

# Muss ich wirklich schon wieder alles testen?

Test-Impact-Analyse mit Teamscale live am Beispiel

Dr. Elmar Jürgens, Dr. Andreas Göb  
CQSE GmbH





## Praxis

Software-Audits

Kontinuierliche Qualitäts-  
und Testkontrolle

 Teamscale

**CQSE** GmbH



## Forschung

16+ **Promotionen** in  
Software Engineering


Eigene **Forschung**


Enger Kontakt zu  
**Universitäten**




com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest - testViewingArchitectureElements

Runs: 1/227

 Errors: 0

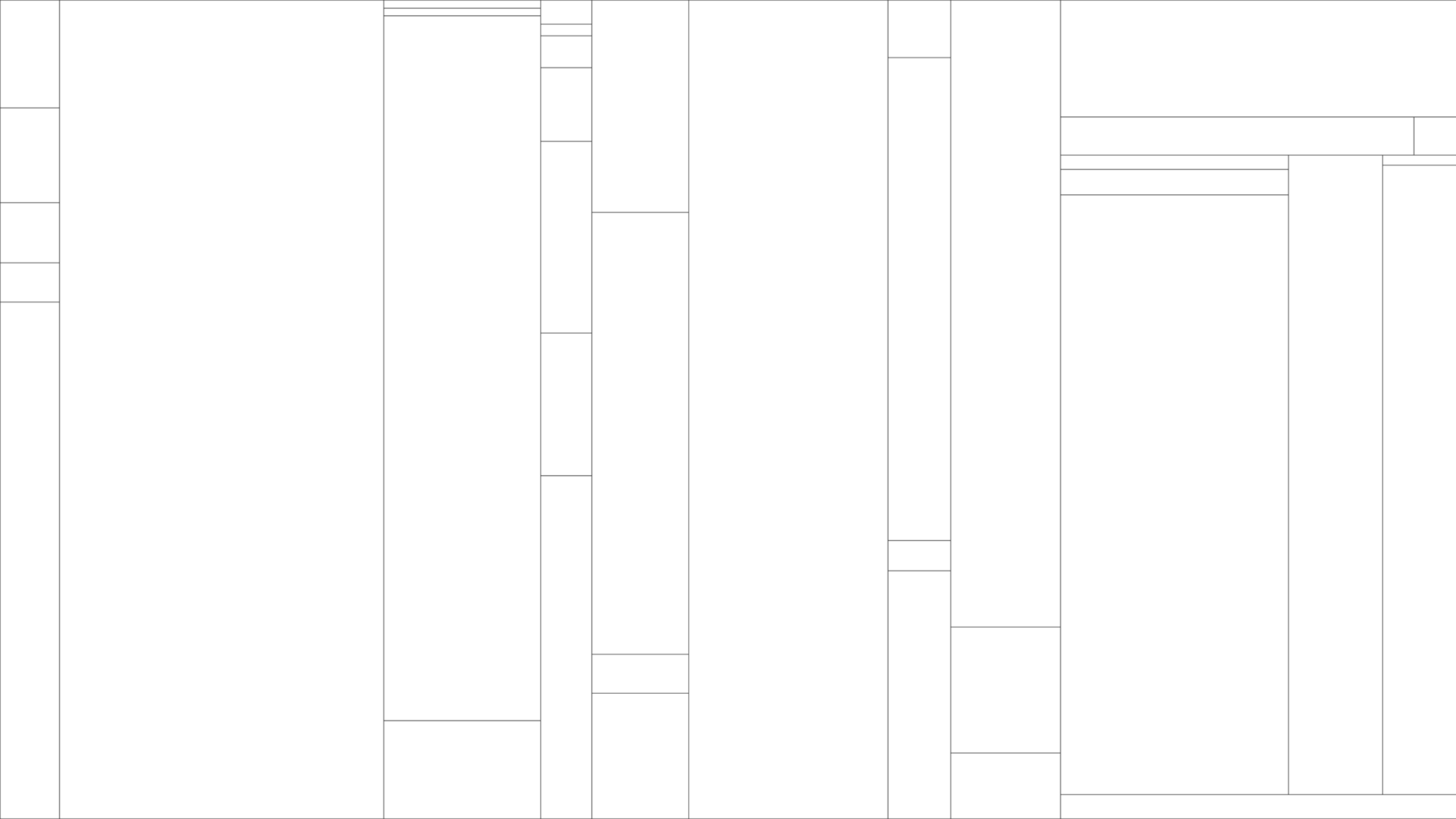
 Failures: 0

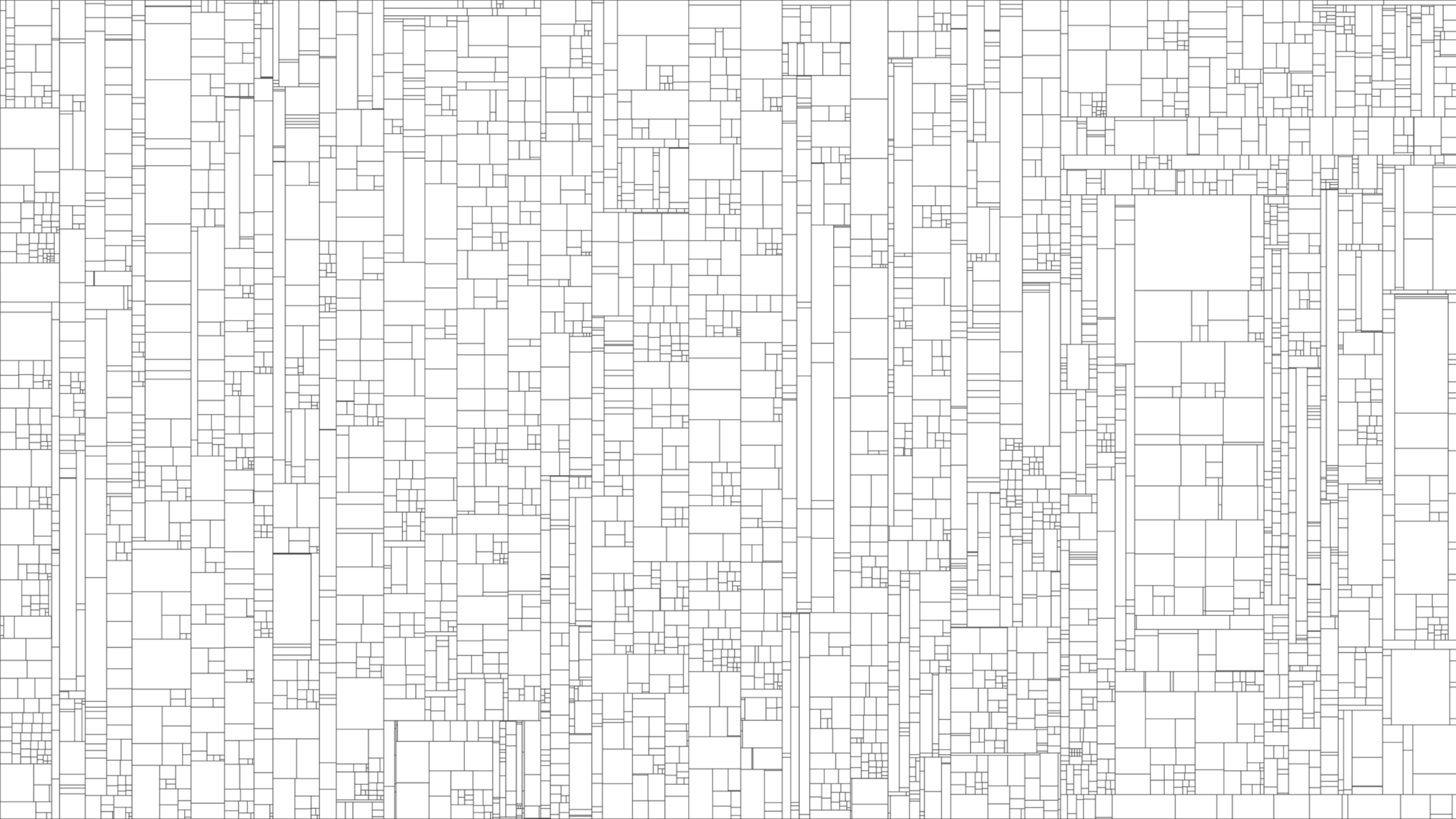
- ▼  com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest [Runner: JUnit 5]
  -  testViewingArchitectureElements
  -  createArchitectureTest
  -  testTimetravelMode
  -  testUndoRedo
  -  testArchitecturePolicyUpdate
  -  testOpenCodeOfOrphans

