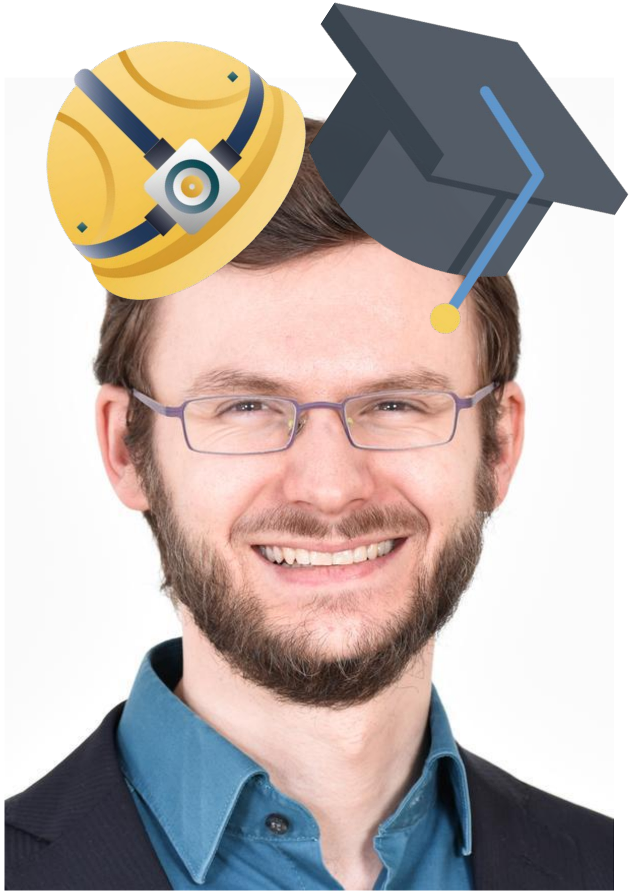
A4Q TESTING SUMMIT

EDITION 2025

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

CQSE

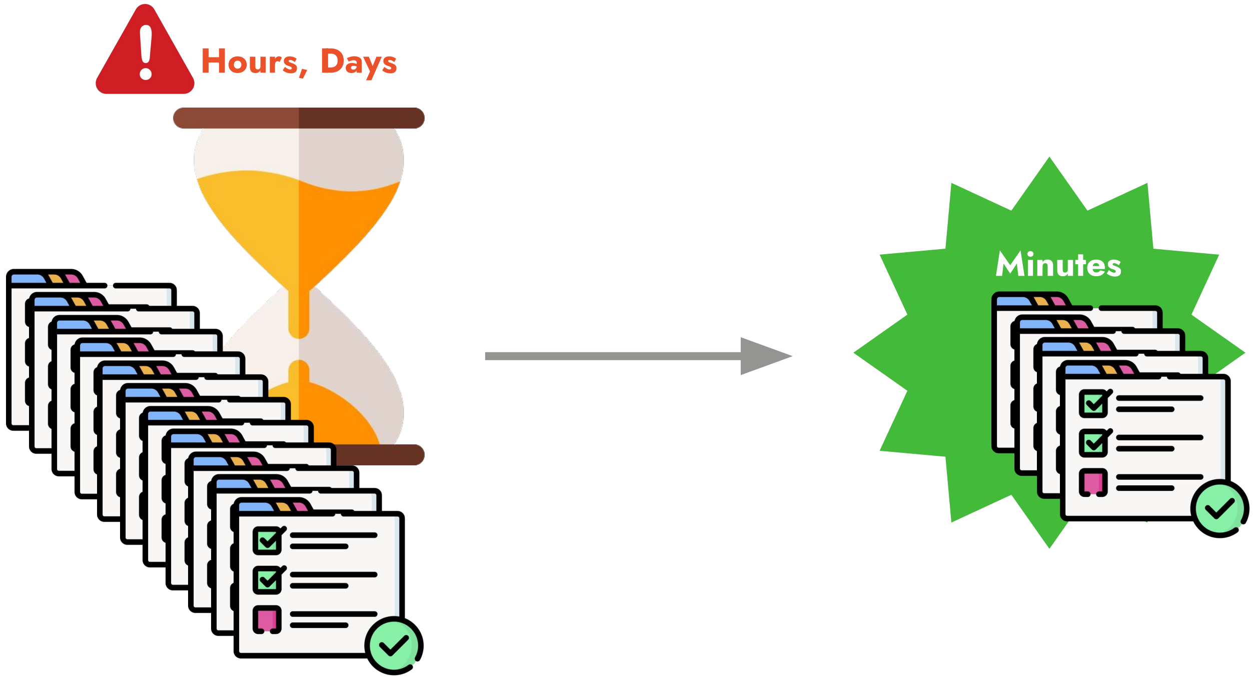
Fabian Streitel



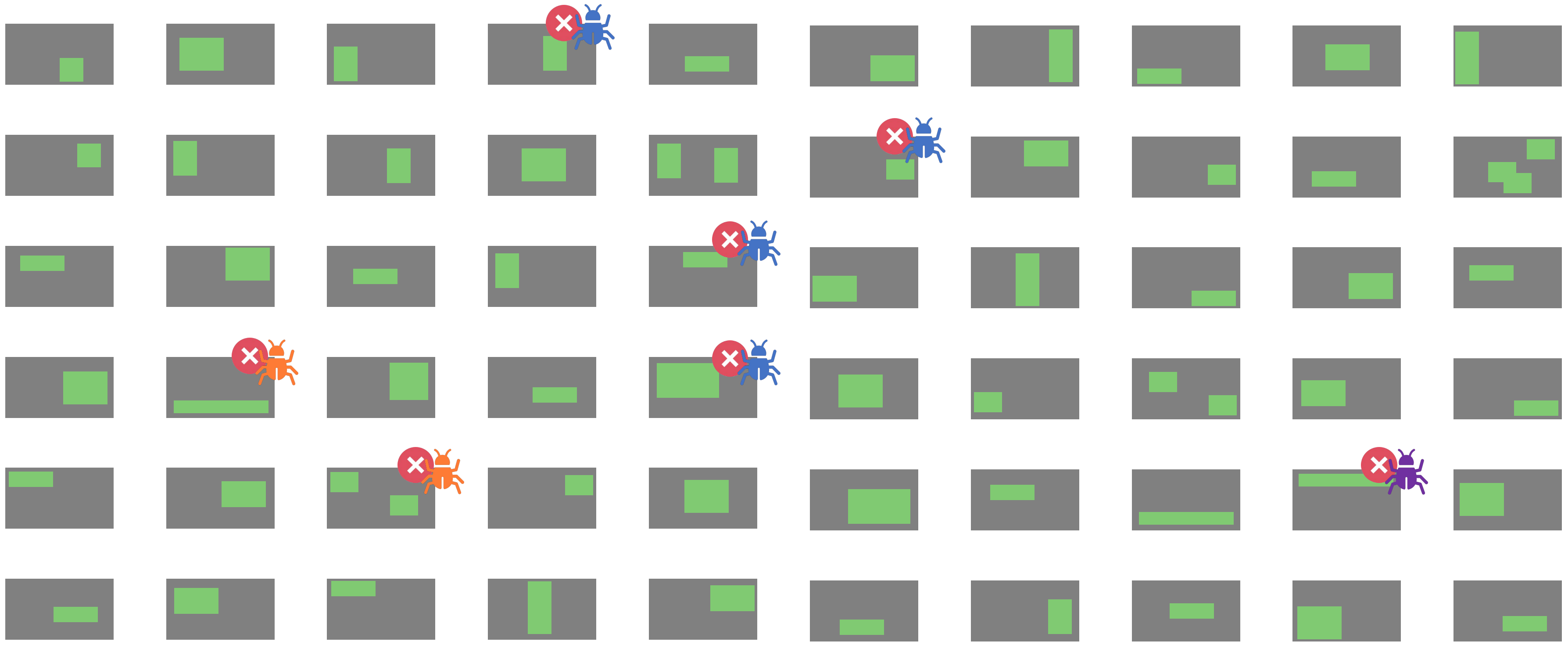
Fabian Streitel

- Team lead “Test Intelligence”
- Supervising **bachelor and master theses**
- For over 7 years: **customer projects**

