AutomationSTAR
ANTWERP ZOO 4-5 NOV. 2026

TRACK TALK

Honey, I Shrunk the Test Suite!

Sebastian Mörtl
PAYBACK Germany

Fabian Streitel
CQSE Germany



The Problem

The Approach

The Proof

The Benefit



Fabian Streitel

- Team Lead „Test Intelligence“
- Supervise **Bachelor and Master theses**
- **Analysed test suites** with dozens of teams

Teamscale

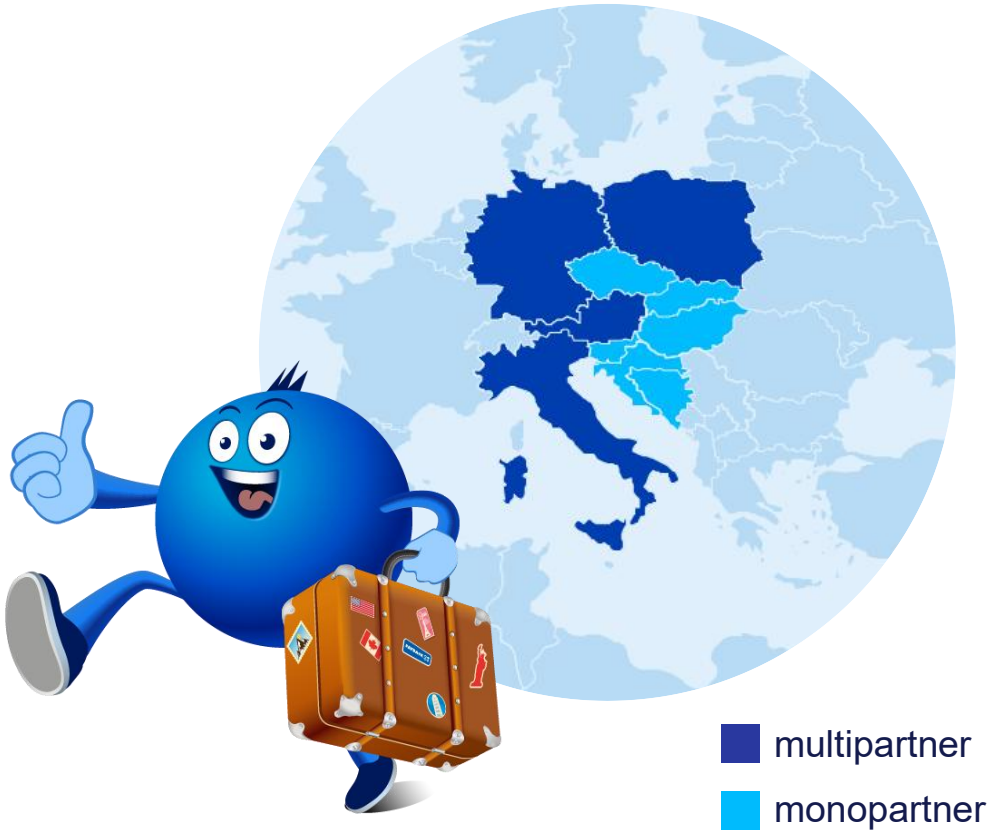


Sebastian Mörtl

- Test Architect
- Leader of the PAYBACK QA Guild

PAYBACK

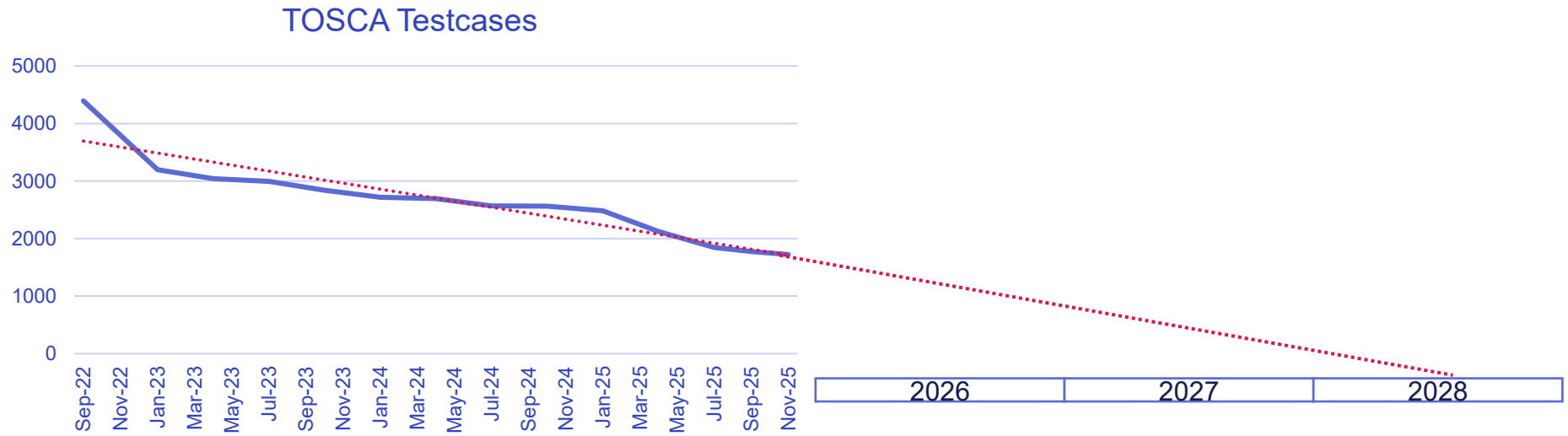
- **Largest bonus program** in Germany
- **9 out of 10** Germans know the brand
- More than **700 partners**
- **35 M active users**
- **18 M** active **mobile app-users**
- PAYBACK is active in **10 countries**



The problem

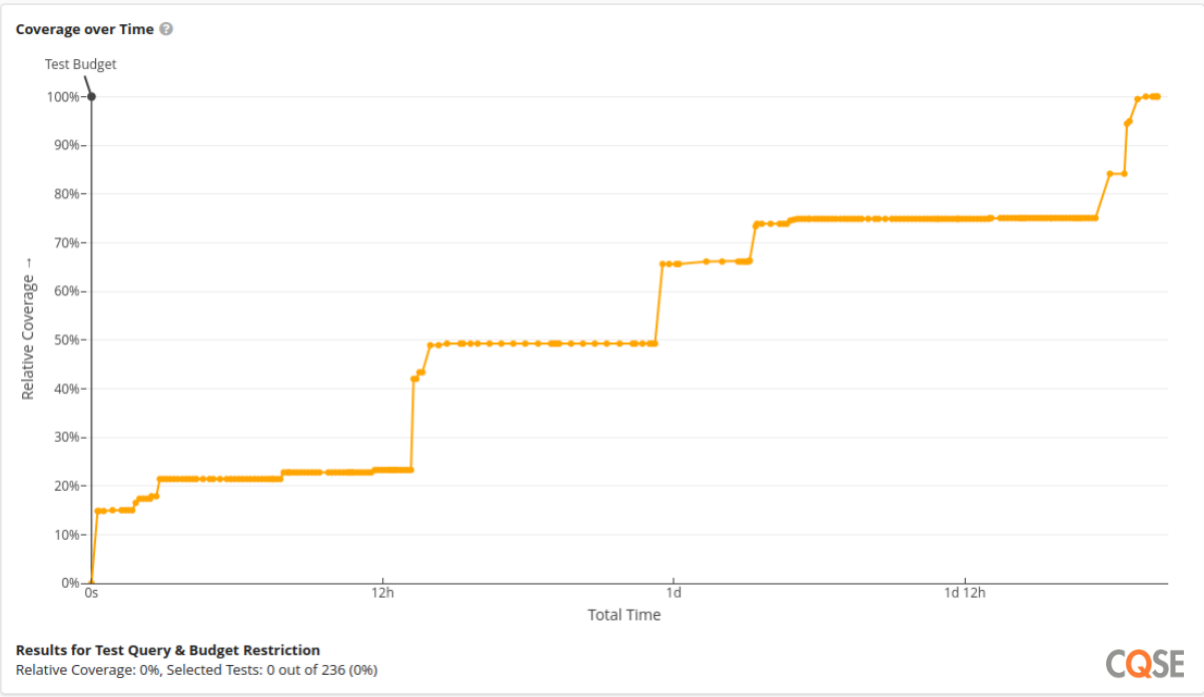
The situation in 2025

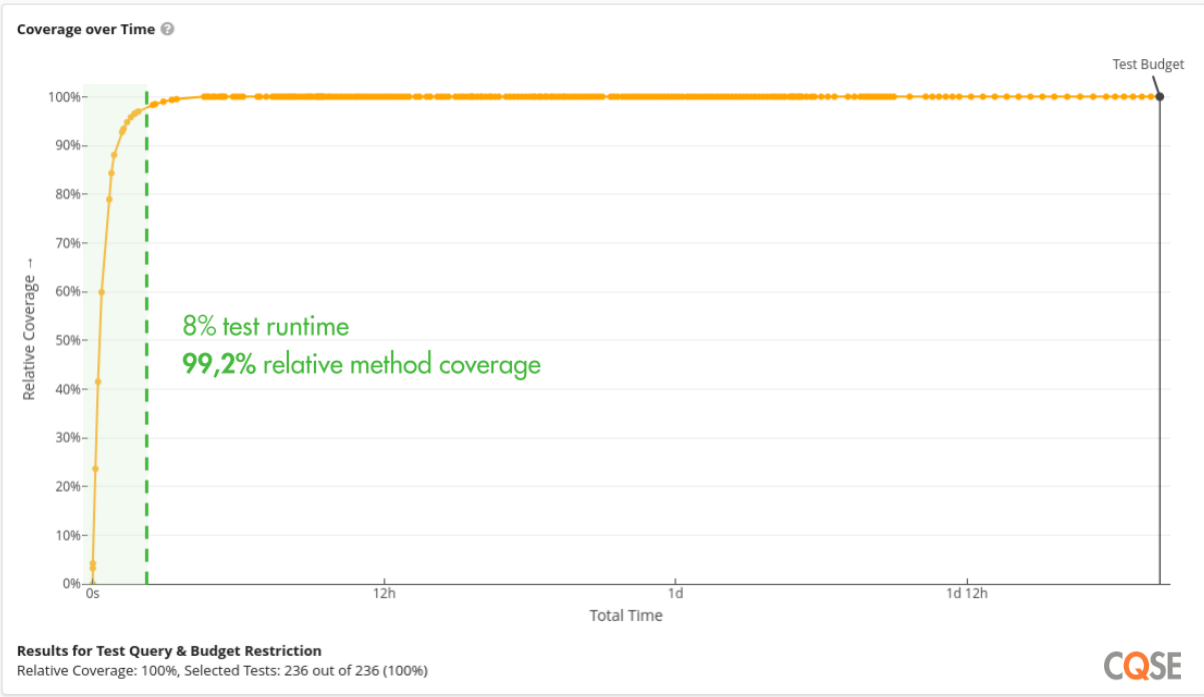
- TOSCA decommissioning decided in 2022
- No further updates → version completely outdated and out of support
- New automated tests implemented in other tools
- No active initiative to migrate remaining TOSCA test cases
- Each **bi-weekly** release costs **3 - 5 person days** executing, analysing, re-executing long running flaky TOSCA suites
- Coverage of TOSCA tests is unclear vs. coverage in our new test tools
- Migration too slow → **1723 test cases** left in November 2025



