Von Clean Code zu Clean Software

Dr. Elmar Jürgens

CQSE

Continuous Quality in Software Engineering

Über Mich

Forschung

- Clone Detection, Architekturanalyse, …
- PC Mitglied von MSR, ICPC, ICSE, …

Beratung

- Gründer
- Qualitäts-Bewertung & Qualitäts-Controlling

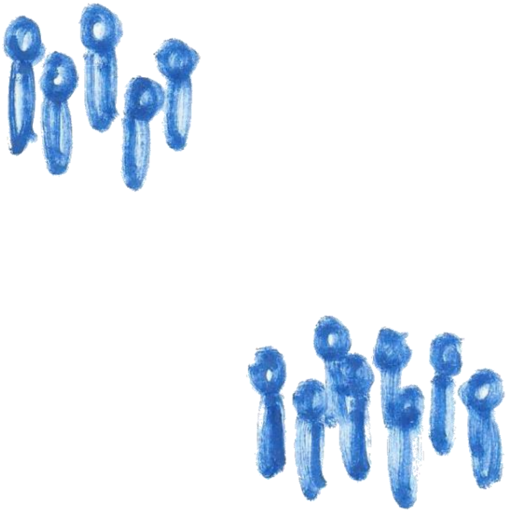
Gesellschaft für Informatik

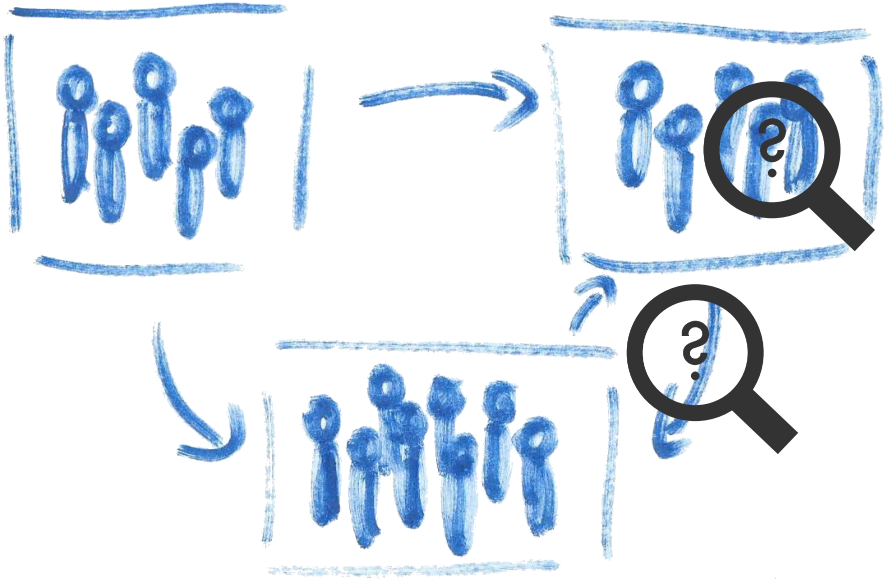
- Zum Junior-Fellow ernannt
- Erfahrungsaustausch Forschung <-> Praxis

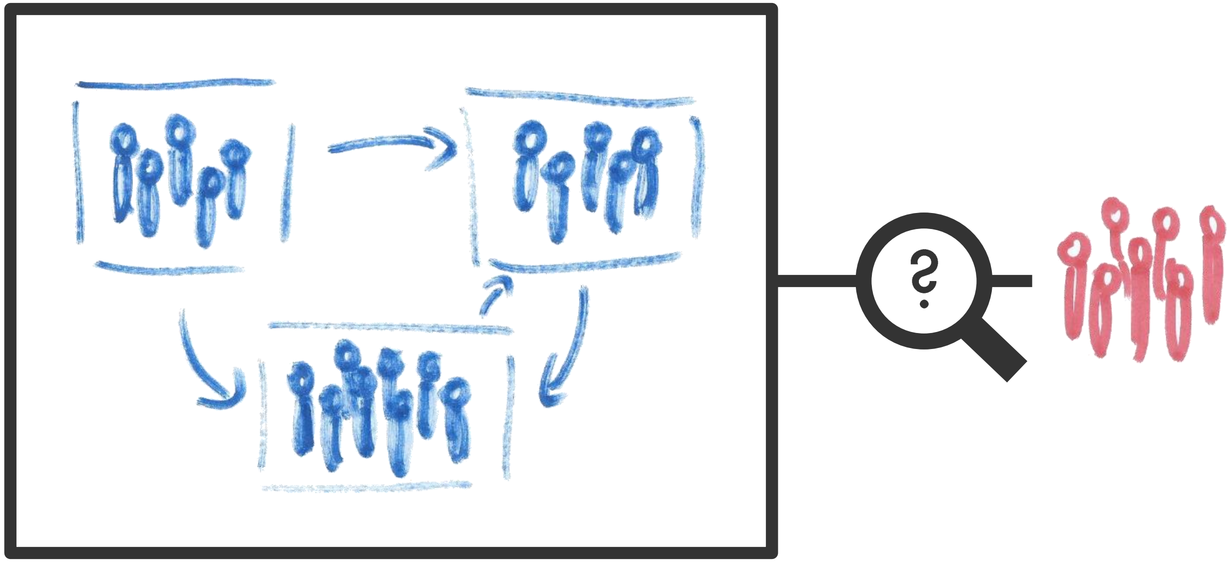
Conway's „Law"