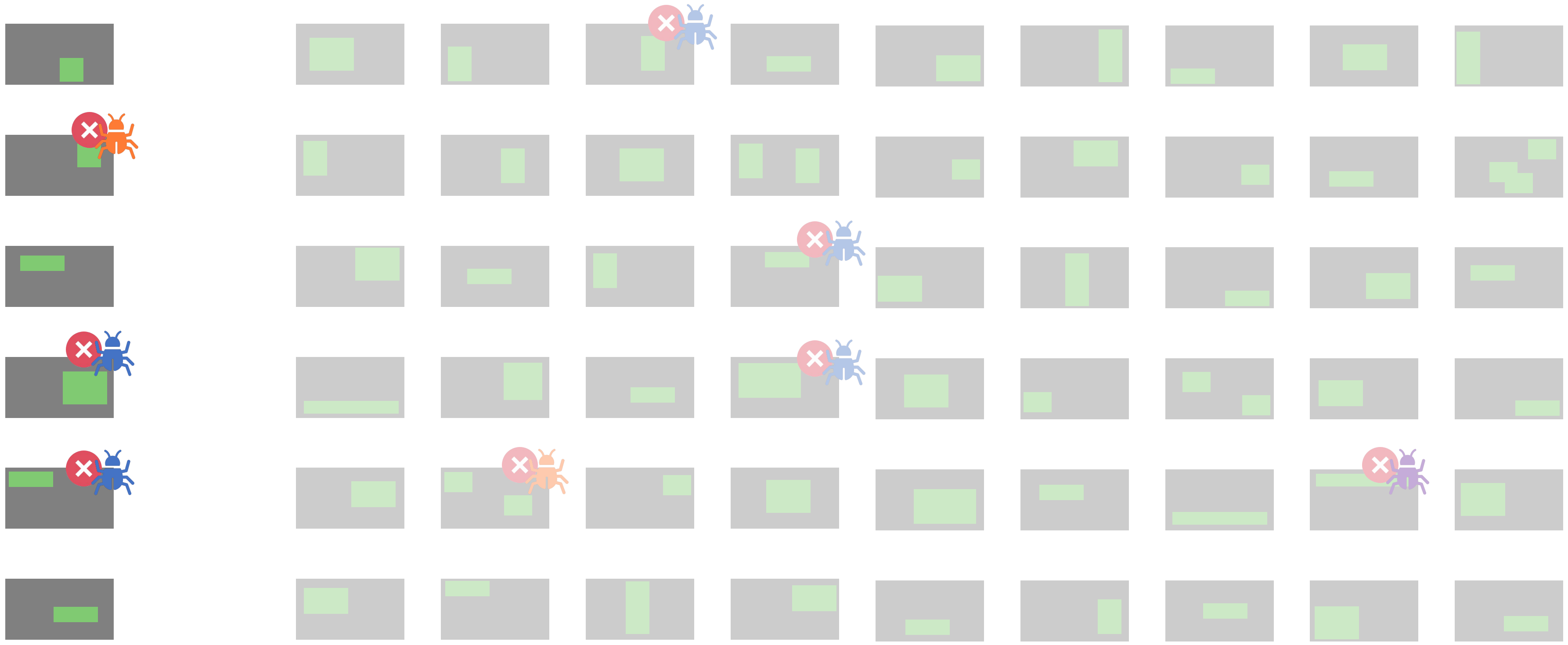
CQSE Teamscale



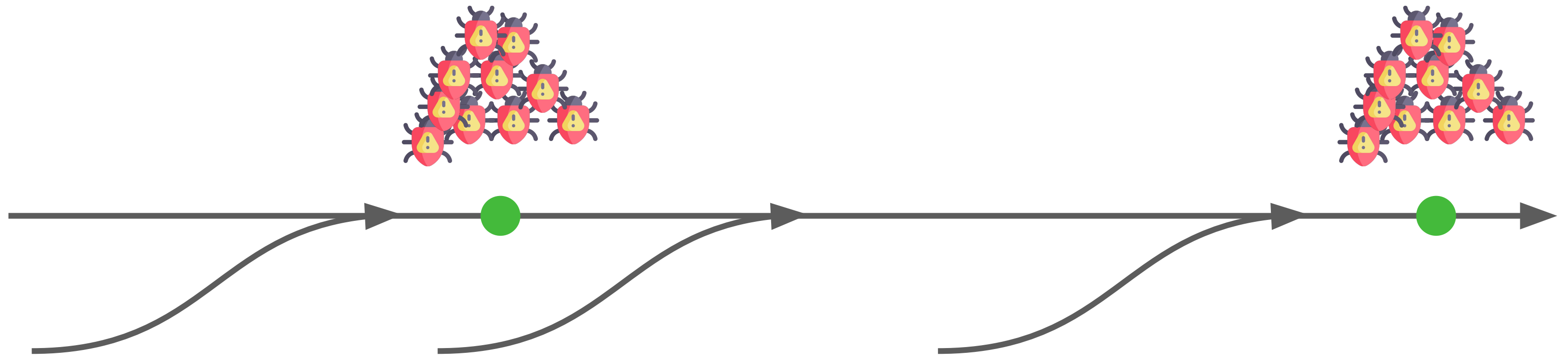
manual, automated, E2E, …



Test Selection

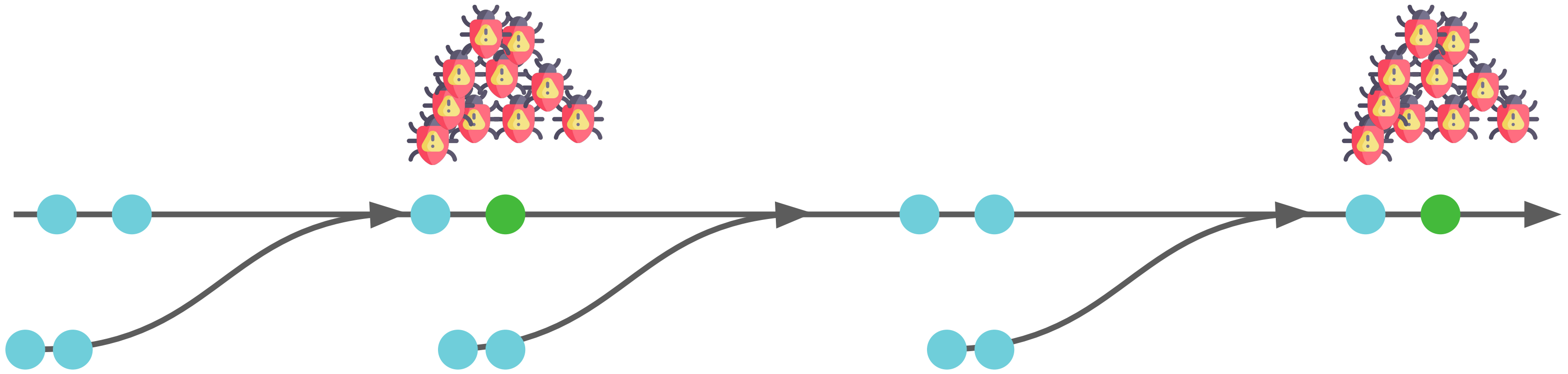


Shift Left



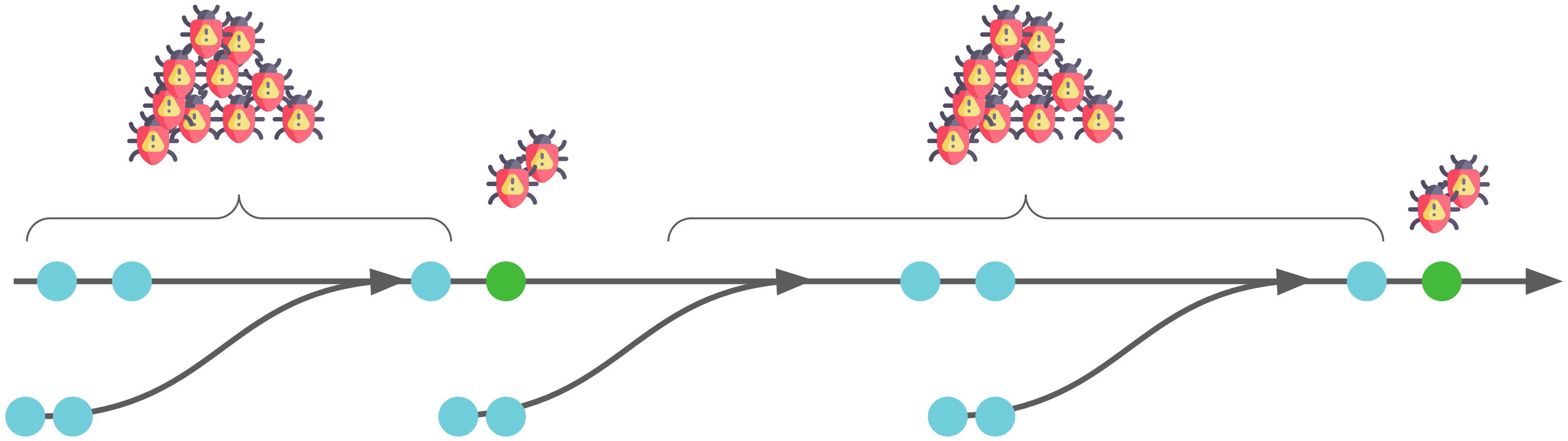
Complete Test Run

Shift Left

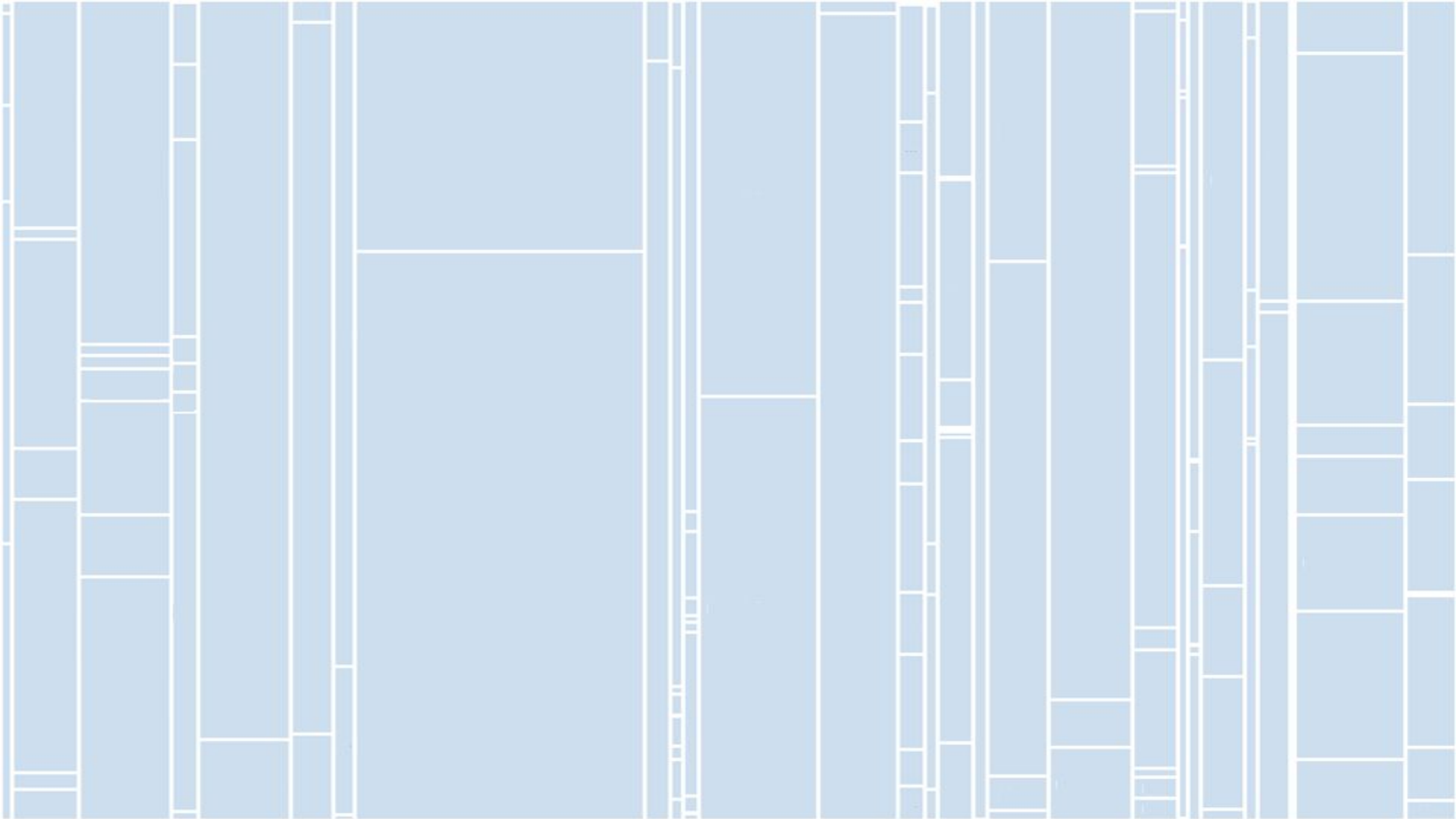


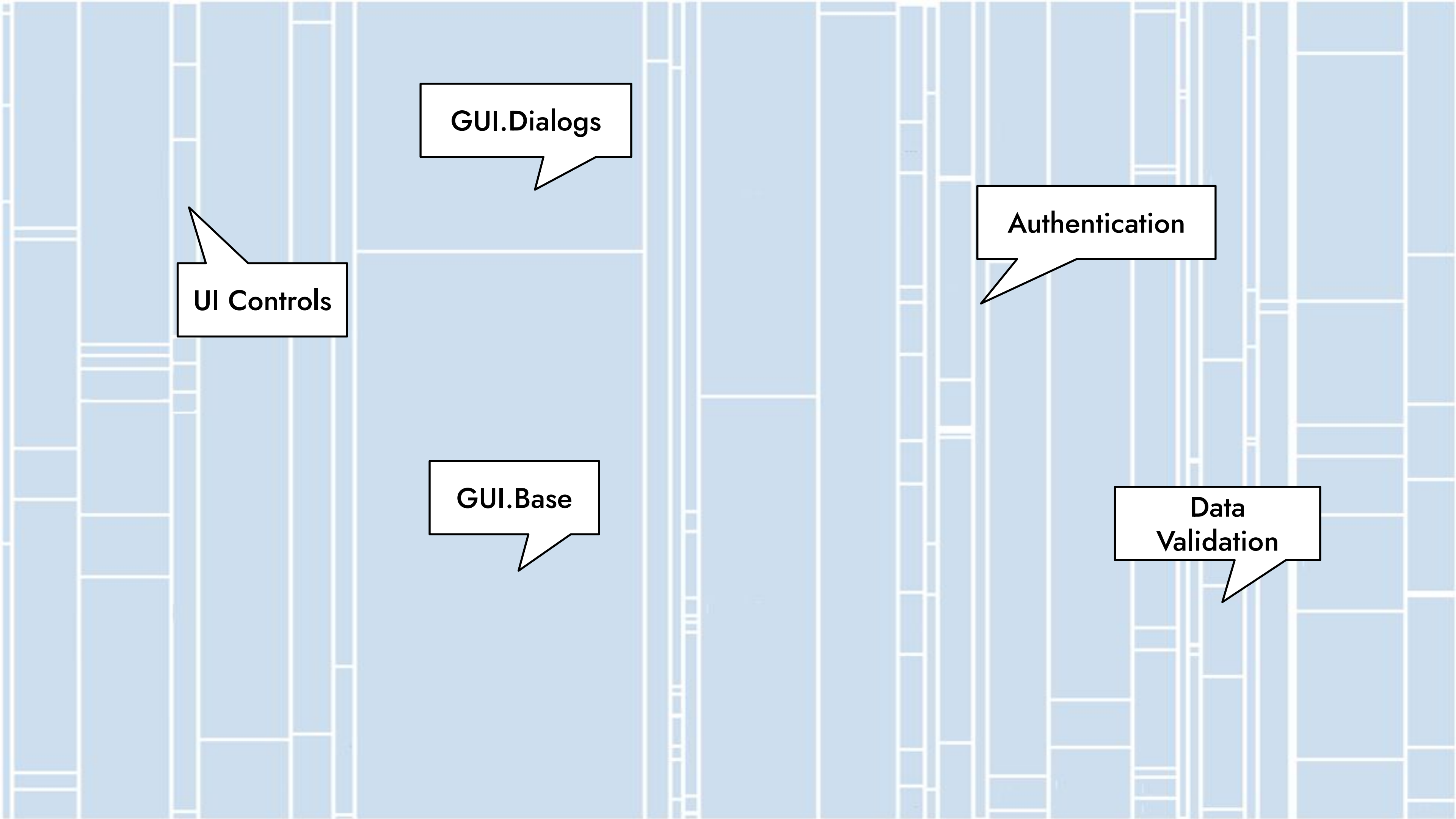
Complete Test Run
Smart Selection

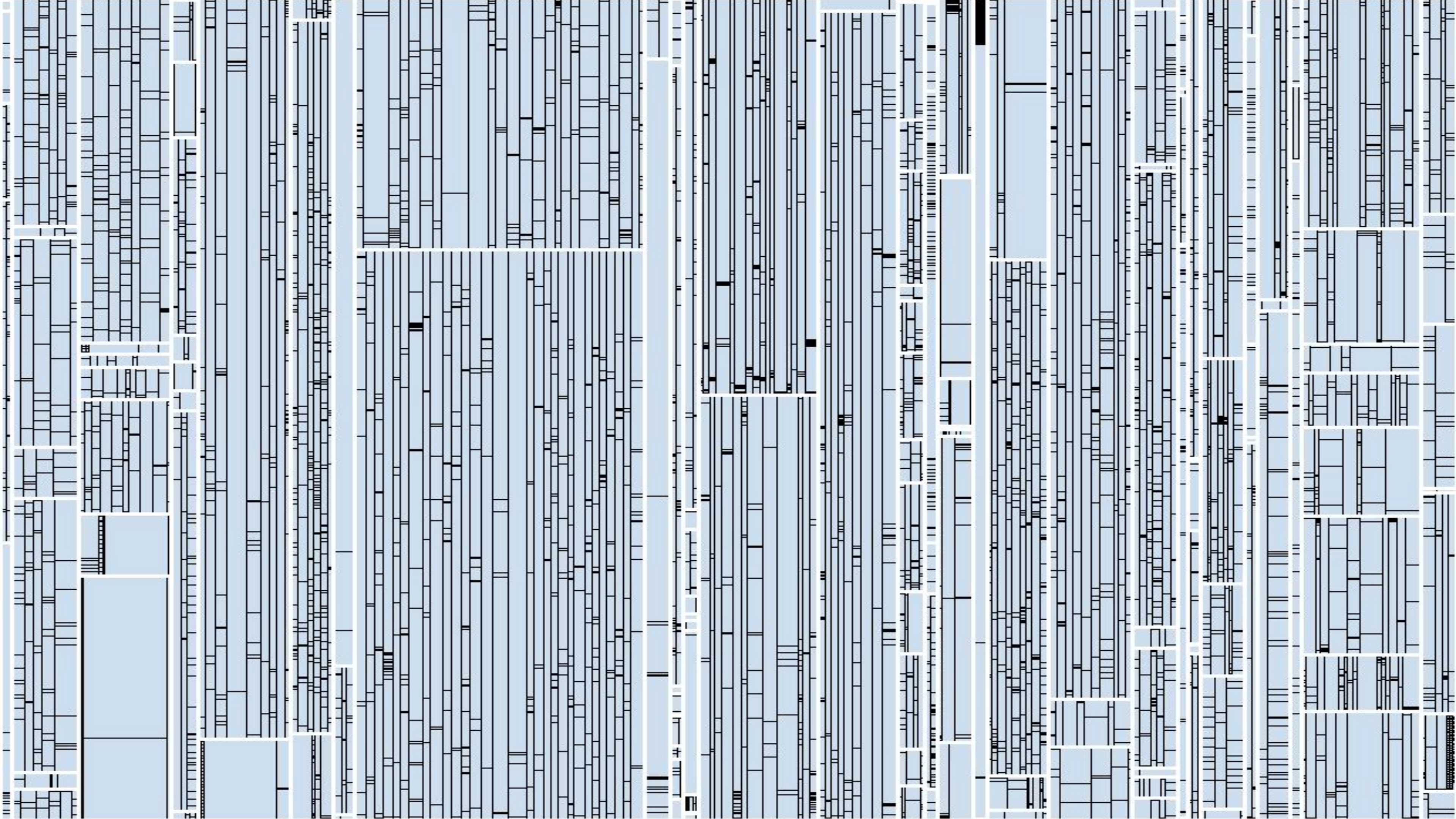
Shift Left

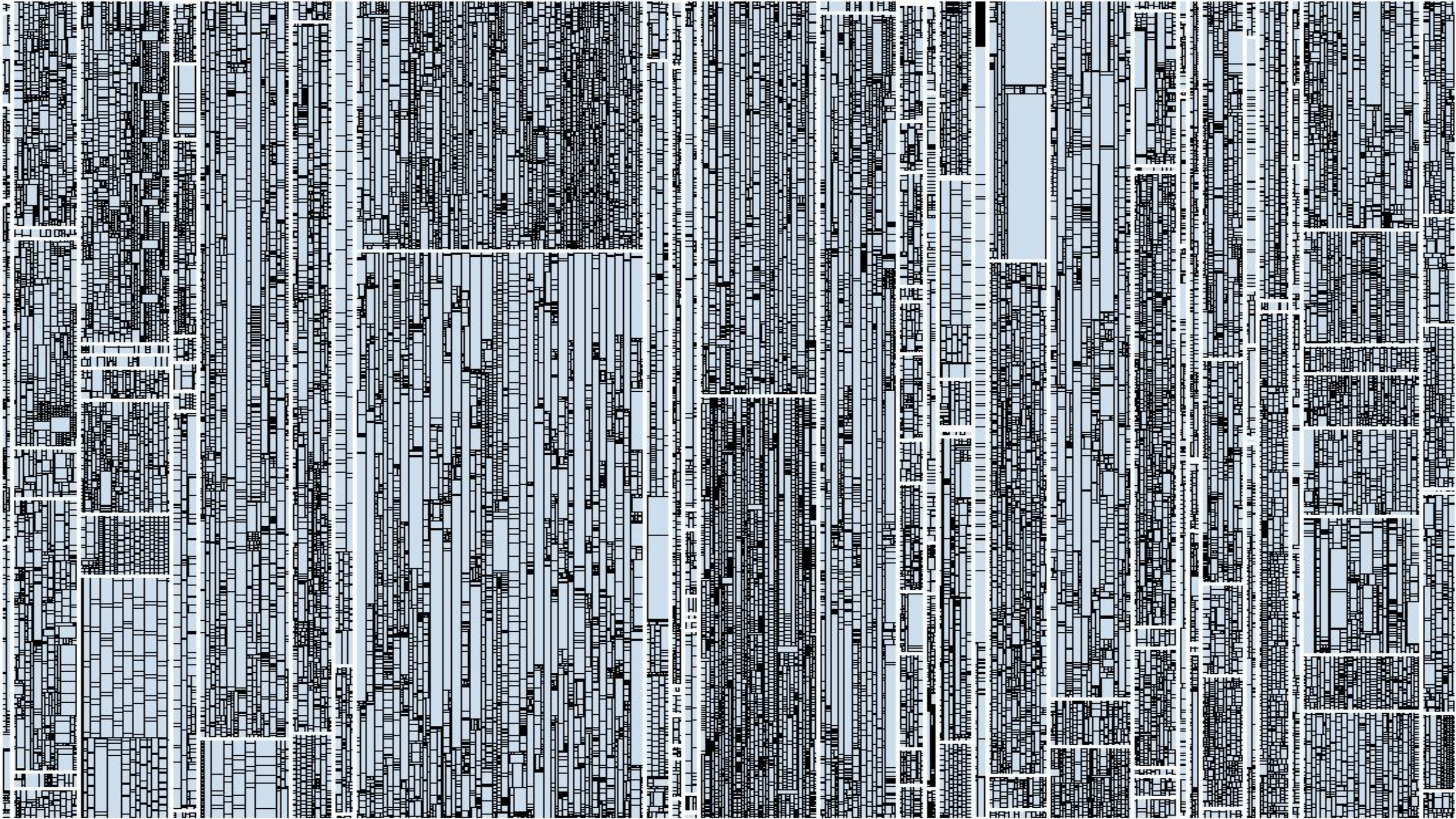


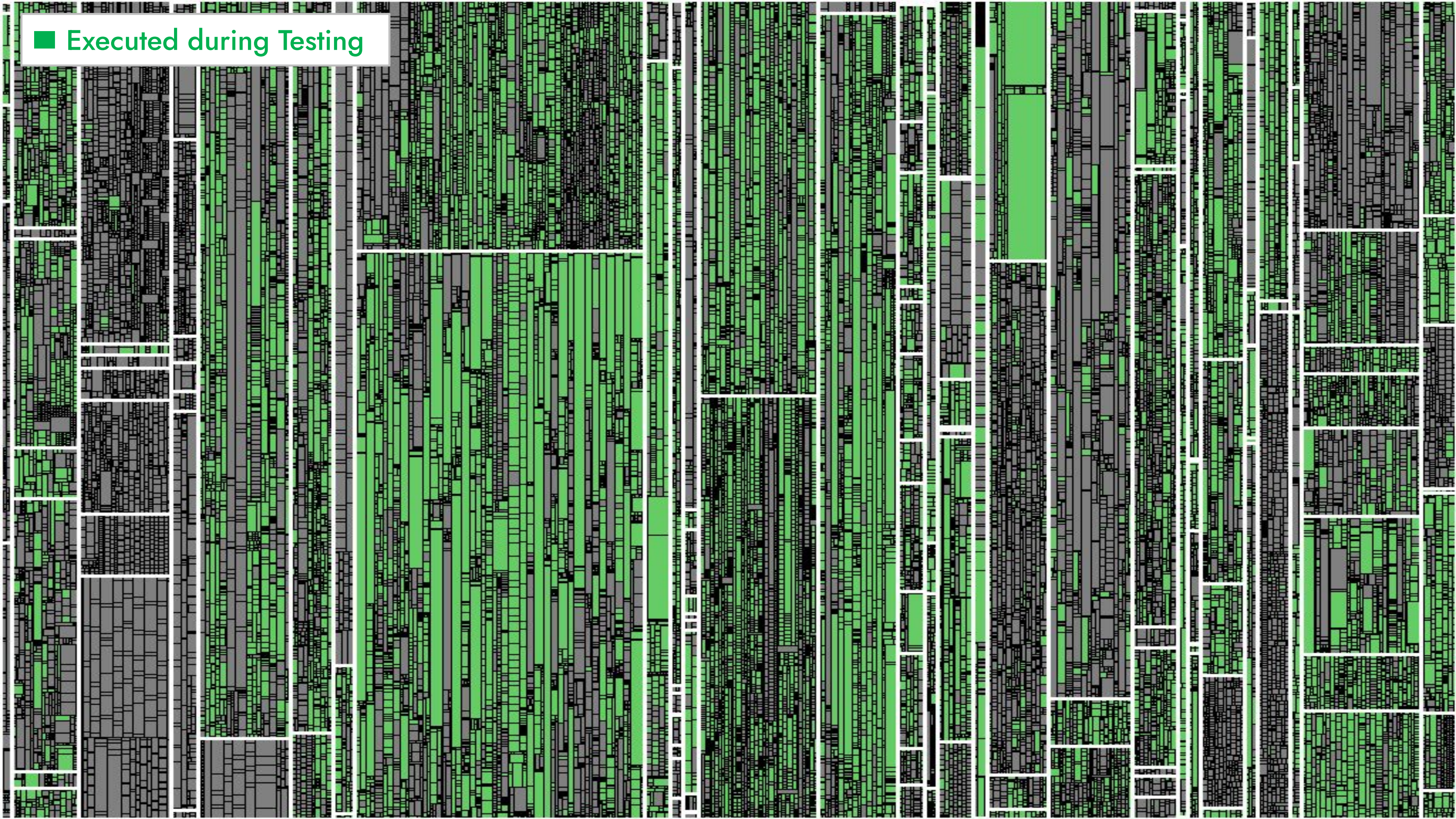
Complete Test Run
Smart Selection

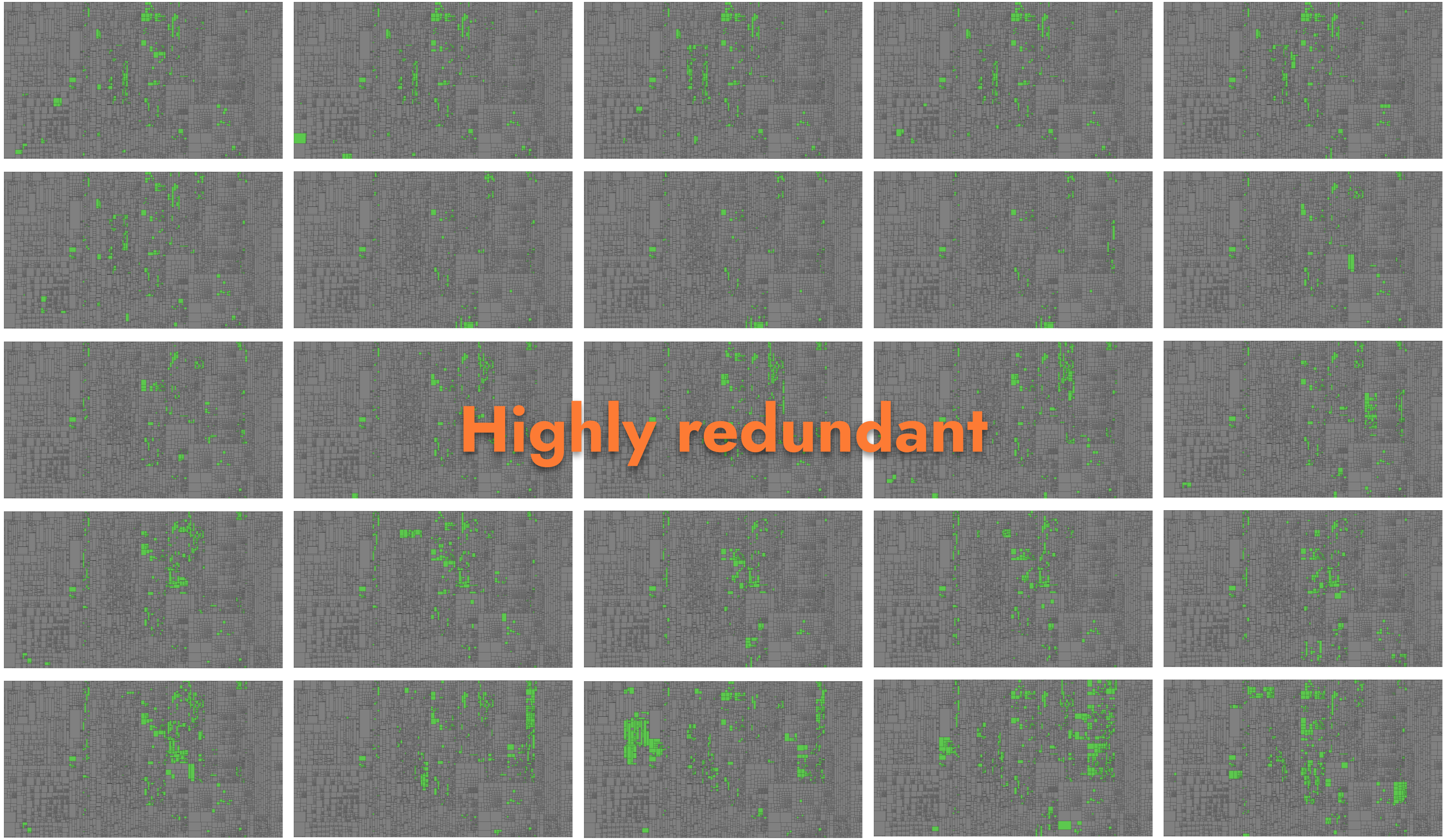



GUI.Dialogs
UI Controls
Authentication
GUI.Base
Data Validation

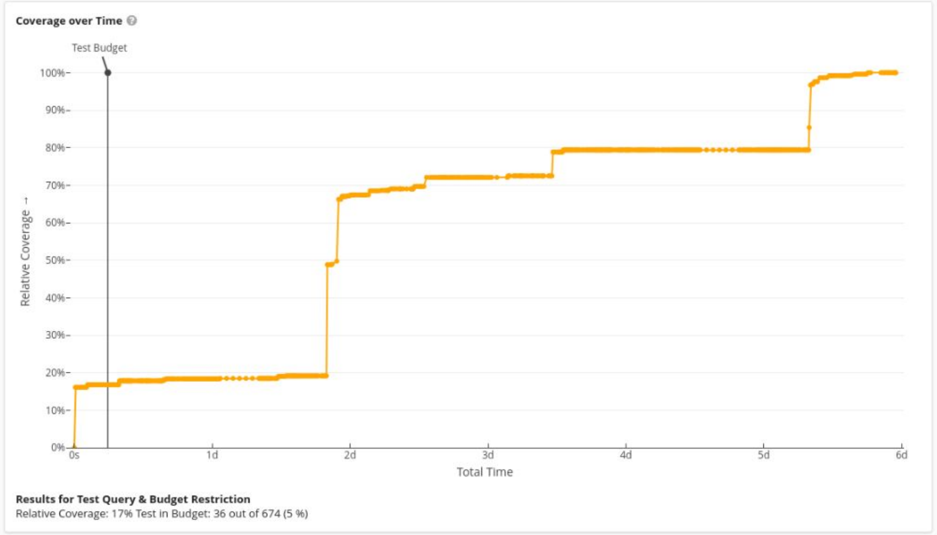




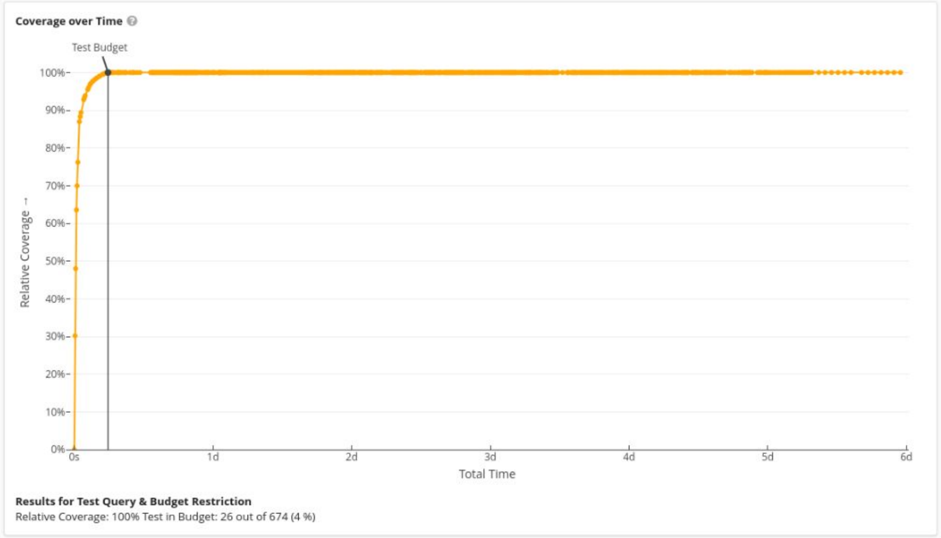

🟩 Executed during Testing



Highly redundant

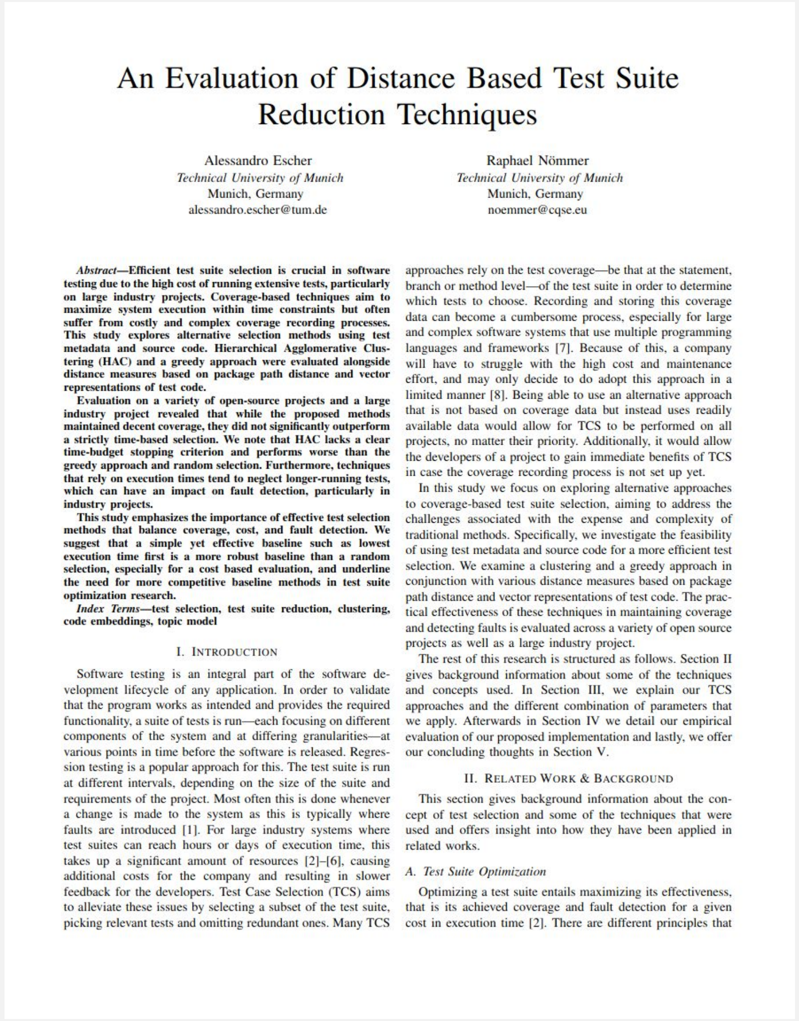


Results for Test Query & Budget Restriction
Relative Coverage: 17% Test in Budget: 36 out of 674 (5 %)



Results for Test Query & Budget Restriction
Relative Coverage: 100% Test in Budget: 26 out of 674 (4 %)

Idea: **use tests that are most dissimilar**



An Evaluation of Distance Based Test Suite Reduction Techniques

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

Raphael Nömmer
Technical University of Munich
Munich, Germany
noemmer@cqse.eu

***Abstract*—Efficient test suite selection is crucial in software testing due to the high cost of running extensive tests, particularly on large industry projects. Coverage-based techniques aim to maximize system execution within time constraints but often suffer from costly and complex coverage recording processes. This study explores alternative selection methods using test metadata and source code. Hierarchical Agglomerative Clustering (HAC) and a greedy approach were evaluated alongside distance measures based on package path distance and vector representations of test code.**

Evaluation on a variety of open-source projects and a large industry project revealed that while the proposed methods maintained decent coverage, they did not significantly outperform a strictly time-based selection. We note that HAC lacks a clear time-budget stopping criterion and performs worse than the greedy approach and random selection. Furthermore, techniques that rely on execution times tend to neglect longer-running tests, which can have an impact on fault detection, particularly in industry projects.

This study emphasizes the importance of effective test selection methods that balance coverage, cost, and fault detection. We suggest that a simple yet effective baseline such as lowest execution time first is a more robust baseline than a random selection, especially for a cost based evaluation, and underline the need for more competitive baseline methods in test suite optimization research.

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

I. Introduction

