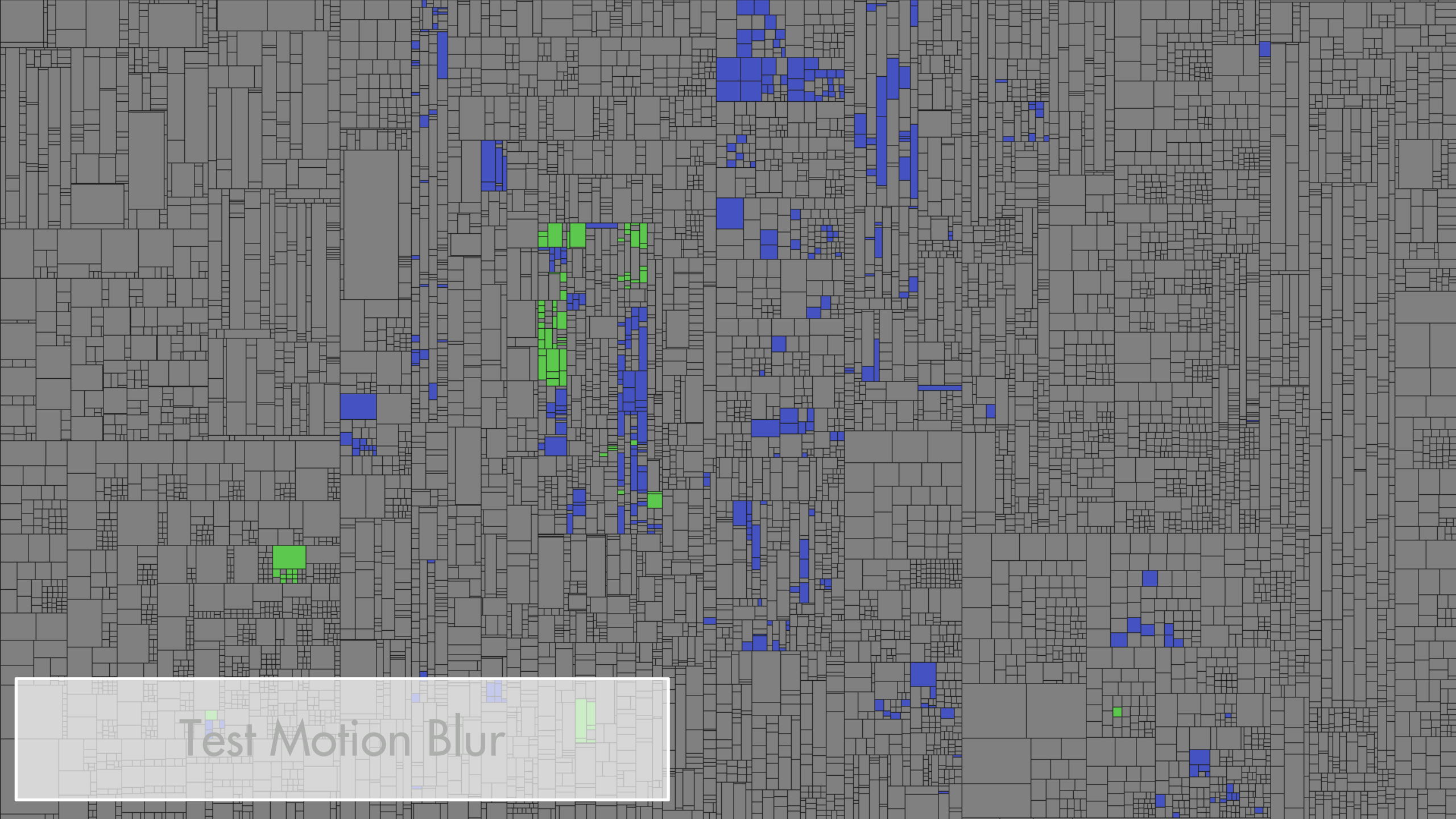

Picture1.png - Pixelitor 4.2.3
File Edit Layer Select Image Color Filter View Help
Sample Only the Active Layer/Mask
Untitled1
Picture1.png
Layers
Opacity: 100 % Normal
layer 1
Picture1.png was opened.
Zoom: 100%
Fit
100%