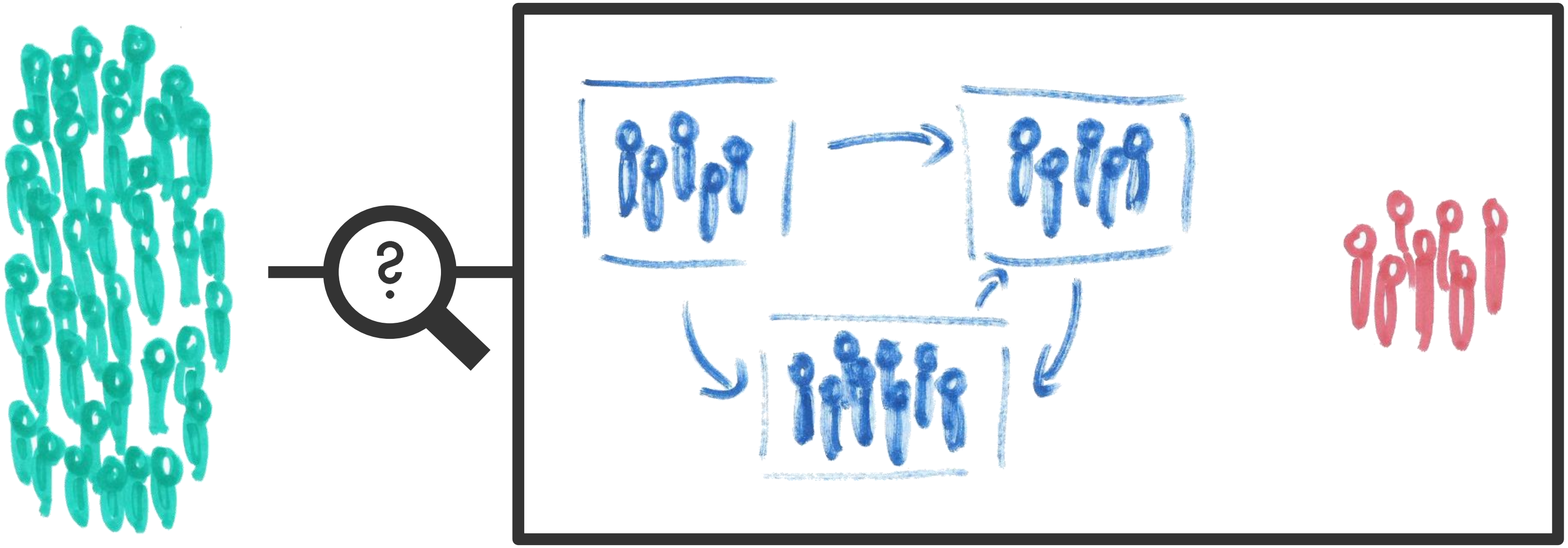
Organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations.