Software testing is an integral part of the software development lifecycle of any application. In order to validate that the program works as intended and provides the required functionality, a suite of tests is run—each focusing on different components of the system and at differing granularities—at various points in time before the software is released. Regression testing is a popular approach for this. The test suite is run at different intervals, depending on the size of the suite and requirements of the project. Most often this is done whenever a change is made to the system as this is typically where faults are introduced [1]. For large industry systems where test suites can reach hours or days of execution time, this takes up a significant amount of resources [2]–[6], causing additional costs for the company and resulting in slower feedback for the developers. Test Case Selection (TCS) aims to alleviate these issues by selecting a subset of the test suite, picking relevant tests and omitting redundant ones. Many TCS approaches rely on the test coverage—be that at the statement, branch or method level—of the test suite in order to determine which tests to choose. Recording and storing this coverage data can become a cumbersome process, especially for large and complex software systems that use multiple programming languages and frameworks [7]. Because of this, a company will have to struggle with the high cost and maintenance effort, and may only decide to do adopt this approach in a limited manner [8]. Being able to use an alternative approach that is not based on coverage data but instead uses readily available data would allow for TCS to be performed on all projects, no matter their priority. Additionally, it would allow the developers of a project to gain immediate benefits of TCS in case the coverage recording process is not set up yet.

In this study we focus on exploring alternative approaches to coverage-based test suite selection, aiming to address the challenges associated with the expense and complexity of traditional methods. Specifically, we investigate the feasibility of using test metadata and source code for a more efficient test selection. We examine a clustering and a greedy approach in conjunction with various distance measures based on package path distance and vector representations of test code. The practical effectiveness of these techniques in maintaining coverage and detecting faults is evaluated across a variety of open source projects as well as a large industry project.

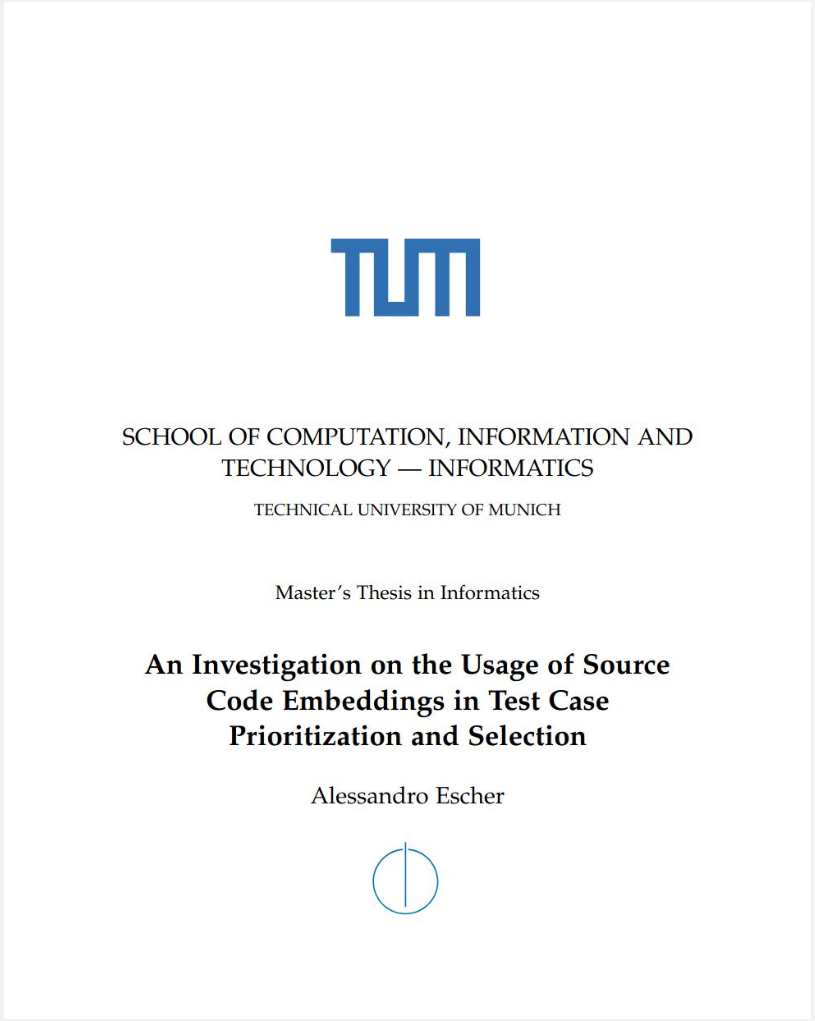
The rest of this research is structured as follows. Section II gives background information about some of the techniques and concepts used. In Section III, we explain our TCS approaches and the different combination of parameters that we apply. Afterwards in Section IV we detail our empirical evaluation of our proposed implementation and lastly, we offer our concluding thoughts in Section V.

II. Related Work & Background

This section gives background information about the concept of test selection and some of the techniques that were used and offers insight into how they have been applied in related works.

A. Test Suite Optimization

Optimizing a test suite entails maximizing its effectiveness, that is its achieved coverage and fault detection for a given cost in execution time [2]. There are different principles that



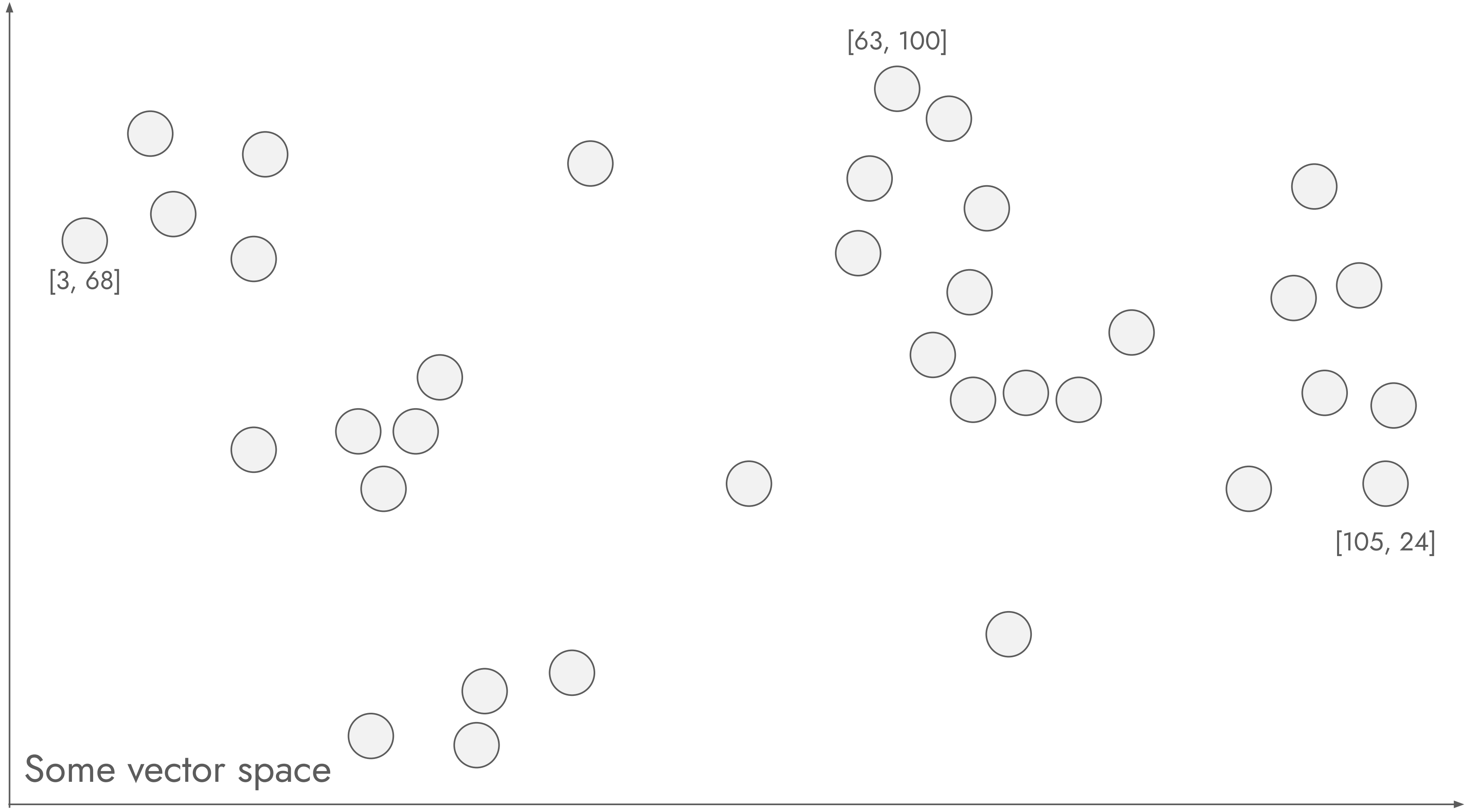
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