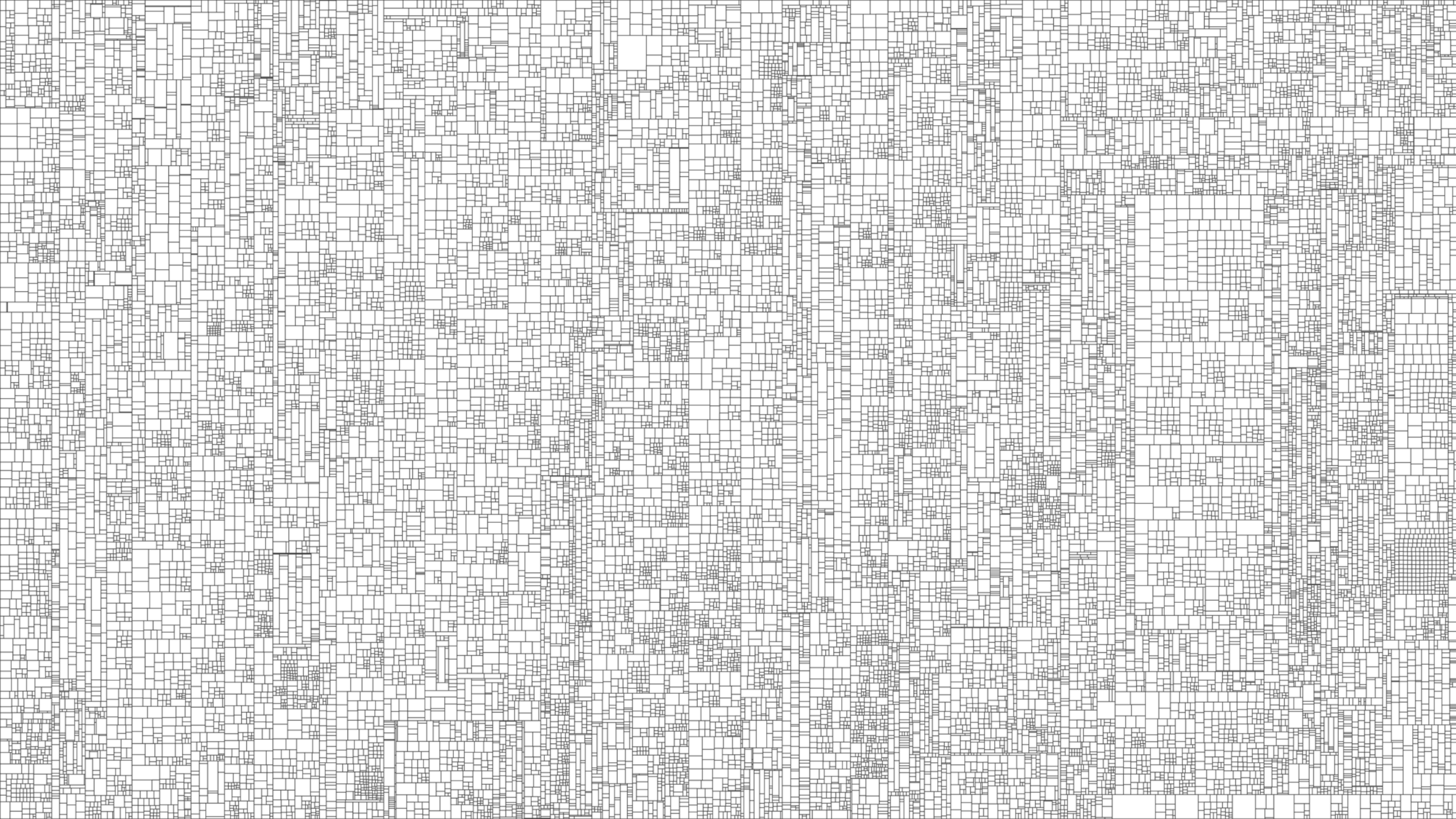
 Failure Trace

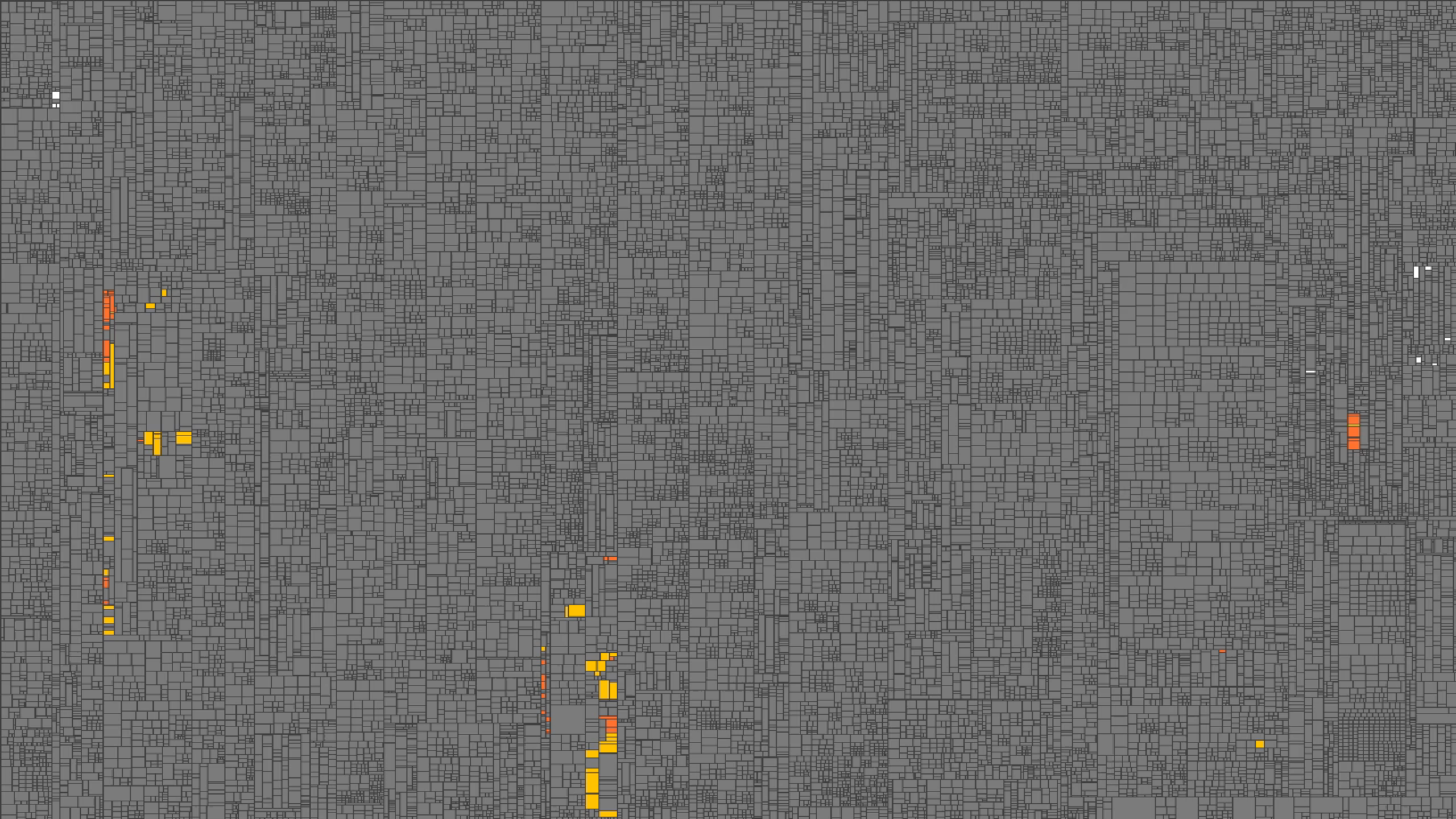




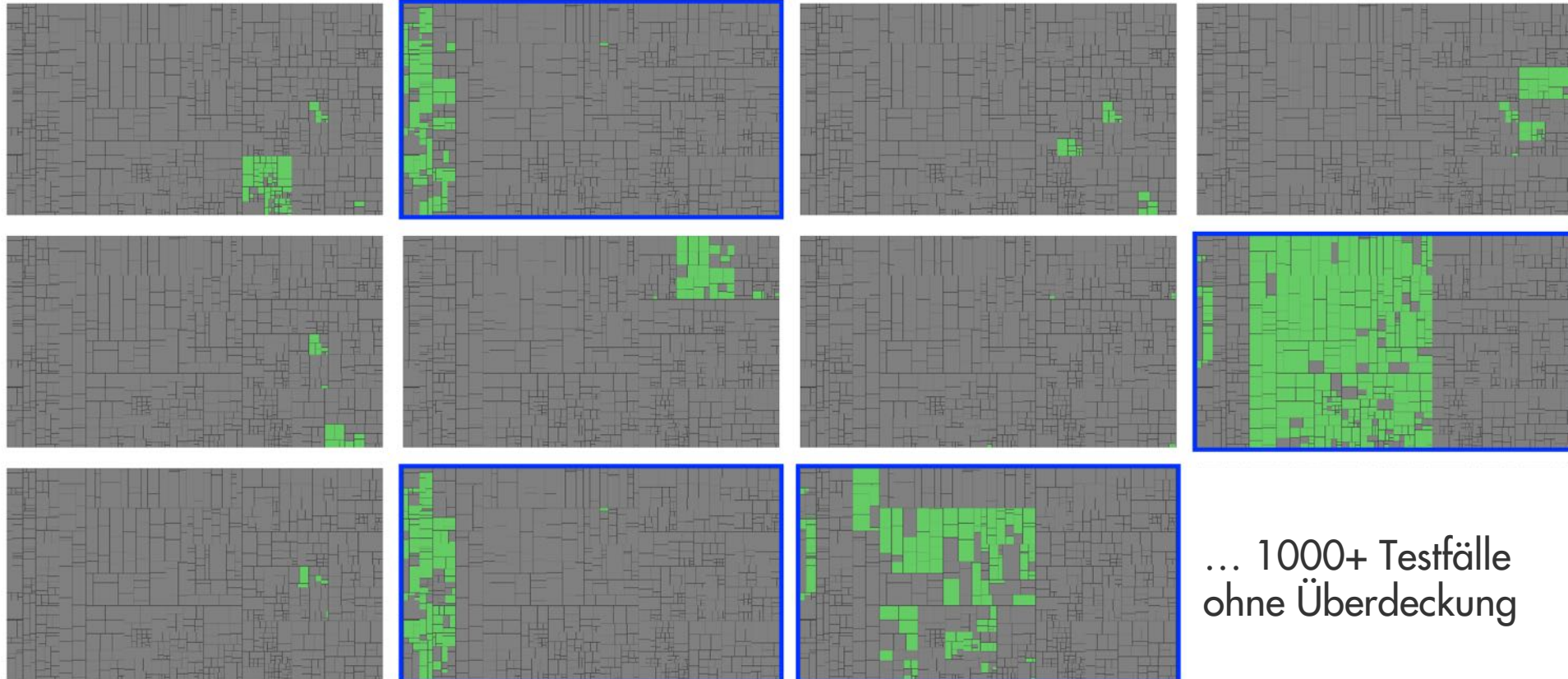
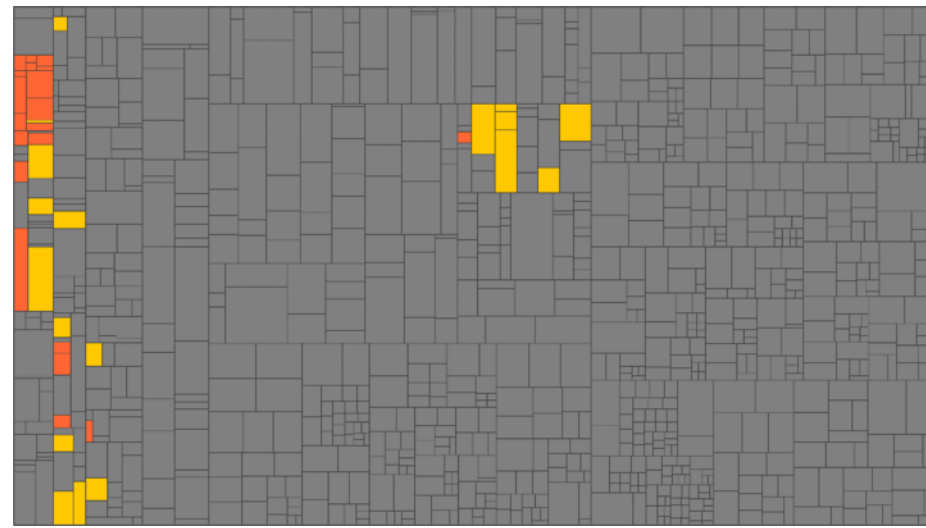