Melvin Conway, 1968

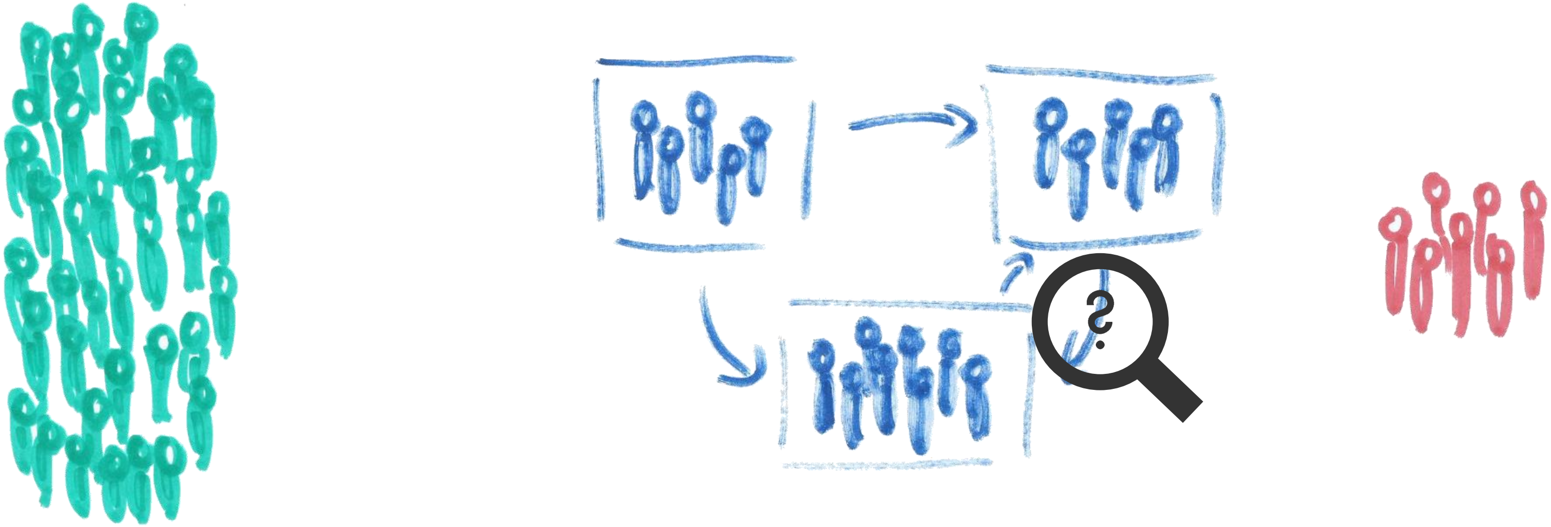


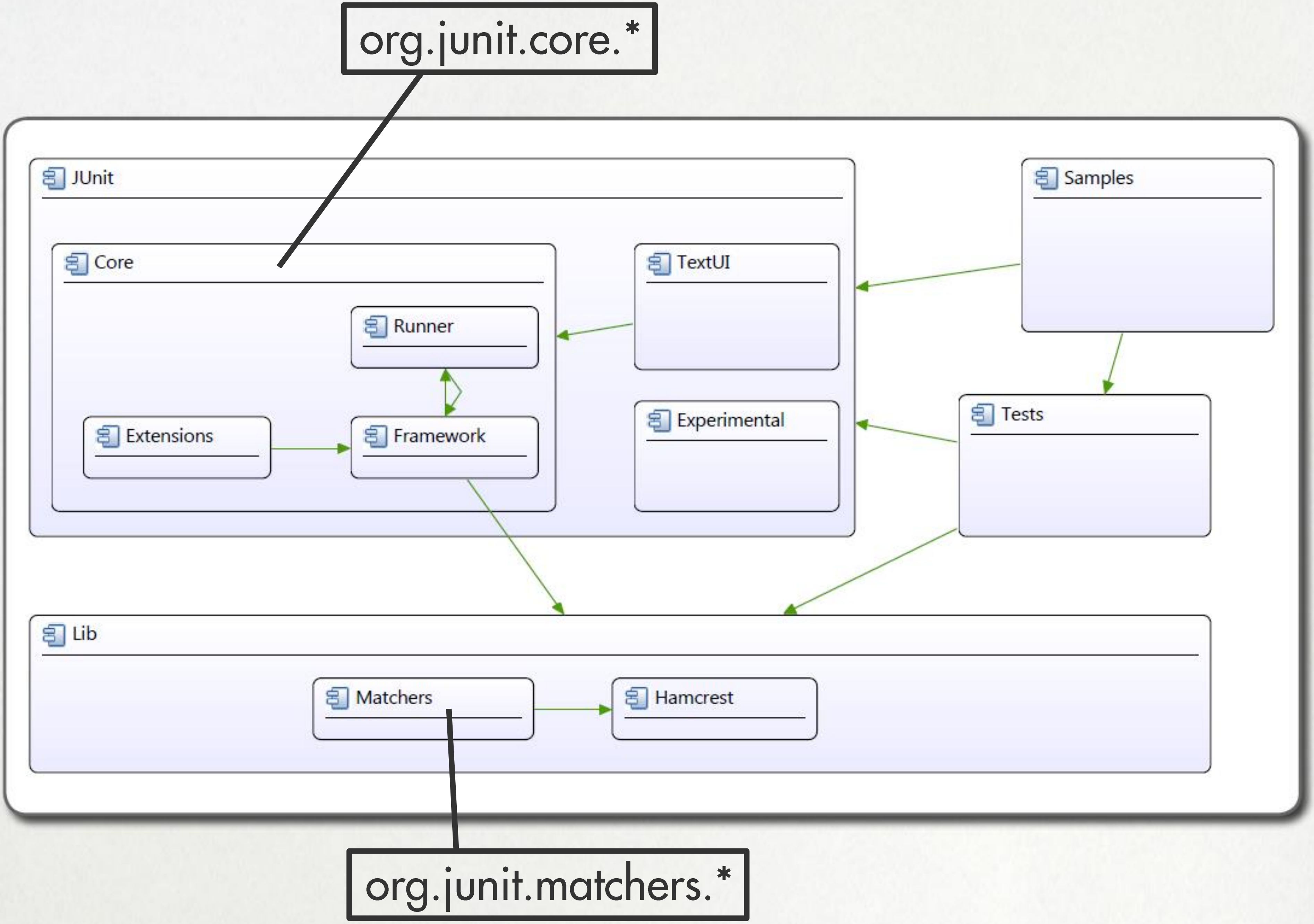






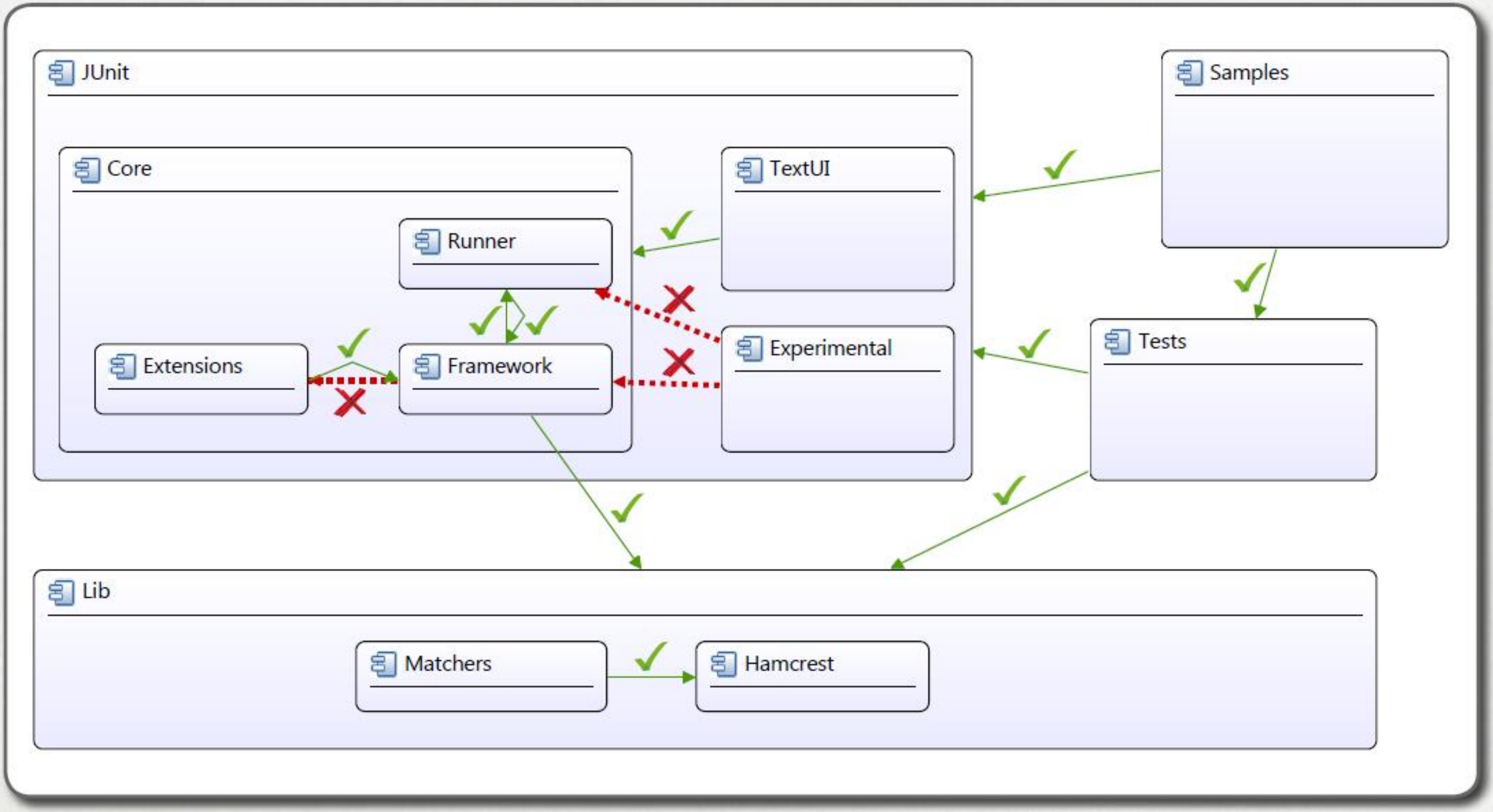
Was soll das?