The origin of the idea

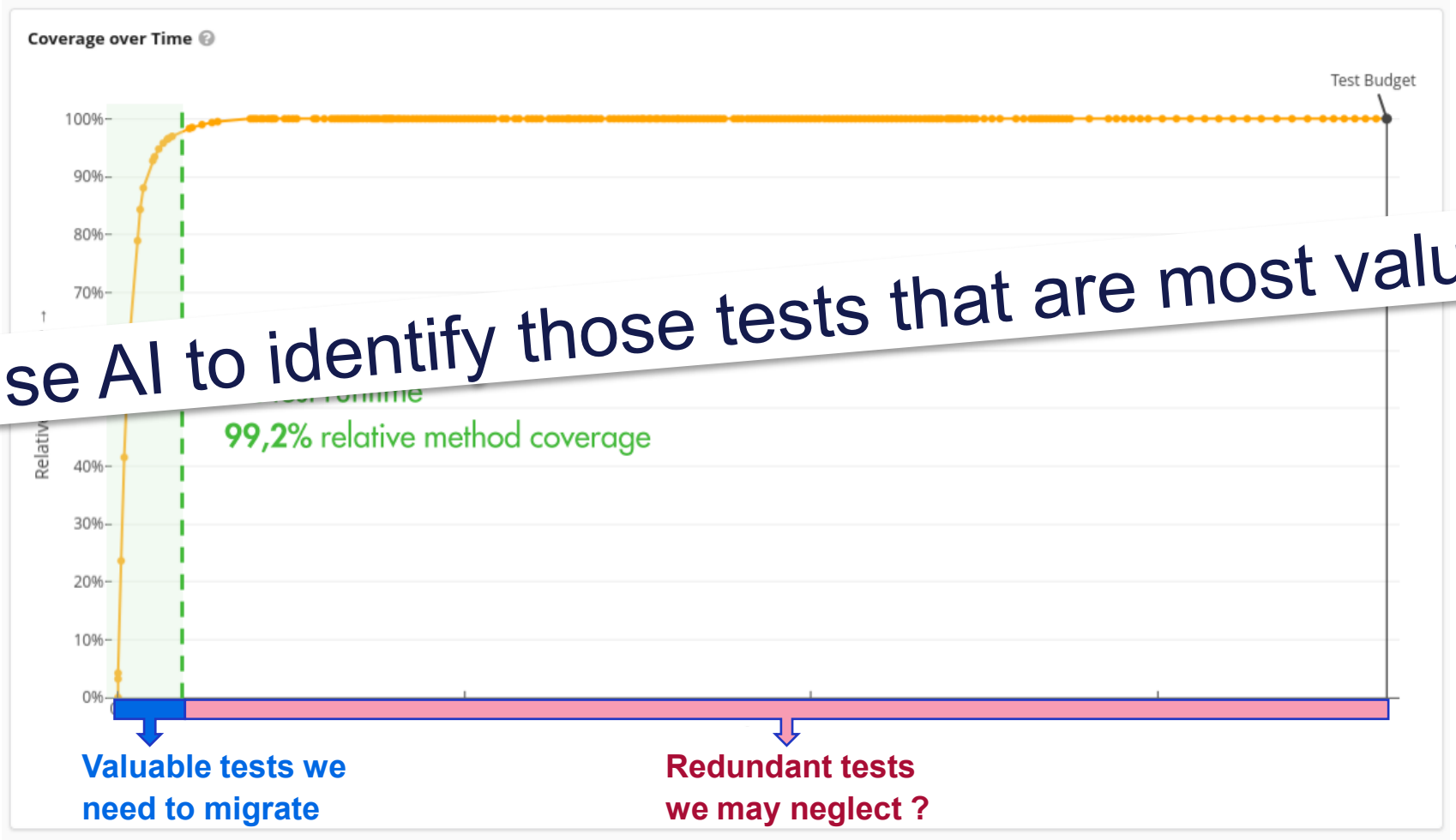
- EuroSTAR conference 2025 in Edinburgh
- Elmar Jürgens, Teamscale: **How well can AI help us find more bugs faster**





The idea

- Reduce the number of TOSCA tests that need to be migrated



Use AI to identify those tests that are most valuable

The project

- Get the help and tooling from 

- Use an AI-based clustering mechanism to select a suitable subset of TOSCA tests.
- Verify the selection through coverage measurements.
- Speed up the tool decommissioning.

The approach

Theoretical / scientific background

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

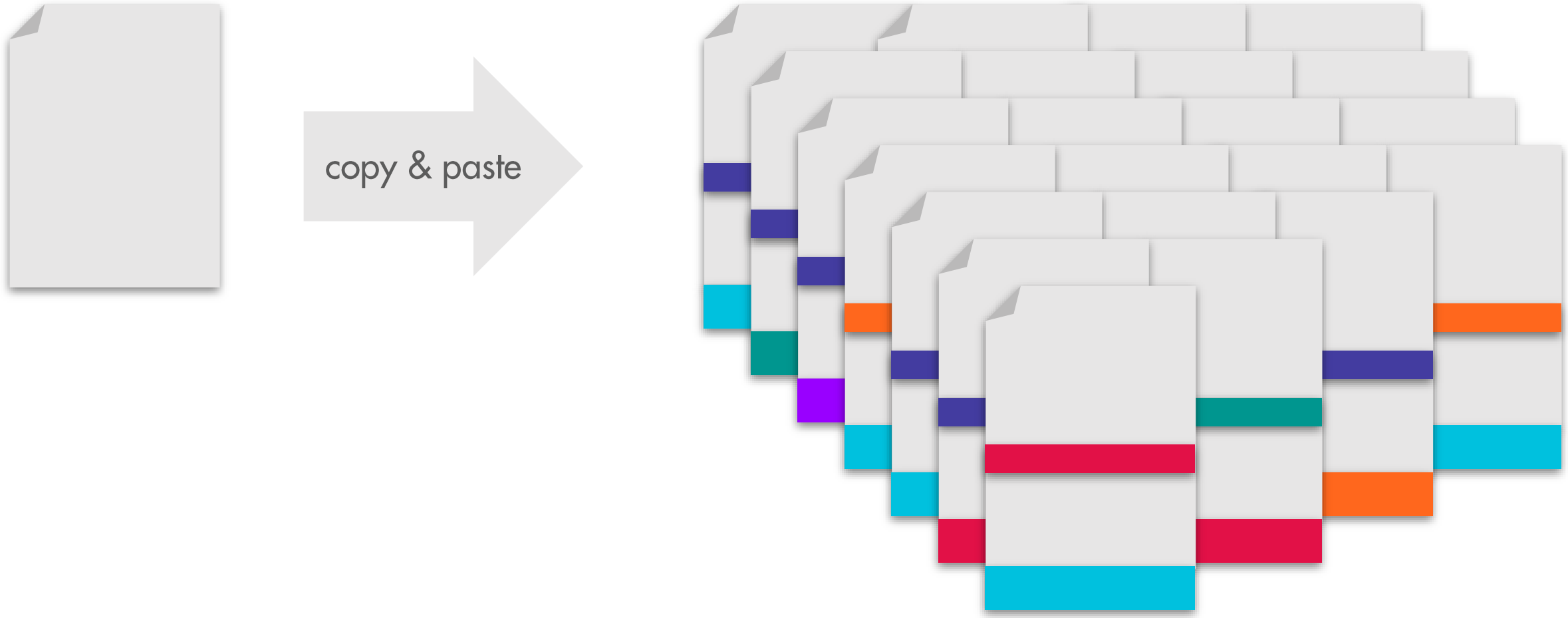
Master's Thesis in Informatics

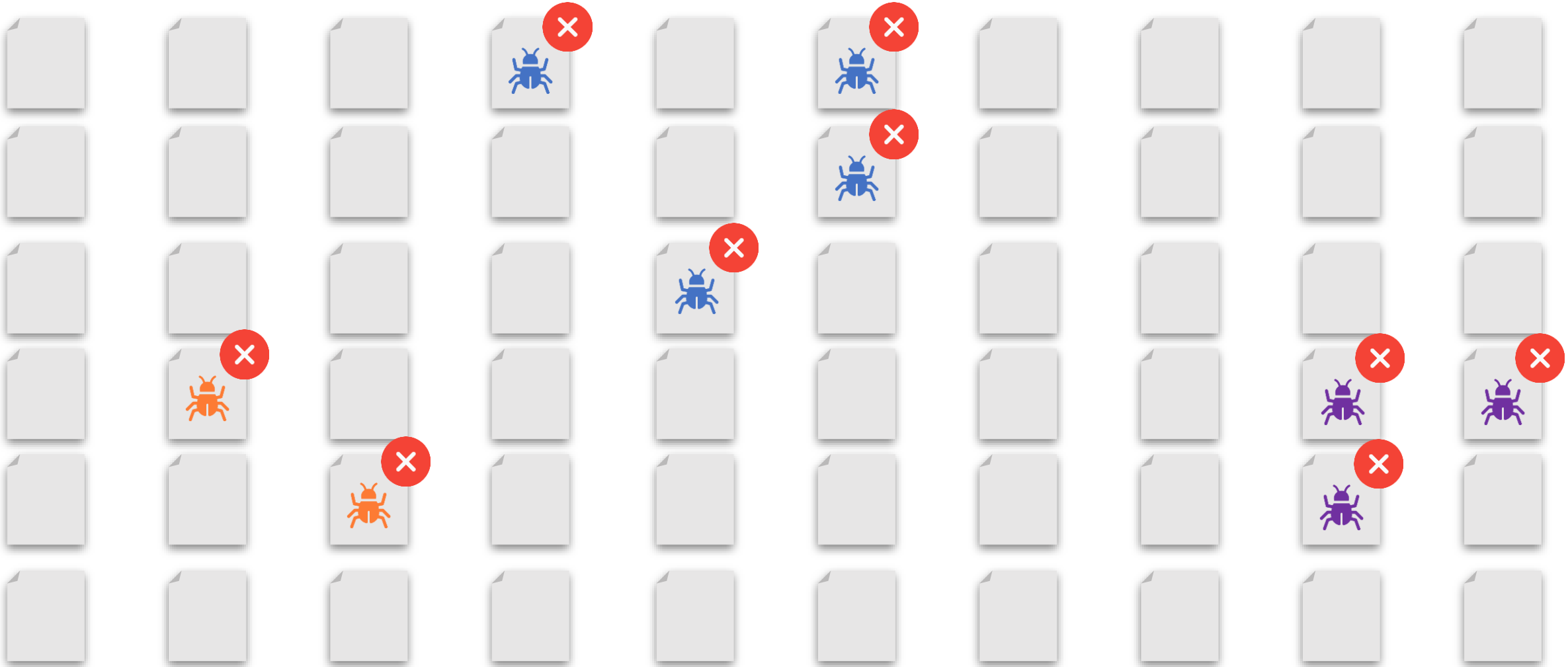
An Investigation on the Usage of Source Code Embeddings in Test Case Prioritization and Selection