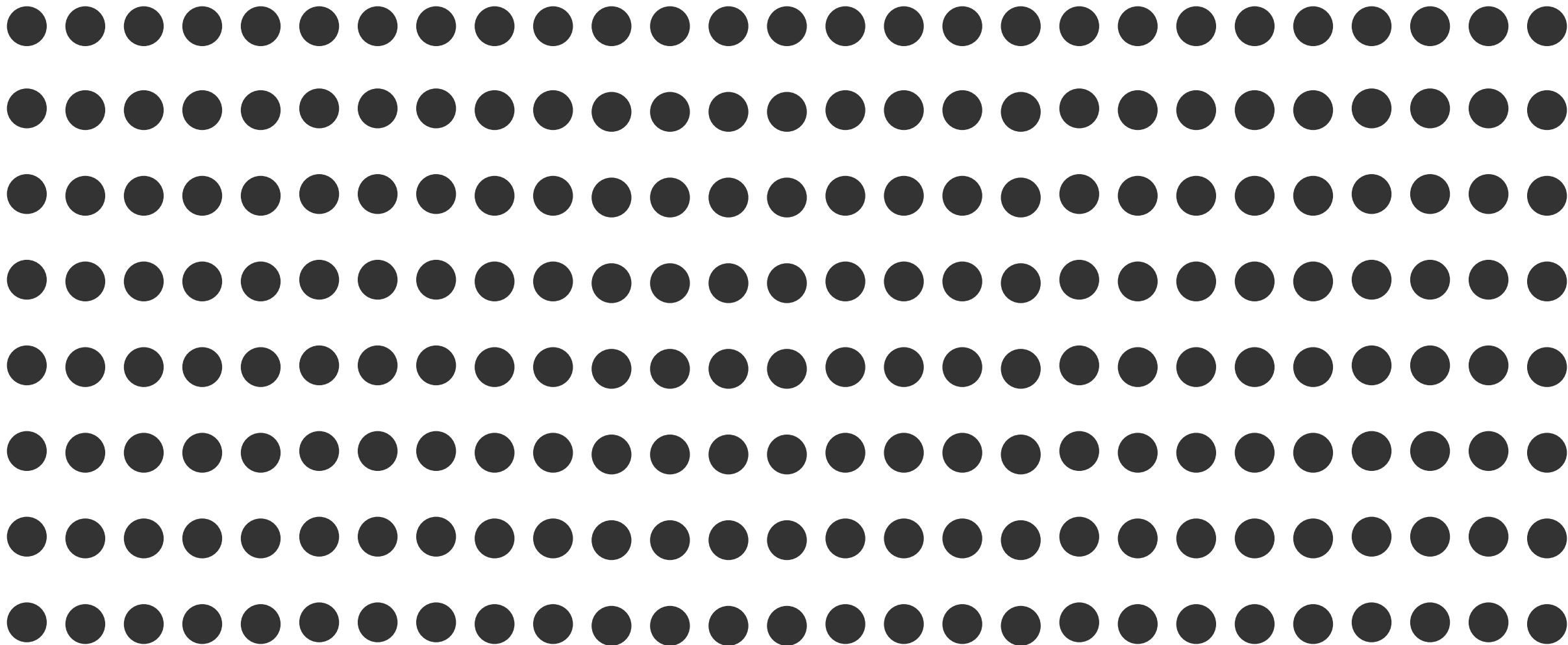






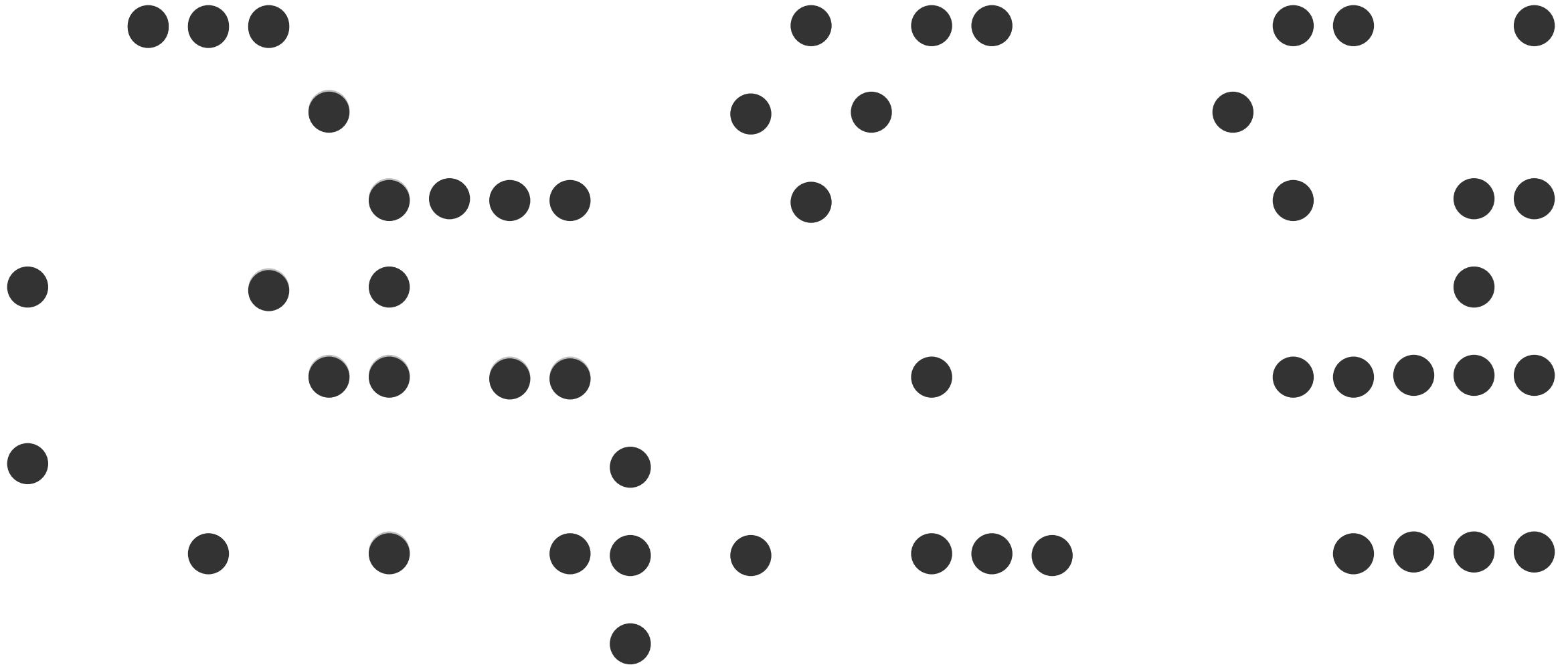


... 1000+ Testfälle  
ohne Überdeckung

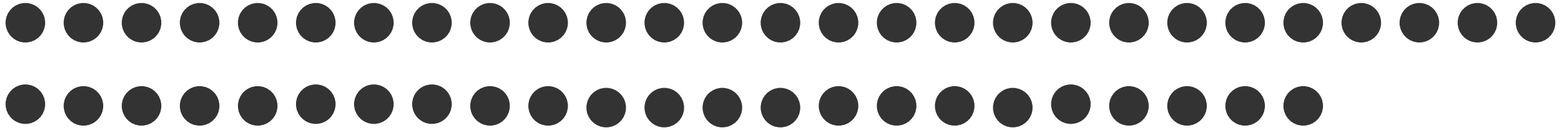