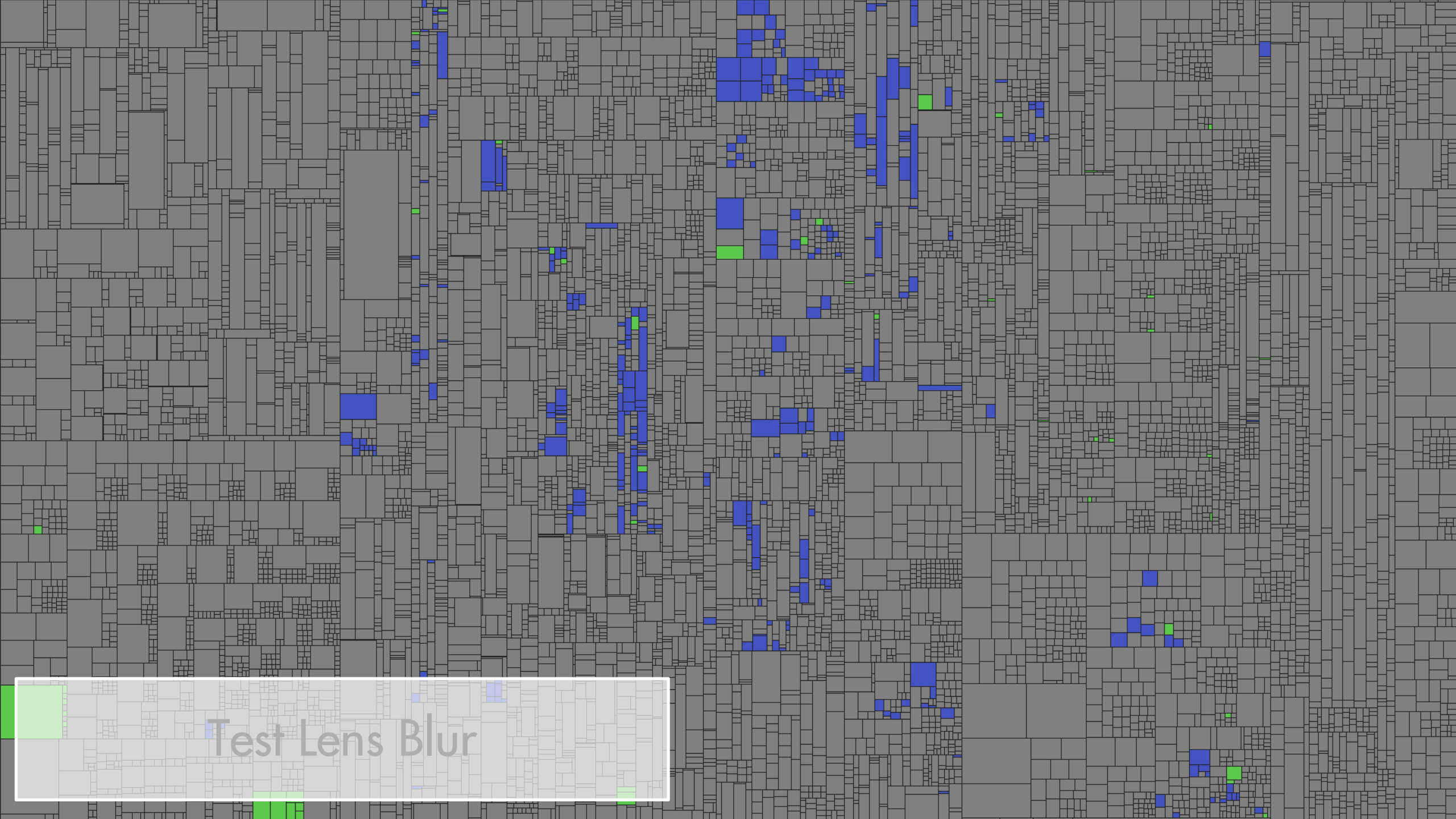




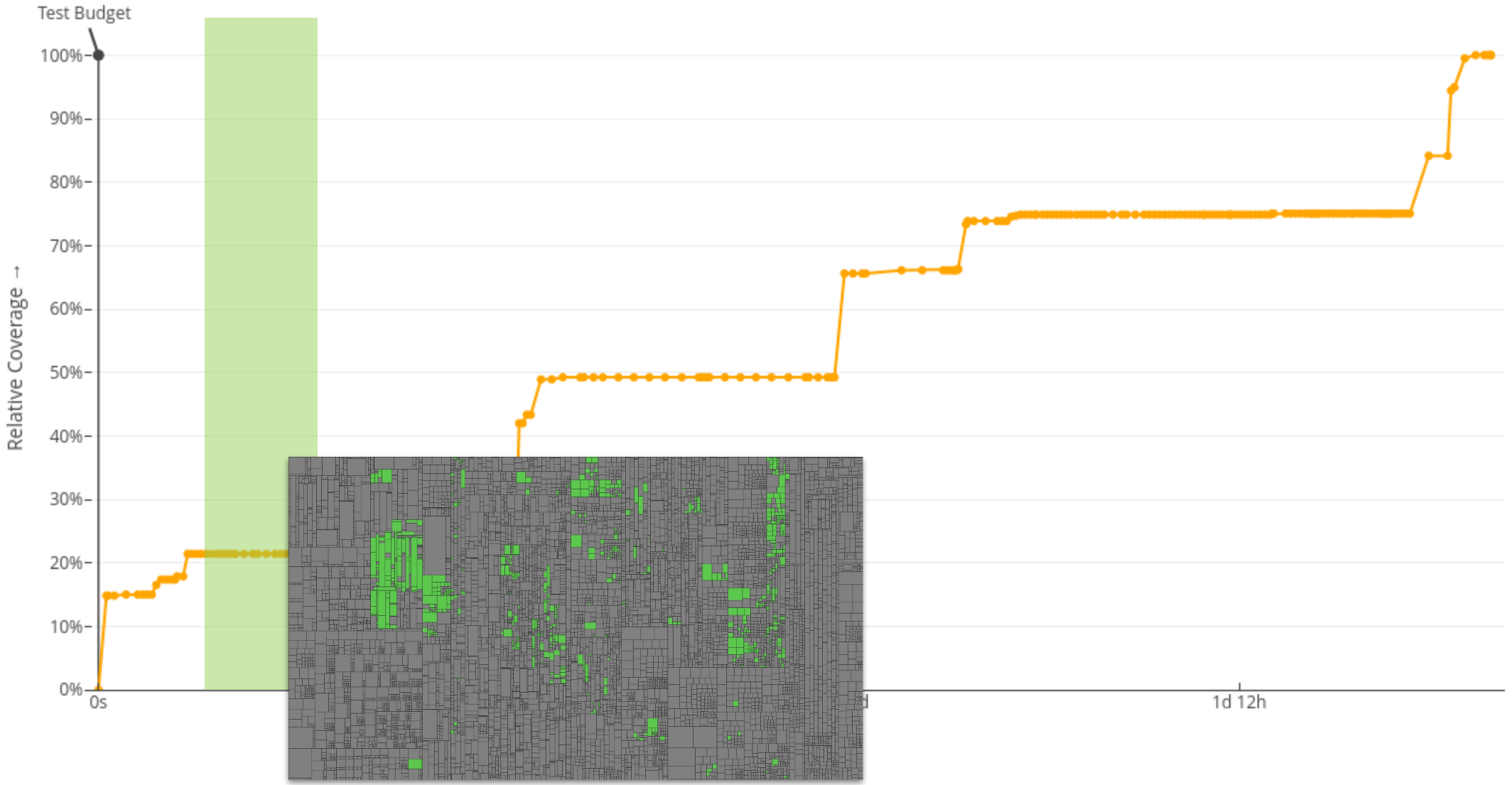


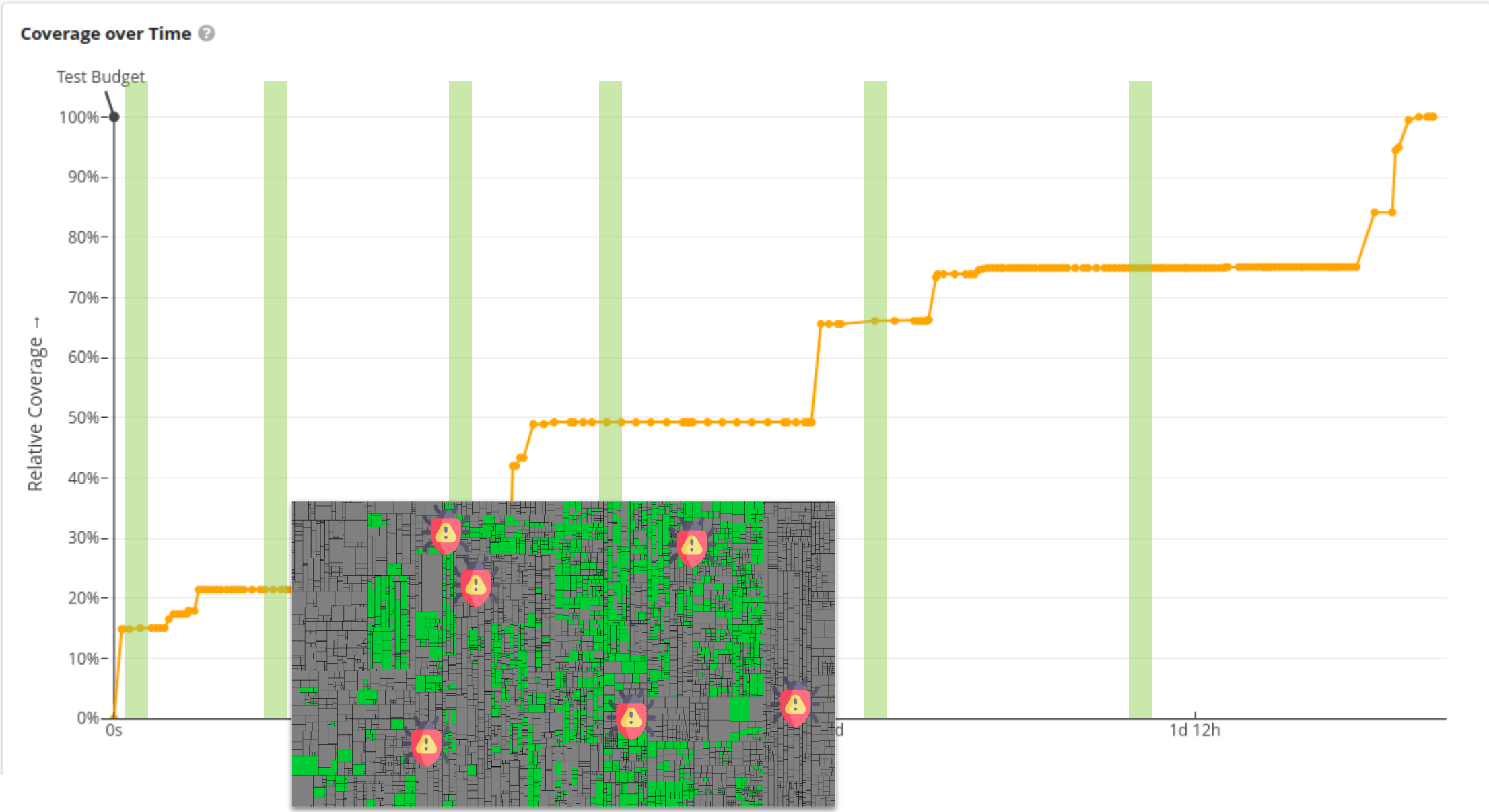

Test Gaussian Blur


Test Motion Blur


Test Lens Blur

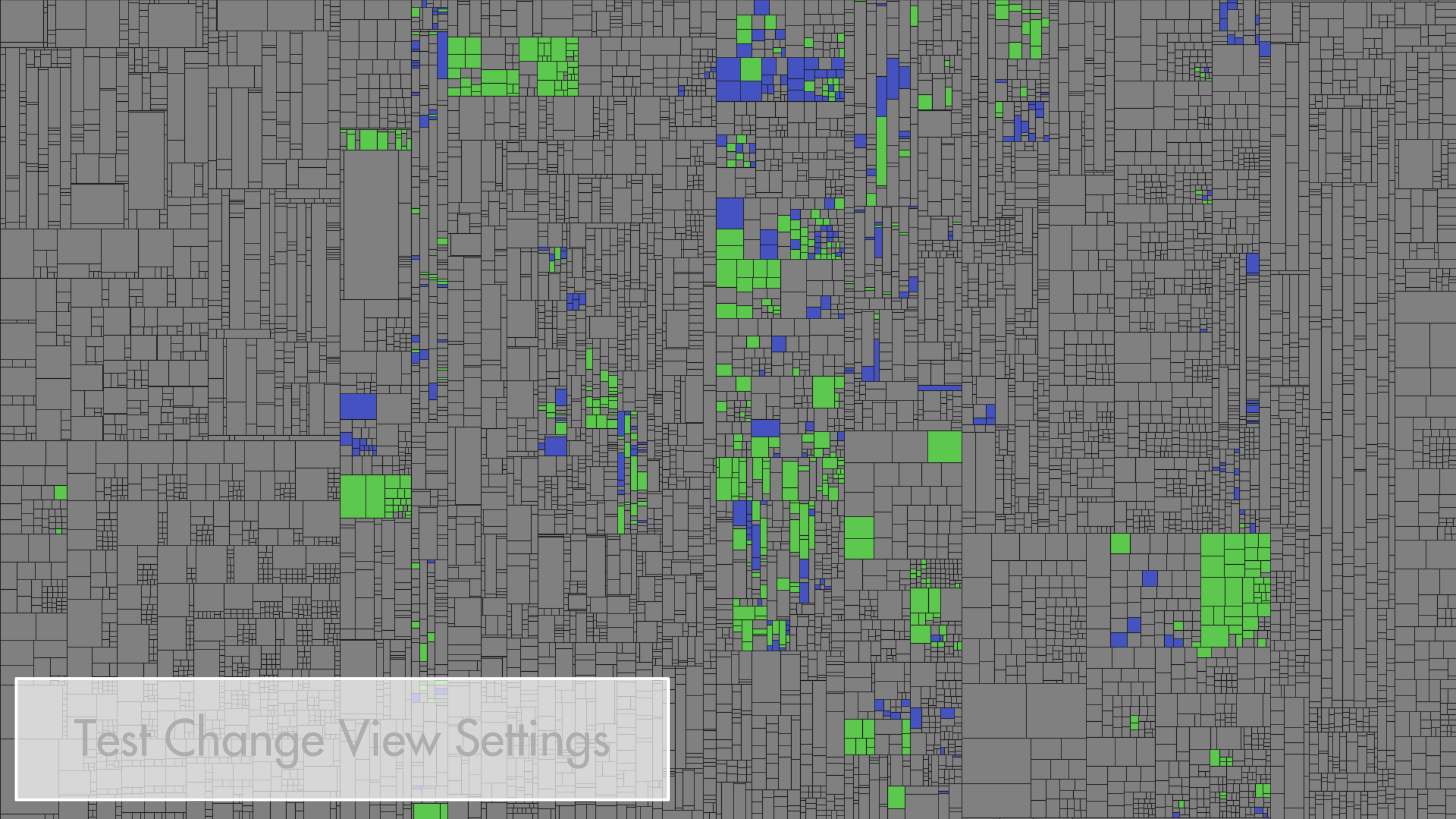
Coverage over Time

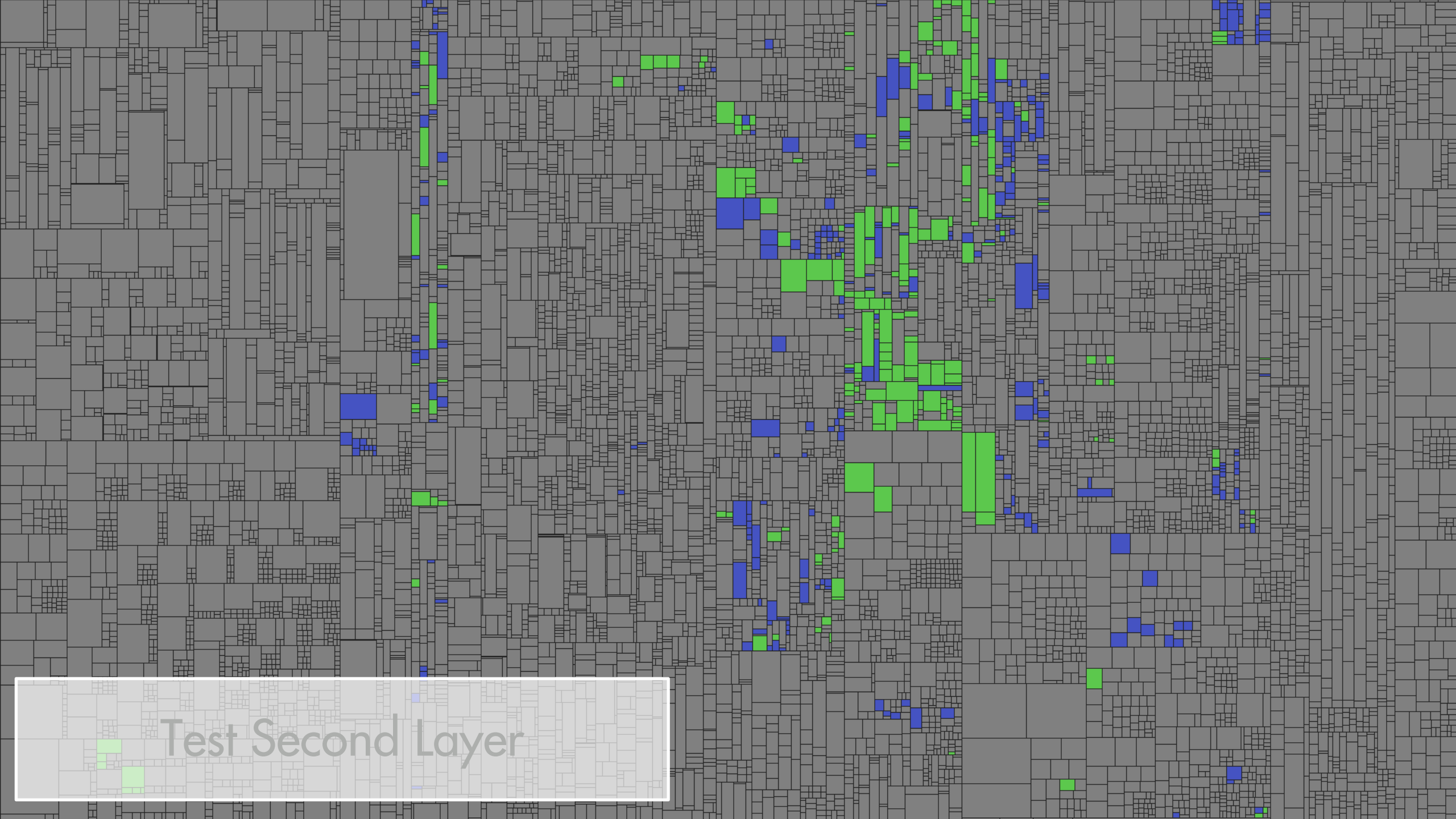




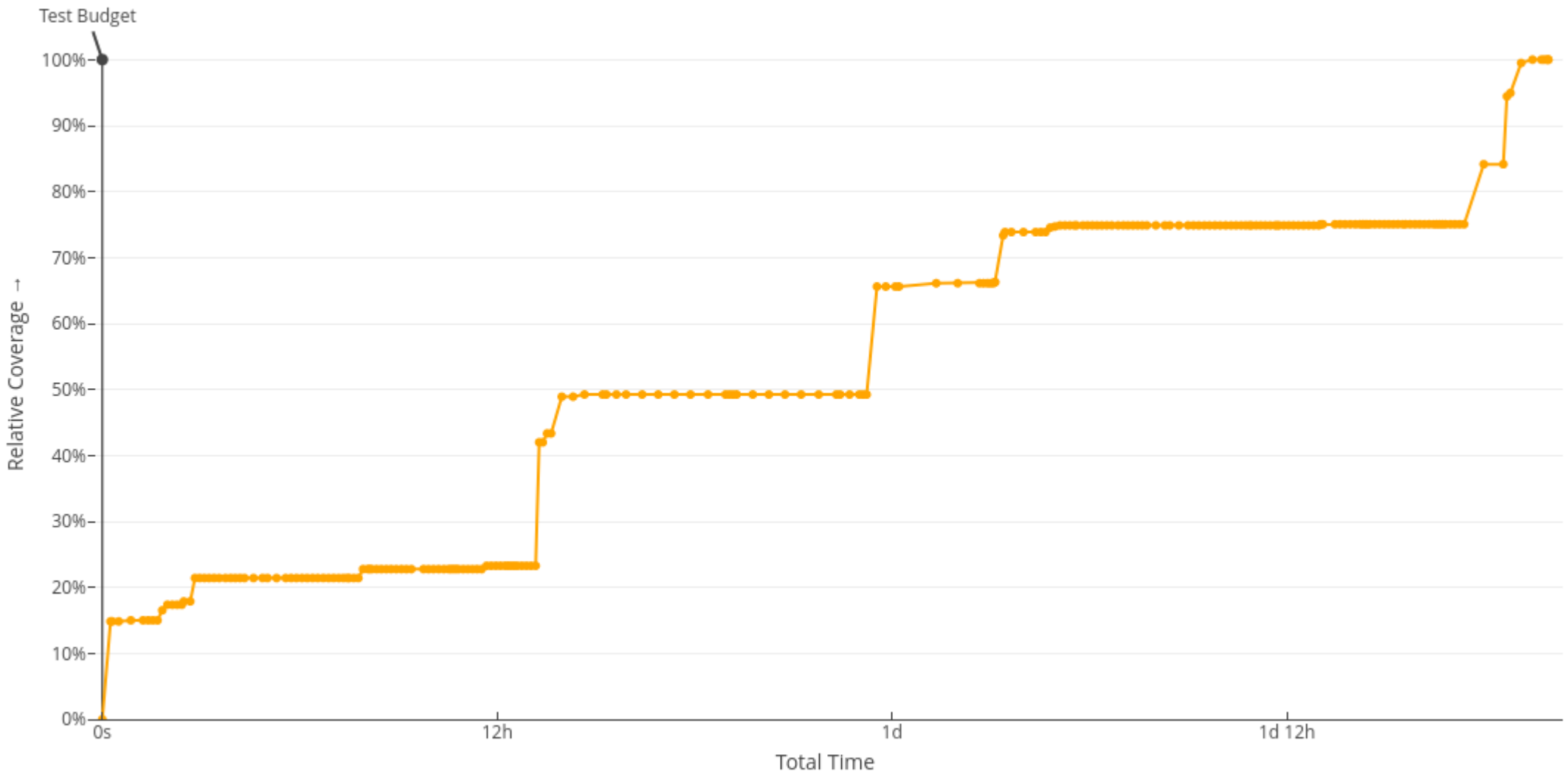
Idea: Select the most dissimilar tests


Test Create and Modify
Selection


Test Change View Settings

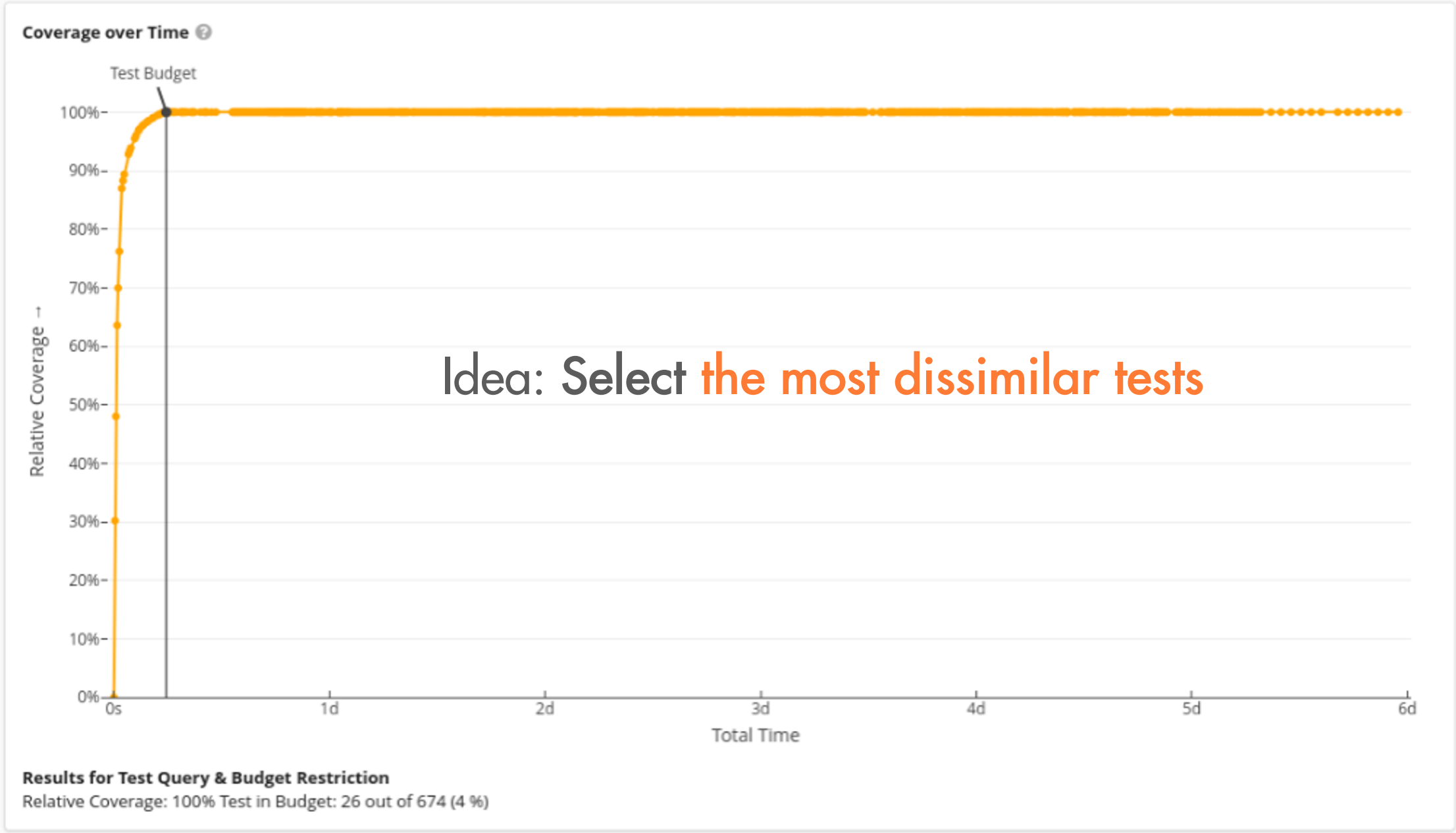

Test Second Layer

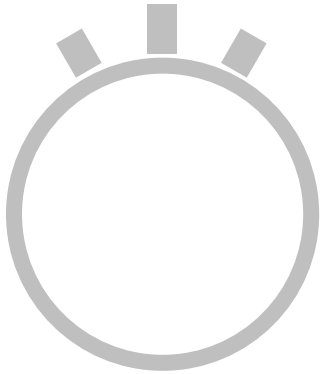
Coverage over Time

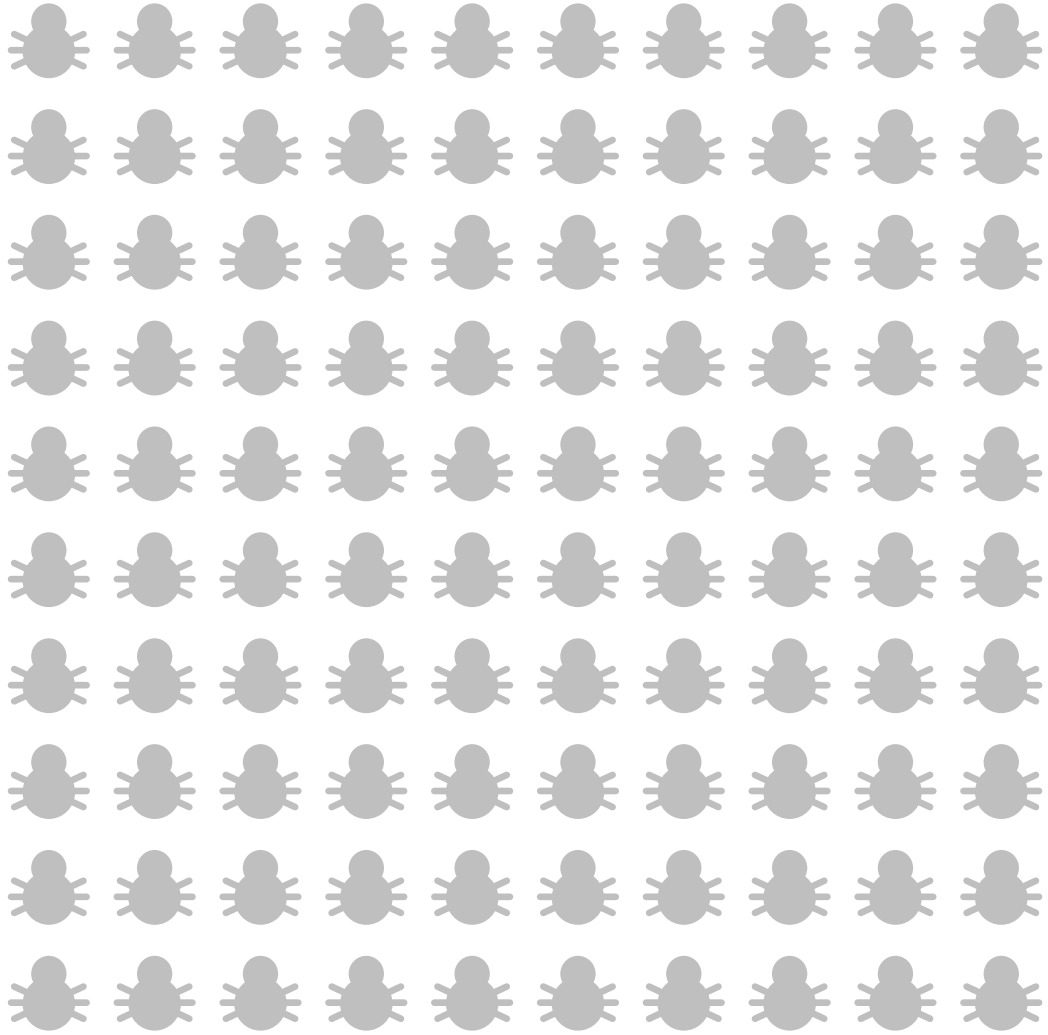


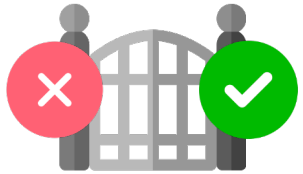