Alessandro Escher

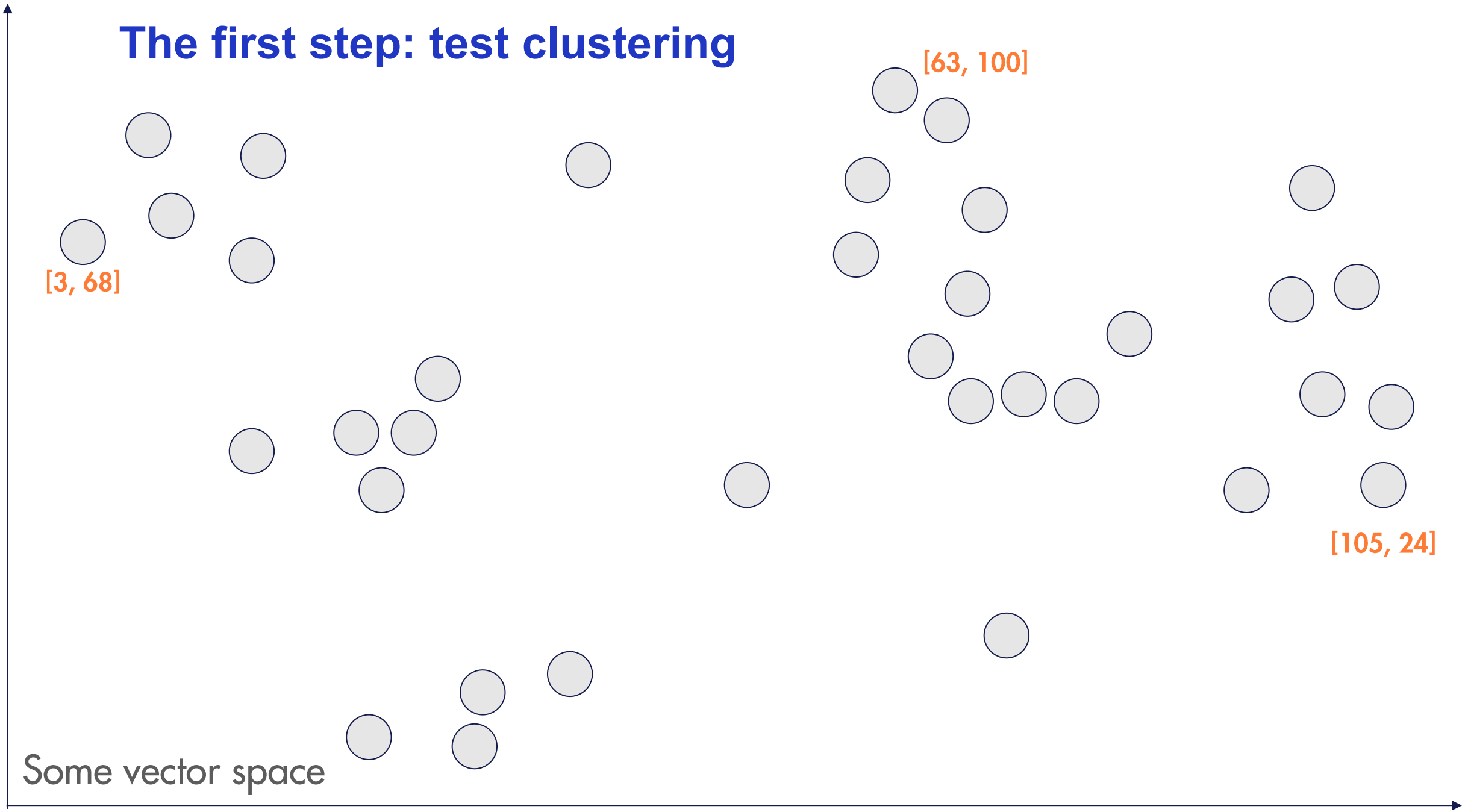


How did the test suite grow so big?