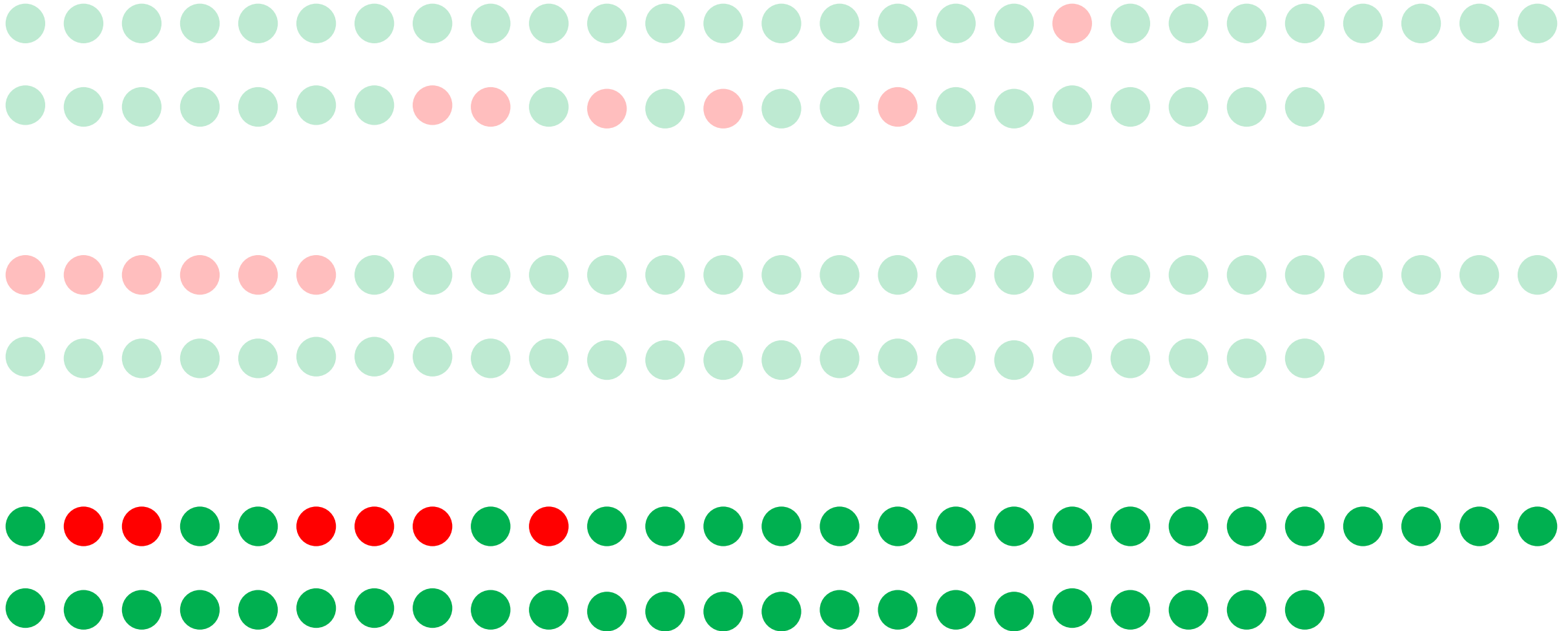
# Schritt 1: Selektion betroffener Testfälle

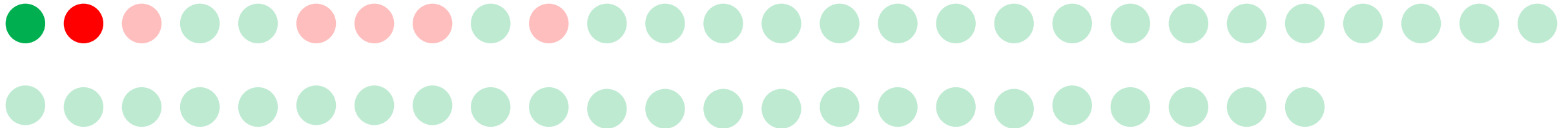
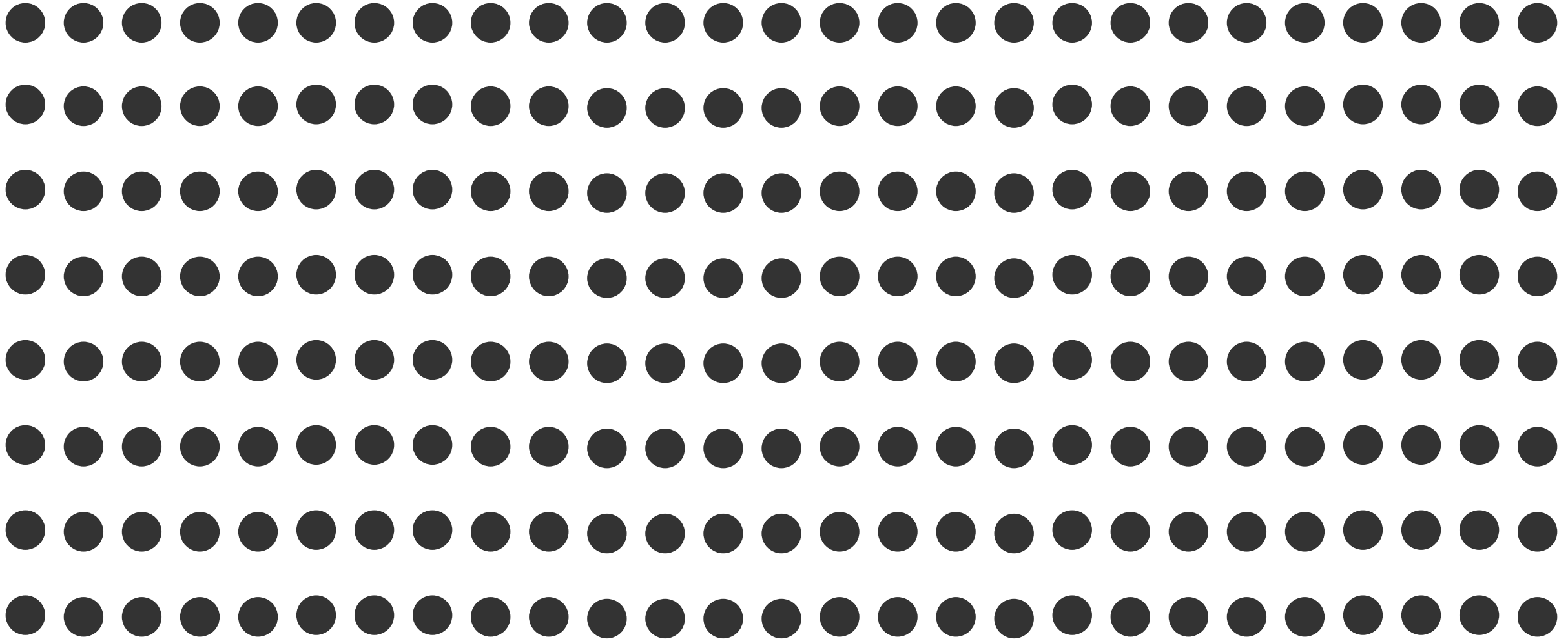