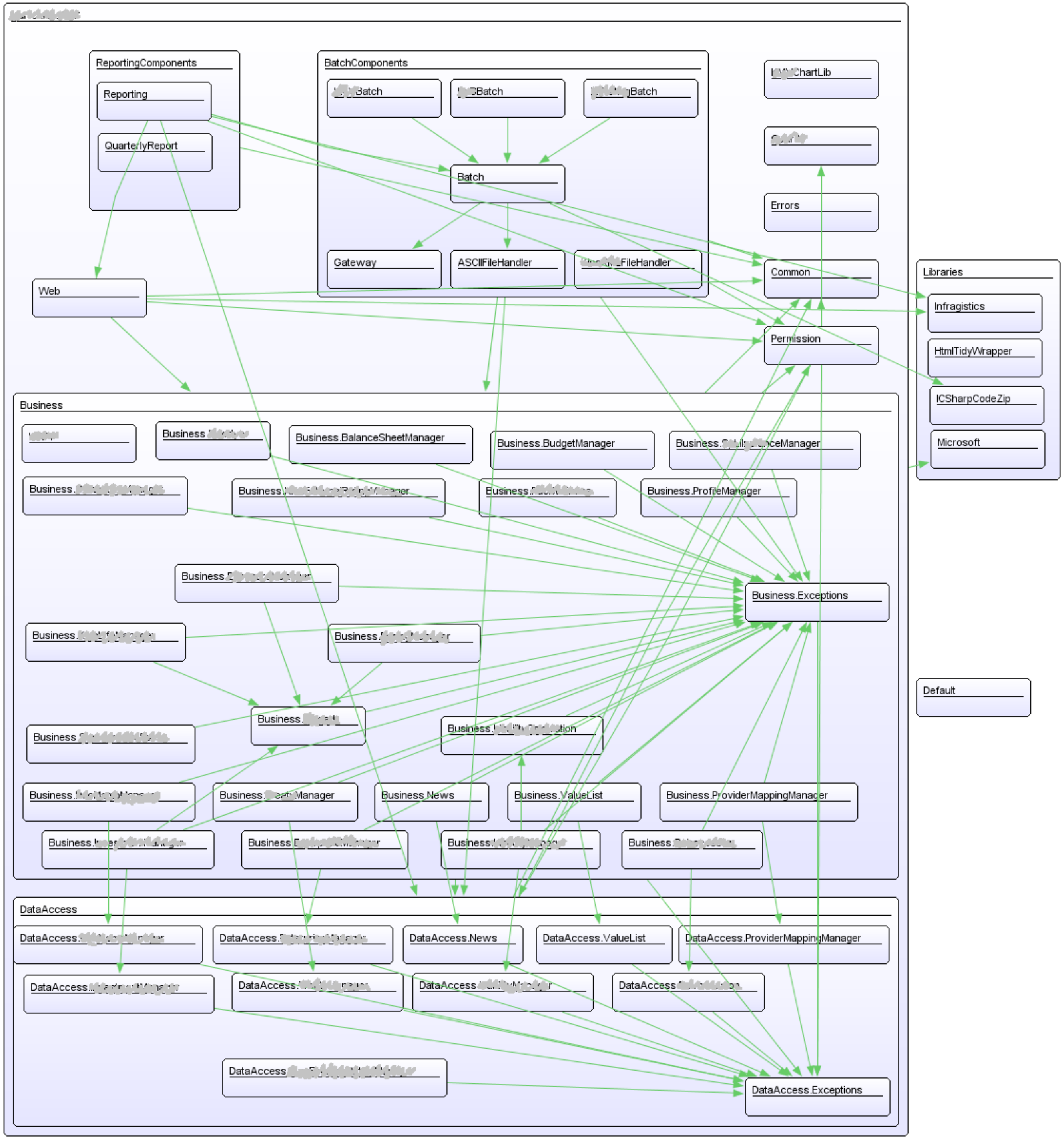


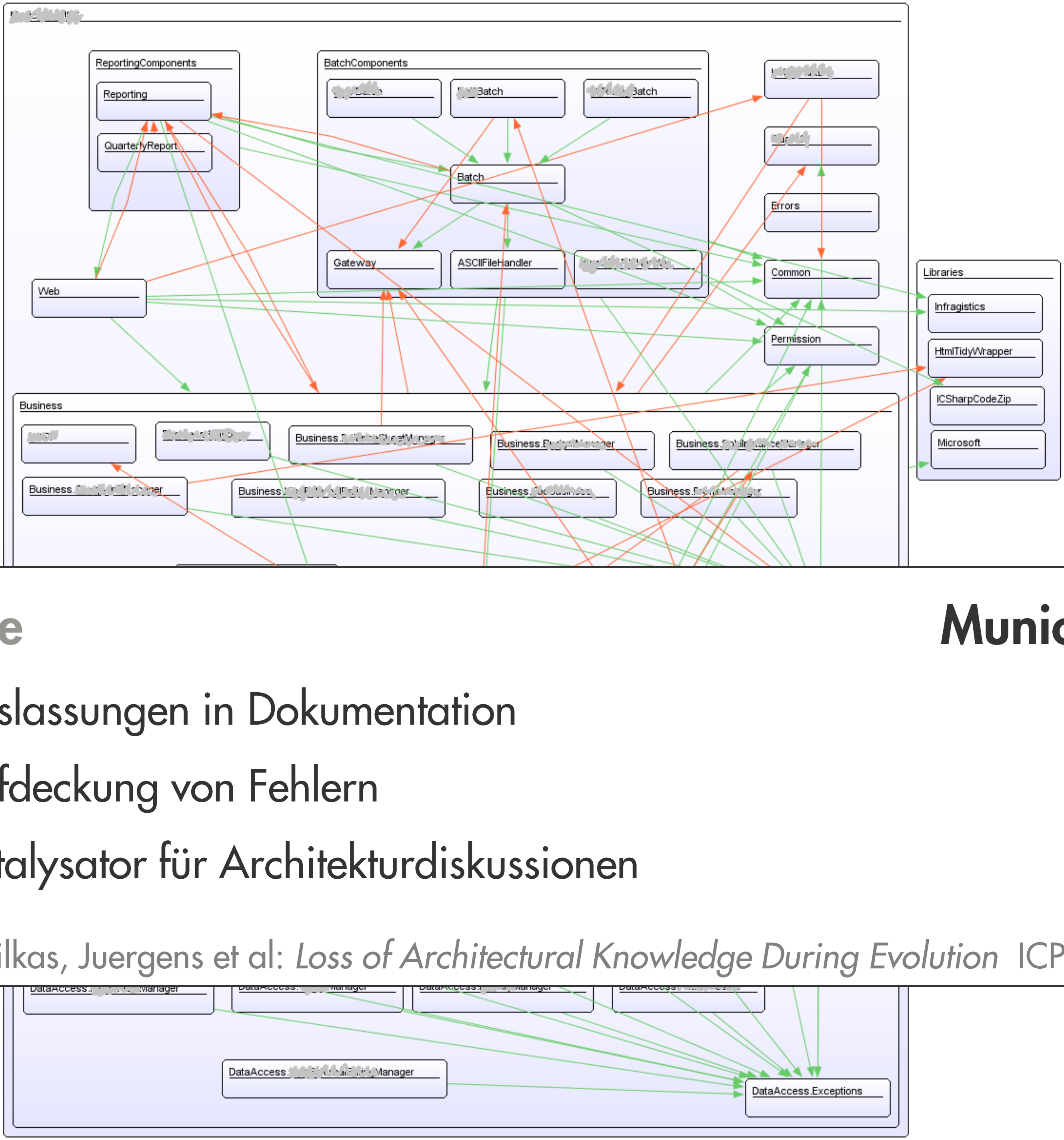


org.junit.core.*

org.junit.matchers.*


JUnit
Core
Runner
Extensions
Framework
TextUI
Experimental
Samples
Tests
Lib
Matchers
Hamcrest

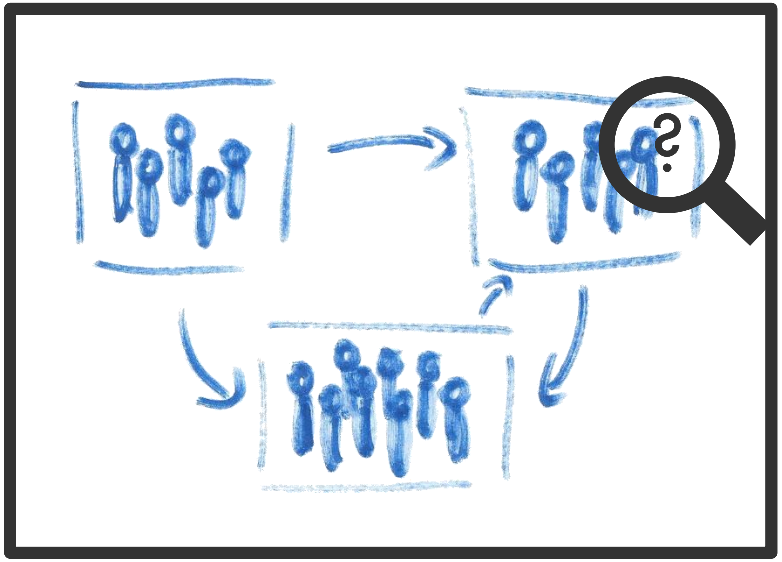

[redacted]
ReportingComponents
Reporting
QuarterlyReport
BatchComponents
[redacted]Batch
[redacted]Batch
[redacted]Batch
Batch
Gateway
ASCIIFileHandler
[redacted]FileHandler
[redacted]ChartLib
[redacted]
Errors
Common
Permission
Libraries
Infragistics
HtmlTidyWrapper
ICSharpCodeZip
Microsoft
Web
Default
Business
[redacted]
Business.[redacted]
Business.BalanceSheetManager
Business.BudgetManager
Business.[redacted]ceManager
Business.[redacted]
Business.[redacted]Manager
Business.[redacted]
Business.ProfileManager
Business.[redacted]
Business.Exceptions
Business.[redacted]
Business.[redacted]
Business.[redacted]
Business.[redacted]
Business.[redacted]tion
Business.[redacted]
Business.[redacted]Manager
Business.News
Business.ValueList
Business.ProviderMappingManager
Business.[redacted]
Business.[redacted]
Business.[redacted]
Business.[redacted]
DataAccess
DataAccess.[redacted]
DataAccess.[redacted]
DataAccess.News
DataAccess.ValueList
DataAccess.ProviderMappingManager
DataAccess.[redacted]
DataAccess.[redacted]
DataAccess.[redacted]
DataAccess.[redacted]
DataAccess.[redacted]
DataAccess.Exceptions