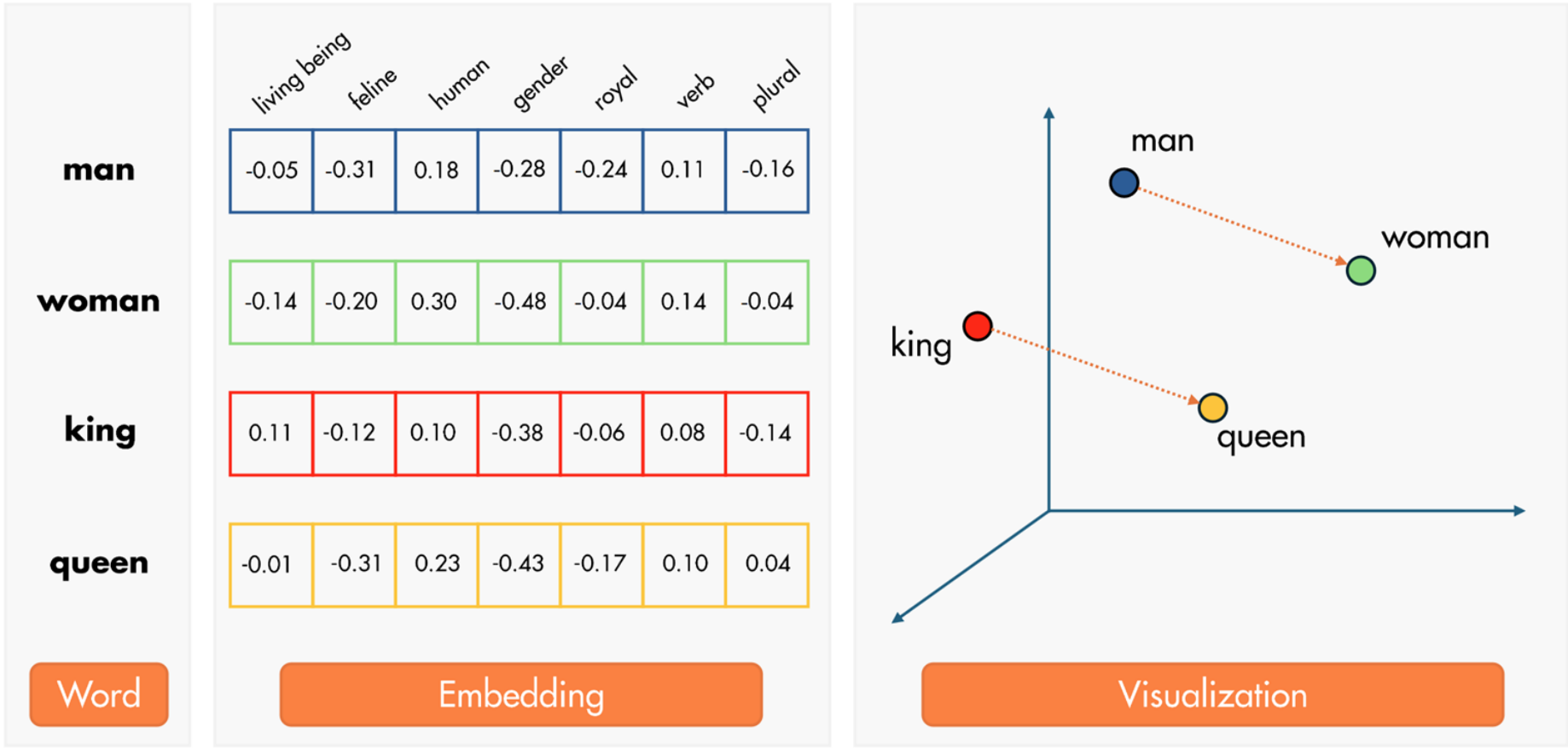


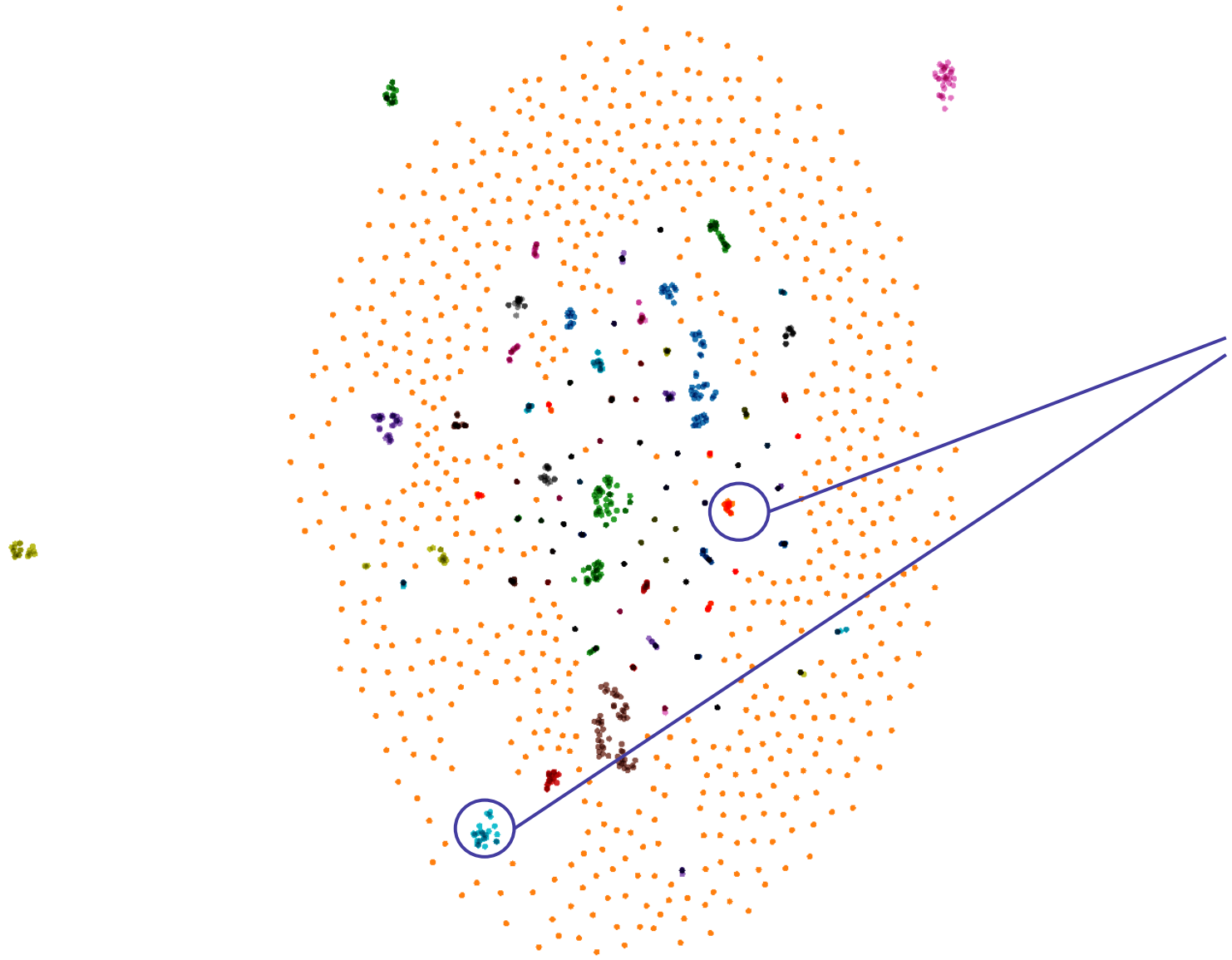


The first step: test clustering



Large Language ("AI") Models





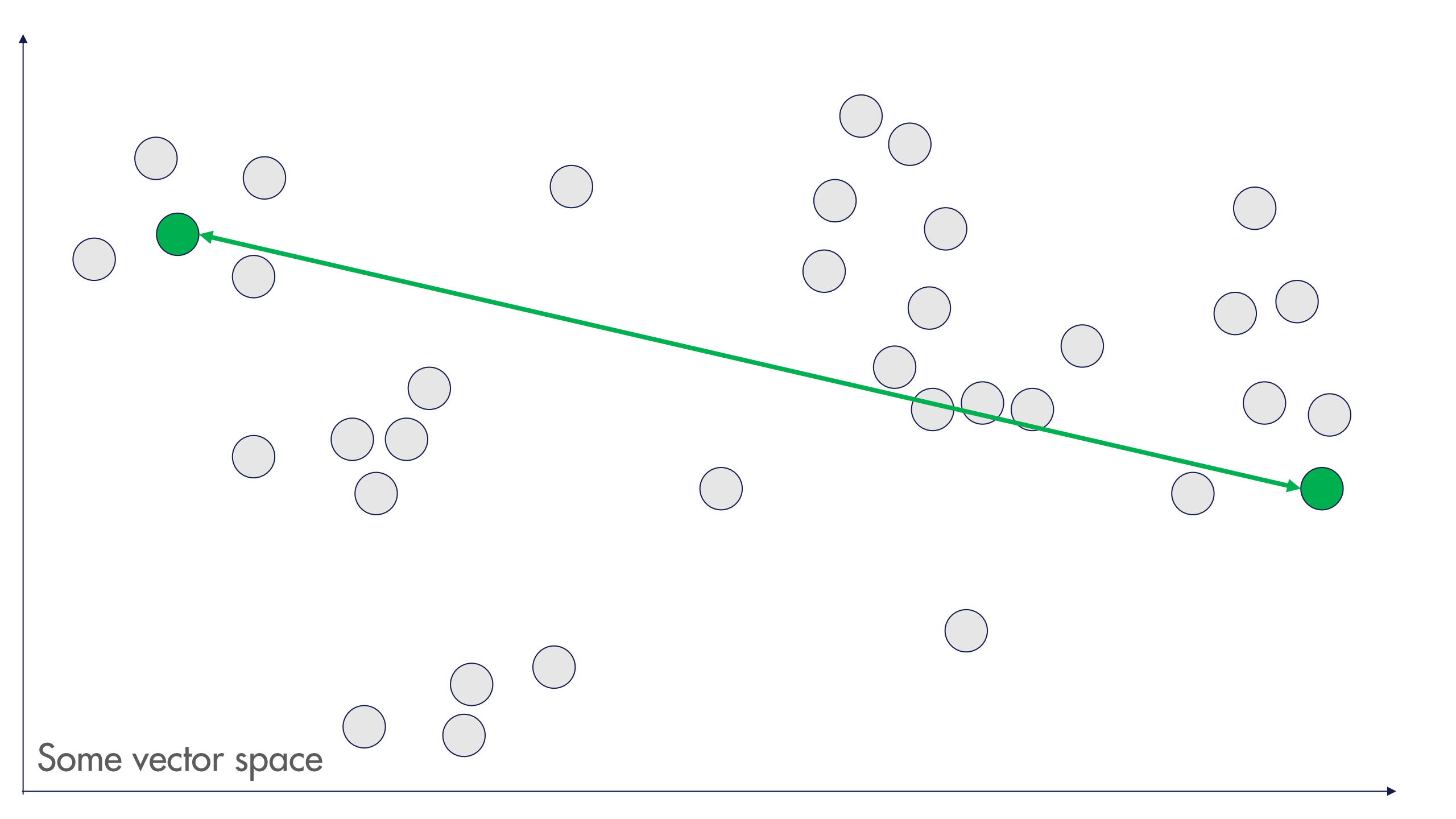
Verification point:

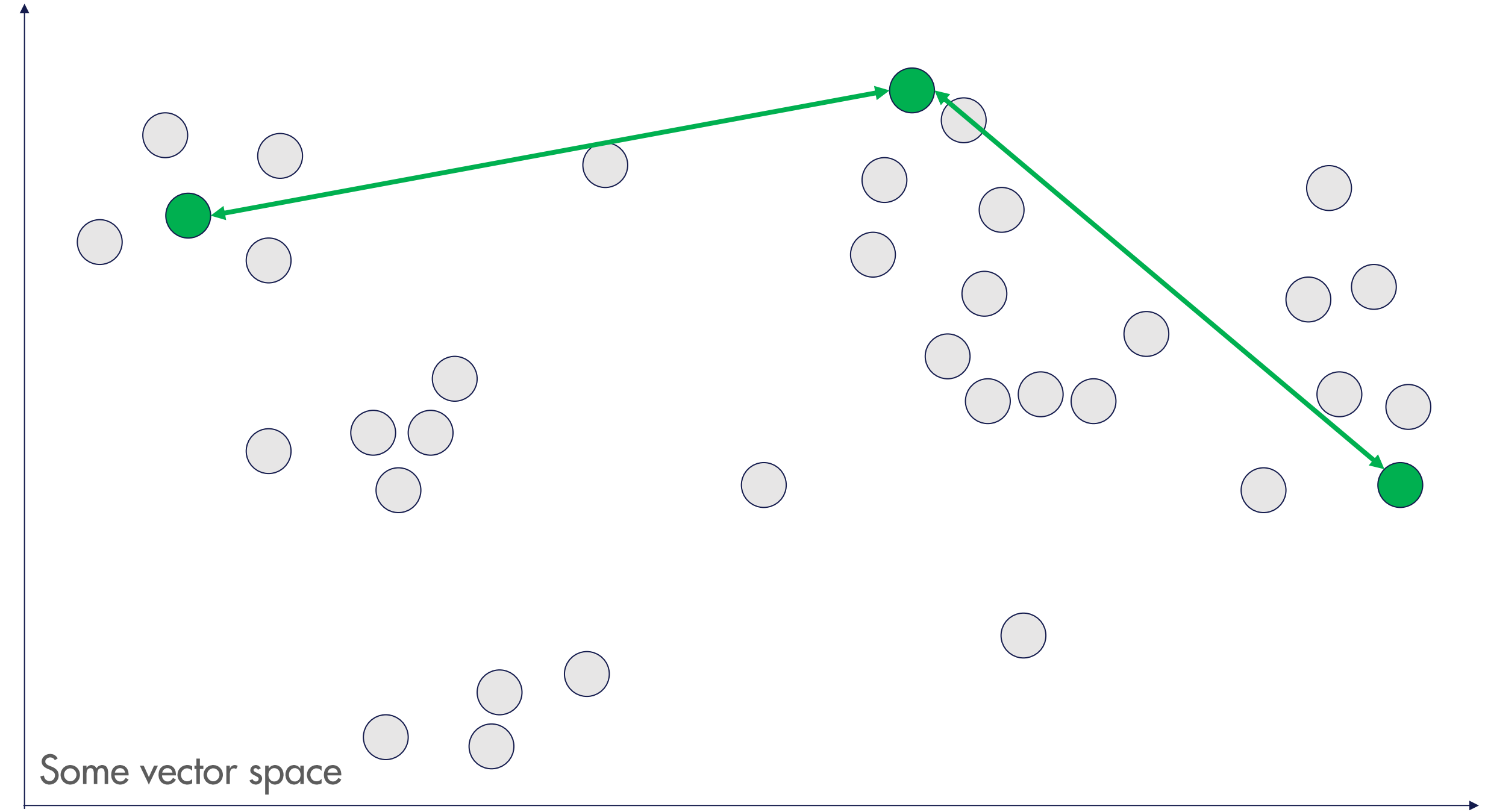
on selected samples

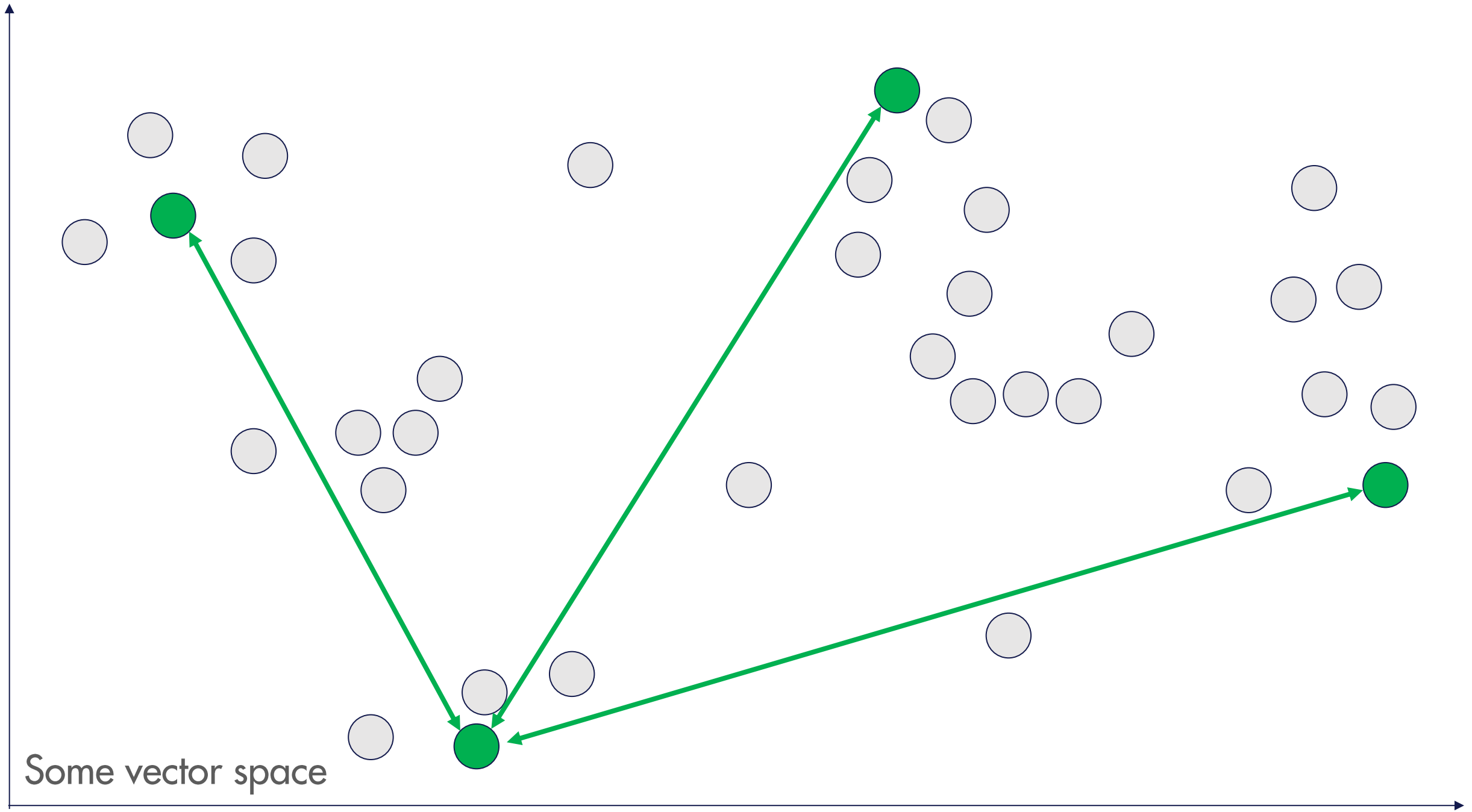
- Are tests in one cluster inside the same test suite?
- Do they check similar things?

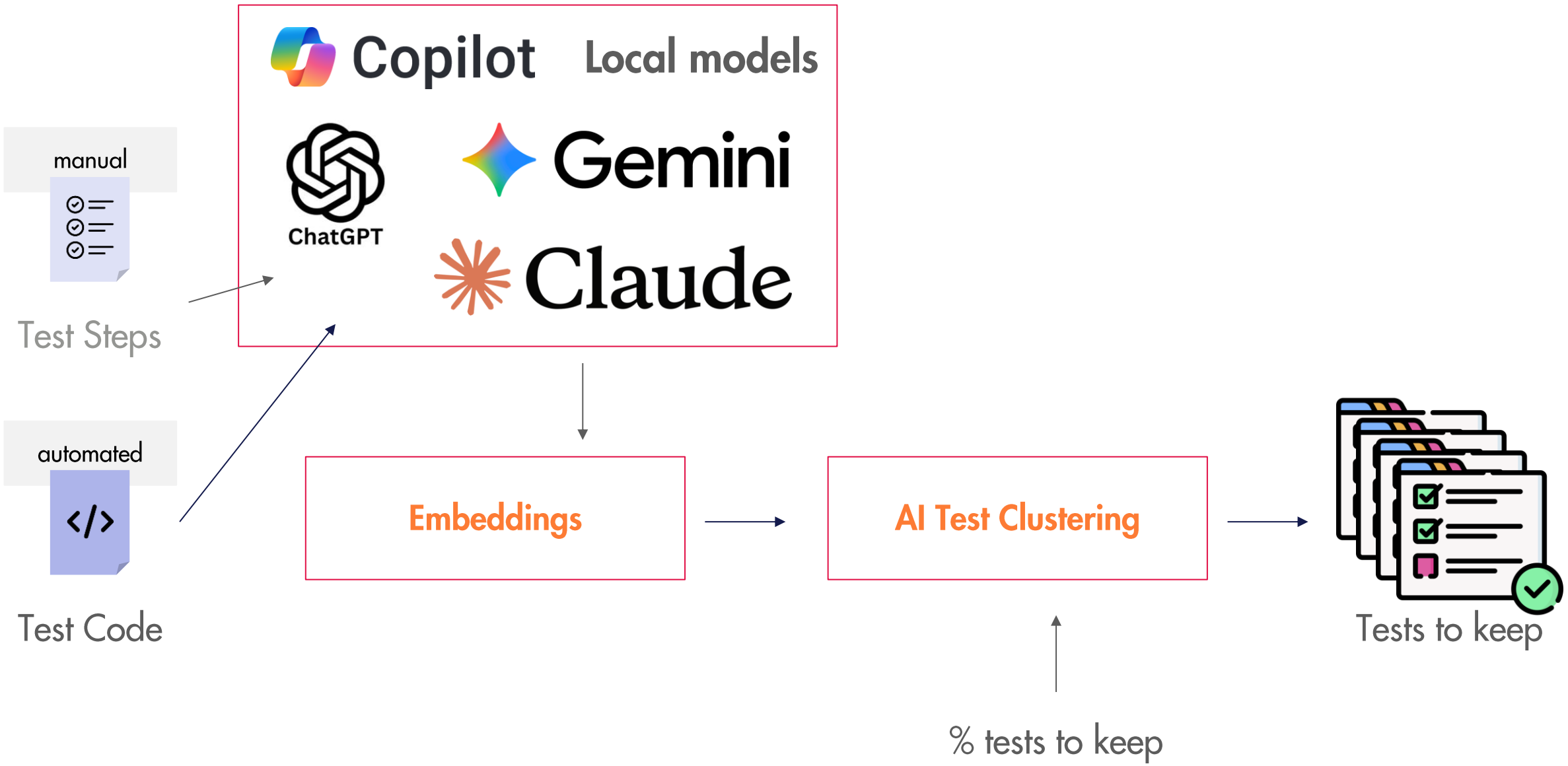
The second step: test ranking




Some vector space


Some vector space


Some vector space


Copilot
Local models
ChatGPT
Gemini
Claude
manual
Test Steps
automated
Test Code
Embeddings
AI Test Clustering
% tests to keep
Tests to keep

The second step: test ranking

Test ranking result:

List with all testcases sorted by dissimilarity

1	Testsuite_7/Testcase_2
2	Testsuite_1/Testcase_41
3	Testsuite_3/Testcase_12
4	Testsuite_2/Testcase_17
5	Testsuite_7/Testcase_10
6	Testsuite_5/Testcase_5
7	Testsuite_1/Testcase_15
8	Testsuite_8/Testcase_19
9	Testsuite_4/Testcase_32
10	Testsuite_3/Testcase_1
11	Testsuite_6/Testcase_41
12	Testsuite_7/Testcase_6
13	Testsuite_2/Testcase_2
14	Testsuite_5/Testcase_14
15	Testsuite_7/Testcase_18
16	Testsuite_7/Testcase_21
17	Testsuite_1/Testcase_32
18	Testsuite_6/Testcase_5
…	*1,723 test cases total*

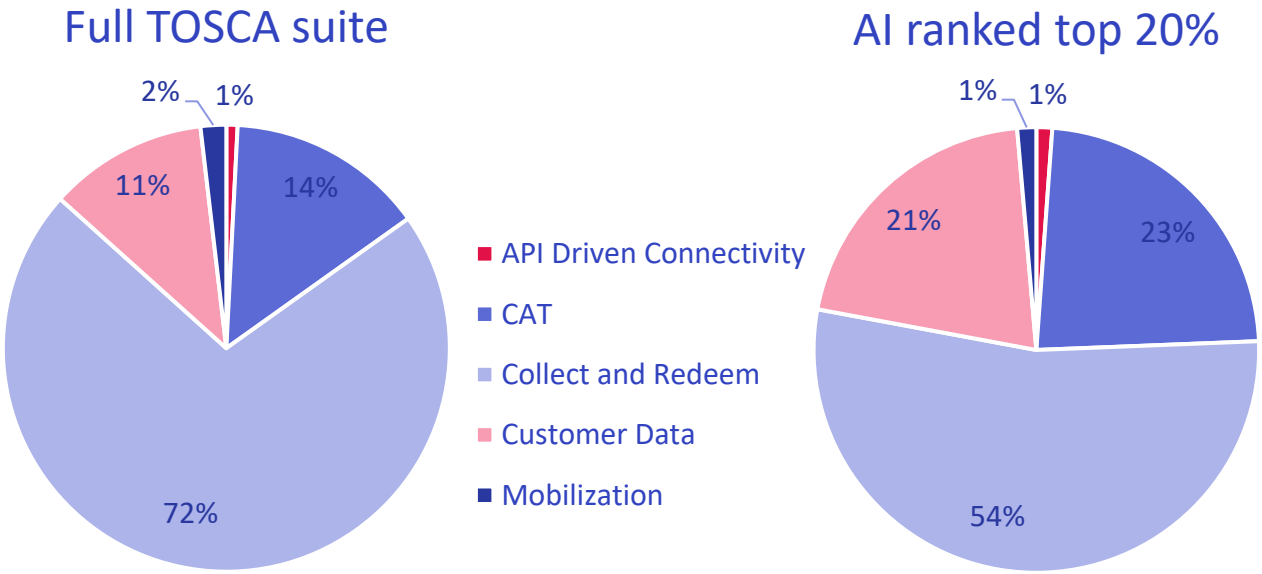
dissimilar

20% sample

similar

Verification point:

did the test ranking work?