# Schritt 1: Selektion betroffener Testfälle



## Schritt 2: Priorisierung selektierter Testfälle







Alle Tests



Testumgebung



Testfallspezifische  
Coverage und Laufzeiten



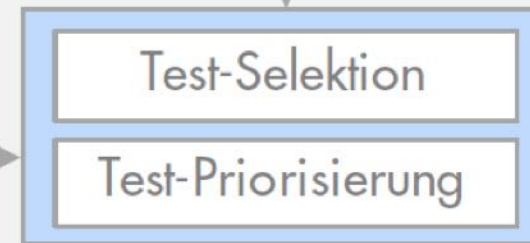
1



Versionskontrollsystem



Änderungen



Änderungsrelevante,  
geordnete Teilmenge  
der Tests



2



80% der fehlerhaften Builds in 1% der Zeit erkannt

90% der fehlerhaften Builds in 2% der Zeit erkannt

---

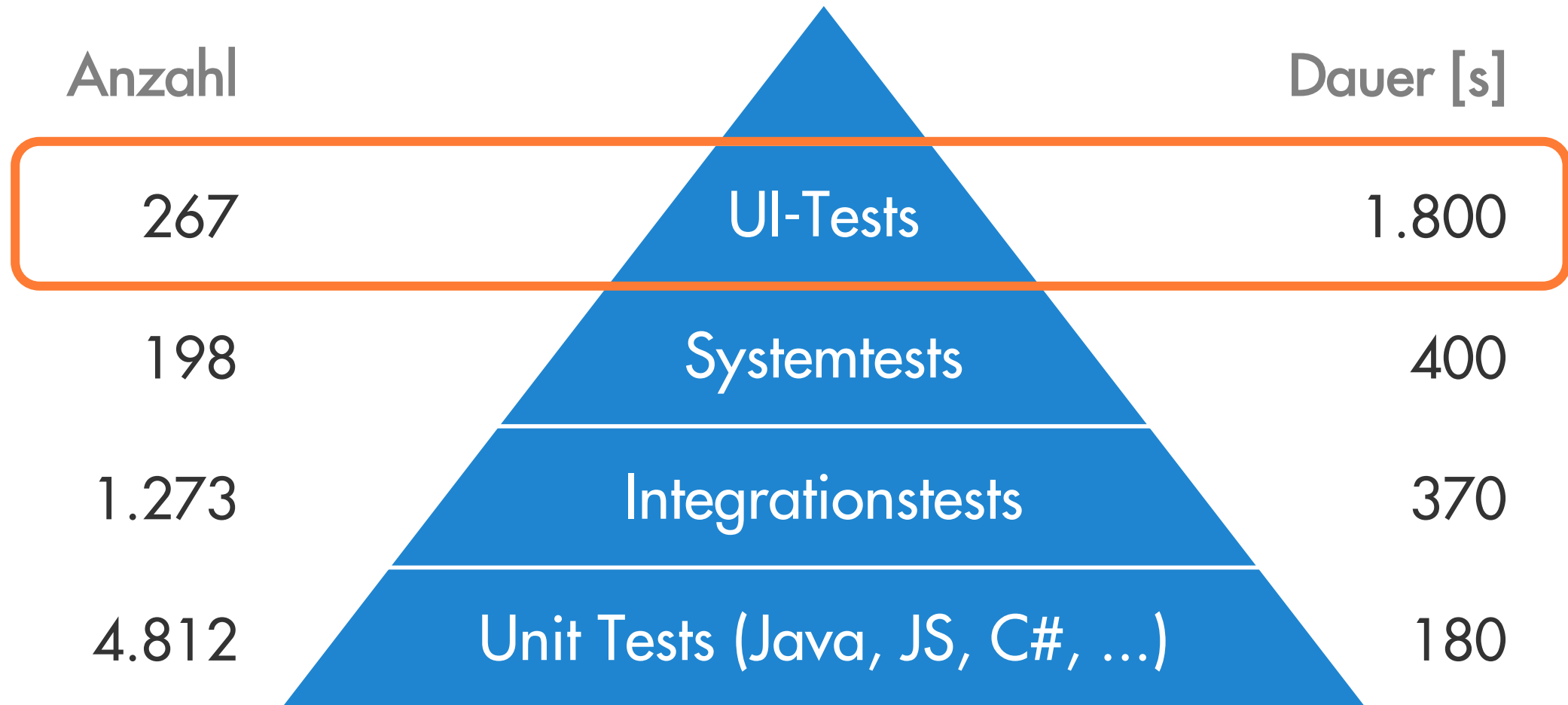


Demo



























[www.teamscale.com](http://www.teamscale.com)

# Wie wir Teamscale testen



























# External Credentials

[+ New External Credentials](#)

| System Type  | Account Name   | Actions   |
|--|--|---|
|  Git                | ABAP Repositories (file:///work-dir/sap.abap.system/)          |          |
|  Git                | CQSE abapGit (https://github.com/cqse/abap-dev.git)            |          |
|  Multiple Systems   | CQSE-JIRA (https://jira.cqse.eu/)                              |          |
|  GitHub           | Github Public (https://github.com/)                            |    |
|  Git              | Github SSH (git@github.com:cqse/teamscale-profiler-dotnet.git) |    |
|  Multiple Systems | Gitlab (https://git.cqse.eu/cqse/teamscale.git)                |    |

# External Credentials

[+ New External Credentials](#)

| System Type  | Account Name   | Actions   | Projects                                 |
|--|--|---|--|
|  Git                | ABAP Repositories (file:///work-dir/sap.abap.system/)          |          |  |
|  Git                | CQSE abapGit (https://github.com/cqse/abap-dev.git)            |          |  |
|  Multiple Systems   | CQSE-JIRA (https://jira.cqse.eu/)                              |          | <a href="#">abap-cqi and 2 more</a>      |
|  GitHub           | Github Public (https://github.com/)                            |    | <a href="#">python-client and 3 more</a> |
|  Git              | Github SSH (git@github.com:cqse/teamscale-profiler-dotnet.git) |    |  |
|  Multiple Systems | Gitlab (https://git.cqse.eu/cqse/teamscale.git)                |    | <a href="#">cqse-all-qc</a>              |