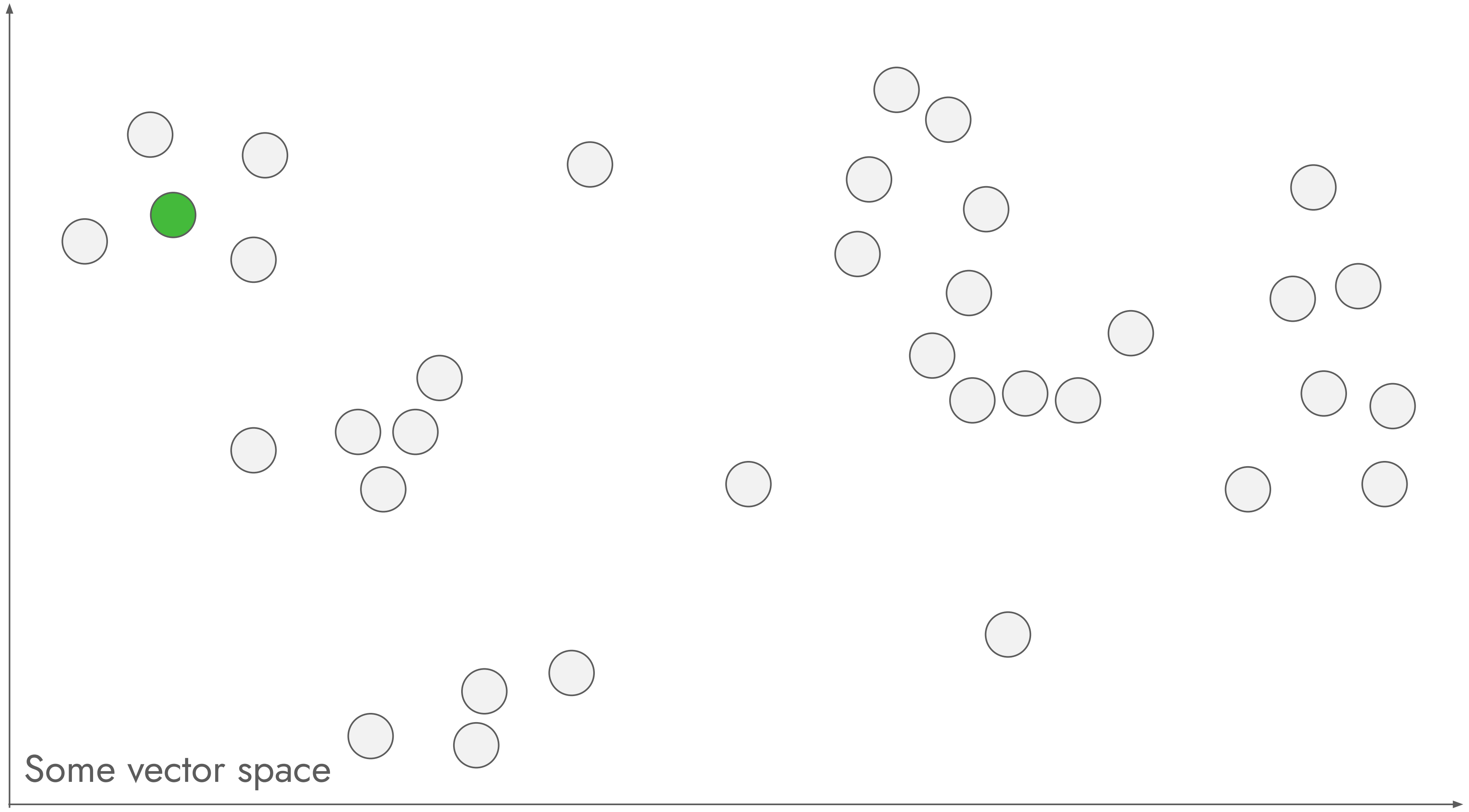
TECHNICAL UNIVERSITY OF MUNICH

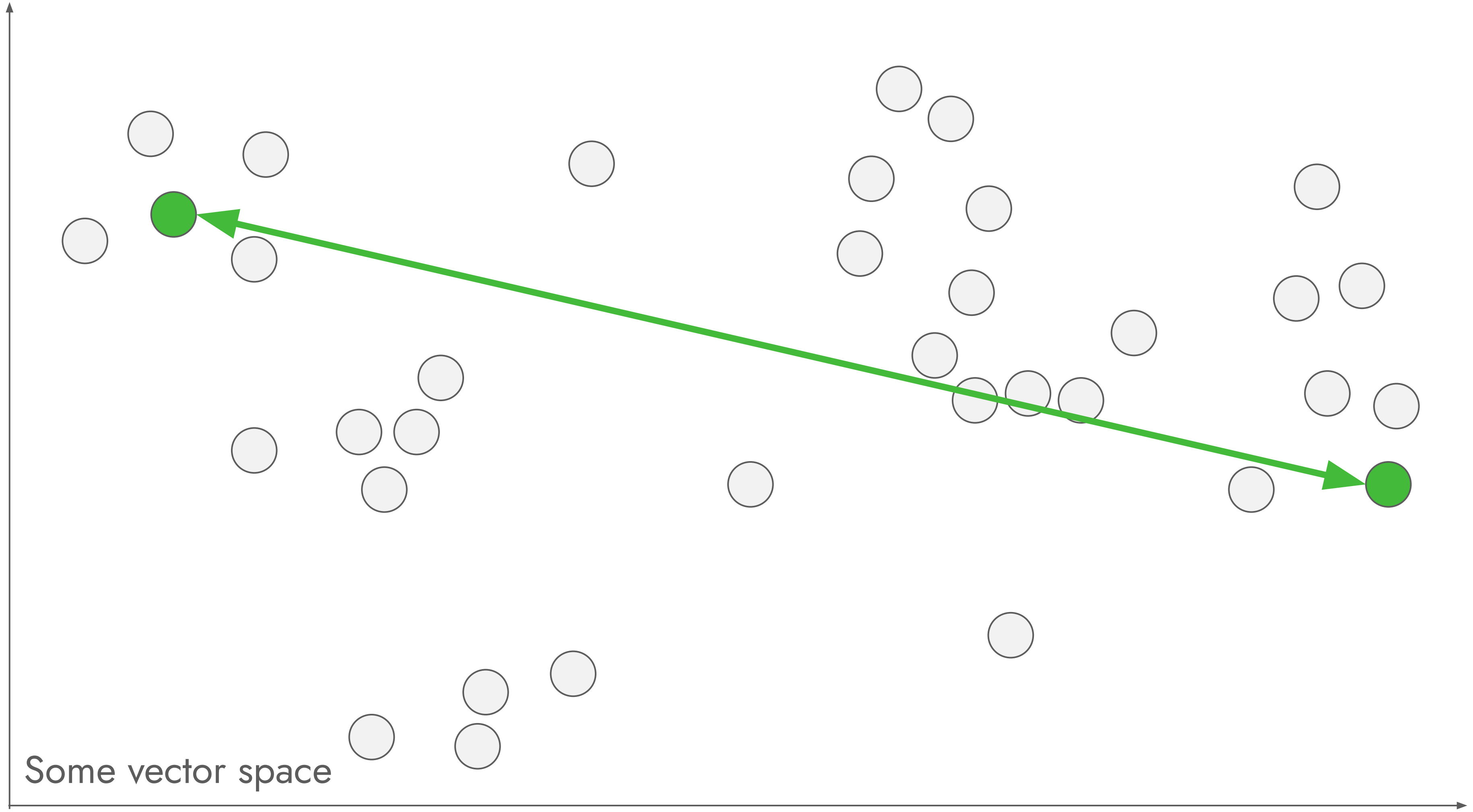
Master's Thesis in Informatics

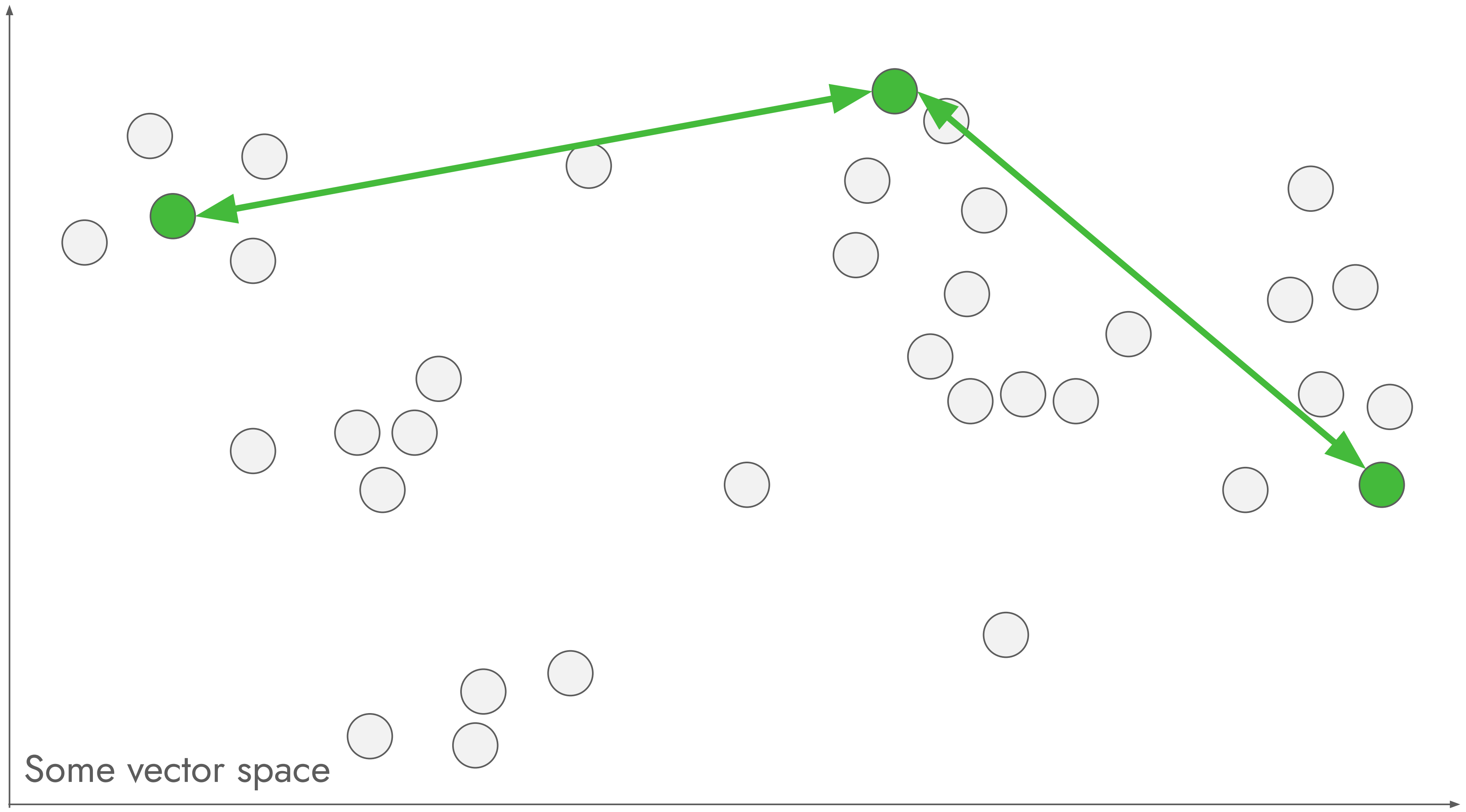
An Investigation on the Usage of Source Code Embeddings in Test Case Prioritization and Selection