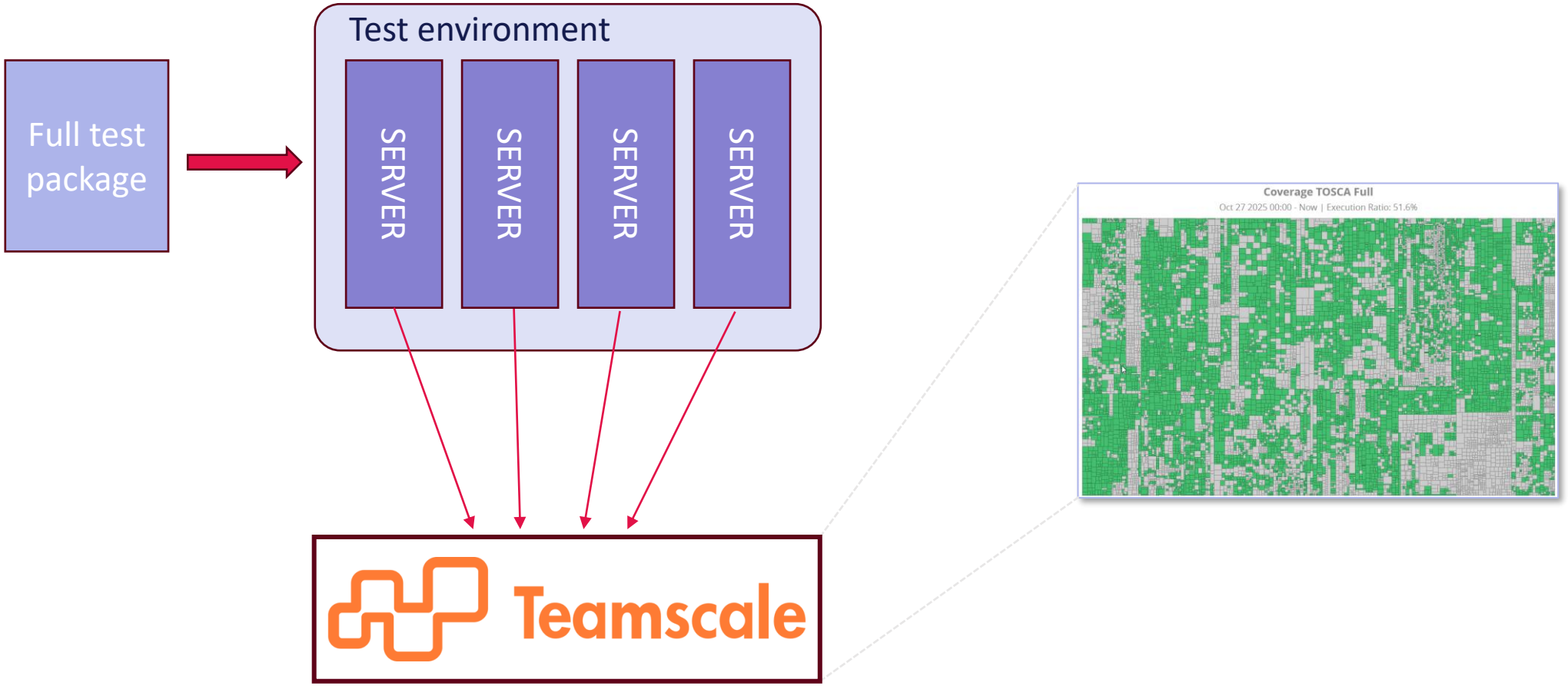


Testcases in Collect&Redeem are more similar to each other.
Testcases in CAT and Customer Data cover a bigger variety.

The proof

The third step: coverage measurement

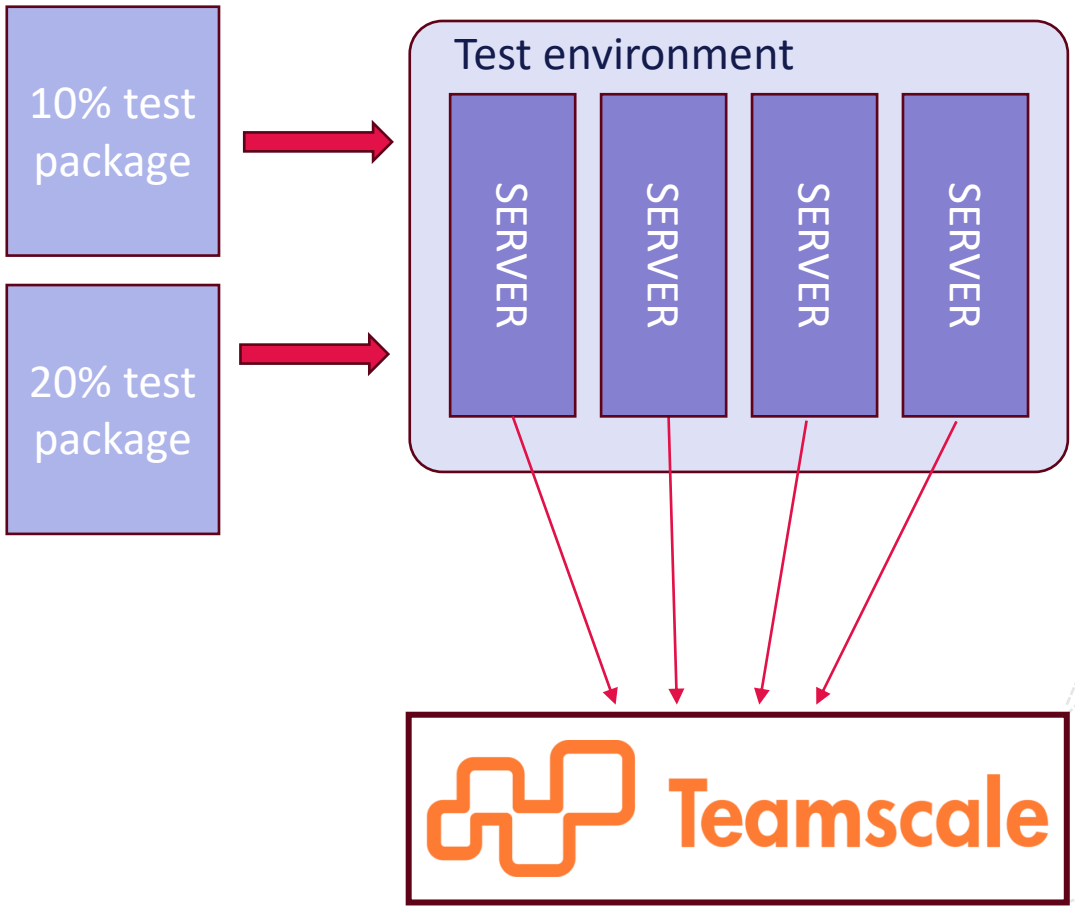
3.1. Perform coverage measurements with 100% of tests.