Teamscale / TS-15717

## Account credentials: show list of projects that use the credential

- Edit
- Comment
- Assign
- More ▾
- Back to New

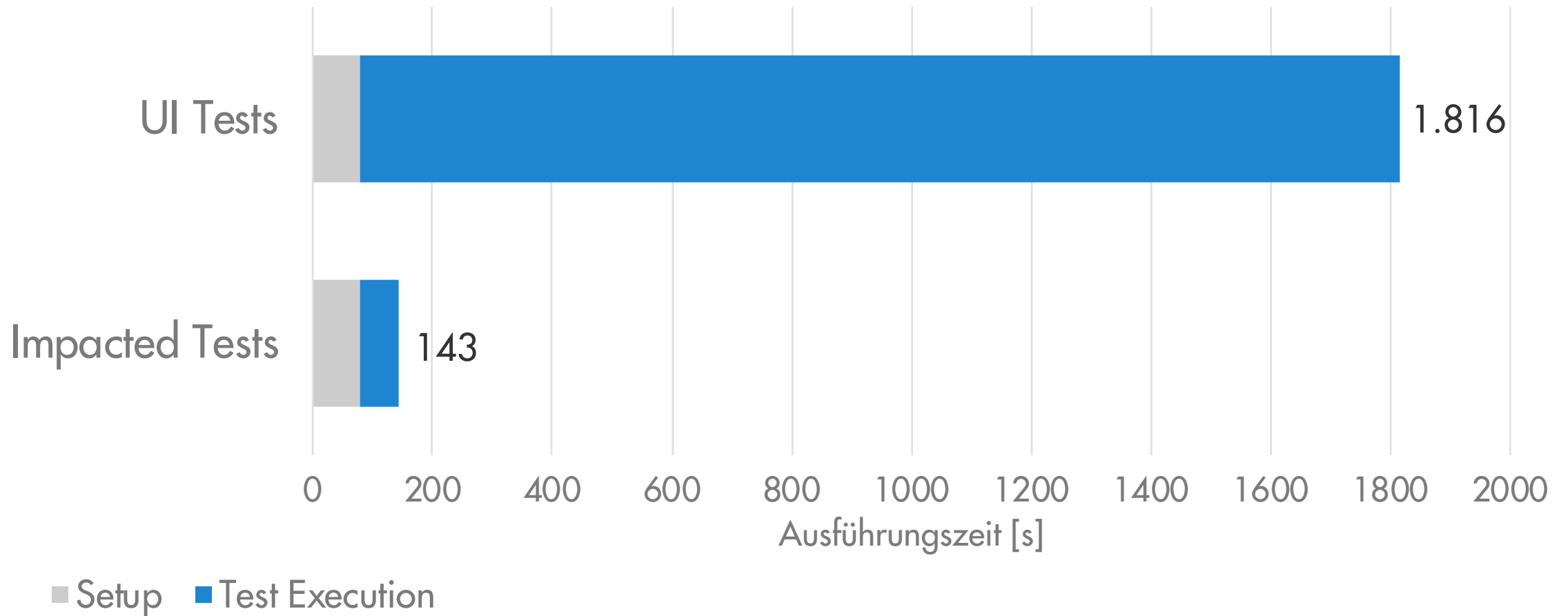
### Details

|                |   |                |                               |
|----------------|---|----------------|-------------------------------|
| Type:          | <span>+</span> Feature  | Status:        | <b>DONE</b> (View Workflow)   |
| Priority:      | <span>↓</span> Normal   | Resolution:    | Green                         |
| Component/s:   | <a href="#">Web Interface</a>   | Fix Version/s: | <a href="#">Teamscale 4.6</a> |
| Labels:        | <a href="#">accounts</a> <a href="#">administration</a> <a href="#">easy</a> <a href="#">students</a>                       |                |                               |
| Merge Request: | <a href="https://git.cqse.eu/cqse/teamscale/merge_requests/3679">https://git.cqse.eu/cqse/teamscale/merge_requests/3679</a> |                |                               |
| Customer:      | [REDACTED]  |                |                               |
| PDash Task:    | #4887   |                |                               |
| Change Log:    | Account credentials show a list of their assigned projects  |                |                               |

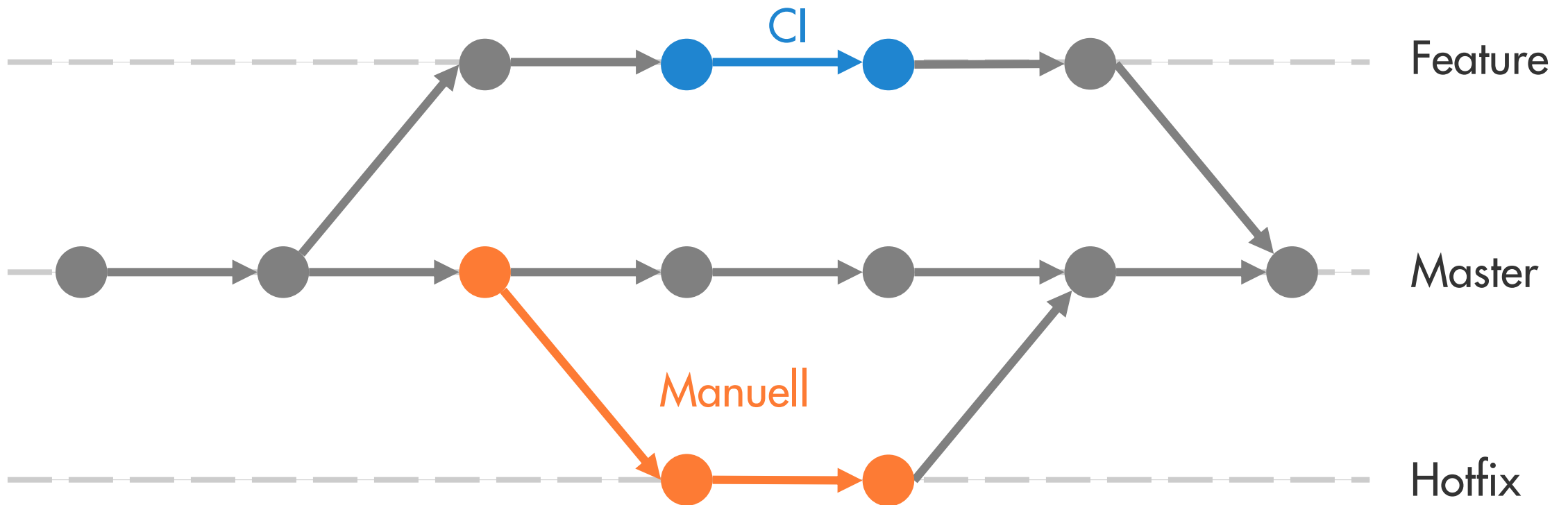
### Description

It would be useful for an admin to know which TS projects are configured to use a particular credential. Especially to locate unused credentials that can be deleted. This could be shown as a short list next to the credential name, similar to how we do this for the analysis profiles.