Results for Test Query & Budget Restriction

Relative Coverage: 0%, Selected Tests: 0 out of 236 (0%)

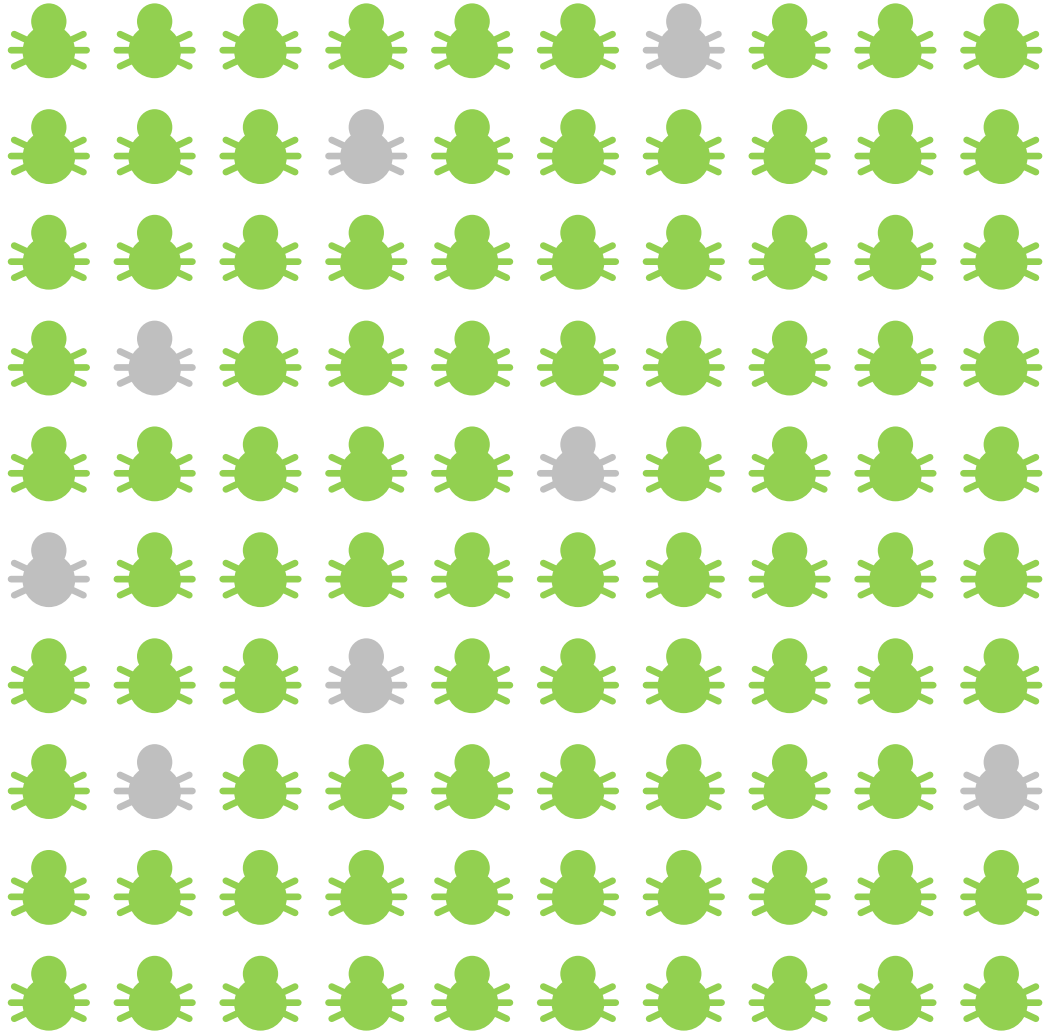

Coverage over Time
Test Budget
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Relative Coverage ↑
0s
1d
2d
3d
4d
5d
6d
Total Time
Idea: Select the most dissimilar tests
Results for Test Query & Budget Restriction
Relative Coverage: 100% Test in Budget: 26 out of 674 (4 %)







Pareto Optimization



Demo

www.teamscale.com

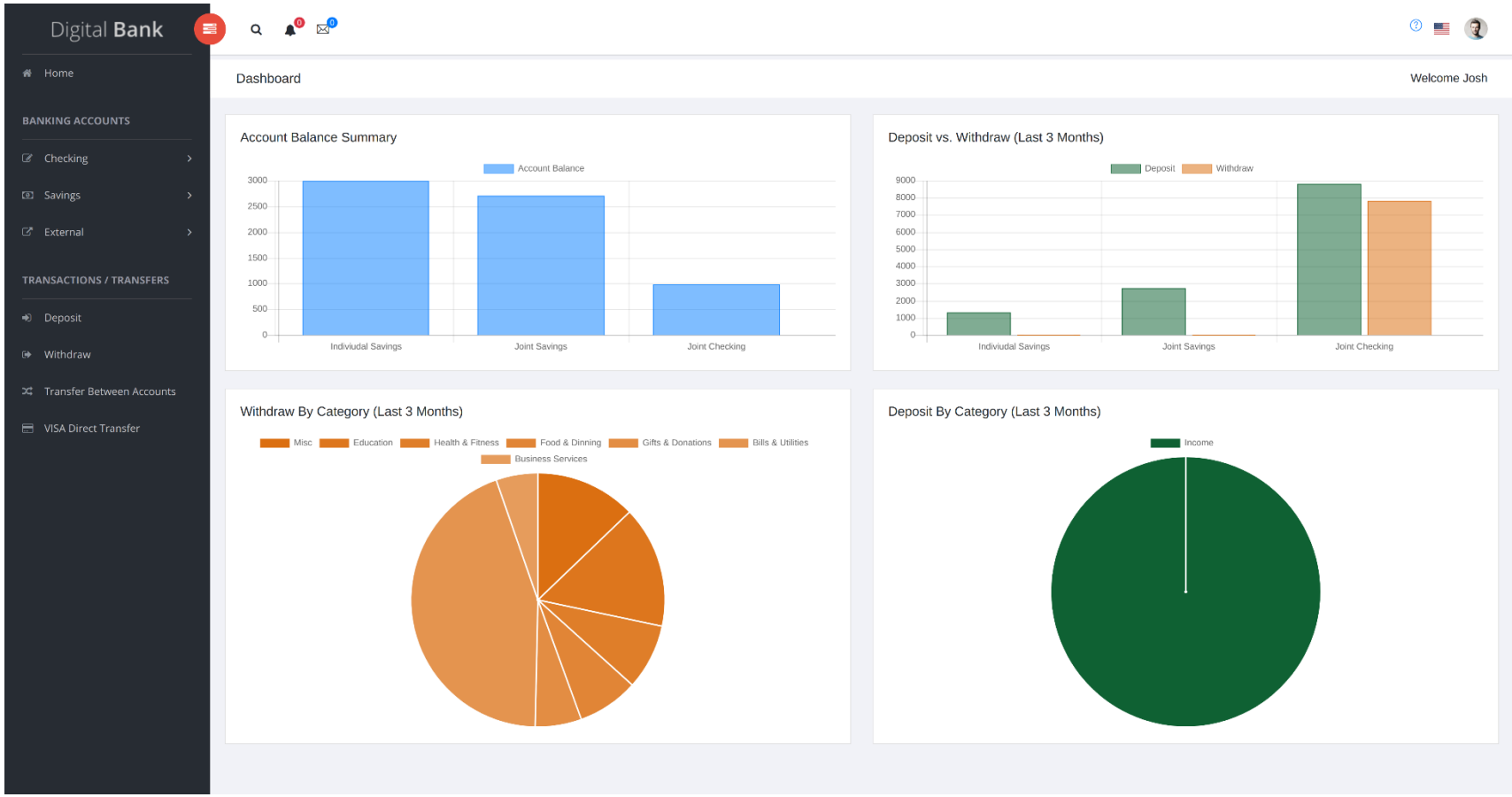
Ramona Kühn

Test Intelligence Team

Onboarding new customers

CQSE Teamscale

Digital Bank: Digital Banking Application (Java)