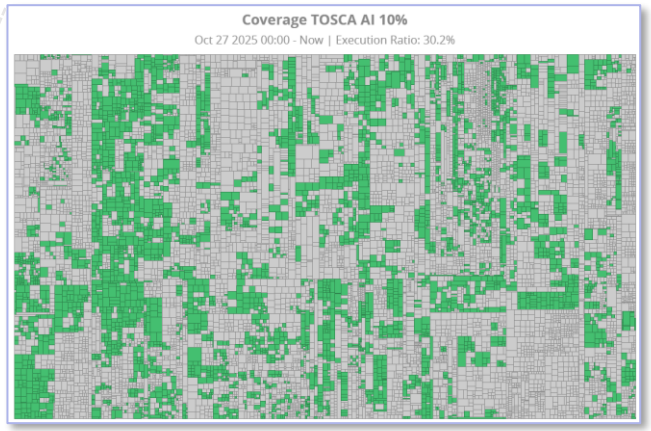


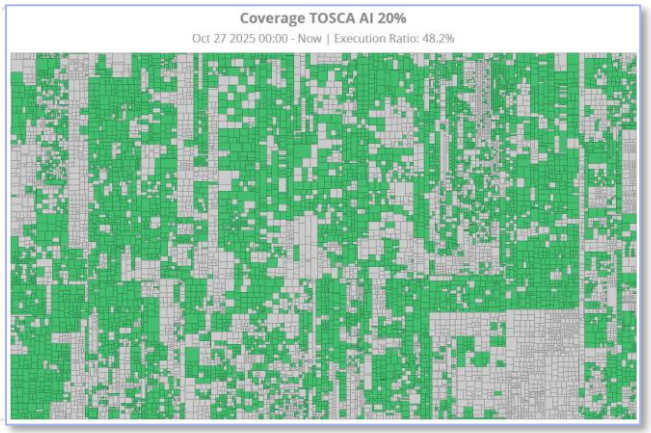
The third step: coverage measurement

3.2. Perform coverage measurements with top 10% of AI ranked tests.

3.3. Perform coverage measurements with top 20% of AI ranked tests.





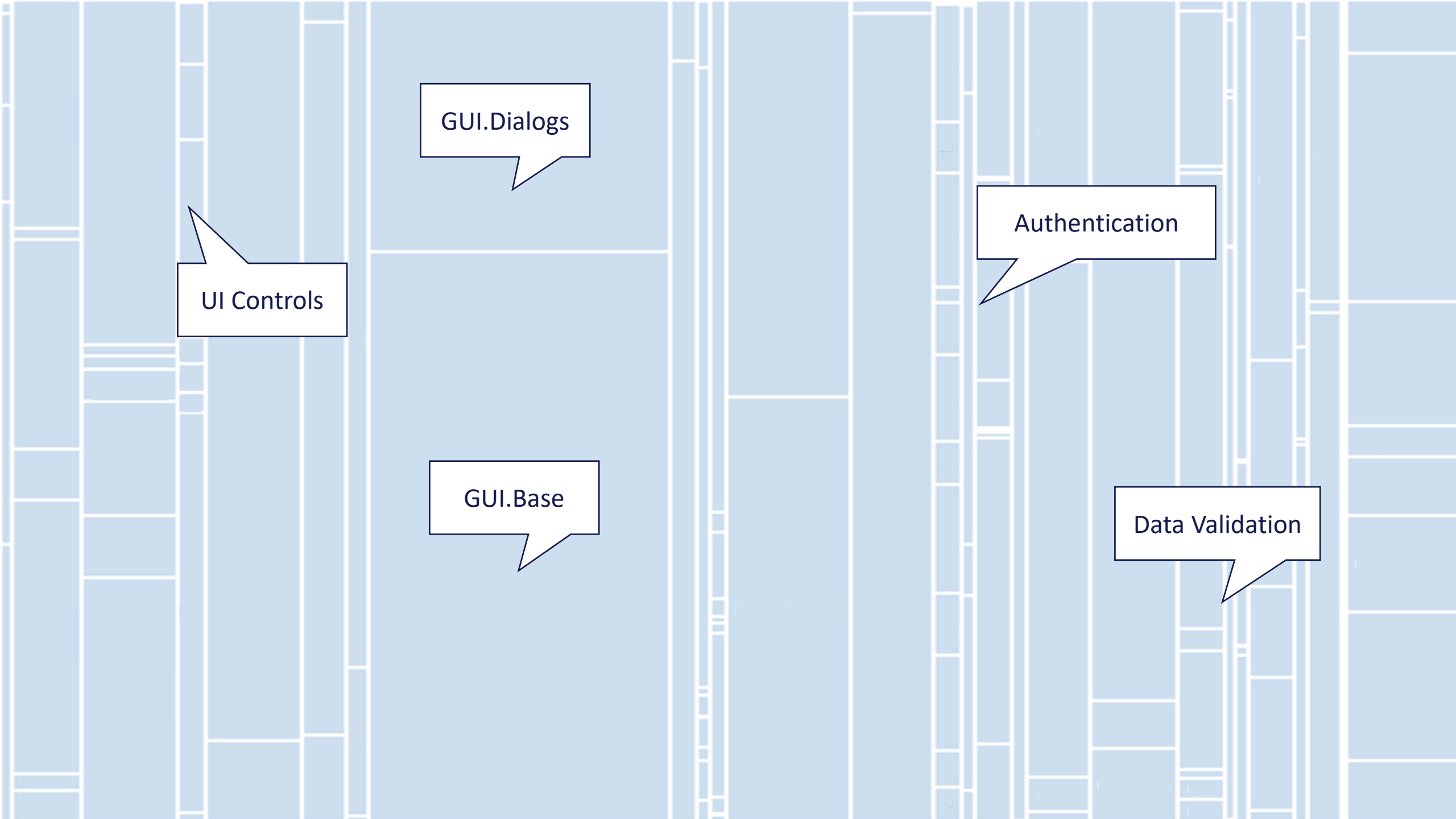


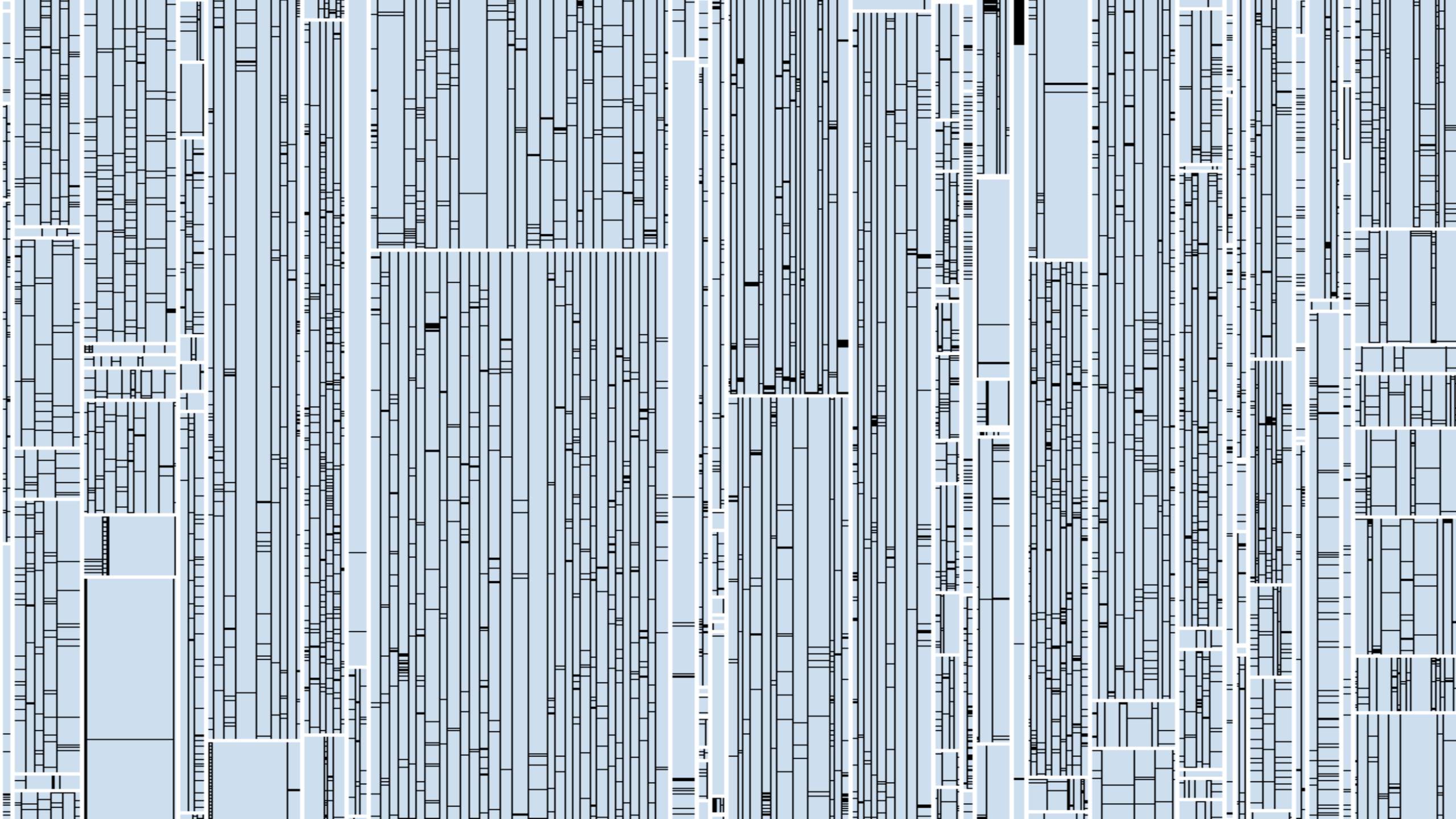
The challenges

- Exclusive usage slots on a shared test environment are scarce!
  → measurements were run during nights and weekends

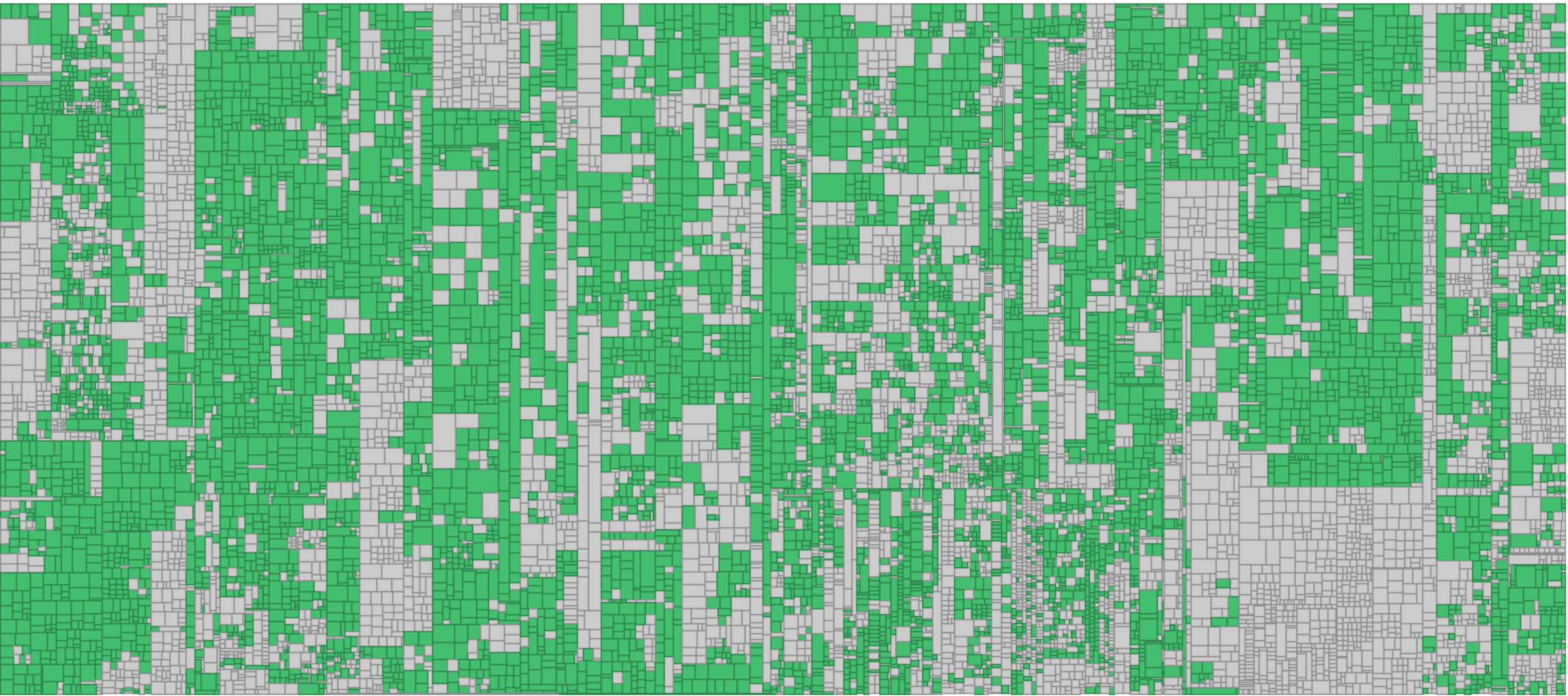
- Our TOSCA tests are very flaky and long-running
  → many re-runs were required

- Other tests executed during the measurement runs “polluted” the results
  → several runs had to be repeated