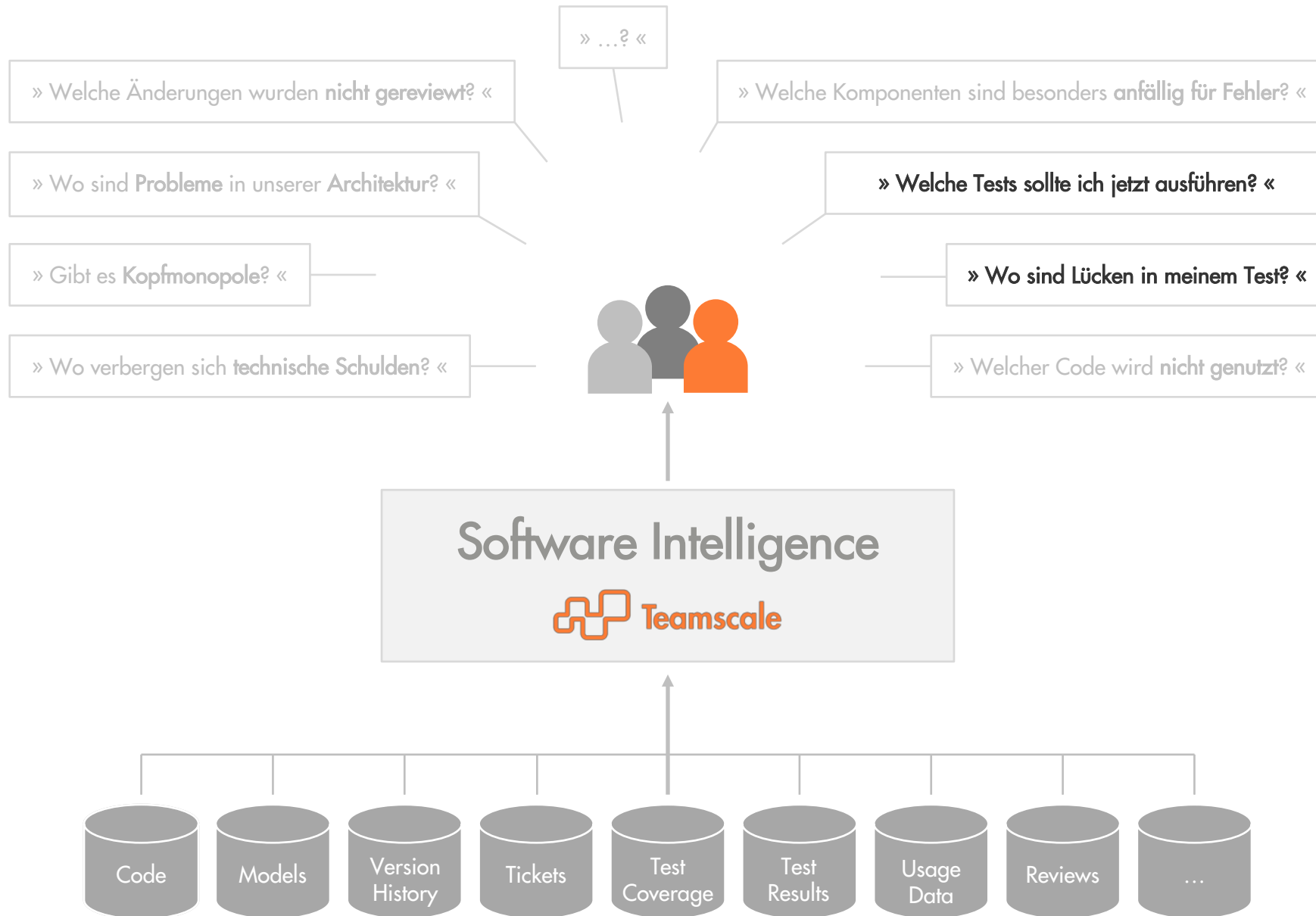
# Zusammenfassung Beispiel



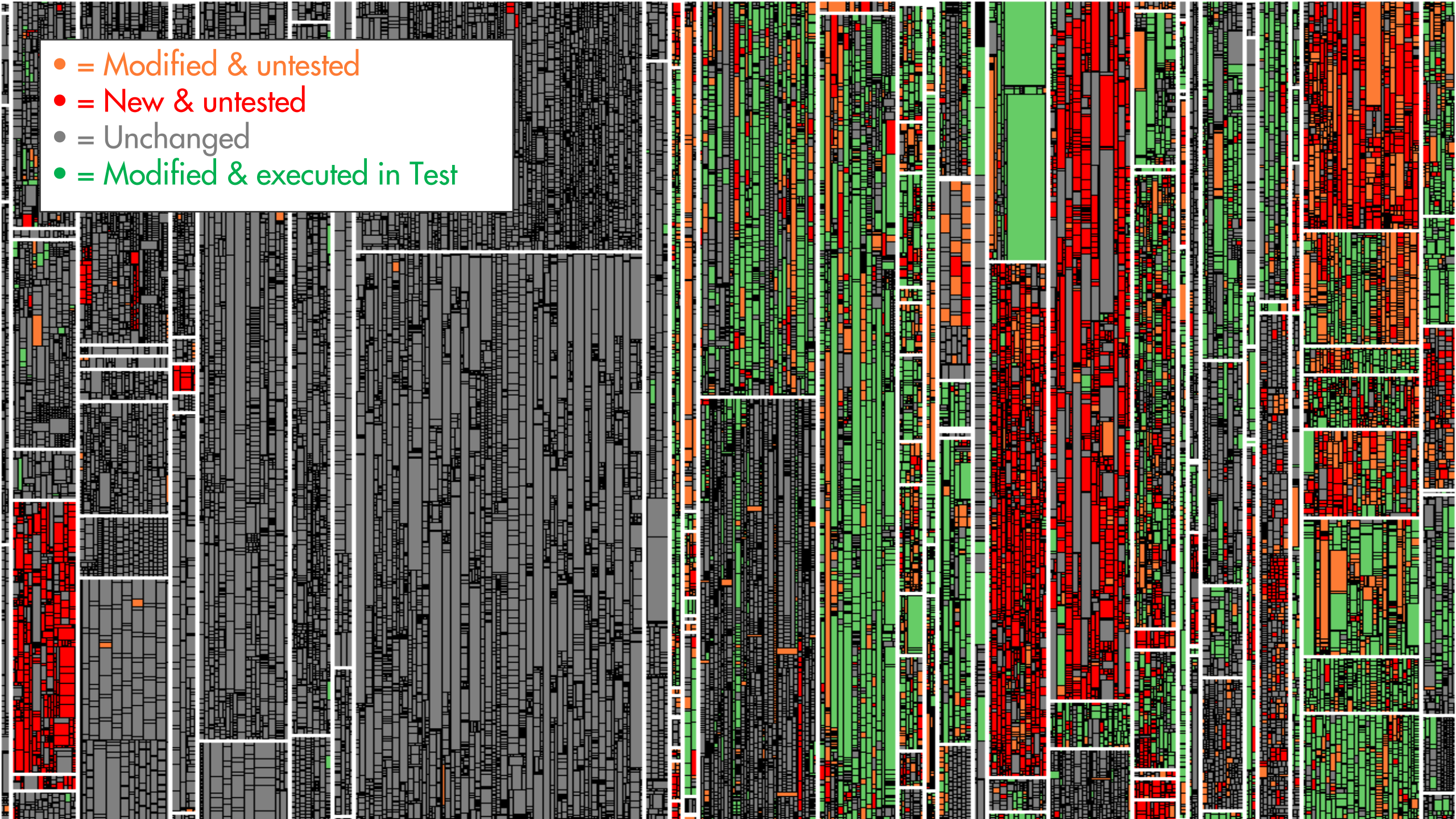
# Beliebige Zeiträume und Branches





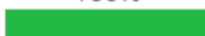
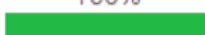
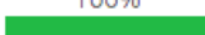
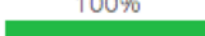
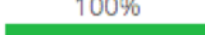
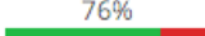