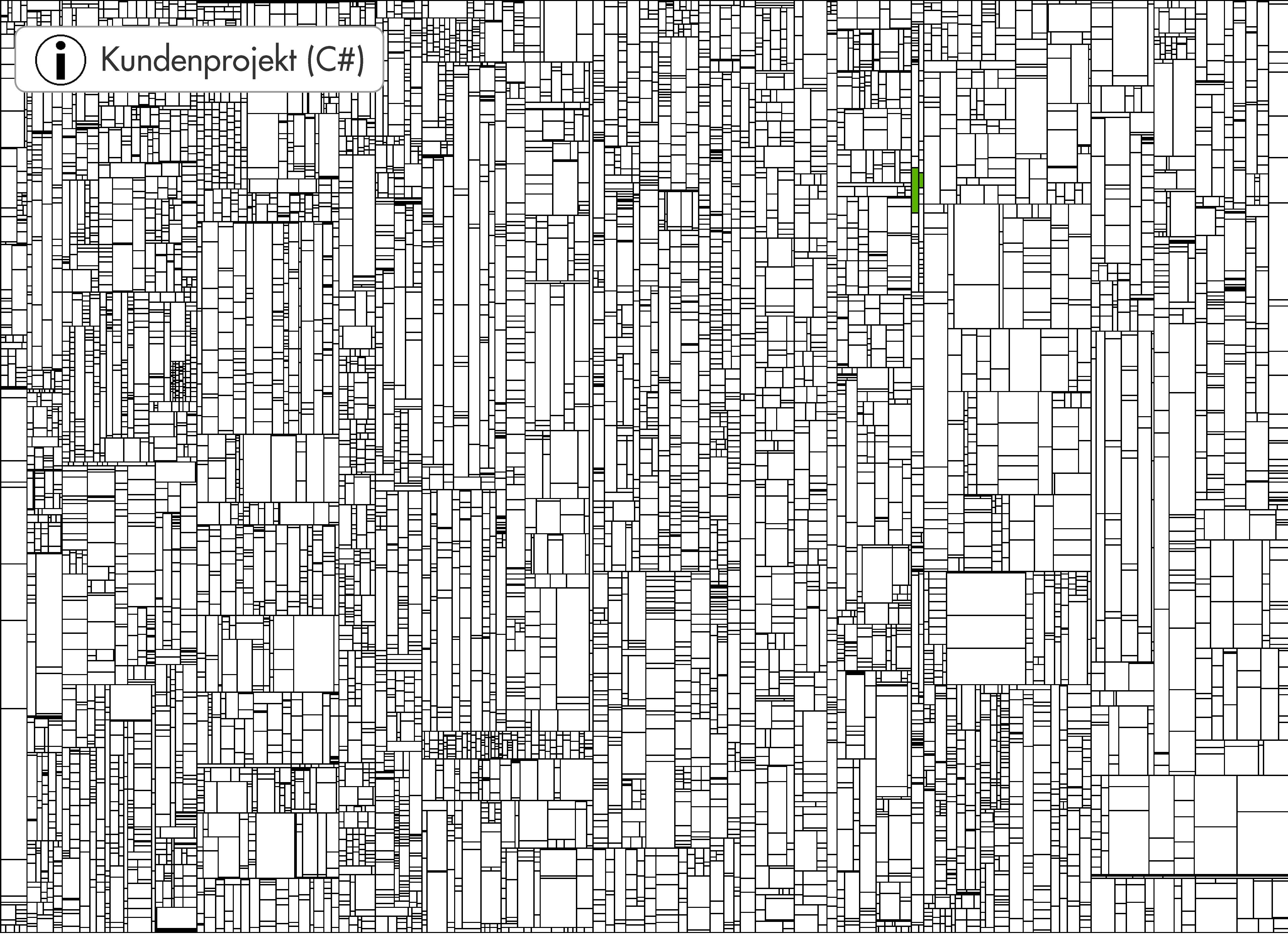


Studie **Munich Re**

- Auslassungen in Dokumentation
- Aufdeckung von Fehlern
- Katalysator für Architekturdiskussionen

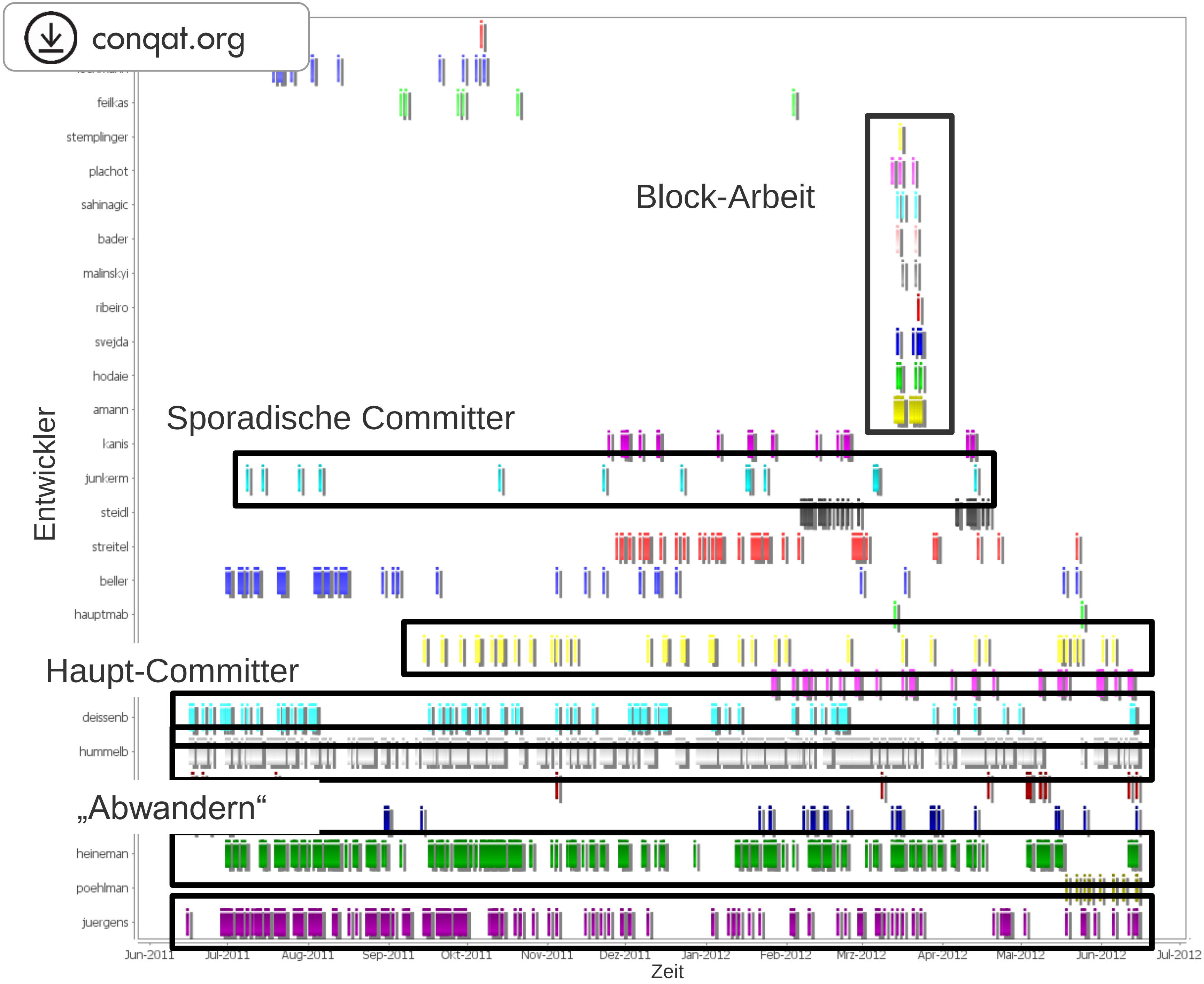
Feilkas, Juergens et al: *Loss of Architectural Knowledge During Evolution* ICPC 2009