GUI.Dialogs
UI Controls
GUI.Base
Authentication
Data Validation



All Tosca Tests



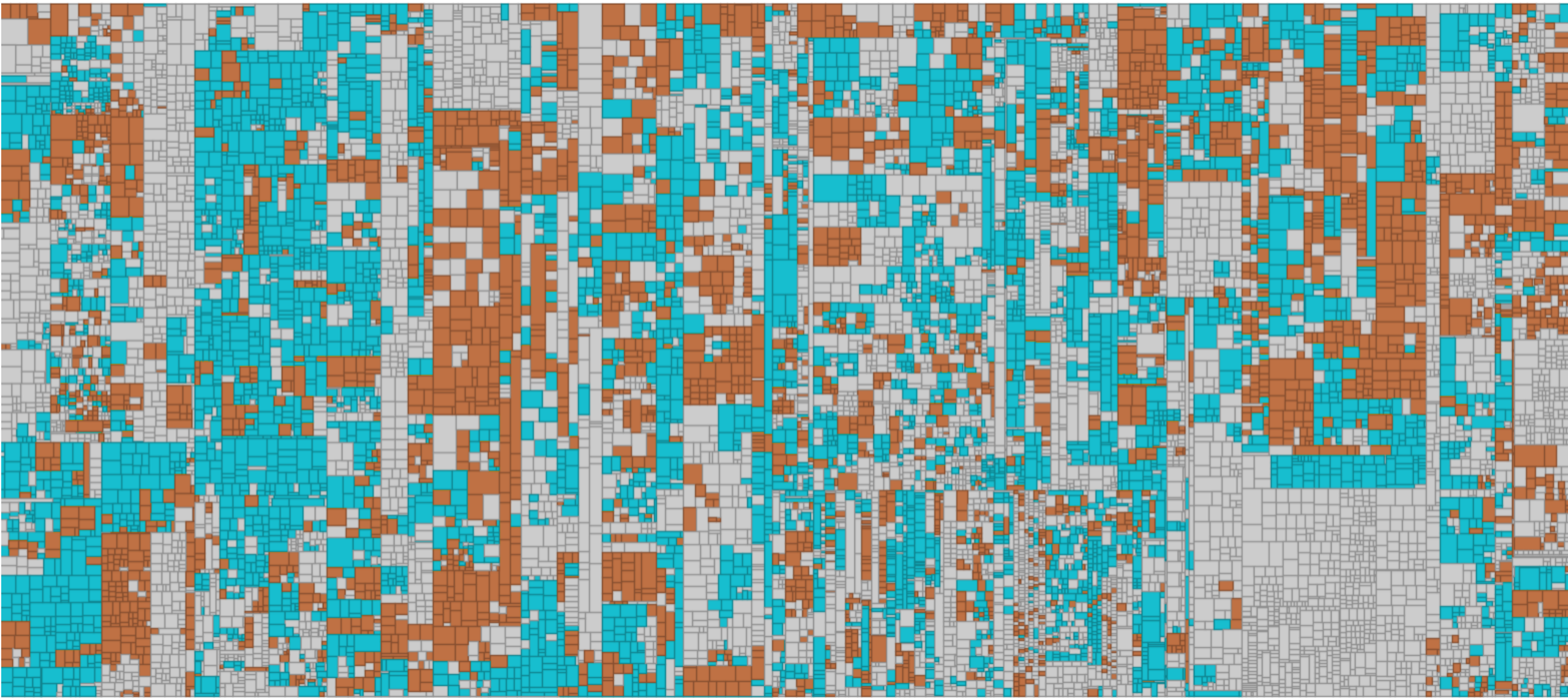
AI ranked top 10% of the Tests



Covered by **10% AI Test Cluster**

Covered by remaining 90%

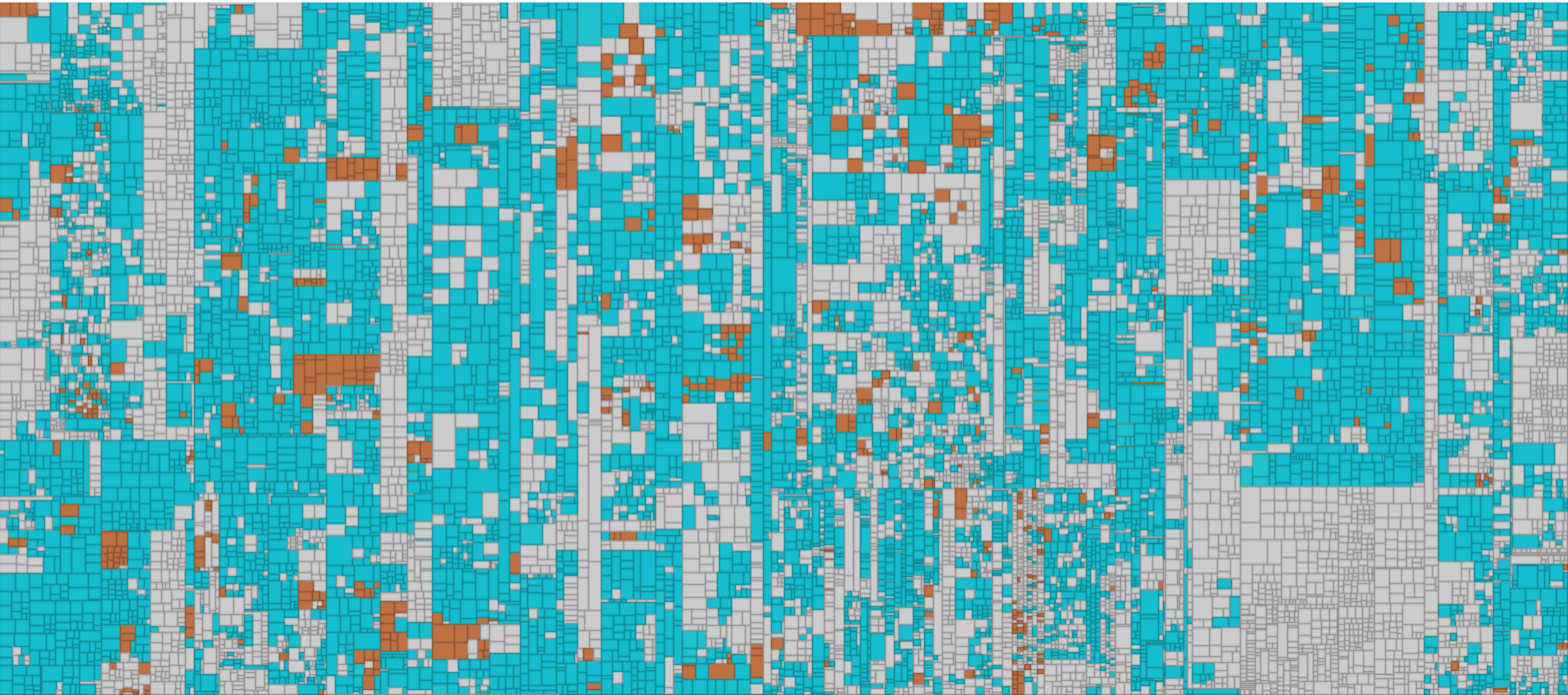
Retained only 59% of the original code coverage



Covered by **20% AI Test Cluster**

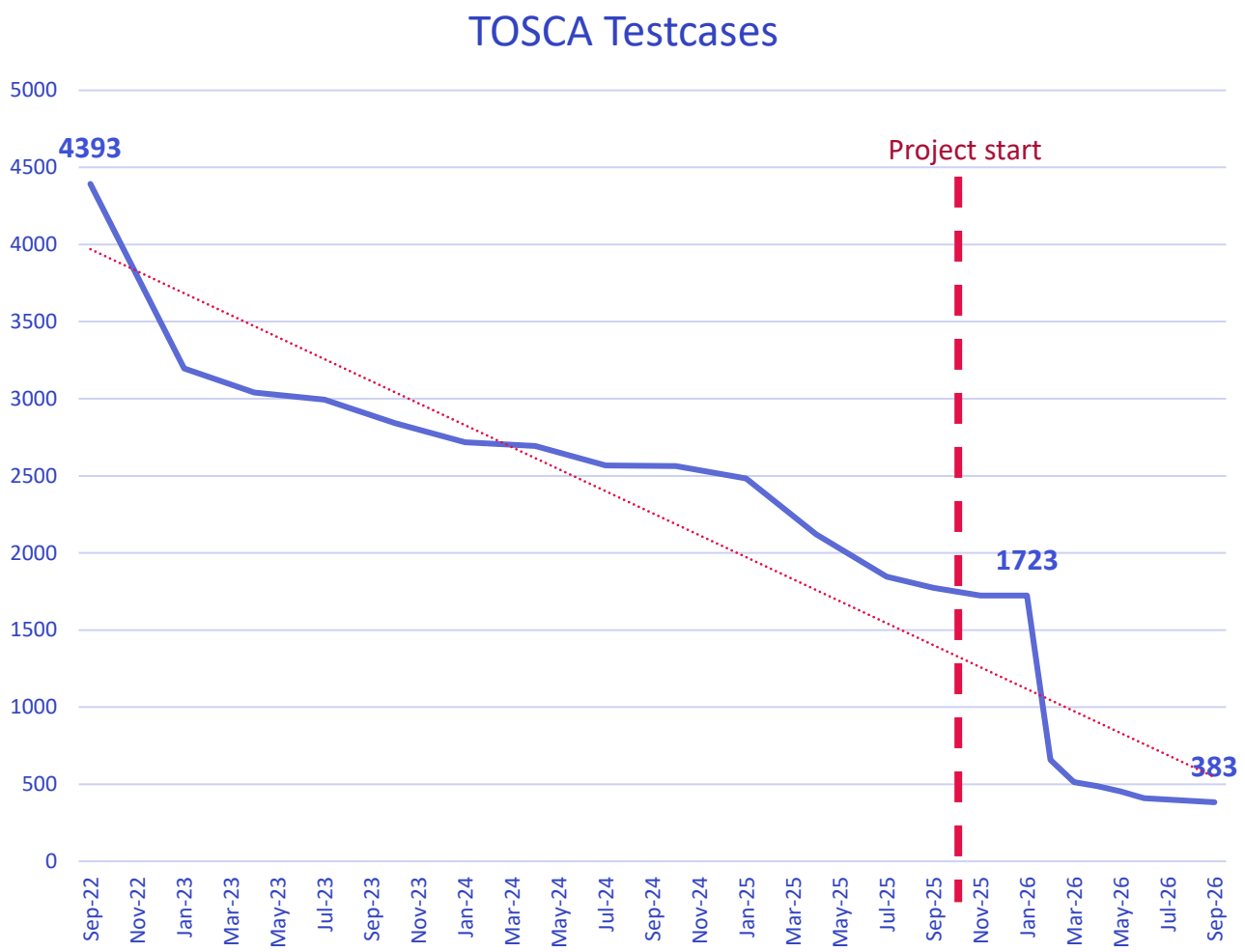
Covered by remaining 80%

Retained **93% of the original test coverage**



The benefit

The final step: use the results



- Results reviewed with the responsible teams
- Teams decide per test: remove or migrate

~4 person days saved per bi-weekly release

~400 person days of migration effort avoided

−78% test case reduction since project start (1723 → 383)

Thanks for Listening

Get the slides and future talks





Please Rate This Session

in the Conference App

Use the  in this session's listing on the Agenda

Thank You!

Questions?

Email sebastian.moertl@payback.net
fabian.streitel@cqse.eu

Social linkedin.com/in/smrtlmunich
linkedin.com/in/fabian-streitel-353a44b4