digisic/digitalbank-gen-one

ABAP

Ada

C#

C/C++

Cobol

Delphi

ESQL

Fortran

Go

Gosu

Groovy

HANA SQLScript / View

IEC 61131-3 ST

Java

JavaScript / TypeScript

Kotlin

Matlab

Objective-C

Open CL

OScript

PHP

PL/SQL

Python

Simulink / StateFlow

Swift

Transact-SQL

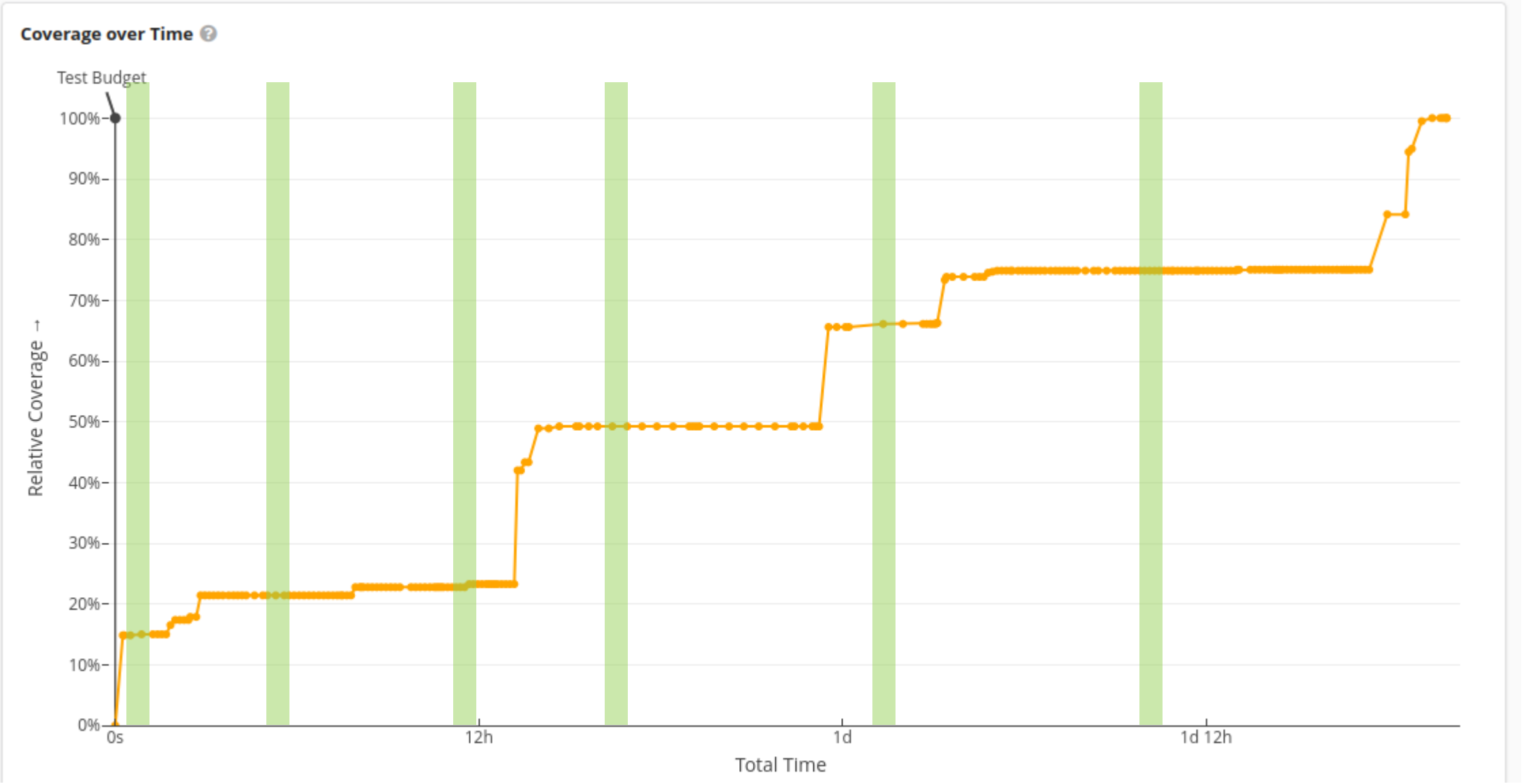
Visual Basic .NET

Xtend

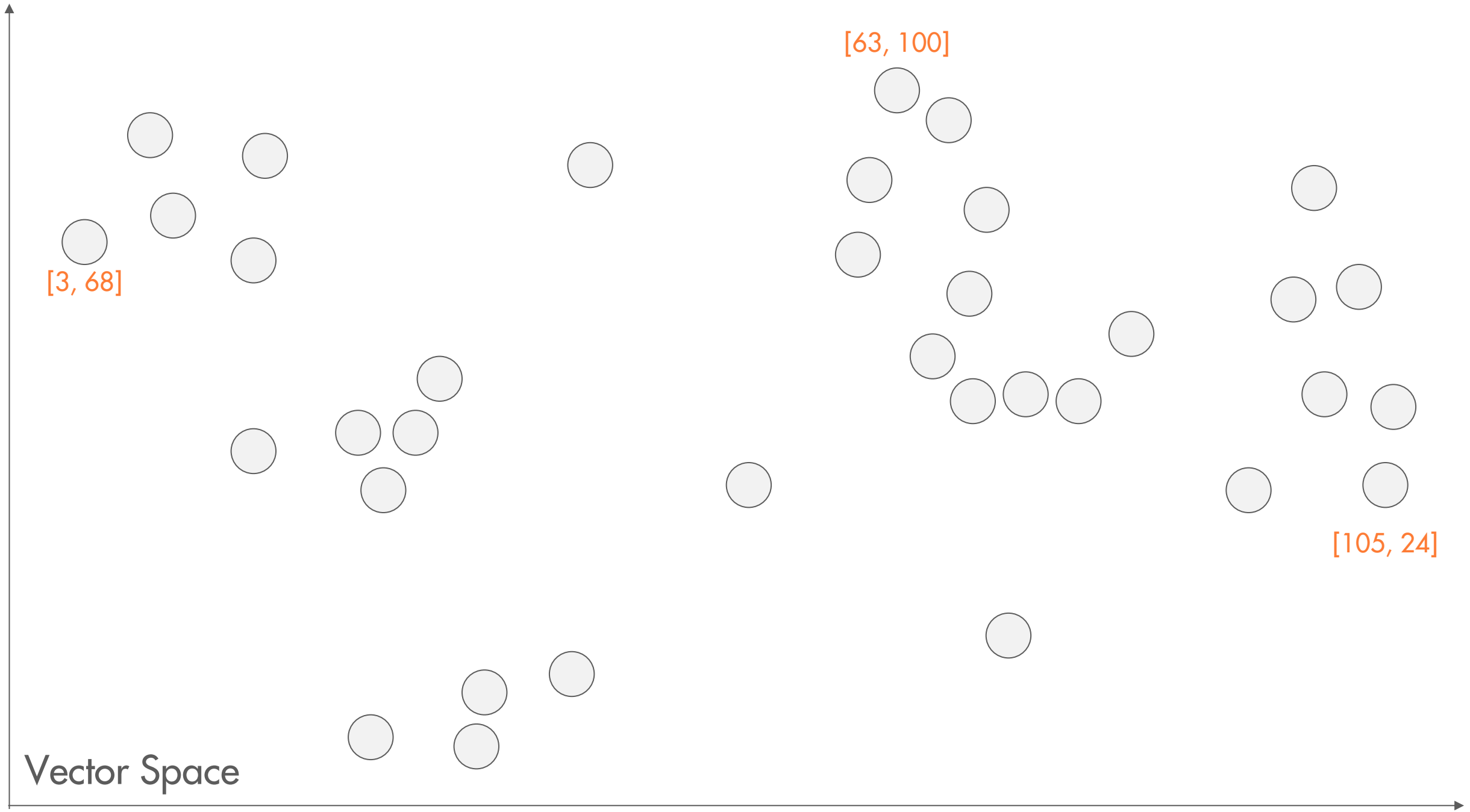
Disadvantages of Coverage-Based Approaches