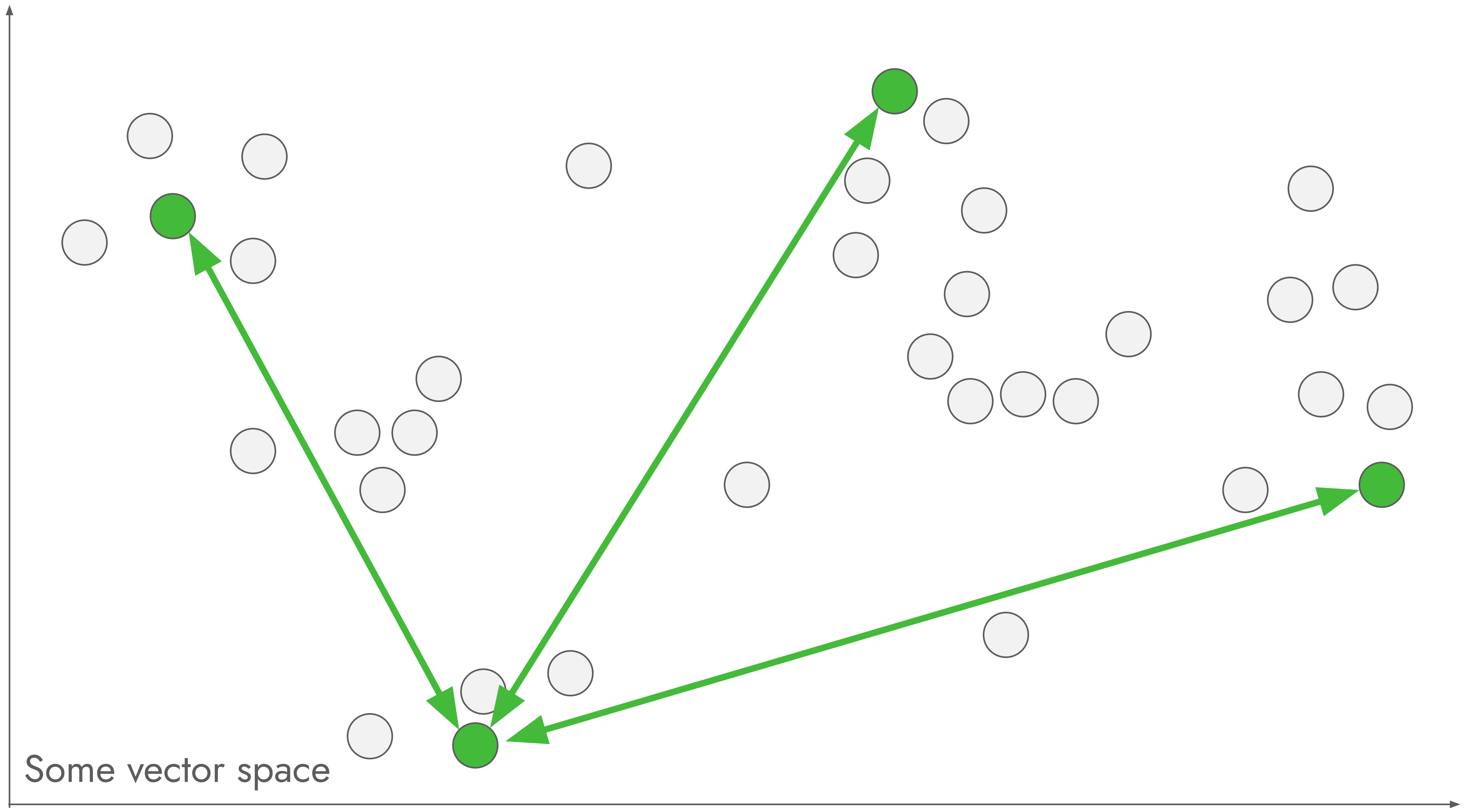
Alessandro Escher


[63, 100]
[3, 68]
[105, 24]
Some vector space

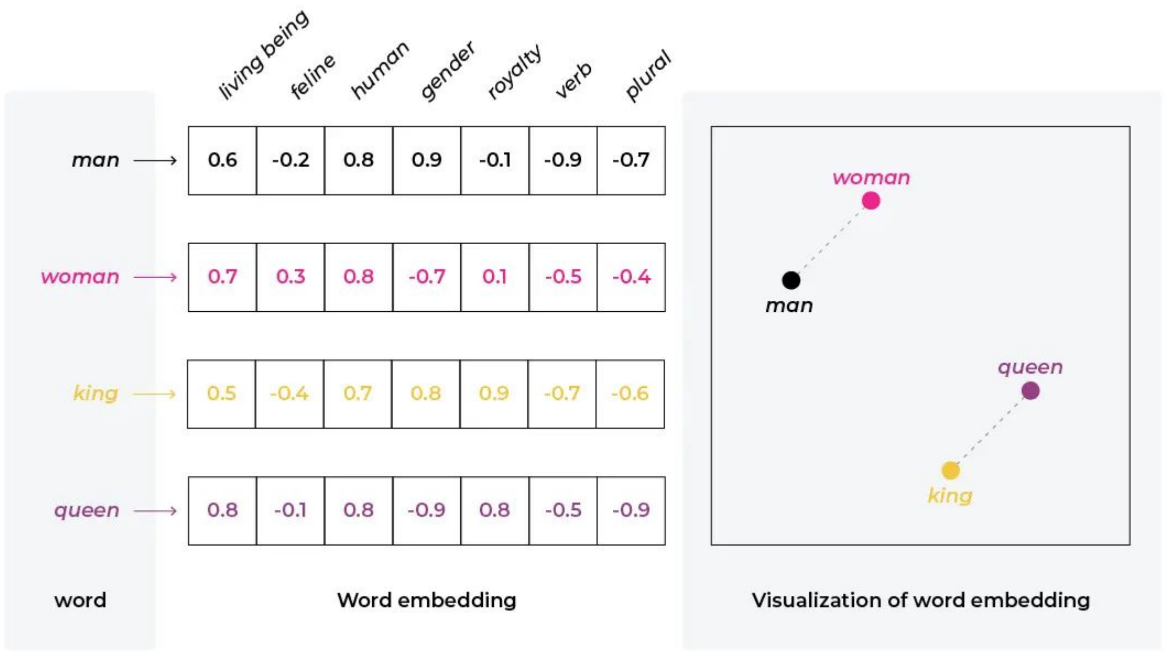

Some vector space


Some vector space


Some vector space

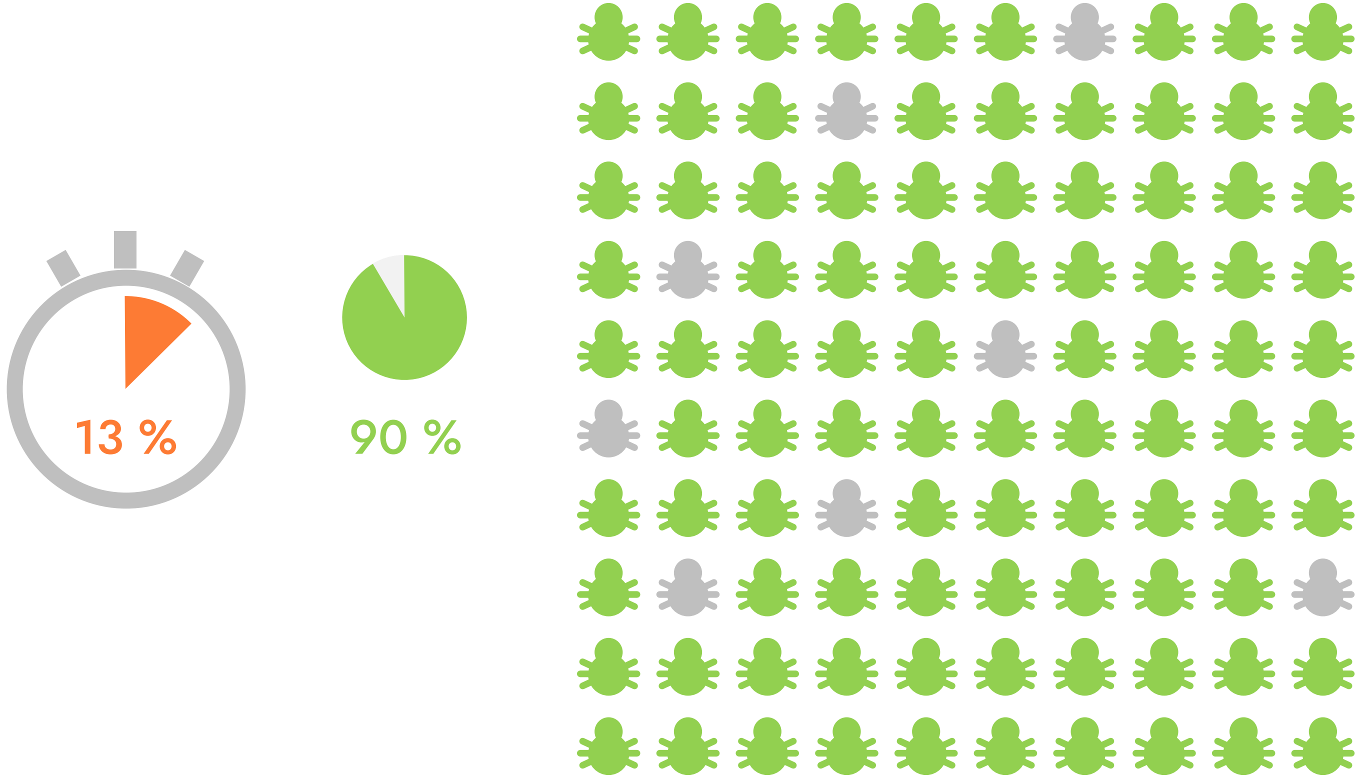

Some vector space

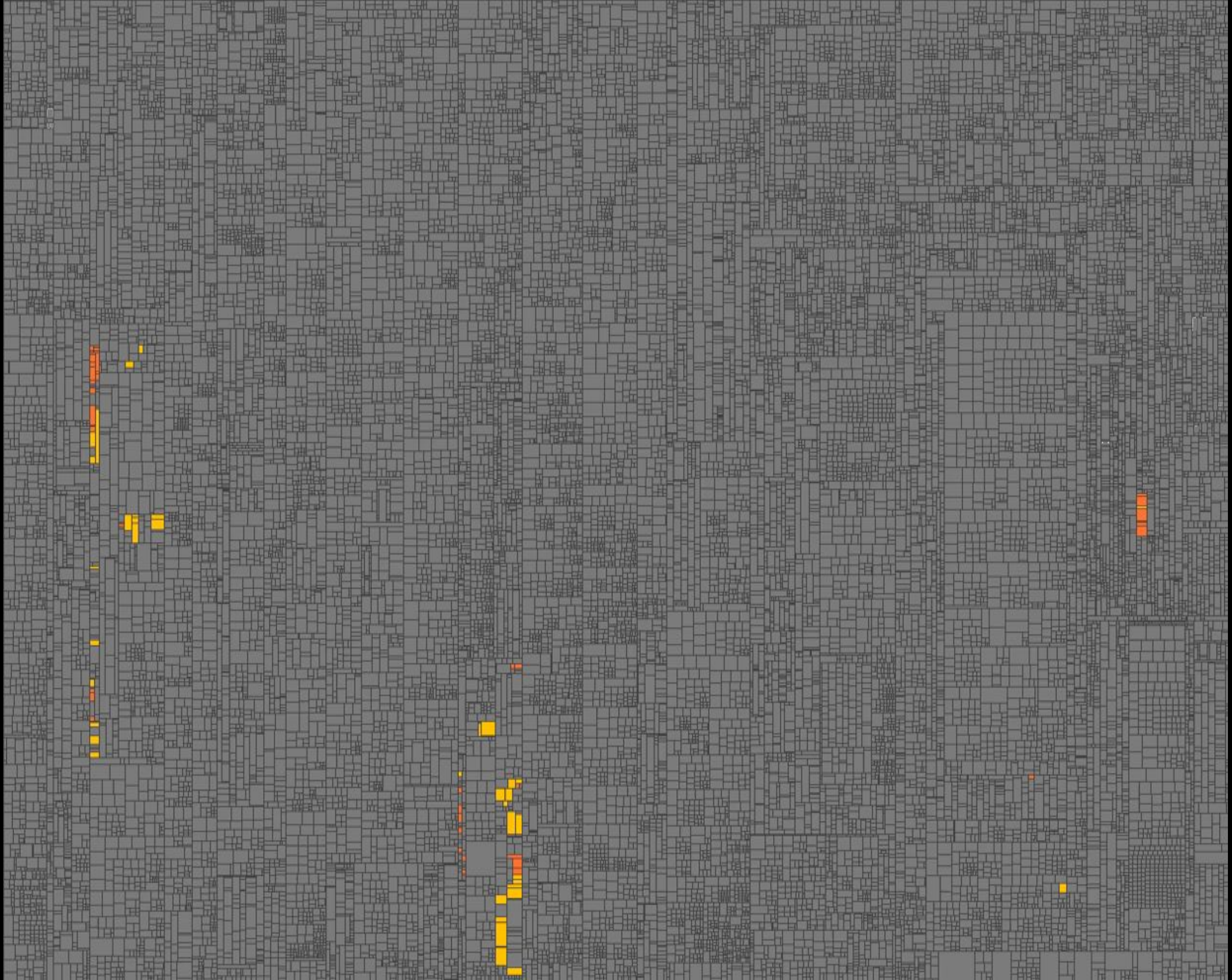
Large Language ("AI") Models

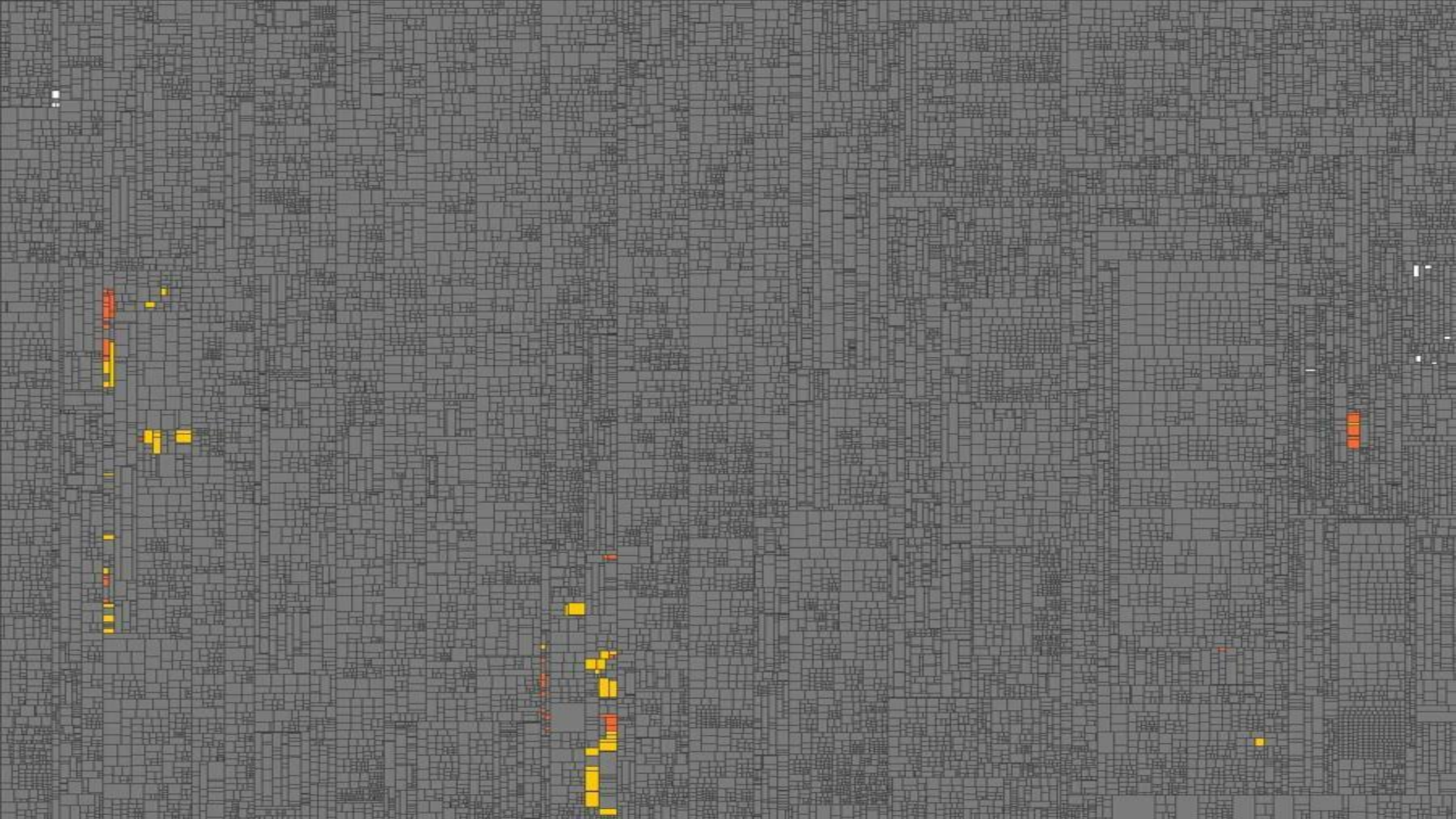


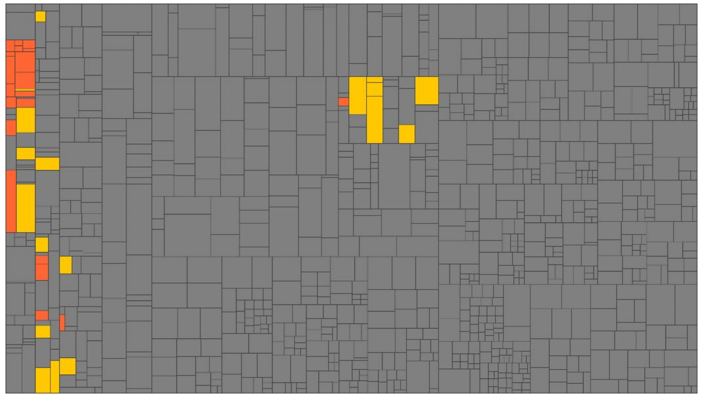
word	living being	feline	human	gender	royalty	verb	plural
man	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman	0.7	0.3	0.8	-0.7	0.1	-0.5	-0.4
king	0.5	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen	0.8	-0.1	0.8	-0.9	0.8	-0.5	-0.9

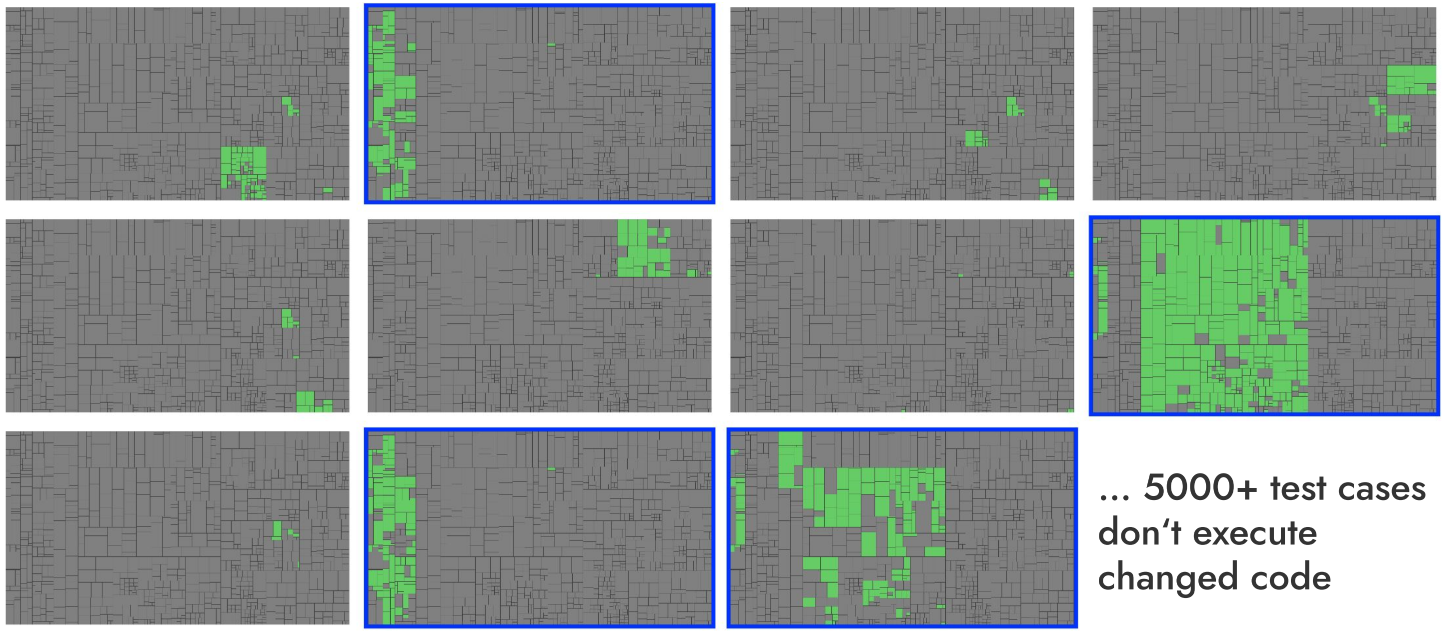
AI Test Clustering





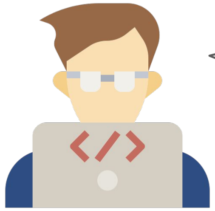






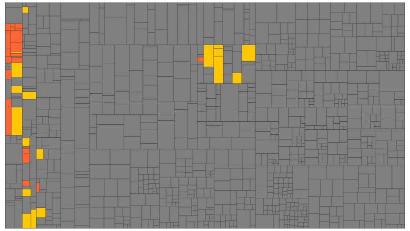
... 5000+ test cases don‘t execute changed code

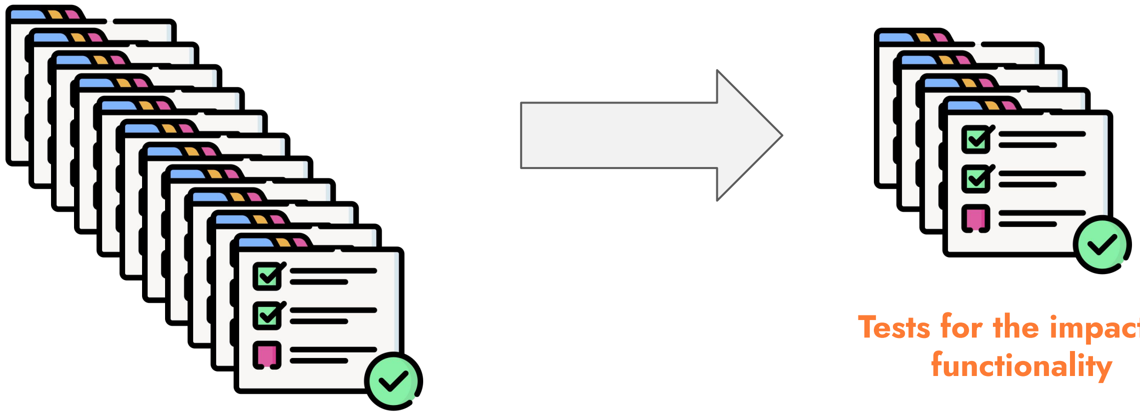
Change-based Testing



We changed **login, accounting and search.**

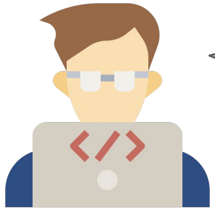
(user story, pull request, release, ...)





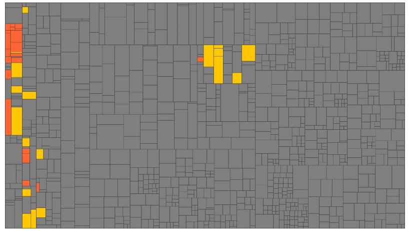
Tests for the impacted functionality

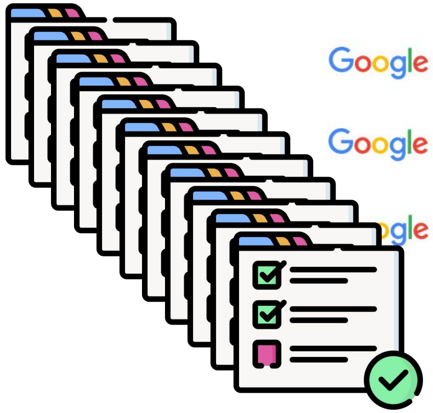
Change-based Testing

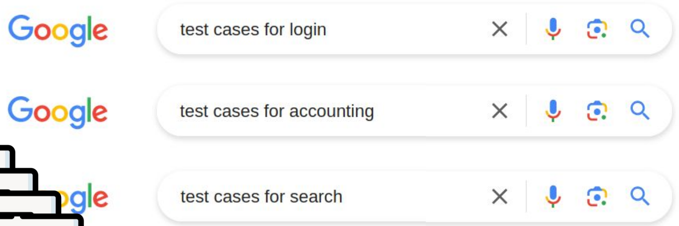


We changed **login, accounting and search.**

(user story, pull request, release, ...)