Kundenprojekt (C#)

conqat.org



Block-Arbeit

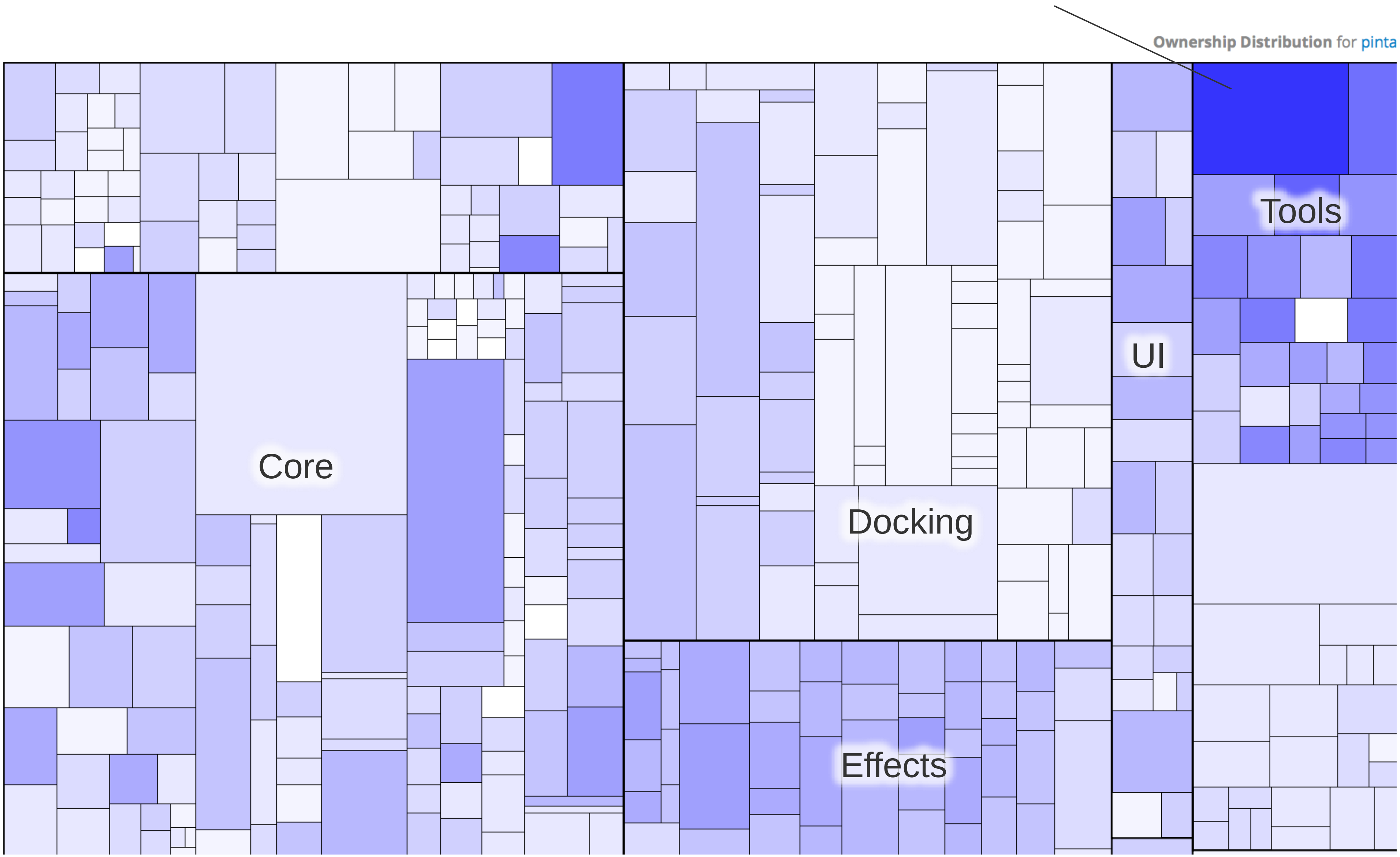
Sporadische Committer

Haupt-Committer

„Abwandern“

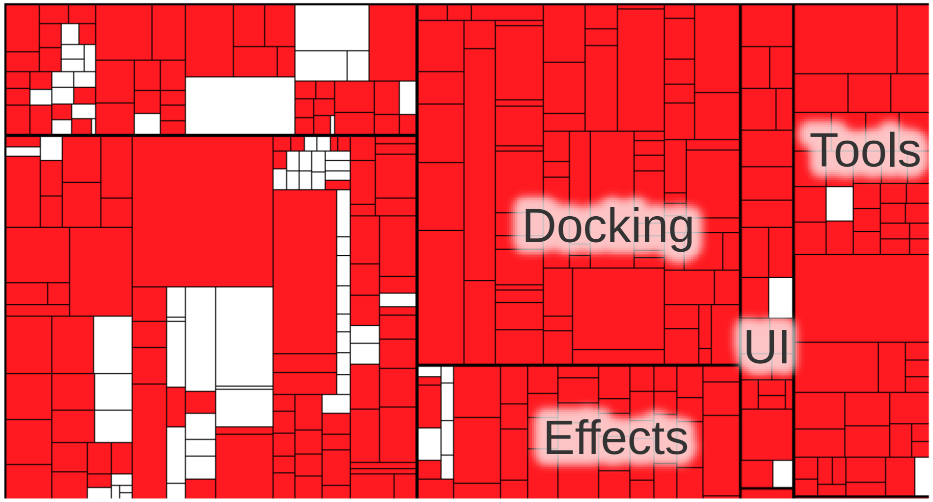
PintaProject/Pinta

TextTool.cs (17 Entwickler)