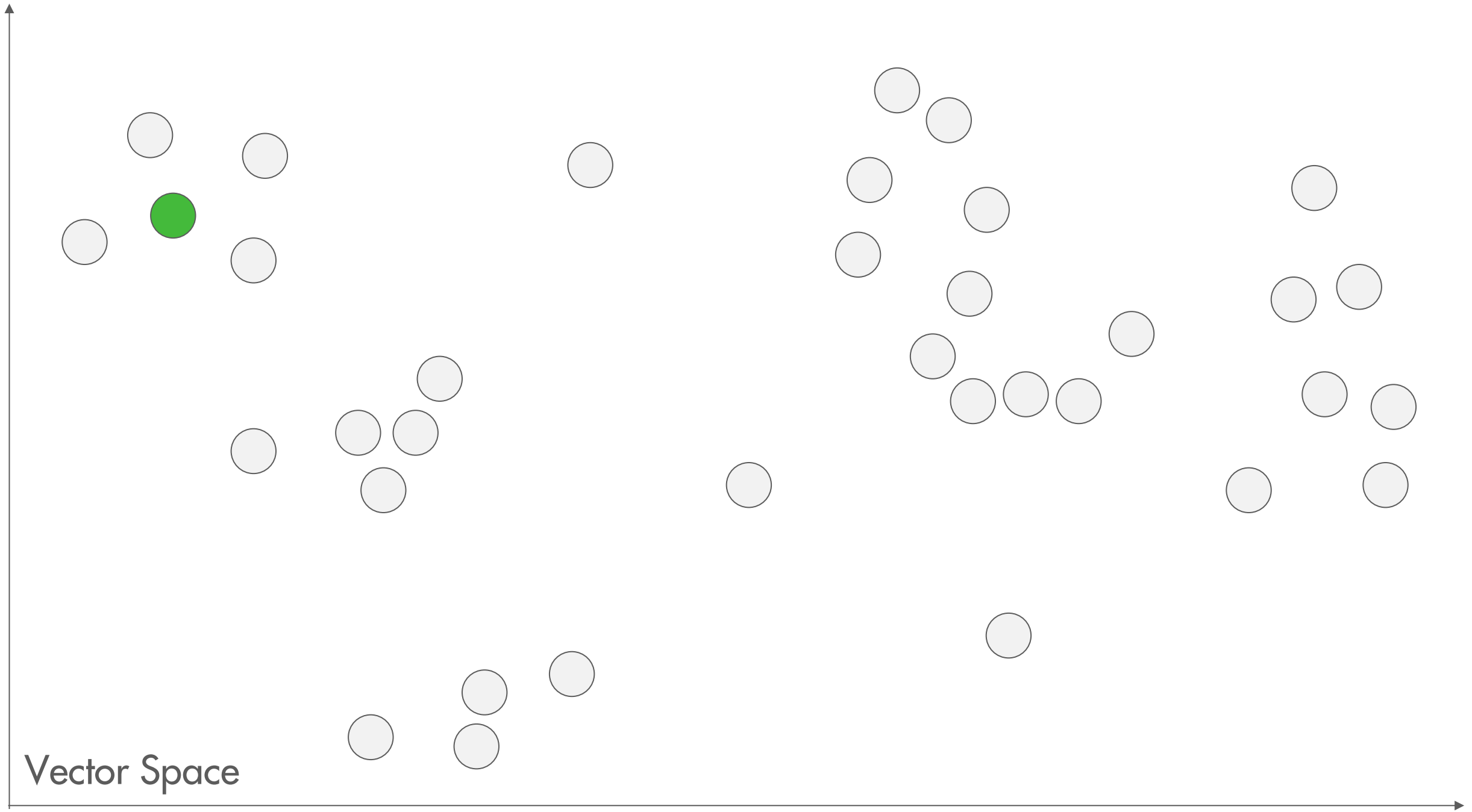
High Setup and Maintenance Effort

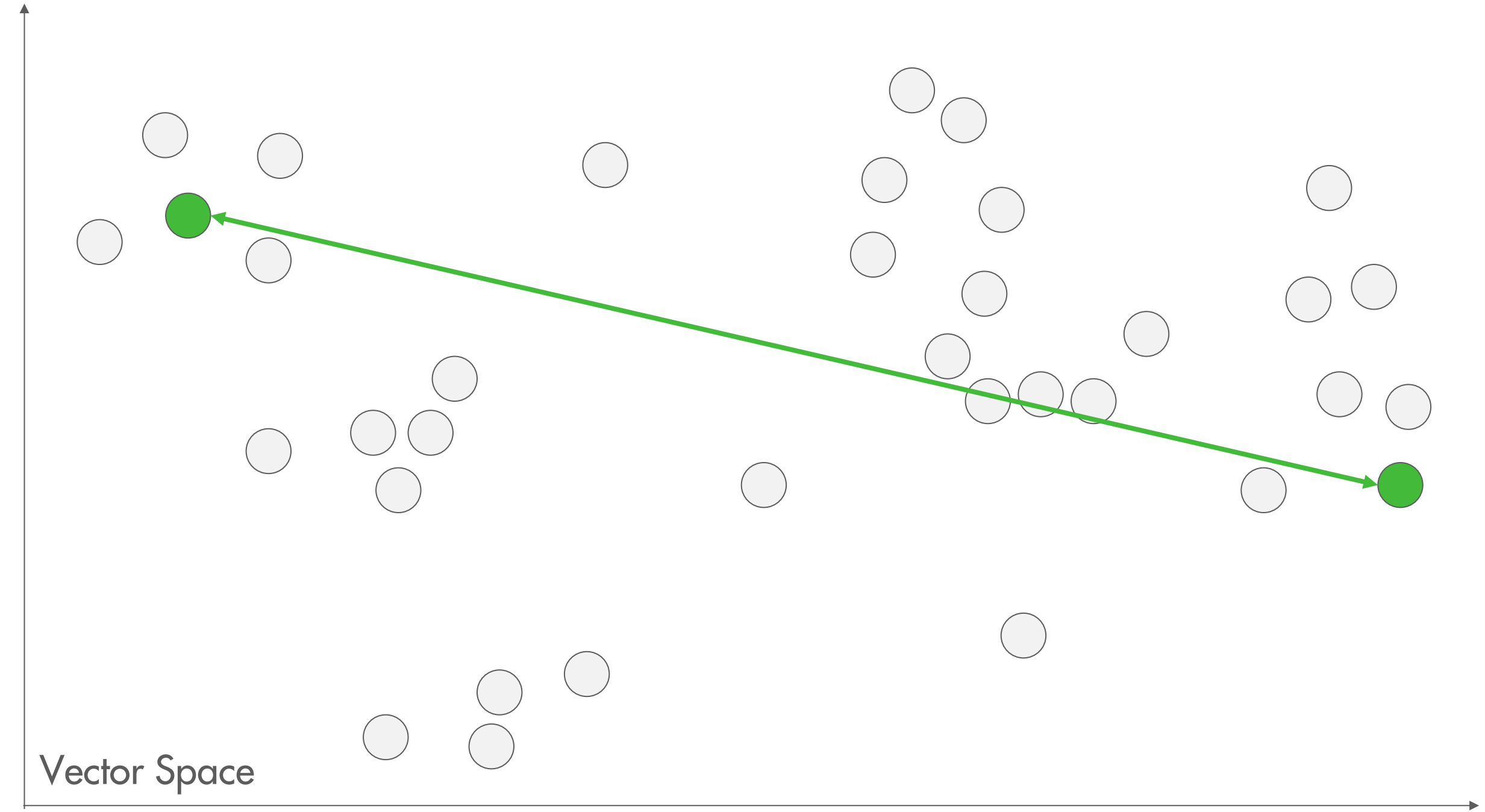
High Complexity

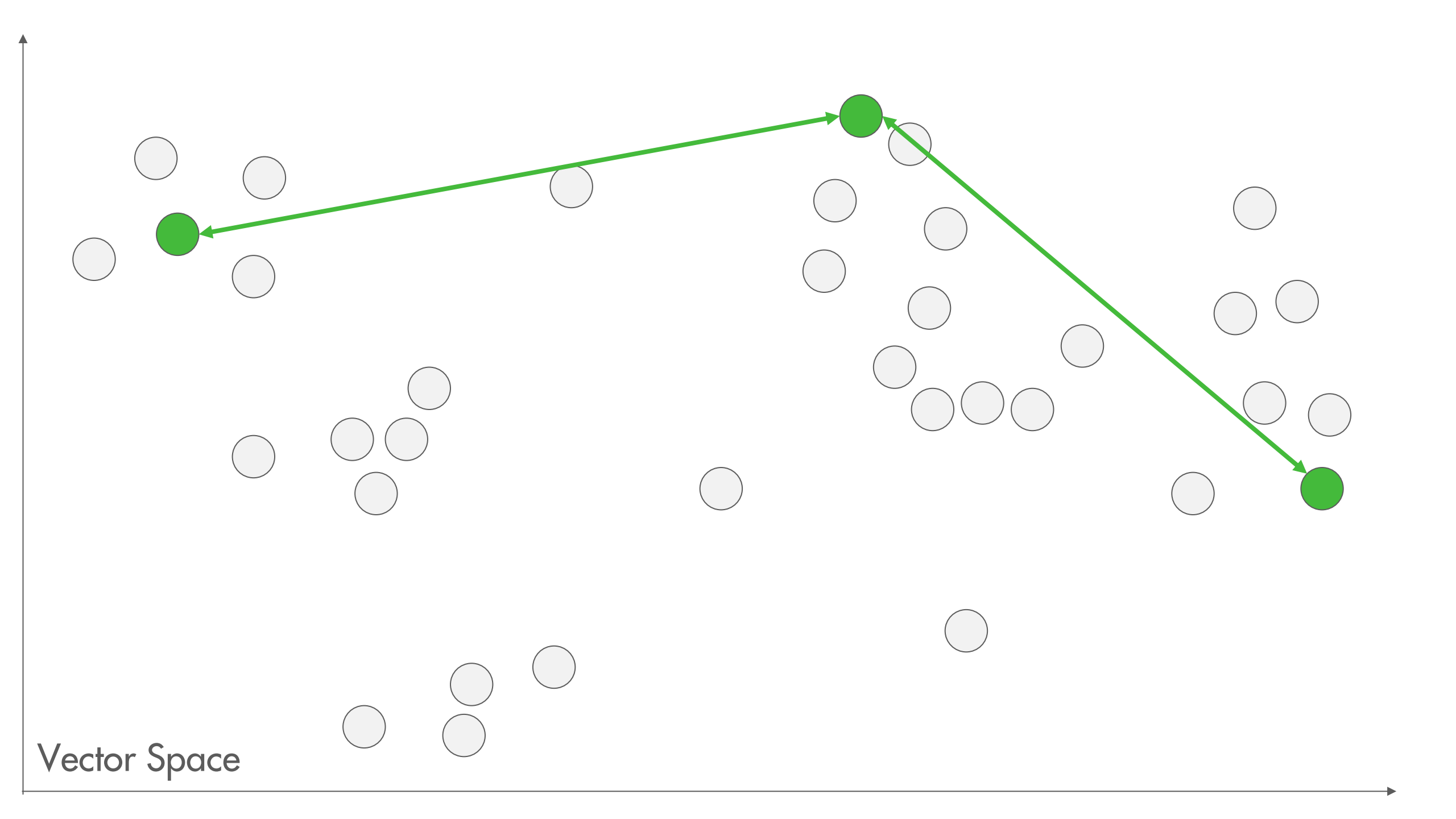


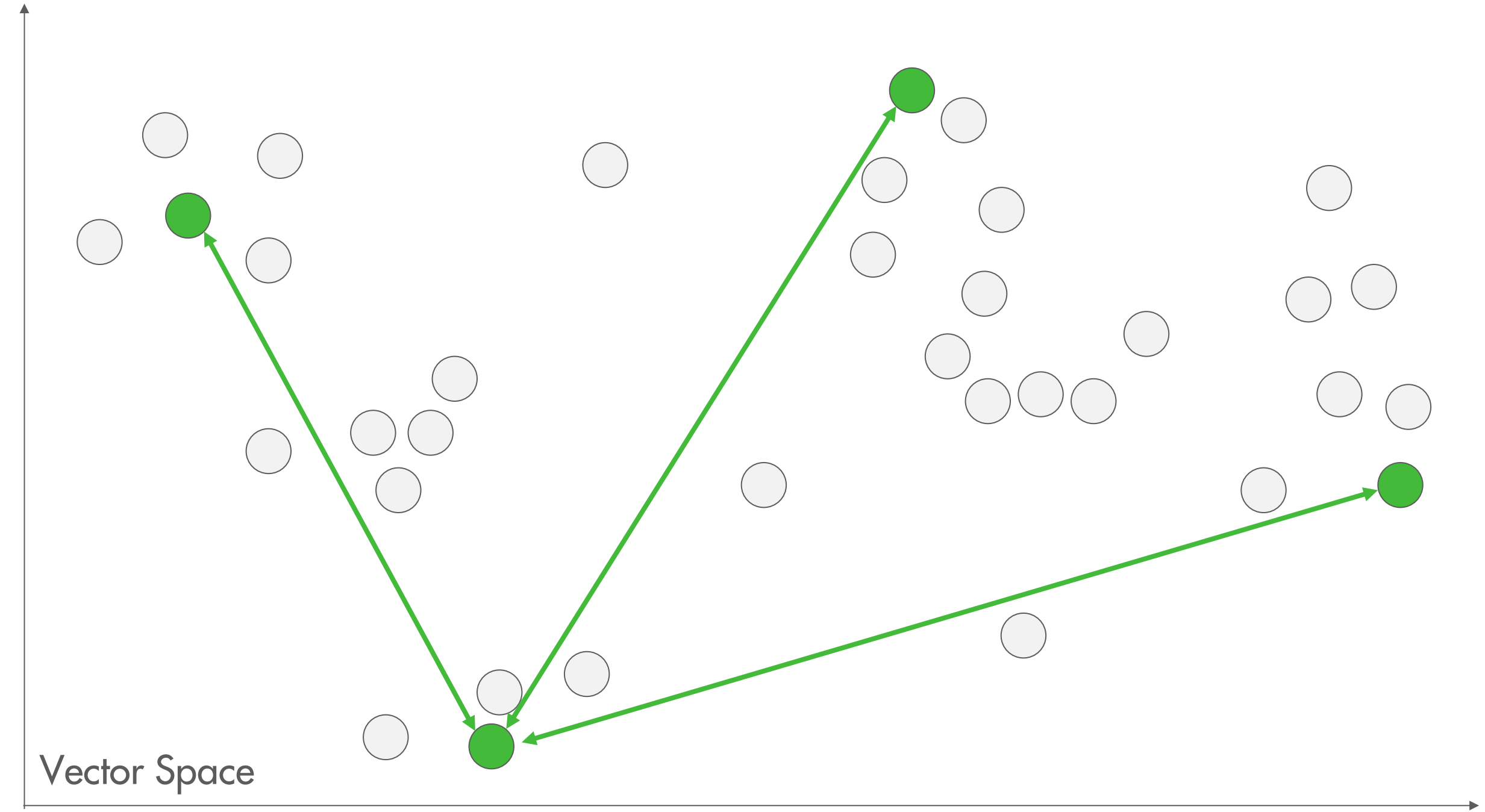
Idea: Select the most dissimilar tests


[63, 100]
[3, 68]
[105, 24]
Vector Space


Vector Space


Vector Space


Vector Space

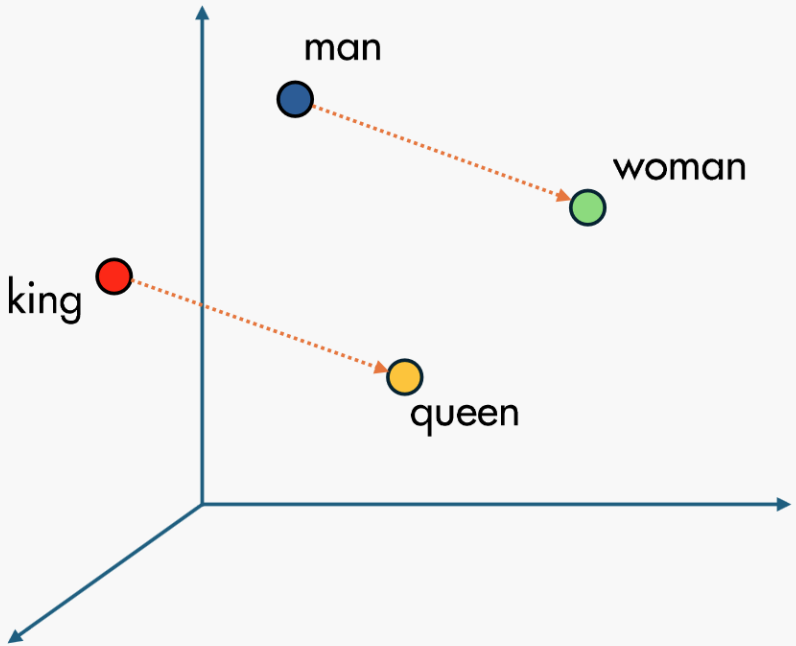

Vector Space

Large Language Models (AI)

	living being	feline	human	gender	royal	verb	plural
man	-0.05	-0.31	0.18	-0.28	-0.24	0.11	-0.16
woman	-0.14	-0.20	0.30	-0.48	-0.04	0.14	-0.04
king	0.11	-0.12	0.10	-0.38	-0.06	0.08	-0.14
queen	-0.01	-0.31	0.23	-0.43	-0.17	0.10	0.04