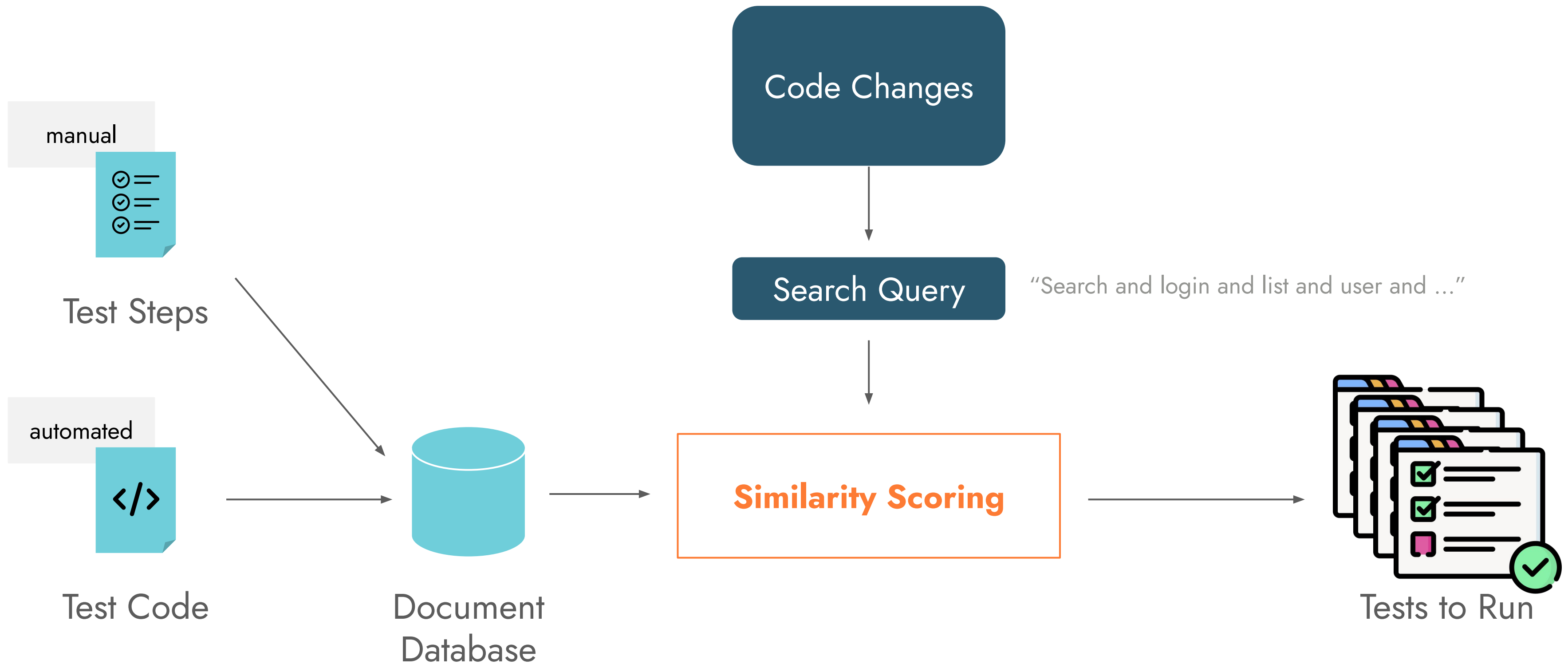


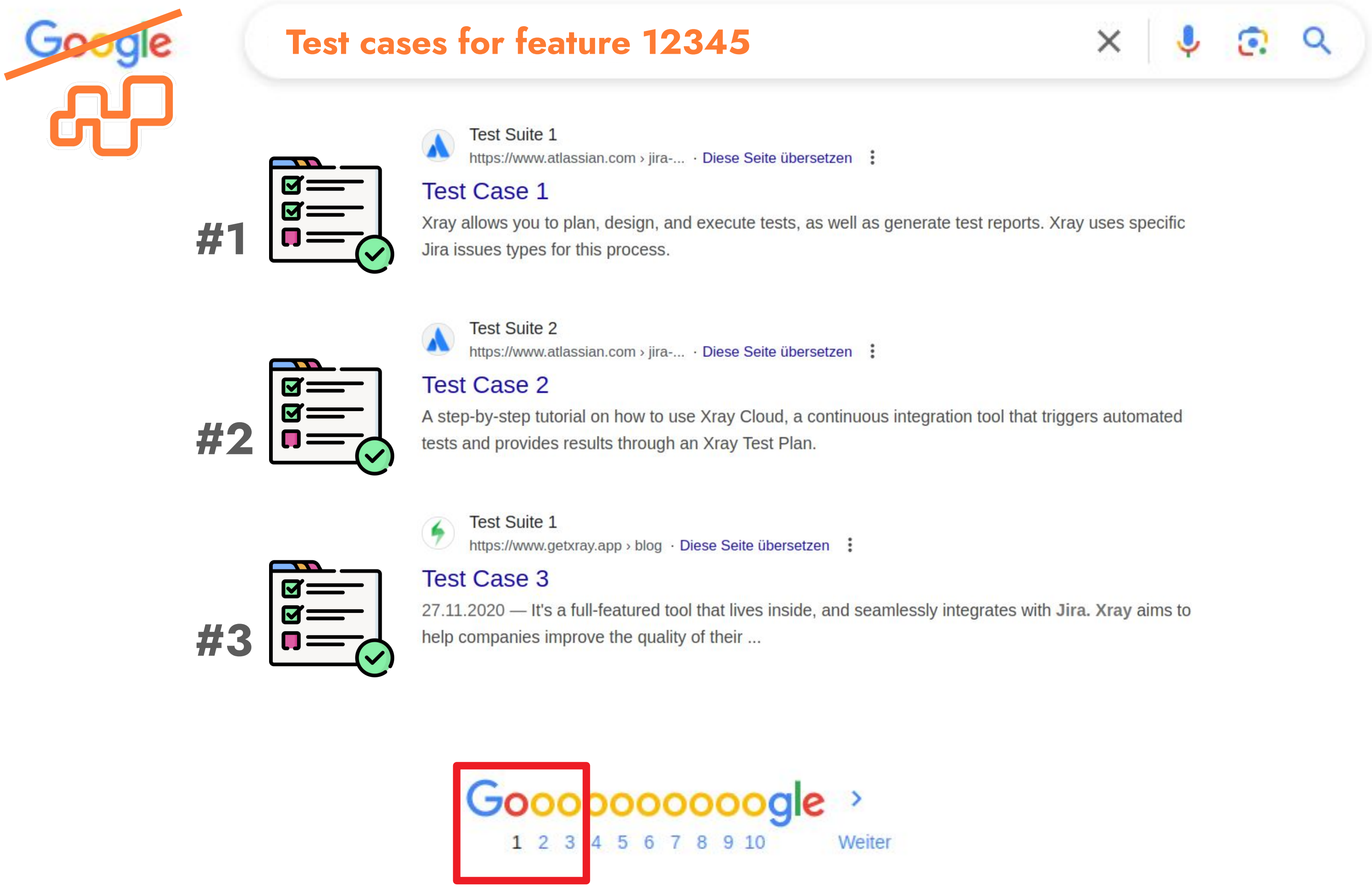

test cases for login

test cases for accounting

test cases for search

Tests for the impacted functionality


manual
Test Steps
automated
Test Code
Document Database
Code Changes
Search Query
"Search and login and list and user and ..."
Similarity Scoring
Tests to Run



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.

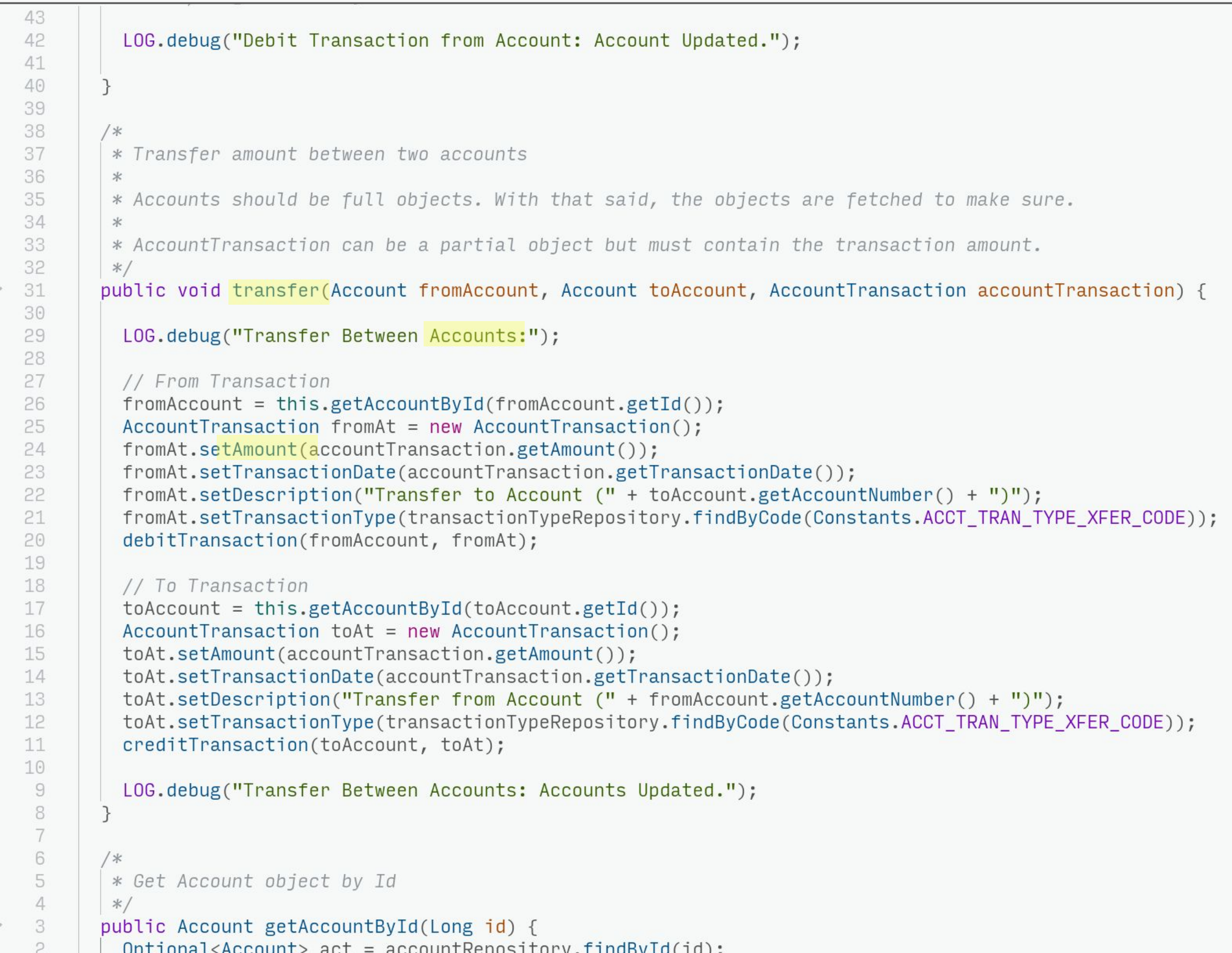
#3

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 ...

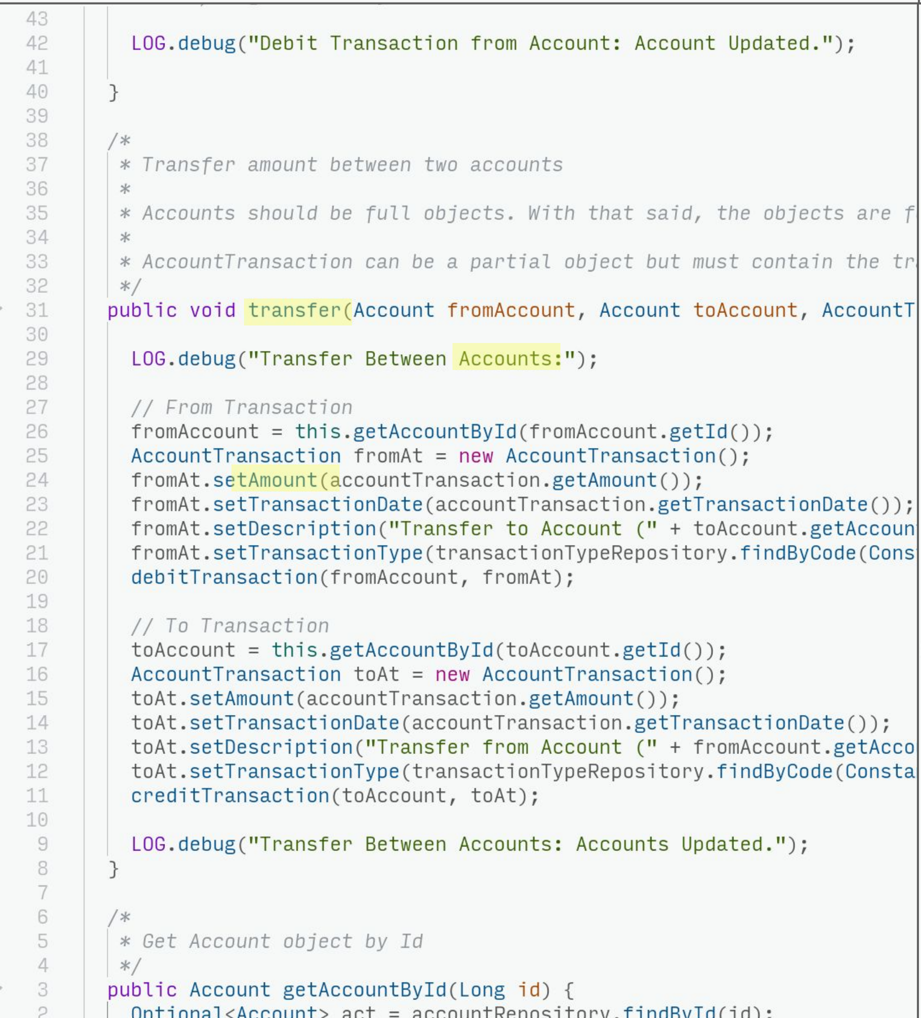
Changed Code



```java
  LOG.debug("Debit Transaction from Account: Account Updated.");

}

/*
 * Transfer amount between two accounts
 *
 * Accounts should be full objects. With that said, the objects are fetched to make sure.
 *
 * AccountTransaction can be a partial object but must contain the transaction amount.
 */
public void transfer(Account fromAccount, Account toAccount, AccountTransaction accountTransaction) {

  LOG.debug("Transfer Between Accounts:");

  // From Transaction
  fromAccount = this.getAccountById(fromAccount.getId());
  AccountTransaction fromAt = new AccountTransaction();
  fromAt.setAmount(accountTransaction.getAmount());
  fromAt.setTransactionDate(accountTransaction.getTransactionDate());
  fromAt.setDescription("Transfer to Account (" + toAccount.getAccountNumber() + ")");
  fromAt.setTransactionType(transactionTypeRepository.findByCode(Constants.ACCT_TRAN_TYPE_XFER_CODE));
  debitTransaction(fromAccount, fromAt);

  // To Transaction
  toAccount = this.getAccountById(toAccount.getId());
  AccountTransaction toAt = new AccountTransaction();
  toAt.setAmount(accountTransaction.getAmount());
  toAt.setTransactionDate(accountTransaction.getTransactionDate());
  toAt.setDescription("Transfer from Account (" + fromAccount.getAccountNumber() + ")");
  toAt.setTransactionType(transactionTypeRepository.findByCode(Constants.ACCT_TRAN_TYPE_XFER_CODE));
  creditTransaction(toAccount, toAt);

  LOG.debug("Transfer Between Accounts: Accounts Updated.");
}

/*
 * Get Account object by Id
 */
public Account getAccountById(Long id) {
  Optional<Account> act = accountRepository.findById(id);
```