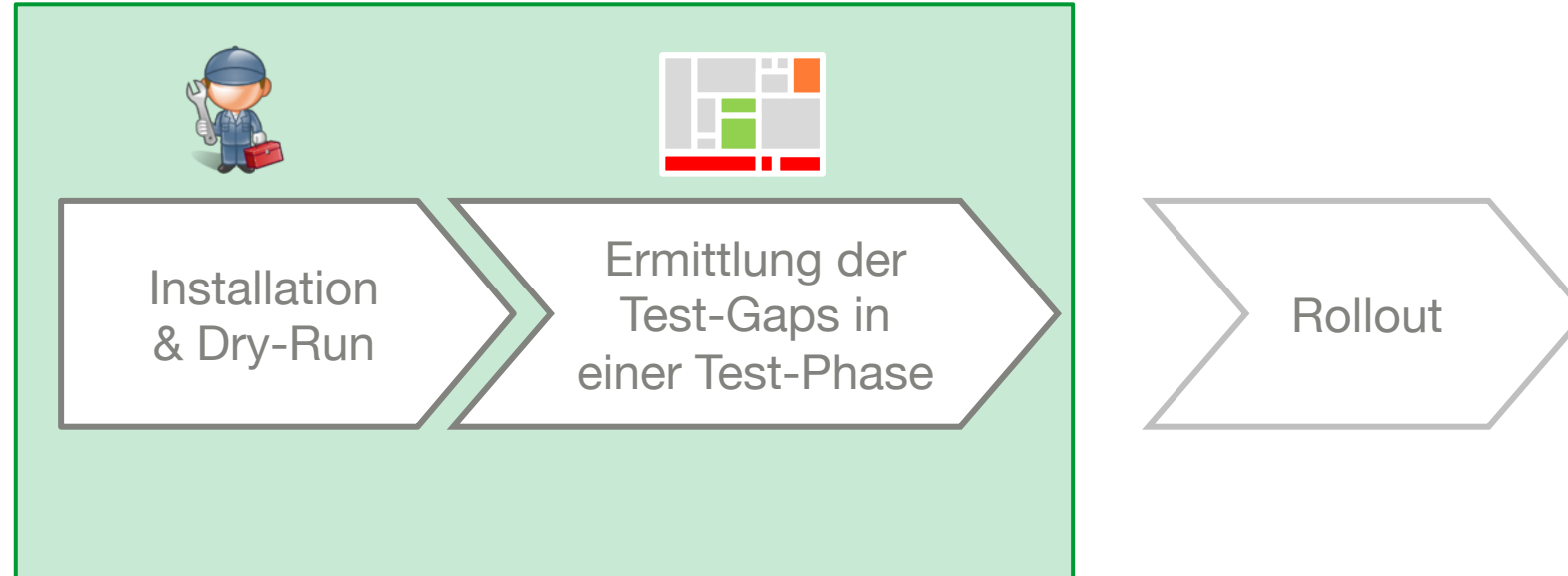


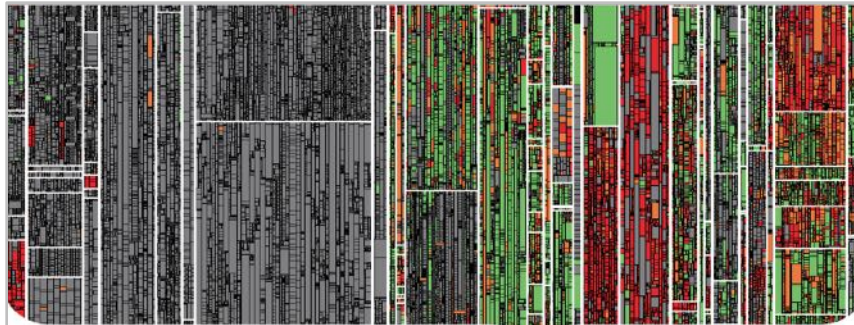
- = Modified & untested
- = New & untested
- = Unchanged
- = Modified & executed in Test



| Issue #                  | Subject  | Ticket Coverage   |
|--------------------------|--|---|
| <a href="#">TS-13689</a> | Undo/Redo for web-based architecture editor  | 70%<br>    |
| <a href="#">TS-13701</a> | Implement metric 'Nesting Depth' for Simulink                                      | 35%<br>    |
| <a href="#">TS-13704</a> | Fix long method finding in TaintAnalysisRunner                                     | 100%<br>   |
| <a href="#">TS-13714</a> | External findings are not registered during first upload                           | 100%<br>   |
| <a href="#">TS-13727</a> | Manual test coverage upload during development                                     | 100%<br>   |
| <a href="#">TS-13731</a> | Tool for transferring Finding Blacklists and Tasks (w/ Findings) between instances | 100%<br>   |
| <a href="#">TS-13734</a> | Cannot set/alter alias without reanalysis  | 100%<br>  |
| <a href="#">TS-13735</a> | Fetch parent relationship of TFS Work Items  | 100%<br> |
| <a href="#">TS-13742</a> | Show data timestamp in project analysis warning                                    | 100%<br> |
| <a href="#">TS-13766</a> | Get rid of ESLintFindingsSynchronizer and TSLintFindingsSynchronizer blocks        | 76%<br>  |

# Pilotierung





# Immer kürzere Testphasen? Mit Ticket Coverage verhindern, dass wichtige Features ungetestet bleiben



## Immer kürzere Testphasen? Mit Ticket Coverage verhindern, dass wichtige Features ungetestet bleiben

Klausur Jürgen  
CQSE Chair  
juergen@cqse.eu

Dieter Pappas  
CQSE Chair  
pappas@cqse.eu

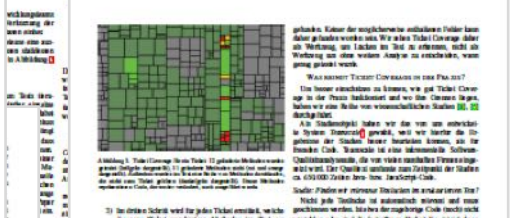
**Zusammenfassung** – In vielen Systemen werden die Release-Zyklen immer kürzer. Dabei wird immer weniger Zeit für Software-Testphasen zur Verfügung. Diese Zeit muss deshalb über mehrere Entwicklungsphasen verteilt werden. Die dabei aufgetretene Herausforderung ist, wie man sicherstellt, dass alle wichtigen Features getestet werden. In diesem Artikel wird ein Ansatz vorgestellt, um diese Herausforderung zu bewältigen. Der Ansatz besteht darin, die Testphasen in kleine, übersichtliche Einheiten zu unterteilen, die jeweils einem bestimmten Feature zugeordnet werden können. Diese Einheiten sind als Tickets bezeichnet. Durch die Verwendung von Tickets kann man sicherstellen, dass alle wichtigen Features getestet werden, bevor ein neues Release in den Markt gebracht wird.

**Qualitätsmanagement von Teams**  
Bei großen Systemen stellen sich immer mehr Teams vor, die die Qualität des Produkts sicherstellen müssen. In der Praxis ist es oft schwierig, die Qualität zu gewährleisten, wenn man viele Teams hat. Ein Ansatz, um die Qualität zu gewährleisten, ist die Verwendung von Tickets. Tickets sind kleine Einheiten von Arbeit, die einem Team zugeordnet werden können. Durch die Verwendung von Tickets kann man sicherstellen, dass alle wichtigen Features getestet werden, bevor ein neues Release in den Markt gebracht wird.

**Das Ticket-Analyse-Werkzeug**  
Das Ticket-Analyse-Werkzeug ist ein Werkzeug, das die Tickets analysiert und die Ergebnisse in einer übersichtlichen Weise darstellt. Es ermöglicht, die Tickets zu gruppieren und die Ergebnisse zu visualisieren. Das Ticket-Analyse-Werkzeug ist ein wichtiges Werkzeug für die Qualitätssicherung von Teams.

**Die Herausforderung**  
Die Herausforderung besteht darin, die Testphasen in kleine, übersichtliche Einheiten zu unterteilen, die jeweils einem bestimmten Feature zugeordnet werden können. Diese Einheiten sind als Tickets bezeichnet. Durch die Verwendung von Tickets kann man sicherstellen, dass alle wichtigen Features getestet werden, bevor ein neues Release in den Markt gebracht wird.

**Die Lösung**  
Die Lösung besteht darin, die Testphasen in kleine, übersichtliche Einheiten zu unterteilen, die jeweils einem bestimmten Feature zugeordnet werden können. Diese Einheiten sind als Tickets bezeichnet. Durch die Verwendung von Tickets kann man sicherstellen, dass alle wichtigen Features getestet werden, bevor ein neues Release in den Markt gebracht wird.



| Modul   | Test Coverage | Ungetestet | Ergebnis    |
|---------|---------------|------------|-------------|
| Modul A | 100%          | 0          | erfolgreich |
| Modul B | 80%           | 20%        | erfolgreich |
| Modul C | 60%           | 40%        | erfolgreich |
| Modul D | 40%           | 60%        | erfolgreich |
| Modul E | 20%           | 80%        | erfolgreich |
| Modul F | 10%           | 90%        | erfolgreich |
| Modul G | 5%            | 95%        | erfolgreich |
| Modul H | 2%            | 98%        | erfolgreich |
| Modul I | 1%            | 99%        | erfolgreich |
| Modul J | 0%            | 100%       | erfolgreich |



**Die Herausforderung**  
Die Herausforderung besteht darin, die Testphasen in kleine, übersichtliche Einheiten zu unterteilen, die jeweils einem bestimmten Feature zugeordnet werden können. Diese Einheiten sind als Tickets bezeichnet. Durch die Verwendung von Tickets kann man sicherstellen, dass alle wichtigen Features getestet werden, bevor ein neues Release in den Markt gebracht wird.

**Die Lösung**  
Die Lösung besteht darin, die Testphasen in kleine, übersichtliche Einheiten zu unterteilen, die jeweils einem bestimmten Feature zugeordnet werden können. Diese Einheiten sind als Tickets bezeichnet. Durch die Verwendung von Tickets kann man sicherstellen, dass alle wichtigen Features getestet werden, bevor ein neues Release in den Markt gebracht wird.

# Software Intelligence



Welche Überraschungen lauern in Ihrer Software?

Welche Änderungen am Code sind ungetestet?

Wurden wichtige Features im Test vernachlässigt?

Welche Tests sind diesmal am wichtigsten?



Was er  
im ex  
Wie  
auf me

chle  
in der  
er?

QS



TAG  
s-tag.de



# Kontakt – Wir freuen uns auf Diskussionen 😊



Dr. Elmar Jürgens · [juergens@cqse.eu](mailto:juergens@cqse.eu) · +49 179 675 3863

Dr. Andreas Göb · [goeb@cqse.eu](mailto:goeb@cqse.eu) · +49 176 101 55225

CQSE GmbH  
Lichtenbergstraße 8  
85748 Garching bei München  
[www.cqse.eu](http://www.cqse.eu)

**CQSE**