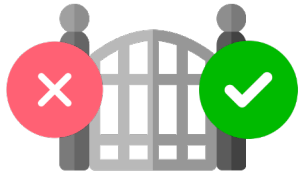
Word

Embedding

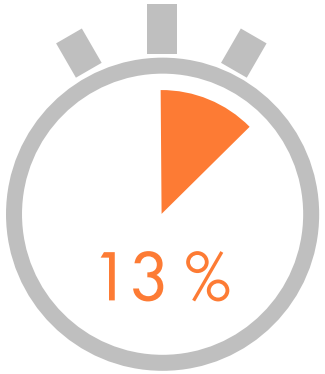


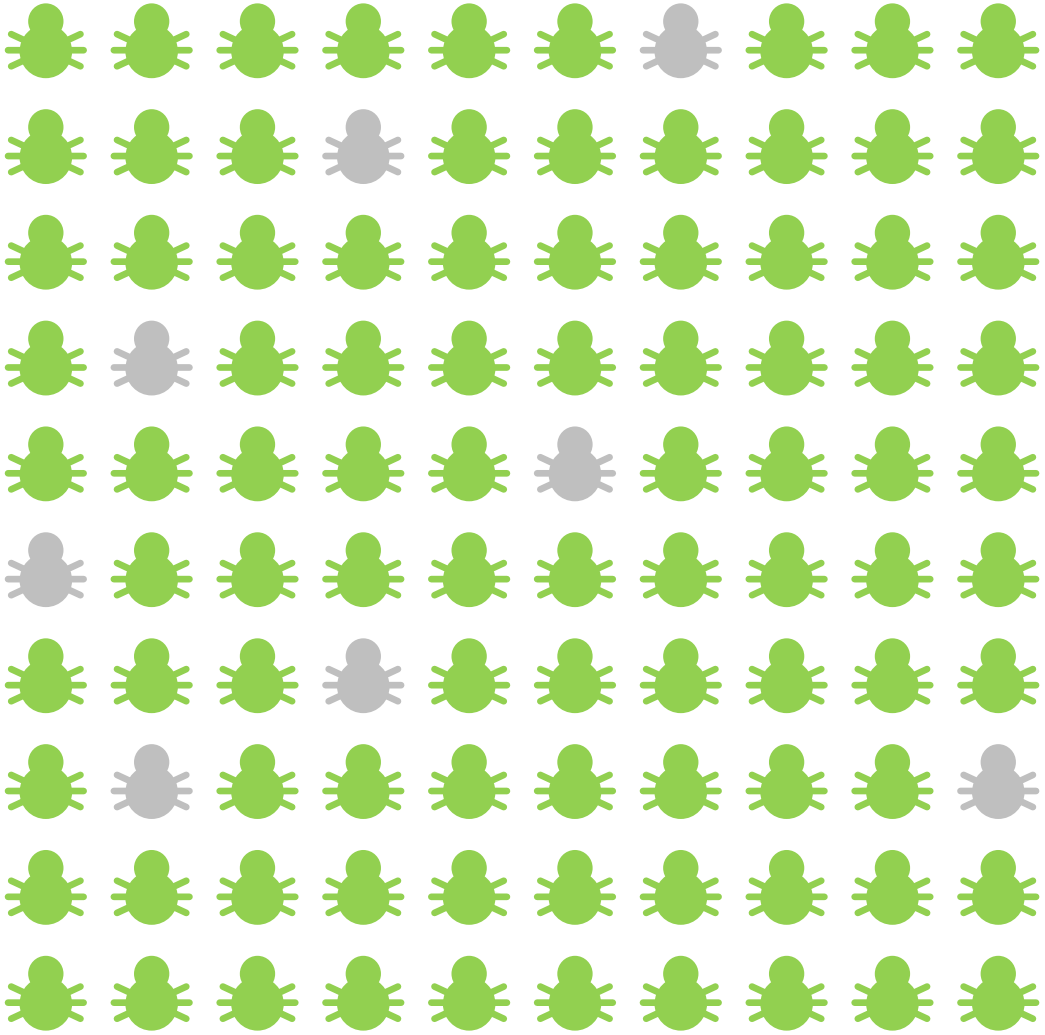
Visualization

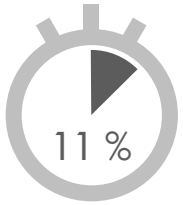

manual
Test Steps
automated
Test Code
AI Test Clustering
A B C D E
Sorted Tests



AI Test Clustering

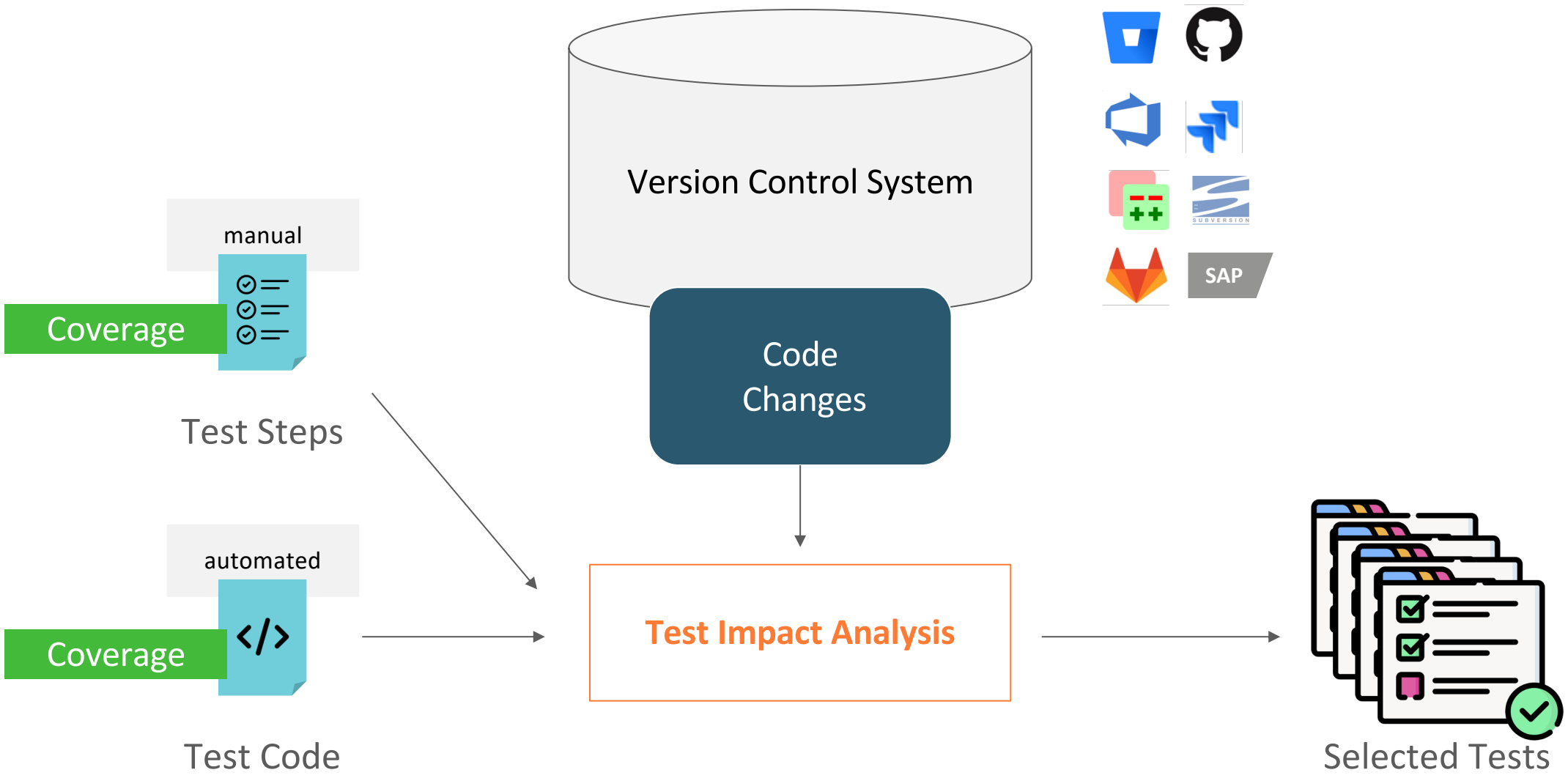


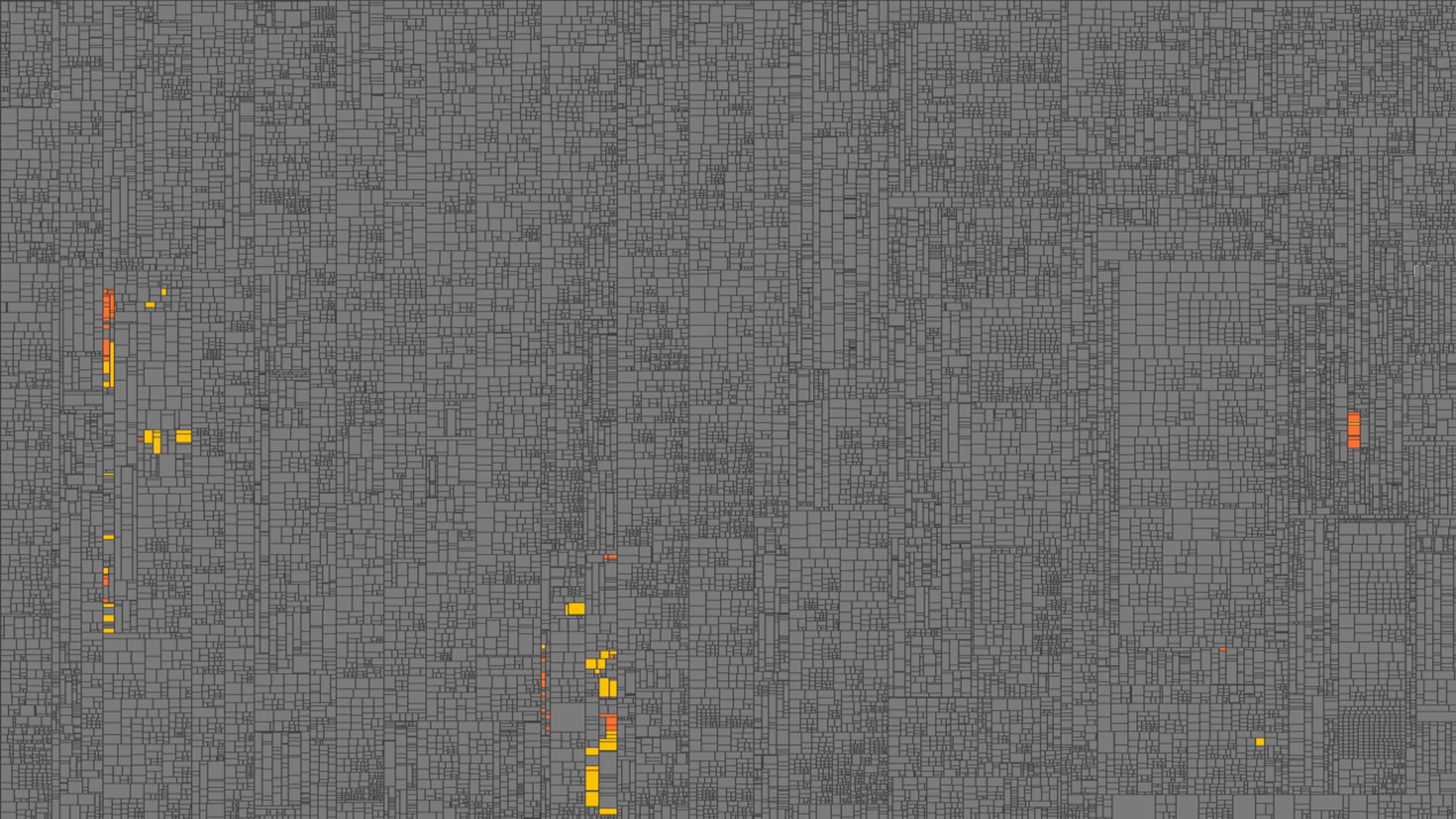


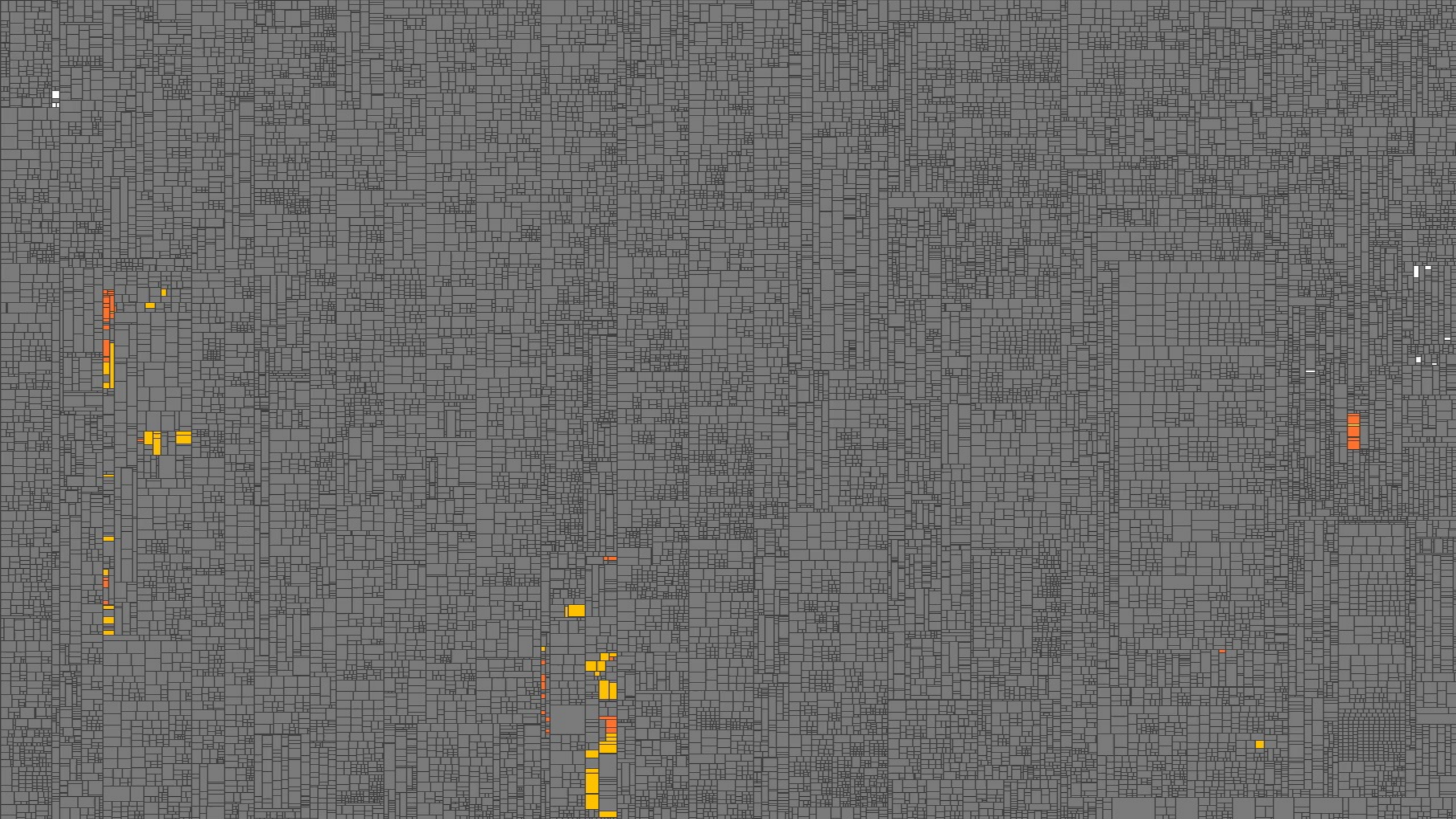


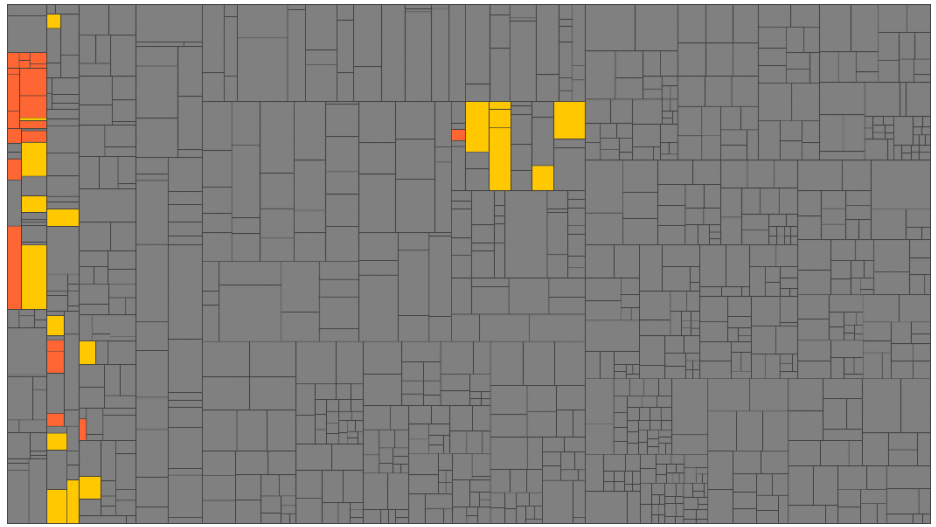
Pareto
Optimization

Change-based Testing


Version Control System
SAP
manual
Coverage
Test Steps
automated
Coverage
Test Code
Code
Changes
Test Impact Analysis
Selected Tests