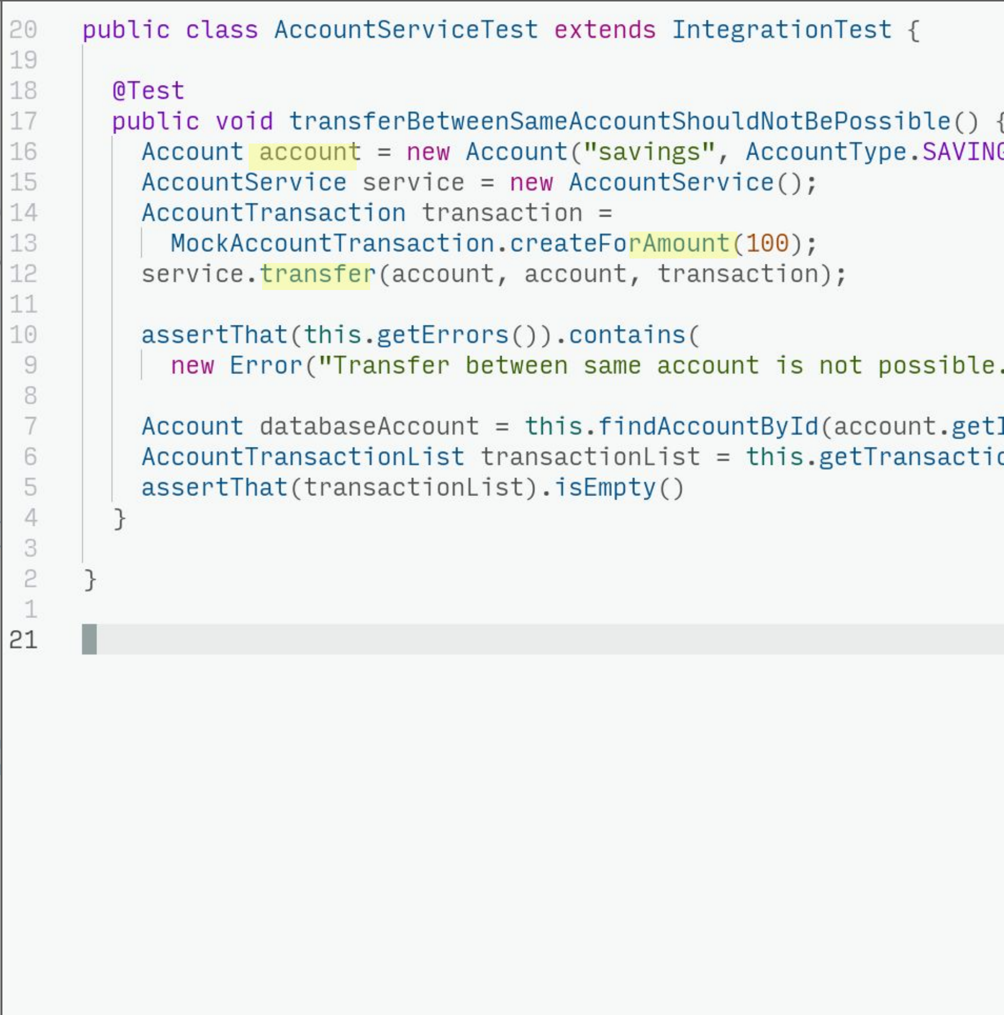
Digibank

Changed Code



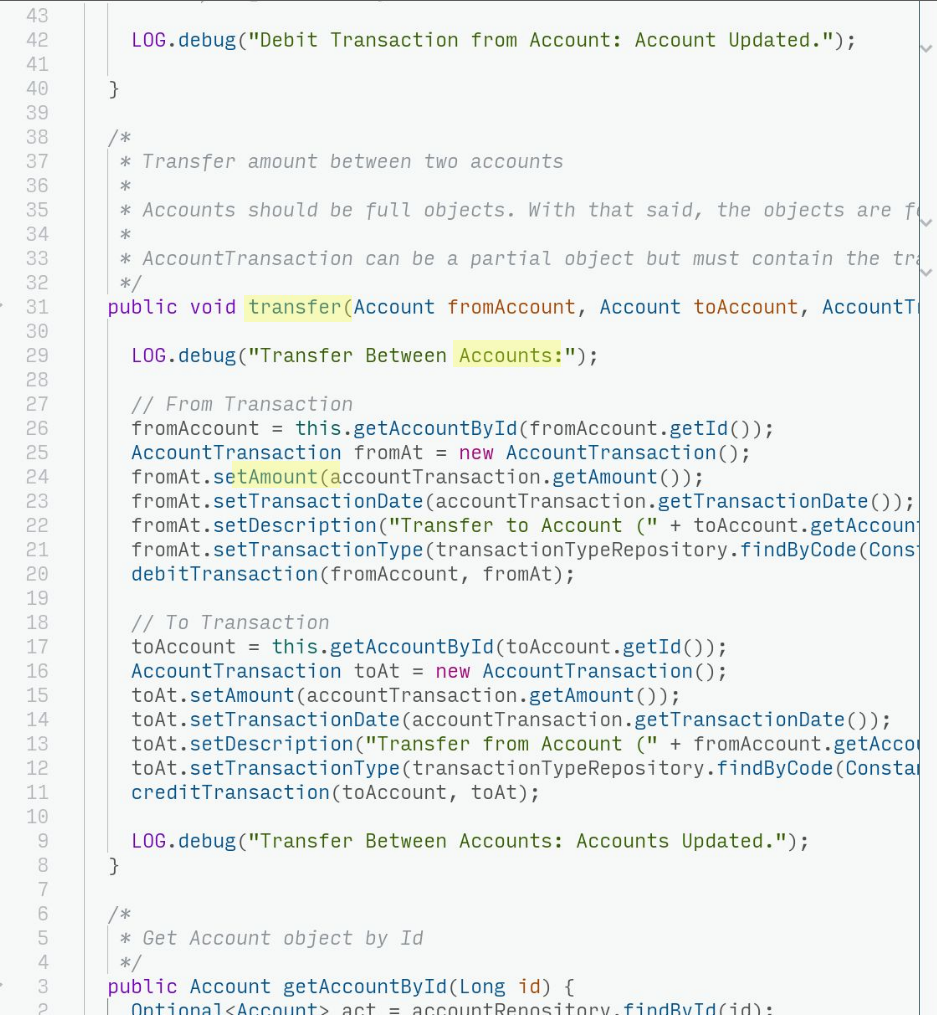
```
    LOG.debug("Debit Transaction from Account: Account Updated.");

  }

  /*
   * Transfer amount between two accounts
   *
   * Accounts should be full objects. With that said, the objects are f
   *
   * AccountTransaction can be a partial object but must contain the tr
   */
  public void transfer(Account fromAccount, Account toAccount, AccountT

    LOG.debug("Transfer Between Accounts:");

    // From Transaction
    fromAccount = this.getAccountById(fromAccount.getId());
    AccountTransaction fromAt = new AccountTransaction();
    fromAt.setAmount(accountTransaction.getAmount());
    fromAt.setTransactionDate(accountTransaction.getTransactionDate());
    fromAt.setDescription("Transfer to Account (" + toAccount.getAccoun
    fromAt.setTransactionType(transactionTypeRepository.findByCode(Cons
    debitTransaction(fromAccount, fromAt);

    // To Transaction
    toAccount = this.getAccountById(toAccount.getId());
    AccountTransaction toAt = new AccountTransaction();
    toAt.setAmount(accountTransaction.getAmount());
    toAt.setTransactionDate(accountTransaction.getTransactionDate());
    toAt.setDescription("Transfer from Account (" + fromAccount.getAcco
    toAt.setTransactionType(transactionTypeRepository.findByCode(Consta
    creditTransaction(toAccount, toAt);

    LOG.debug("Transfer Between Accounts: Accounts Updated.");
  }

  /*
   * Get Account object by Id
   */
  public Account getAccountById(Long id) {
    Optional<Account> act = accountRepository.findById(id);
```

Code Test



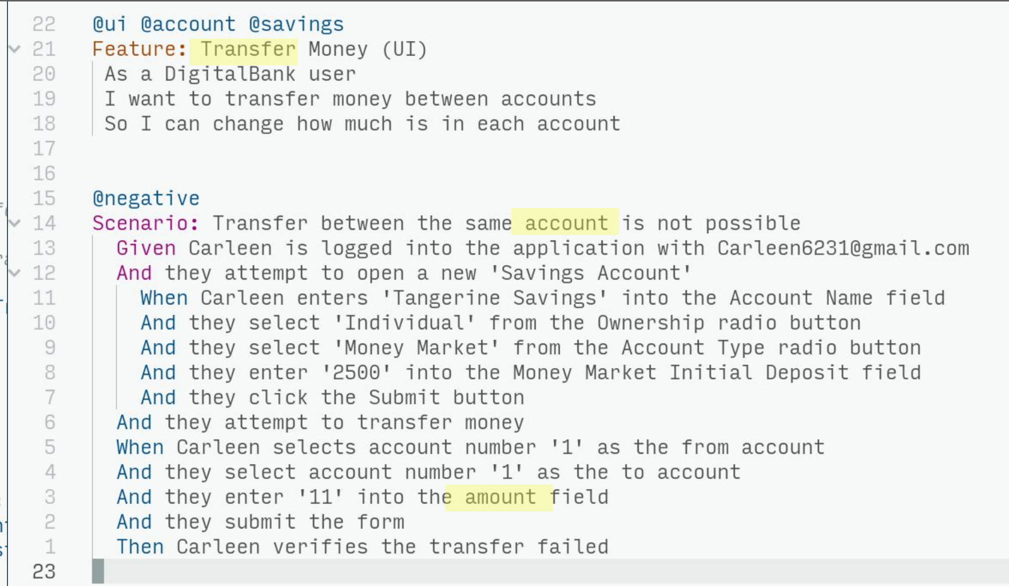
```
public class AccountServiceTest extends IntegrationTest {

  @Test
  public void transferBetweenSameAccountShouldNotBePossible() {
    Account account = new Account("savings", AccountType.SAVING
    AccountService service = new AccountService();
    AccountTransaction transaction =
      MockAccountTransaction.createForAmount(100);
    service.transfer(account, account, transaction);

    assertThat(this.getErrors()).contains(
      new Error("Transfer between same account is not possible.

    Account databaseAccount = this.findAccountById(account.getI
    AccountTransactionList transactionList = this.getTransactio
    assertThat(transactionList).isEmpty()
  }

}

```

Changed Code



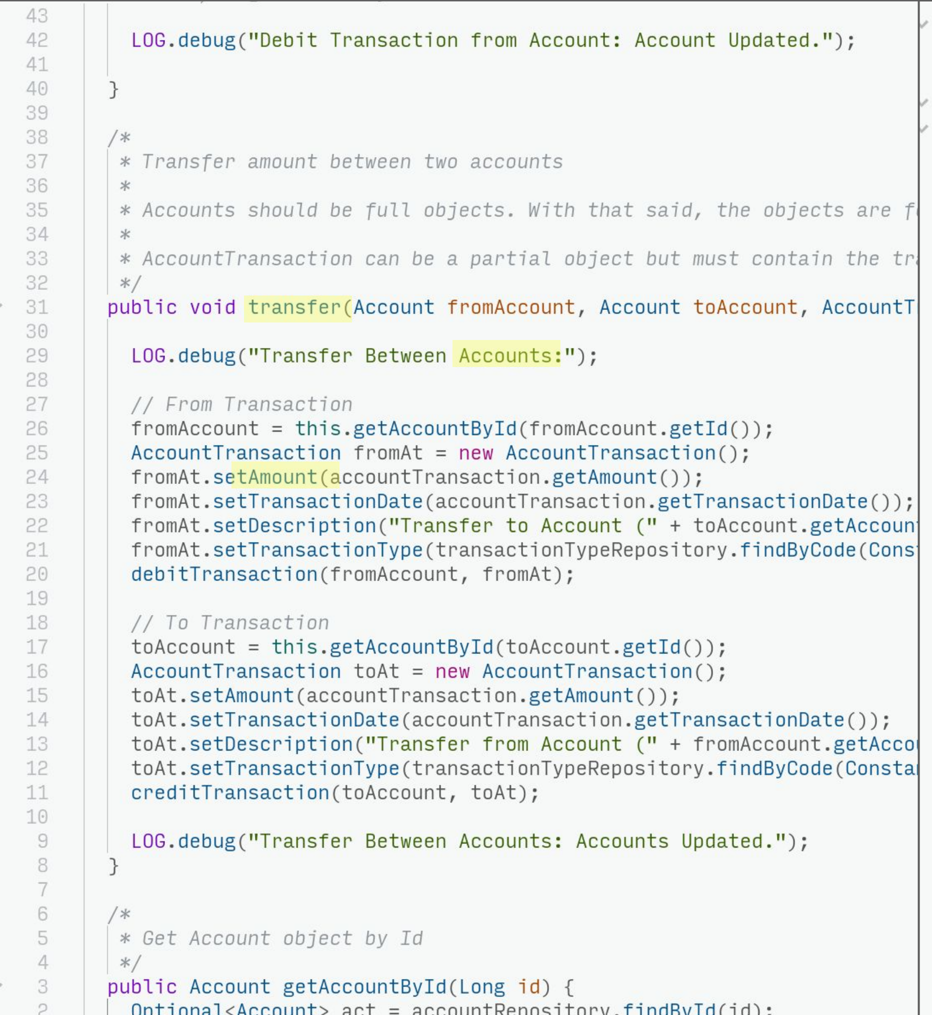
```
    LOG.debug("Debit Transaction from Account: Account Updated.");

  }

  /*
   * Transfer amount between two accounts
   *
   * Accounts should be full objects. With that said, the objects are f
   *
   * AccountTransaction can be a partial object but must contain the tr
   */
  public void transfer(Account fromAccount, Account toAccount, AccountT

    LOG.debug("Transfer Between Accounts:");

    // From Transaction
    fromAccount = this.getAccountById(fromAccount.getId());
    AccountTransaction fromAt = new AccountTransaction();
    fromAt.setAmount(accountTransaction.getAmount());
    fromAt.setTransactionDate(accountTransaction.getTransactionDate());
    fromAt.setDescription("Transfer to Account (" + toAccount.getAccoun
    fromAt.setTransactionType(transactionTypeRepository.findByCode(Cons
    debitTransaction(fromAccount, fromAt);

    // To Transaction
    toAccount = this.getAccountById(toAccount.getId());
    AccountTransaction toAt = new AccountTransaction();
    toAt.setAmount(accountTransaction.getAmount());
    toAt.setTransactionDate(accountTransaction.getTransactionDate());
    toAt.setDescription("Transfer from Account (" + fromAccount.getAcco
    toAt.setTransactionType(transactionTypeRepository.findByCode(Consta
    creditTransaction(toAccount, toAt);

    LOG.debug("Transfer Between Accounts: Accounts Updated.");
  }

  /*
   * Get Account object by Id
   */
  public Account getAccountById(Long id) {
    Optional<Account> act = accountRepository.findById(id);
```

Cucumber Test



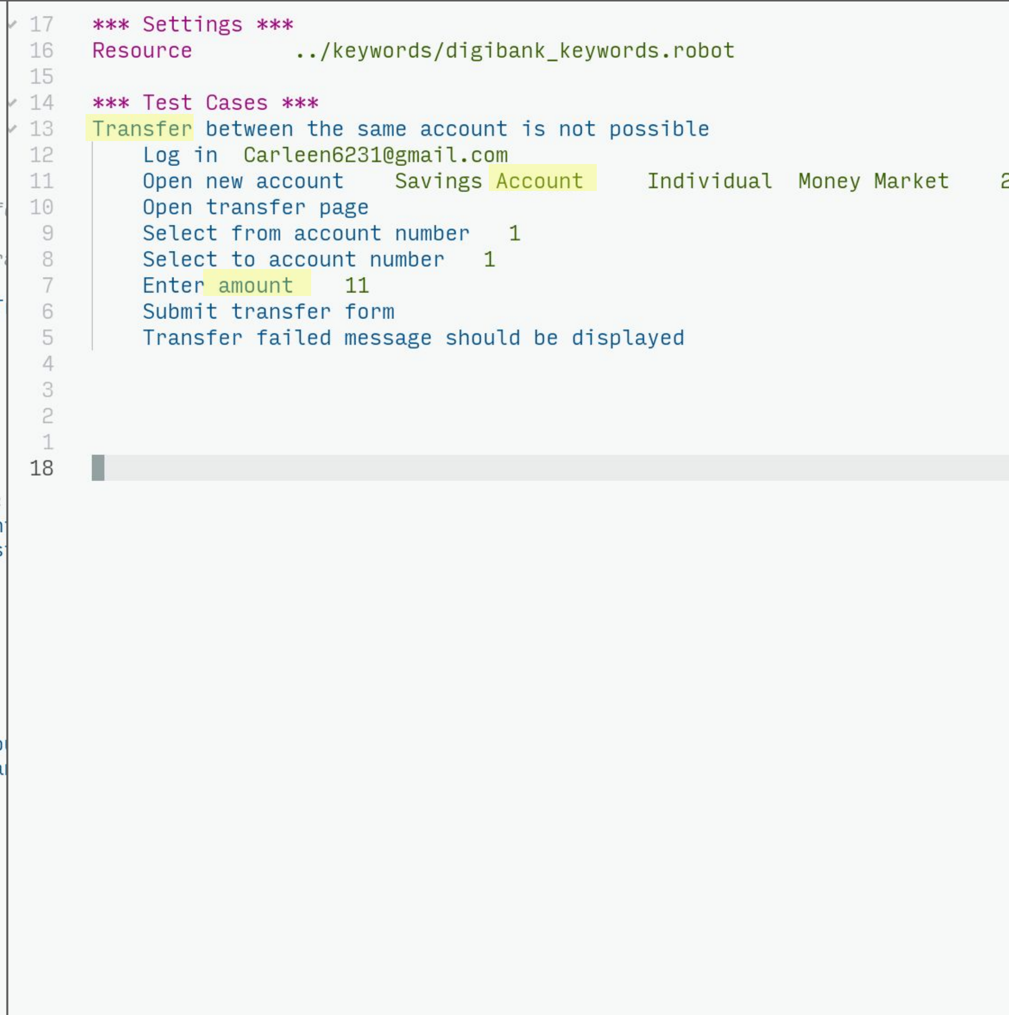
```
@ui @account @savings
Feature: Transfer Money (UI)
 As a DigitalBank user
 I want to transfer money between accounts
 So I can change how much is in each account


@negative
Scenario: Transfer between the same account is not possible
  Given Carleen is logged into the application with Carleen6231@gmail.com
  And they attempt to open a new 'Savings Account'
    When Carleen enters 'Tangerine Savings' into the Account Name field
    And they select 'Individual' from the Ownership radio button
    And they select 'Money Market' from the Account Type radio button
    And they enter '2500' into the Money Market Initial Deposit field
    And they click the Submit button
  And they attempt to transfer money
  When Carleen selects account number '1' as the from account
  And they select account number '1' as the to account
  And they enter '11' into the amount field
  And they submit the form
  Then Carleen verifies the transfer failed
```