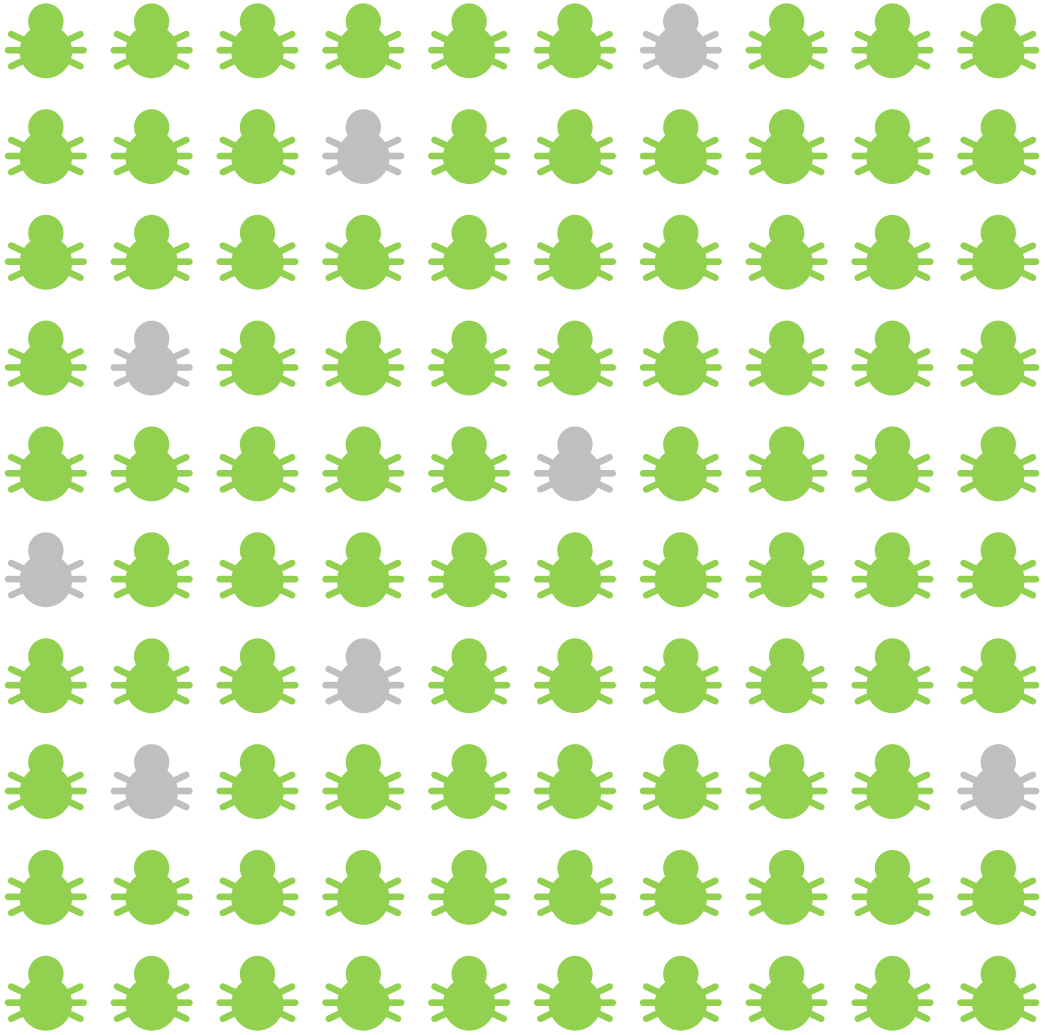


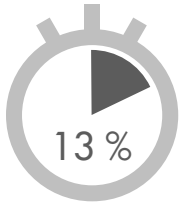


Idea: Select tests that
fit the changes

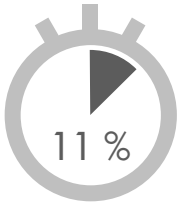
… 5000+ Test Cases without Coverage

Test Impact Analysis





AI Test Clustering



Pareto Optimization

Demo

www.teamscale.com

In Progress

Issue 133749 - Add QuickSave view

Updated: Dec 15 2018 14:20

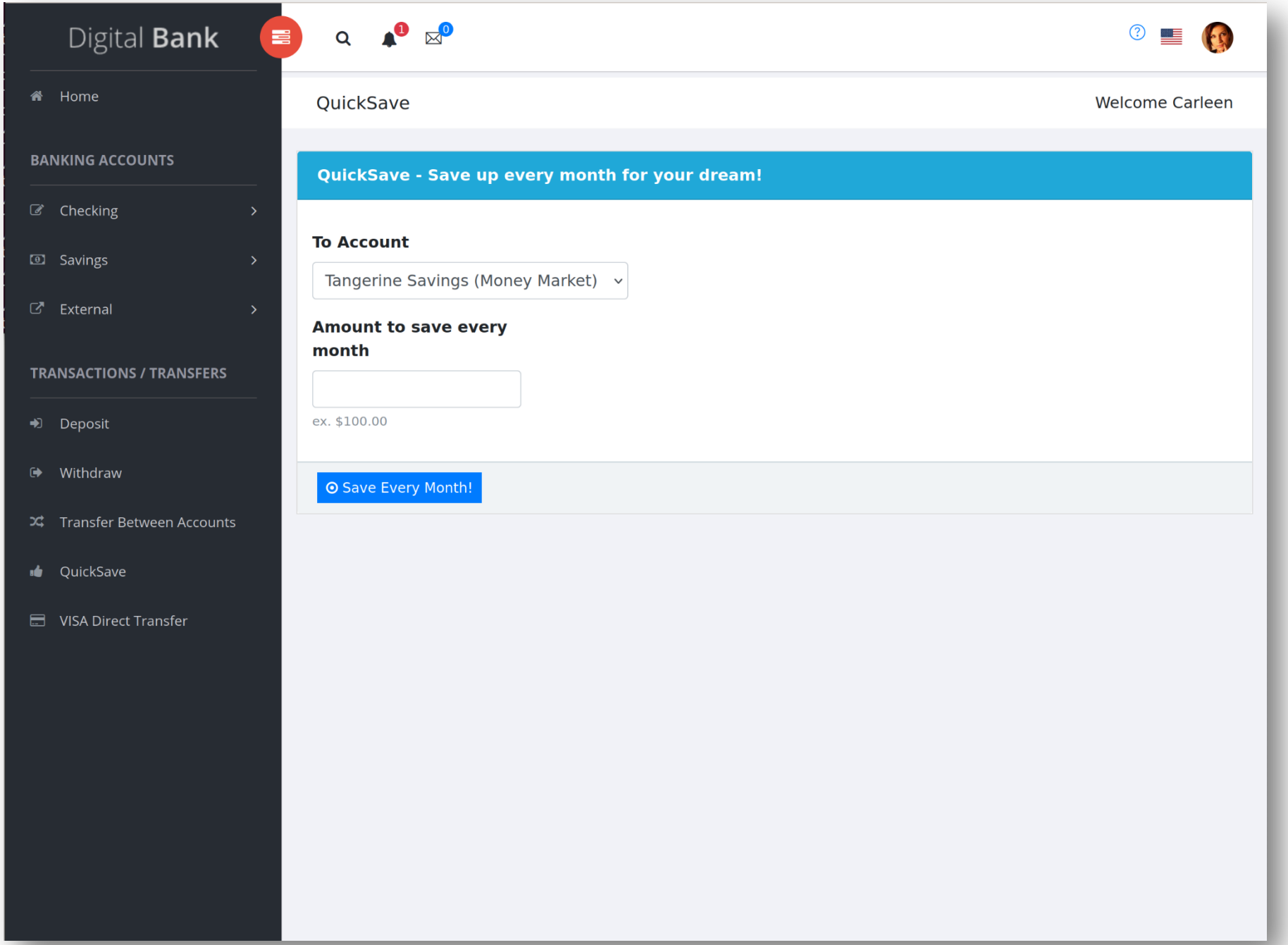
Creator:  Mark Asbury (on Dec 15 2018 14:20)

Assignee:  Andrew Smith

Type	Priority	Fix Version	Component	QA-Contact
Feature	High	DigiBank 2.2.0	Backend	qa

Description

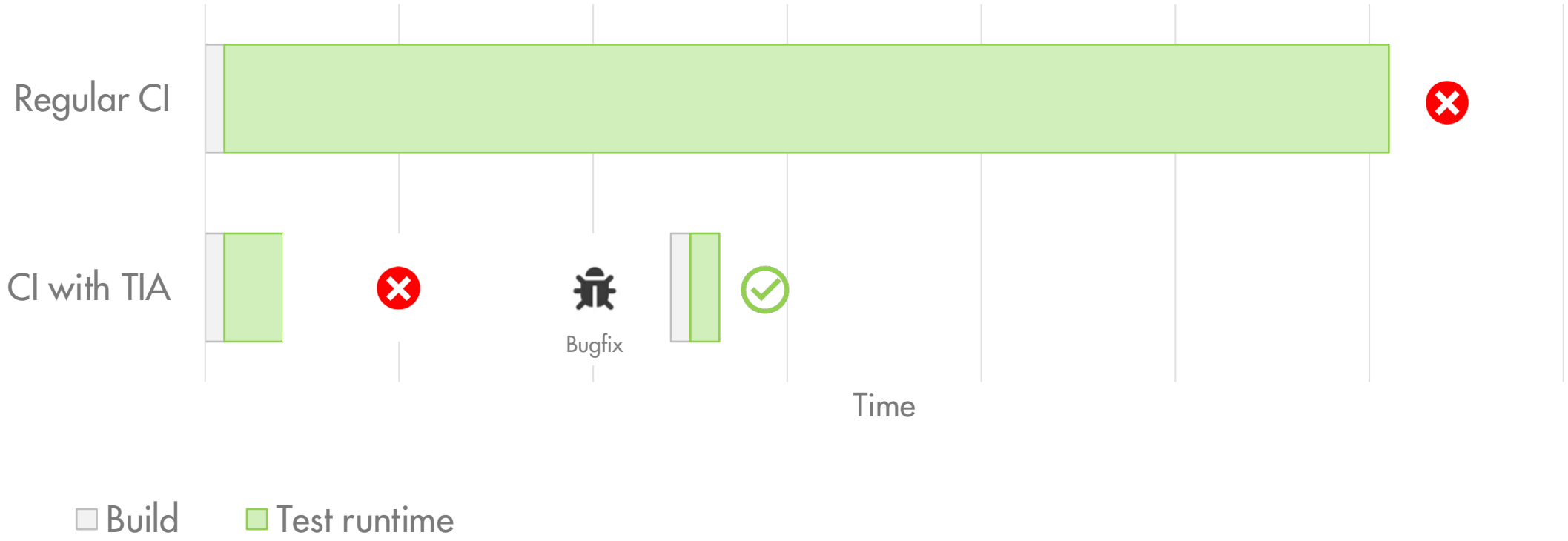
Allow users to easily save 100€ to their savings account. This should just take one click of a button.


Digital Bank
Home
BANKING ACCOUNTS
Checking
Savings
External
TRANSACTIONS / TRANSFERS
Deposit
Withdraw
Transfer Between Accounts
QuickSave
VISA Direct Transfer
QuickSave
Welcome Carleen
QuickSave - Save up every month for your dream!
To Account
Tangerine Savings (Money Market)
Amount to save every month
ex. $100.00
Save Every Month!

Demo

www.teamscale.com

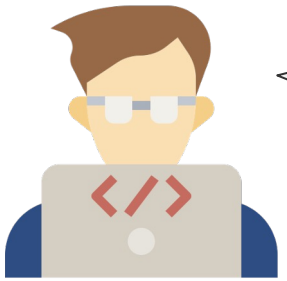
Summary



Disadvantages of Coverage-Based Approaches

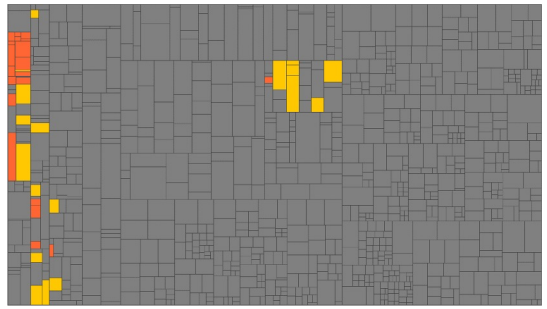
High Setup and Maintenance Effort

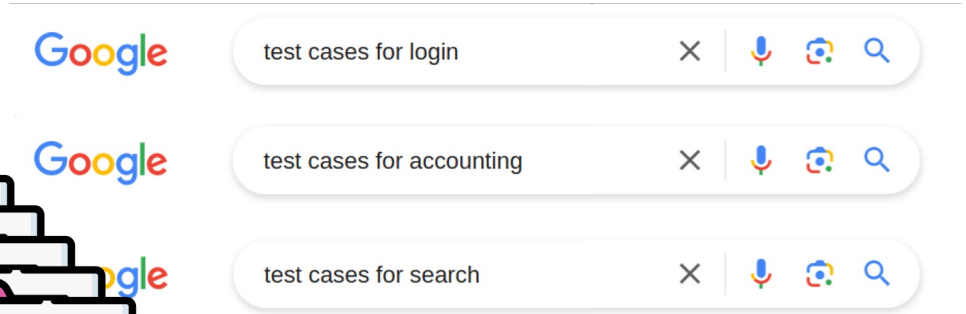
High Complexity

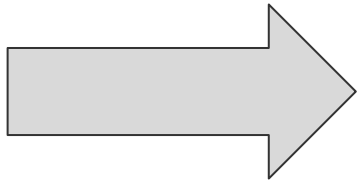


We changed **Login, Accounting and Search**.

(User Story, Pull Request, Release, …)







Tests for the affected functionality

```
125  -                  try (LoggingUtils.LoggingResources ignored = initializeFallbackLogging(options, delayedLogger)) {
126  -                      delayedLogger.errorAndStdErr(
127  -                              e.getMessage() + " The application should start up normally, but NO coverage will be collected!",
128  -                              e);
129  -                      attemptLogAndThrow(delayedLogger);
     132 +                  try (LoggingUtils.LoggingResources ignored = initializeFallbackLogging(options)) {
     133 +                      String message = e.getMessage() + " The application should start up normally, but NO coverage will be collected!";
     134 +                      logAndPrintError(e, message);
130  135                       throw e;
131  136               }
132  137           }
133  138
134  -           initializeLogging(
135  -           Logger logger =
136  -           delayedLogger.
     139 +           initializeLoggi
137  140           HttpUtils.setSh
138  141           return agentOpti
139  142       }
140  143
141  -   private static void att
142  -           // We perform a
143  -           // ensure the c
```

Code Change

Search query

"Search and login and list and user and …"

manual

Test Steps

automated

Test Code

Document DB

Similarity Scoring

Suggested Tests

Changed Code

```
43
42      LOG.debug("Debit Transaction from Account: Account Updated.");
41
40    }
39
38    /*
37     * Transfer amount between two accounts
36     *
35     * Accounts should be full objects. With that said, the objects are fetched to make sure.
34     *
33     * AccountTransaction can be a partial object but must contain the transaction amount.
32     */
31    public void transfer(Account fromAccount, Account toAccount, AccountTransaction accountTransaction) {
30
29      LOG.debug("Transfer Between Accounts:");
28
27      // From Transaction
26      fromAccount = this.getAccountById(fromAccount.getId());
25      AccountTransaction fromAt = new AccountTransaction();
24      fromAt.setAmount(accountTransaction.getAmount());
23      fromAt.setTransactionDate(accountTransaction.getTransactionDate());
22      fromAt.setDescription("Transfer to Account (" + toAccount.getAccountNumber() + ")");
21      fromAt.setTransactionType(transactionTypeRepository.findByCode(Constants.ACCT_TRAN_TYPE_XFER_CODE));
20      debitTransaction(fromAccount, fromAt);
19
18      // To Transaction
17      toAccount = this.getAccountById(toAccount.getId());
16      AccountTransaction toAt = new AccountTransaction();
15      toAt.setAmount(accountTransaction.getAmount());
14      toAt.setTransactionDate(accountTransaction.getTransactionDate());
13      toAt.setDescription("Transfer from Account (" + fromAccount.getAccountNumber() + ")");
12      toAt.setTransactionType(transactionTypeRepository.findByCode(Constants.ACCT_TRAN_TYPE_XFER_CODE));
11      creditTransaction(toAccount, toAt);
10
9       LOG.debug("Transfer Between Accounts: Accounts Updated.");
8     }
7
6     /*
5      * Get Account object by Id
4      */
3     public Account getAccountById(Long id) {
2       Optional<Account> act = accountRepository.findById(id);
```

Digibank

Changed Code

```
43
42      LOG.debug("Debit Transaction from Account: Account Updated.");
41
40    }
39
38    /*
37     * Transfer amount between two accounts
36     *
35     * Accounts should be full objects. With that said, the objects are f
34     *
33     * AccountTransaction can be a partial object but must contain the tr
32     */
31    public void transfer(Account fromAccount, Account toAccount, AccountT
30
29      LOG.debug("Transfer Between Accounts:");
28
27      // From Transaction
26      fromAccount = this.getAccountById(fromAccount.getId());
25      AccountTransaction fromAt = new AccountTransaction();
24      fromAt.setAmount(accountTransaction.getAmount());
23      fromAt.setTransactionDate(accountTransaction.getTransactionDate());
22      fromAt.setDescription("Transfer to Account (" + toAccount.getAccoun
21      fromAt.setTransactionType(transactionTypeRepository.findByCode(Cons
20      debitTransaction(fromAccount, fromAt);
19
18      // To Transaction
17      toAccount = this.getAccountById(toAccount.getId());
16      AccountTransaction toAt = new AccountTransaction();
15      toAt.setAmount(accountTransaction.getAmount());
14      toAt.setTransactionDate(accountTransaction.getTransactionDate());
13      toAt.setDescription("Transfer from Account (" + fromAccount.getAcco
12      toAt.setTransactionType(transactionTypeRepository.findByCode(Consta
11      creditTransaction(toAccount, toAt);
10
9       LOG.debug("Transfer Between Accounts: Accounts Updated.");
8     }
7
6     /*
5      * Get Account object by Id
4      */
3     public Account getAccountById(Long id) {
2       Optional<Account> act = accountRepository.findById(id);
```

Code Test

```
20    public class AccountServiceTest extends IntegrationTest {
19
18      @Test
17      public void transferBetweenSameAccountShouldNotBePossible() {
16        Account account = new Account("savings", AccountType.SAVING
15        AccountService service = new AccountService();
14        AccountTransaction transaction =
13          MockAccountTransaction.createForAmount(100);
12        service.transfer(account, account, transaction);
11
10        assertThat(this.getErrors()).contains(
9           new Error("Transfer between same account is not possible.
8
7         Account databaseAccount = this.findAccountById(account.getI
6         AccountTransactionList transactionList = this.getTransactio
5         assertThat(transactionList).isEmpty()
4       }
3
2     }
1
21
```

Changed Code

```
43
42      LOG.debug("Debit Transaction from Account: Account Updated.");
41
40  }
39
38  /*
37   * Transfer amount between two accounts
36   *
35   * Accounts should be full objects. With that said, the objects are f
34   *
33   * AccountTransaction can be a partial object but must contain the tra
32   */
31  public void transfer(Account fromAccount, Account toAccount, AccountT
30
29      LOG.debug("Transfer Between Accounts:");
28
27      // From Transaction
26      fromAccount = this.getAccountById(fromAccount.getId());
25      AccountTransaction fromAt = new AccountTransaction();
24      fromAt.setAmount(accountTransaction.getAmount());
23      fromAt.setTransactionDate(accountTransaction.getTransactionDate());
22      fromAt.setDescription("Transfer to Account (" + toAccount.getAccoun
21      fromAt.setTransactionType(transactionTypeRepository.findByCode(Cons
20      debitTransaction(fromAccount, fromAt);
19
18      // To Transaction
17      toAccount = this.getAccountById(toAccount.getId());
16      AccountTransaction toAt = new AccountTransaction();
15      toAt.setAmount(accountTransaction.getAmount());
14      toAt.setTransactionDate(accountTransaction.getTransactionDate());
13      toAt.setDescription("Transfer from Account (" + fromAccount.getAcco
12      toAt.setTransactionType(transactionTypeRepository.findByCode(Consta
11      creditTransaction(toAccount, toAt);
10
9       LOG.debug("Transfer Between Accounts: Accounts Updated.");
8   }
7
6   /*
5    * Get Account object by Id
4    */
3   public Account getAccountById(Long id) {
2       Optional<Account> act = accountRepository.findById(id);
```

Cucumber Test

```
22  @ui @account @savings
21  Feature: Transfer Money (UI)
20    As a DigitalBank user
19    I want to transfer money between accounts
18    So I can change how much is in each account
17
16
15  @negative
14  Scenario: Transfer between the same account is not possible
13    Given Carleen is logged into the application with Carleen6231@gmail.com
12    And they attempt to open a new 'Savings Account'
11      When Carleen enters 'Tangerine Savings' into the Account Name field
10      And they select 'Individual' from the Ownership radio button
9       And they select 'Money Market' from the Account Type radio button
8       And they enter '2500' into the Money Market Initial Deposit field
7       And they click the Submit button
6     And they attempt to transfer money
5     When Carleen selects account number '1' as the from account
4     And they select account number '1' as the to account
3     And they enter '11' into the amount field
2     And they submit the form
1     Then Carleen verifies the transfer failed
23
```

Changed Code

```
43
42      LOG.debug("Debit Transaction from Account: Account Updated.");
41
40  }
39
38  /*
37   * Transfer amount between two accounts
36   *
35   * Accounts should be full objects. With that said, the objects are fe
34   *
33   * AccountTransaction can be a partial object but must contain the tra
32   */
31  public void transfer(Account fromAccount, Account toAccount, AccountTr
30
29      LOG.debug("Transfer Between Accounts:");
28
27      // From Transaction
26      fromAccount = this.getAccountById(fromAccount.getId());
25      AccountTransaction fromAt = new AccountTransaction();
24      fromAt.setAmount(accountTransaction.getAmount());
23      fromAt.setTransactionDate(accountTransaction.getTransactionDate());
22      fromAt.setDescription("Transfer to Account (" + toAccount.getAccount
21      fromAt.setTransactionType(transactionTypeRepository.findByCode(Const
20      debitTransaction(fromAccount, fromAt);
19
18      // To Transaction
17      toAccount = this.getAccountById(toAccount.getId());
16      AccountTransaction toAt = new AccountTransaction();
15      toAt.setAmount(accountTransaction.getAmount());
14      toAt.setTransactionDate(accountTransaction.getTransactionDate());
13      toAt.setDescription("Transfer from Account (" + fromAccount.getAccou
12      toAt.setTransactionType(transactionTypeRepository.findByCode(Constan
11      creditTransaction(toAccount, toAt);
10
9       LOG.debug("Transfer Between Accounts: Accounts Updated.");
8   }
7
6   /*
5    * Get Account object by Id
4    */
3   public Account getAccountById(Long id) {
2       Optional<Account> act = accountRepository.findById(id);
```

Robot Test

```
17  *** Settings ***
16  Resource        ../keywords/digibank_keywords.robot
15
14  *** Test Cases ***
13  Transfer between the same account is not possible
12      Log in  Carleen6231@gmail.com
11      Open new account    Savings Account     Individual  Money Market    2
10      Open transfer page
9       Select from account number   1
8       Select to account number   1
7       Enter amount    11
6       Submit transfer form
5       Transfer failed message should be displayed
4
3
2
1
18
```

Geänderter Code

```
43
42      LOG.debug("Debit Transaction from Account: Account Updated.");
41
40  }
39
38  /*
37   * Transfer amount between two accounts
36   *
35   * Accounts should be full objects. With that said, the objects are f
34   *
33   * AccountTransaction can be a partial object but must contain the tra
32   */
31  public void transfer(Account fromAccount, Account toAccount, AccountT
30
29      LOG.debug("Transfer Between Accounts:");
28
27      // From Transaction
26      fromAccount = this.getAccountById(fromAccount.getId());
25      AccountTransaction fromAt = new AccountTransaction();
24      fromAt.setAmount(accountTransaction.getAmount());
23      fromAt.setTransactionDate(accountTransaction.getTransactionDate());
22      fromAt.setDescription("Transfer to Account (" + toAccount.getAccount
21      fromAt.setTransactionType(transactionTypeRepository.findByCode(Const
20      debitTransaction(fromAccount, fromAt);
19
18      // To Transaction
17      toAccount = this.getAccountById(toAccount.getId());
16      AccountTransaction toAt = new AccountTransaction();
15      toAt.setAmount(accountTransaction.getAmount());
14      toAt.setTransactionDate(accountTransaction.getTransactionDate());
13      toAt.setDescription("Transfer from Account (" + fromAccount.getAccou
12      toAt.setTransactionType(transactionTypeRepository.findByCode(Constan
11      creditTransaction(toAccount, toAt);
10
9       LOG.debug("Transfer Between Accounts: Accounts Updated.");
8   }
7
6   /*
5    * Get Account object by Id
4    */
3   public Account getAccountById(Long id) {
2       Optional<Account> act = accountRepository.findById(id);
```

Manual Test

Action	Check
Log in as Carleen6231@gmail.com	
Open a new account: Type Savings Account, Individual In the Money Market Start deposit: 2500	Account was created as specified.
Open the transfer page.	
Select the account from step 2 as both from and to account.	
Enter amount: 11	
Submit the form	Transfer should fail with a message that transfers between the same account are prohibited

Score of a test for a search term =

term frequency * inverse **document frequency**

How often does the term occur in a test?
The more often, the higher the score

How many tests contain this term?
The more specific, the higher the score.

Test Cases for Feature 12345

#1

Test Suite 1
https://www.atlassian.com › jira-... · Diese Seite übersetzen

Test Case 1

Xray allows you to plan, design, and execute tests, as well as generate test reports. Xray uses specific Jira issues types for this process.

#2

Test Suite 2
https://www.atlassian.com › jira-... · Diese Seite übersetzen

Test Case 2

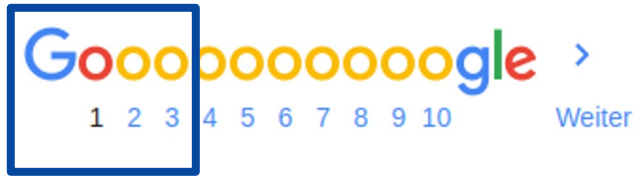
A step-by-step tutorial on how to use Xray Cloud, a continuous integration tool that triggers automated tests and provides results through an Xray Test Plan.

#3

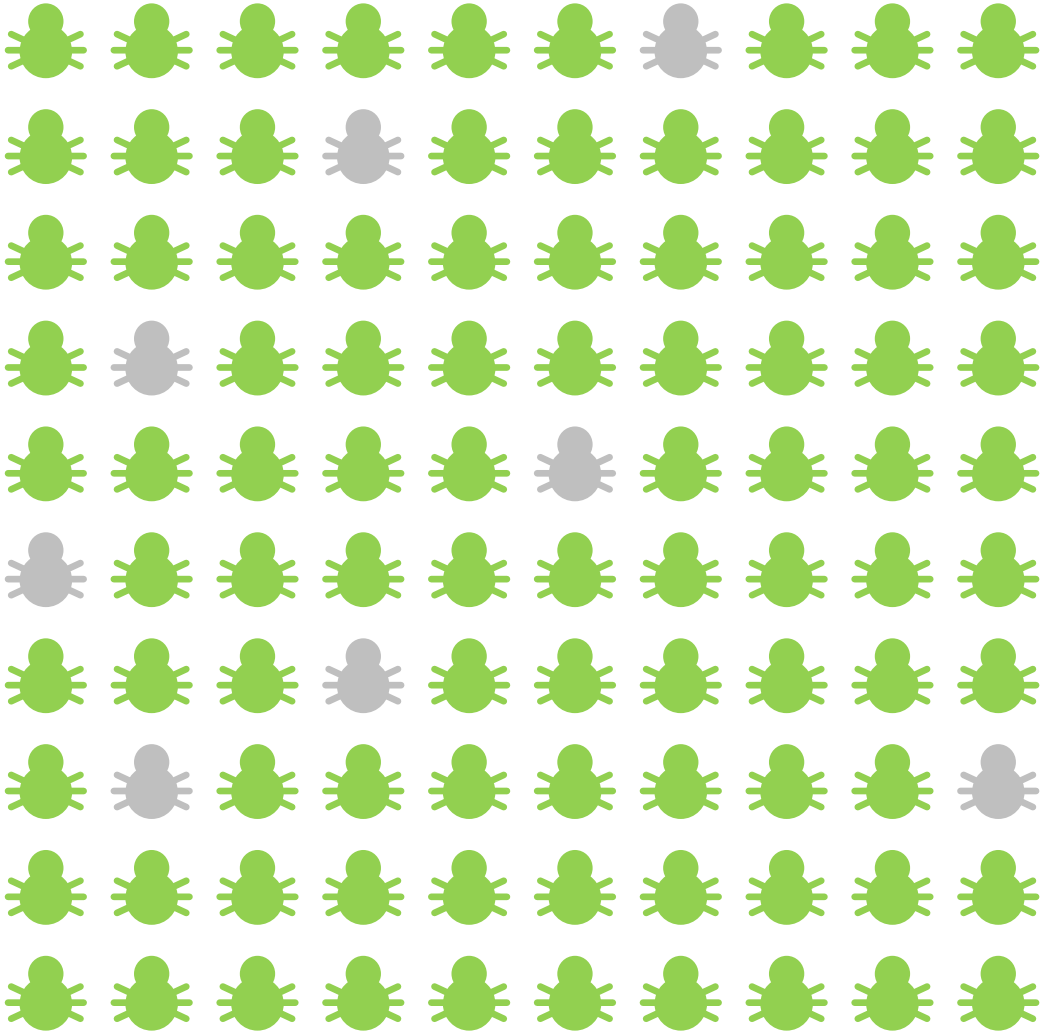
Test Suite 1
https://www.getxray.app › blog · Diese Seite übersetzen

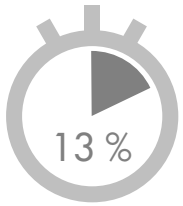
Test Case 3

27.11.2020 — It's a full-featured tool that lives inside, and seamlessly integrates with **Jira. Xray** aims to help companies improve the quality of their ...

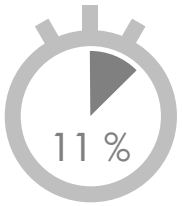


Similarity Scoring

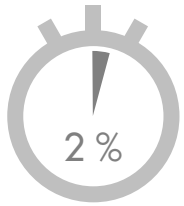




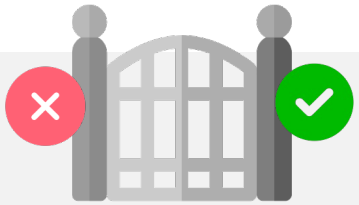
AI Test Clustering



Pareto Optimization



Test Impact Analysis



Quality Gate

Change-based Testing

We find 90% of the bugs in X% of the time

11%

Pareto Optimization

2%

Test Impact Analysis

Test Coverage

Precision & Effort

13%

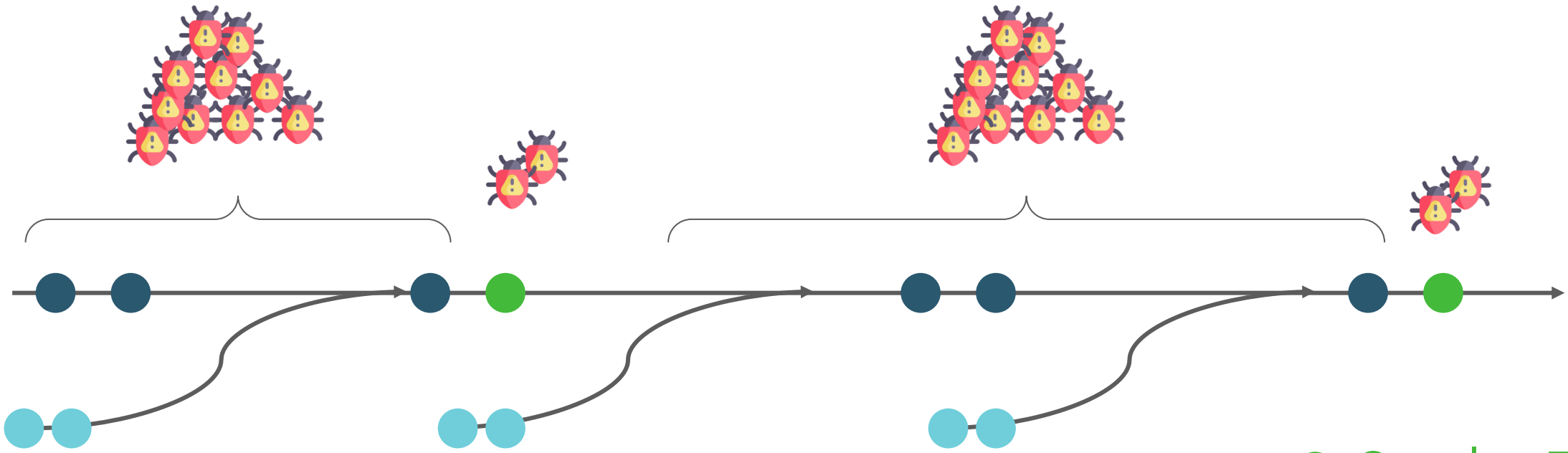
AI Test Clustering

4%

Similarity Scoring

Test Content

Shift Left



Complete Test Run
AI Test Clustering
Similarity Scoring

Teamscale by CQSE

Test Suggestions: Get Started

Tuesday
June 23, 2026
5 to 6 PM CEST / 8 to 9 AM PDT
online and free

Register now: tmscl.me/ts266-getstarted

Ramona Kühn

Raphael Nömmer

Interested in Teamscale for your team?

We're happy to hold a **free workshop for your team**!

Ramona Kühn

Raphael Nömmer

Let's talk!

CQSE GmbH
Centa-Hafenbrädl-Str. 59
81249 Munich

CQSE

Teamscale by CQSE

Get Quality for Three!

How Vestas, Dolby, Rational & others deliver high quality software

Tuesday
June 16, 2026
10:30 AM to 12:00 PM CEST
online and free – with recording

Register now: tmscl.me/si266-ts266

Dr. Sven Amann