Changed Code



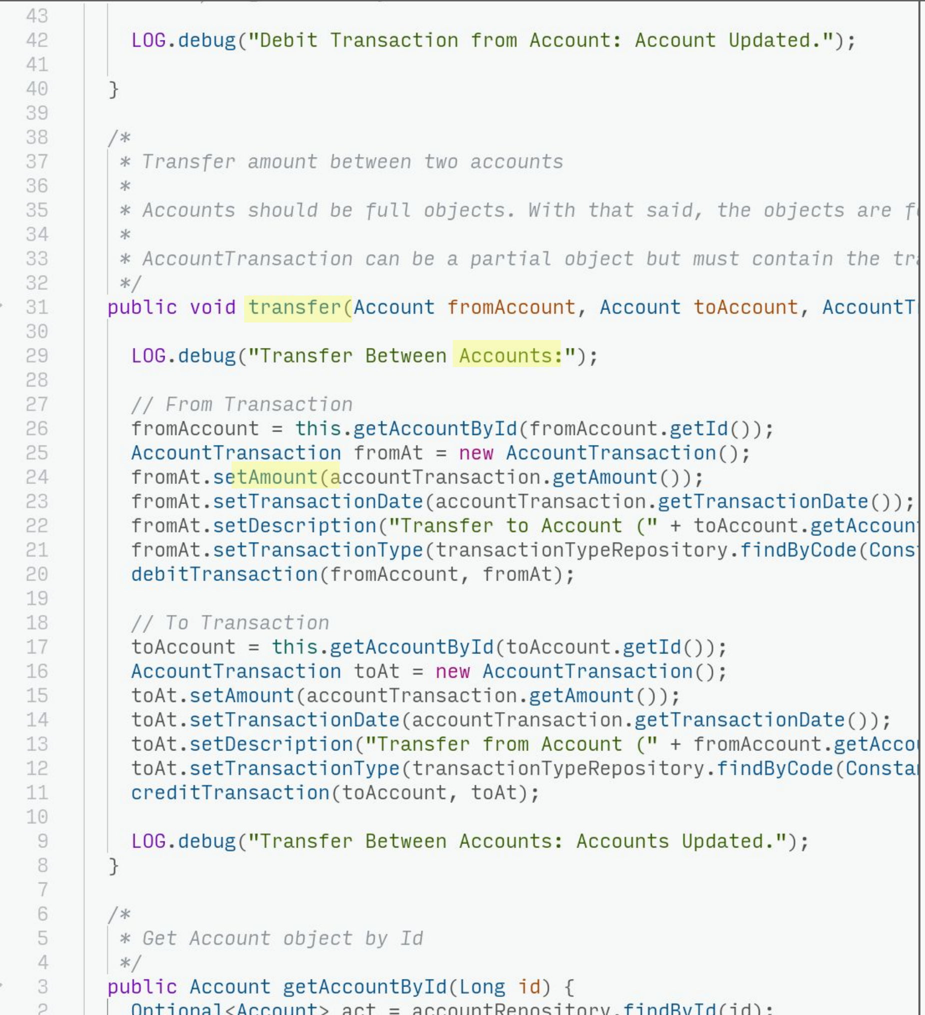
```
    LOG.debug("Debit Transaction from Account: Account Updated.");

  }

  /*
   * Transfer amount between two accounts
   *
   * Accounts should be full objects. With that said, the objects are f
   *
   * AccountTransaction can be a partial object but must contain the tr
   */
  public void transfer(Account fromAccount, Account toAccount, AccountT

    LOG.debug("Transfer Between Accounts:");

    // From Transaction
    fromAccount = this.getAccountById(fromAccount.getId());
    AccountTransaction fromAt = new AccountTransaction();
    fromAt.setAmount(accountTransaction.getAmount());
    fromAt.setTransactionDate(accountTransaction.getTransactionDate());
    fromAt.setDescription("Transfer to Account (" + toAccount.getAccoun
    fromAt.setTransactionType(transactionTypeRepository.findByCode(Cons
    debitTransaction(fromAccount, fromAt);

    // To Transaction
    toAccount = this.getAccountById(toAccount.getId());
    AccountTransaction toAt = new AccountTransaction();
    toAt.setAmount(accountTransaction.getAmount());
    toAt.setTransactionDate(accountTransaction.getTransactionDate());
    toAt.setDescription("Transfer from Account (" + fromAccount.getAcco
    toAt.setTransactionType(transactionTypeRepository.findByCode(Consta
    creditTransaction(toAccount, toAt);

    LOG.debug("Transfer Between Accounts: Accounts Updated.");
  }

  /*
   * Get Account object by Id
   */
  public Account getAccountById(Long id) {
    Optional<Account> act = accountRepository.findById(id);
```

Robot Test



```
*** Settings ***
Resource        ../keywords/digibank_keywords.robot

*** Test Cases ***
Transfer between the same account is not possible
    Log in   Carleen6231@gmail.com
    Open new account    Savings Account     Individual   Money Market    2
    Open transfer page
    Select from account number    1
    Select to account number    1
    Enter amount    11
    Submit transfer form
    Transfer failed message should be displayed
```

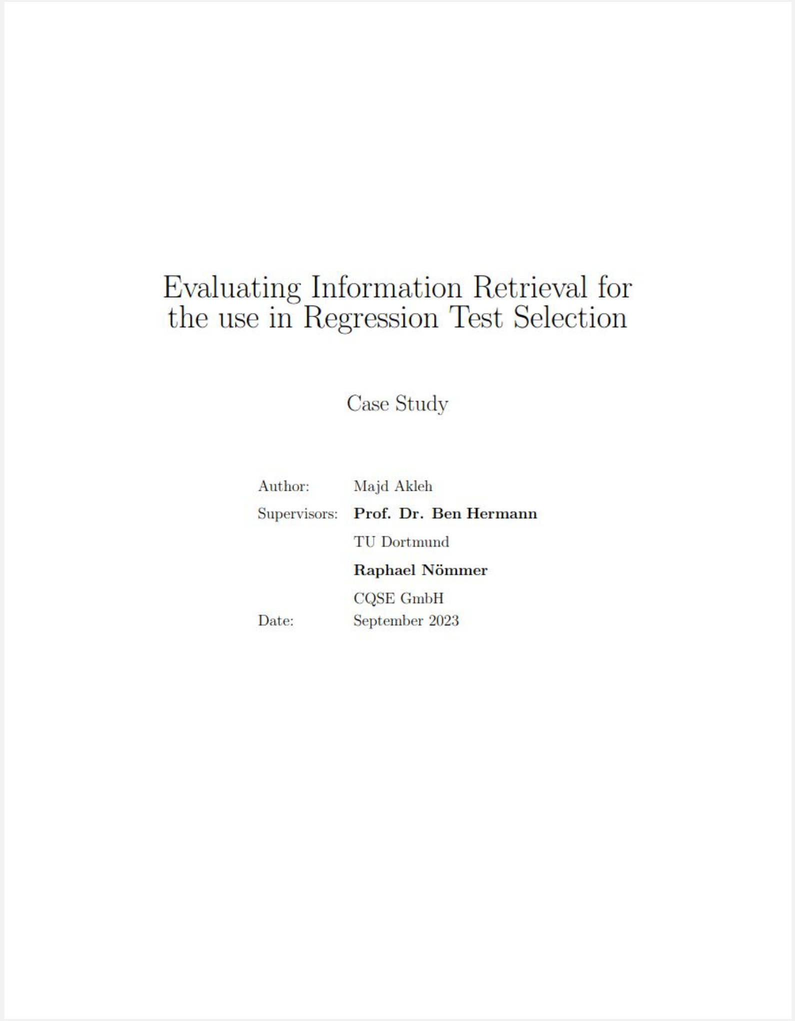
Changed Code



```
    LOG.debug("Debit Transaction from Account: Account Updated.");

  }

  /*
   * Transfer amount between two accounts
   *
   * Accounts should be full objects. With that said, the objects are f
   *
   * AccountTransaction can be a partial object but must contain the tr
   */
  public void transfer(Account fromAccount, Account toAccount, AccountT

    LOG.debug("Transfer Between Accounts:");

    // From Transaction
    fromAccount = this.getAccountById(fromAccount.getId());
    AccountTransaction fromAt = new AccountTransaction();
    fromAt.setAmount(accountTransaction.getAmount());
    fromAt.setTransactionDate(accountTransaction.getTransactionDate());
    fromAt.setDescription("Transfer to Account (" + toAccount.getAccoun
    fromAt.setTransactionType(transactionTypeRepository.findByCode(Cons
    debitTransaction(fromAccount, fromAt);

    // To Transaction
    toAccount = this.getAccountById(toAccount.getId());
    AccountTransaction toAt = new AccountTransaction();
    toAt.setAmount(accountTransaction.getAmount());
    toAt.setTransactionDate(accountTransaction.getTransactionDate());
    toAt.setDescription("Transfer from Account (" + fromAccount.getAcco
    toAt.setTransactionType(transactionTypeRepository.findByCode(Consta
    creditTransaction(toAccount, toAt);

    LOG.debug("Transfer Between Accounts: Accounts Updated.");
  }

  /*
   * Get Account object by Id
   */
  public Account getAccountById(Long id) {
    Optional<Account> act = accountRepository.findById(id);
```

Manual Test

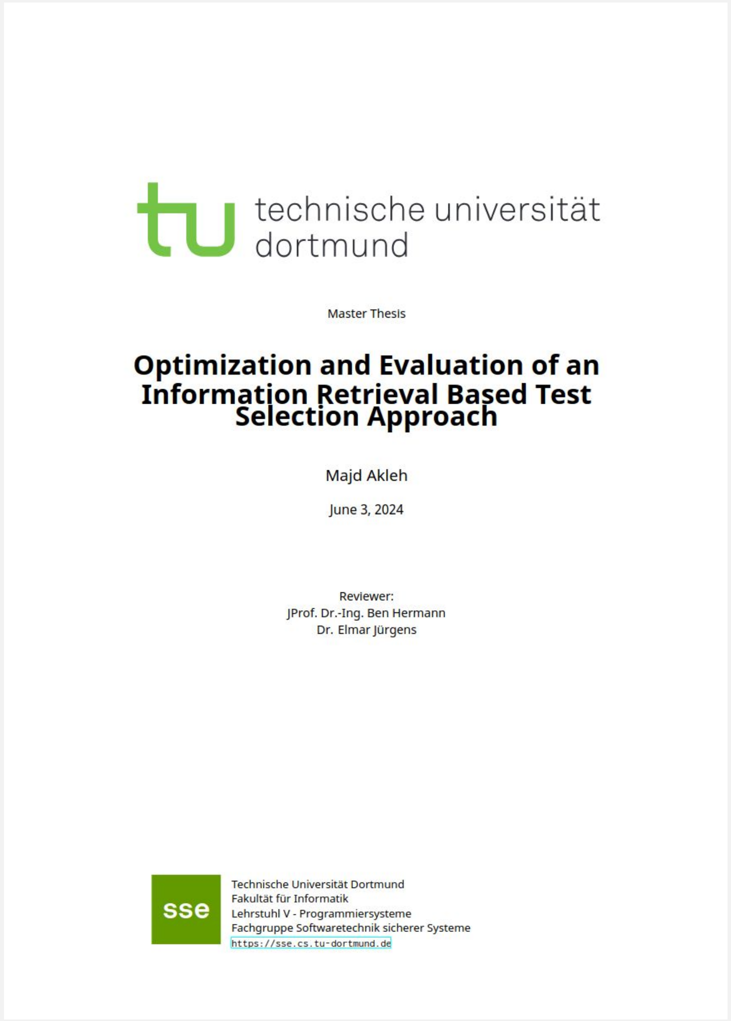
Action	Check
Log in as Carleen6231@gmail.com	
Open a new account: Type Savings Account, Individual In the Money Market Start deposit: 2500	Account was created as specified.
Open the transfer page.	
Select the account from step 2 as both from and to account.	
Enter amount: 11	
Submit the form	Transfer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



Evaluating Information Retrieval for the use in Regression Test Selection

Case Study

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



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

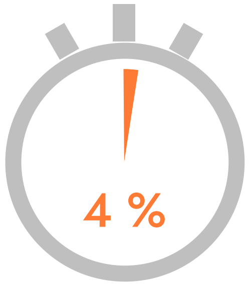
sse

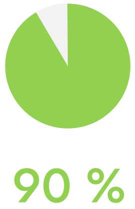
Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl V - Programmiersysteme
Fachgruppe Softwaretechnik sicherer Systeme
https://sse.cs.tu-dortmund.de

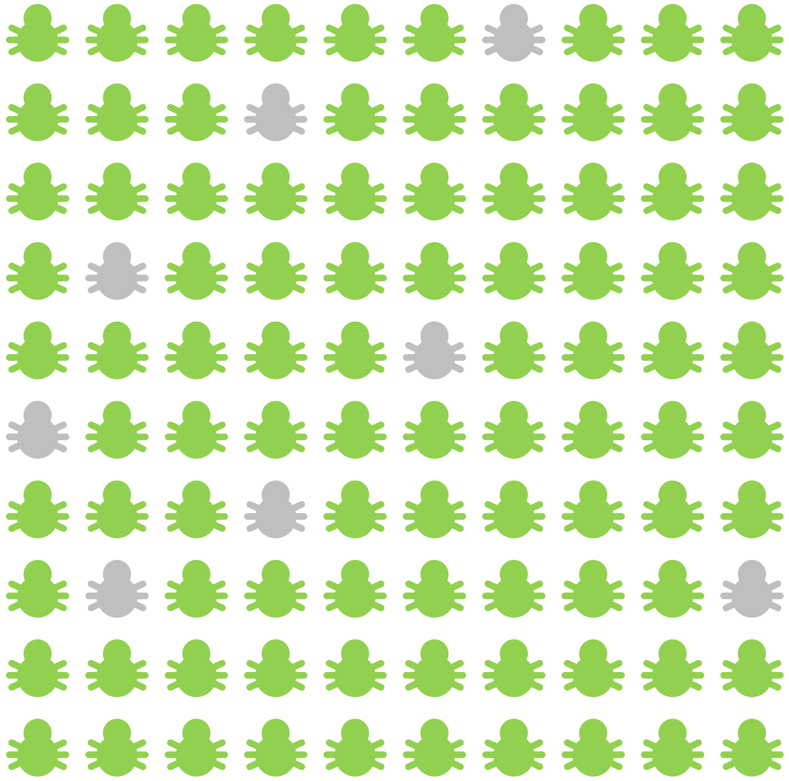
Score of one test for one search term =

$$\underbrace{\textbf{term frequency}}_{\substack{\text{How often the term appears in this test.} \\ \text{We reward repetition of terms in the test.}}} * \text{ inverse } \underbrace{\textbf{document frequency}}_{\substack{\text{How many of the tests contain the term.} \\ \text{We penalize terms that appear in many tests.}}}$$

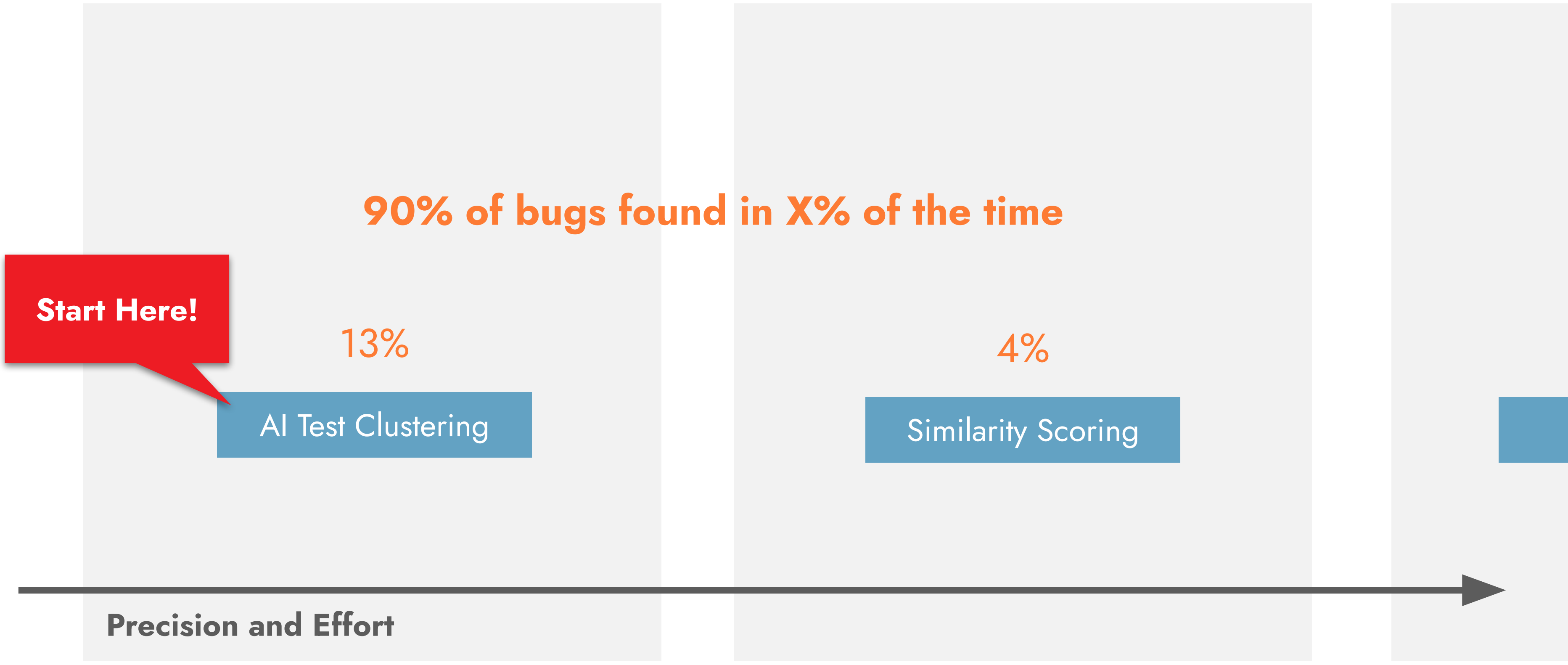
Similarity Scoring



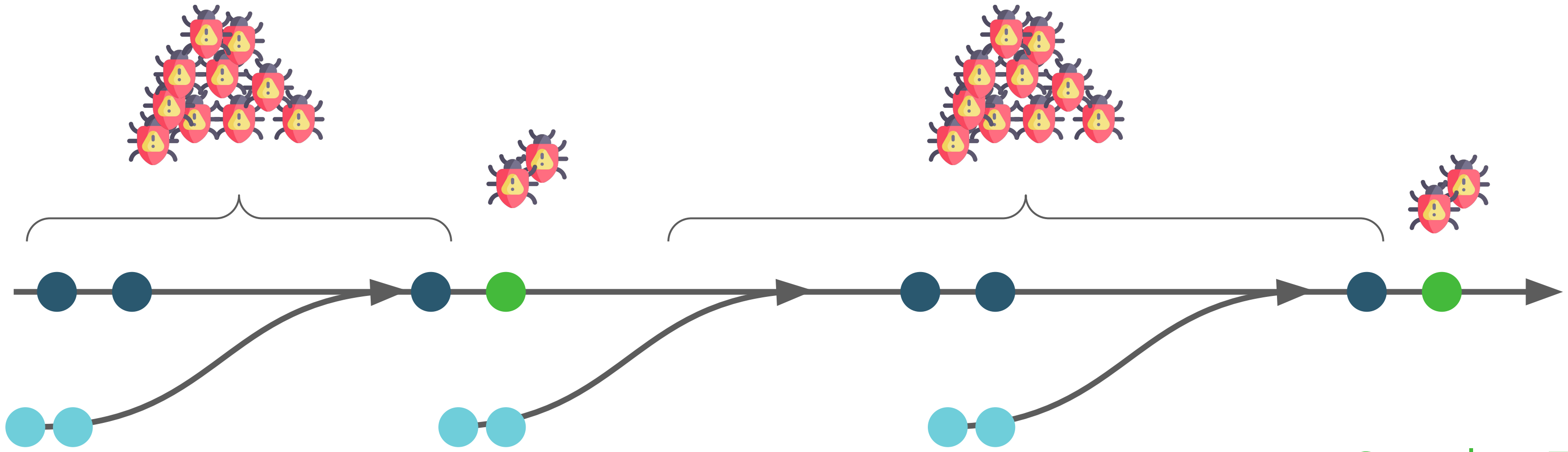




90% of bugs found in X% of the time



Shift Left



Complete Test Run

AI Test Clustering

Similarity Scoring

A4Q TESTING SUMMIT
EDITION 2025

Fabian Streitel

streitel@cqse.eu

CQSE

CQSE GmbH
Centa-Hafenbrädl-Straße 59
81249 München
www.teamscale.com

Contact me!