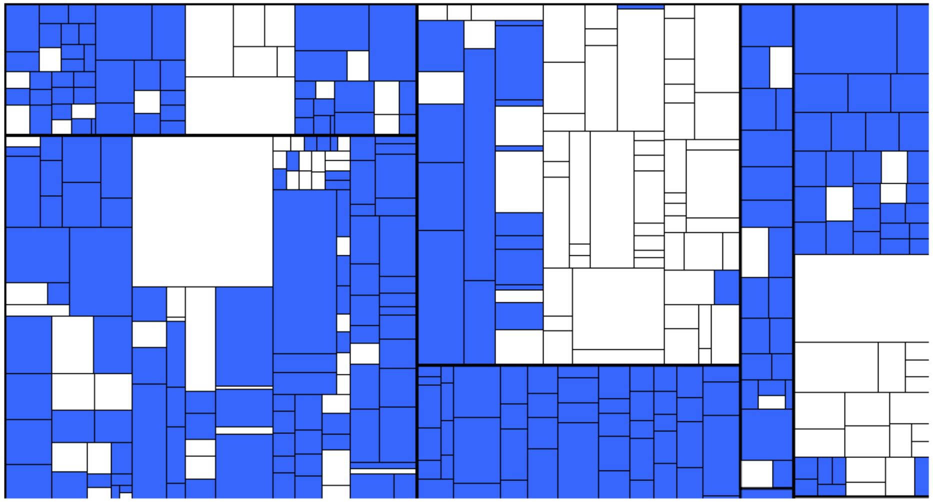


PintaProject/Pinta

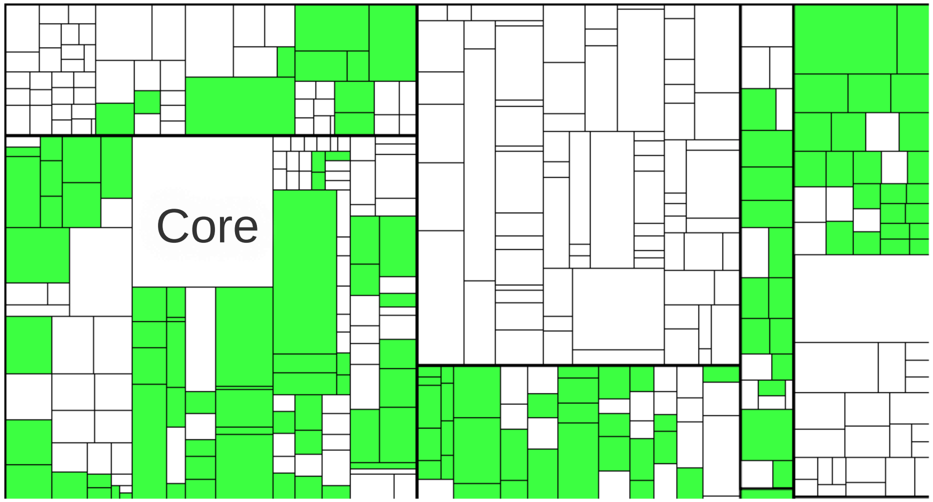
Ownership Cameron White (#1)



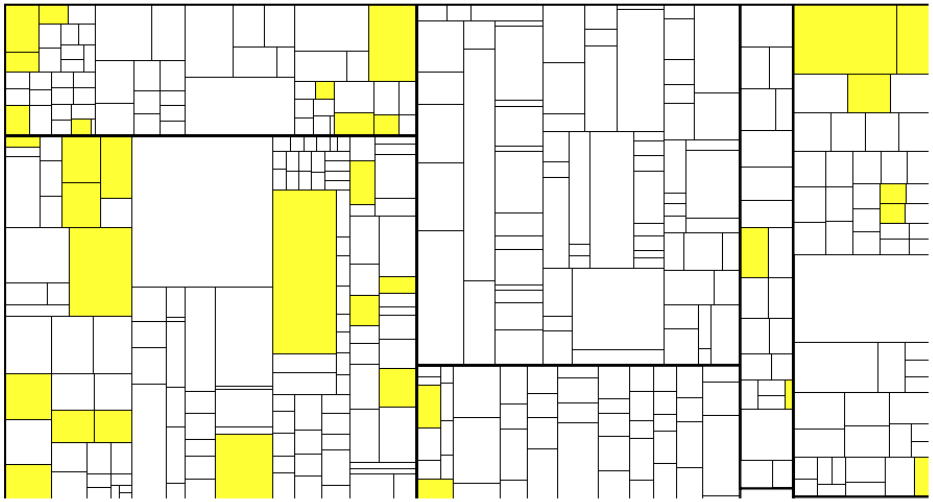
Ownership Jonathan Pobst (#2)

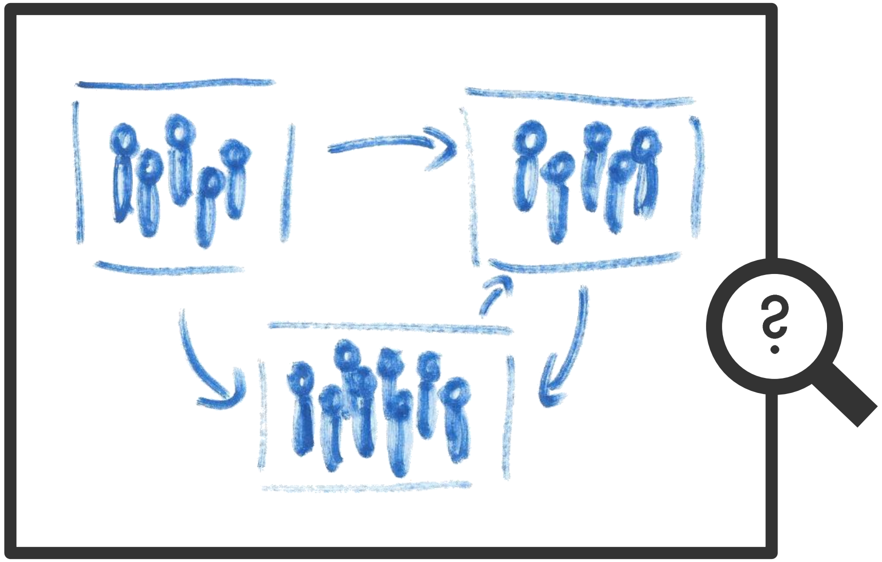


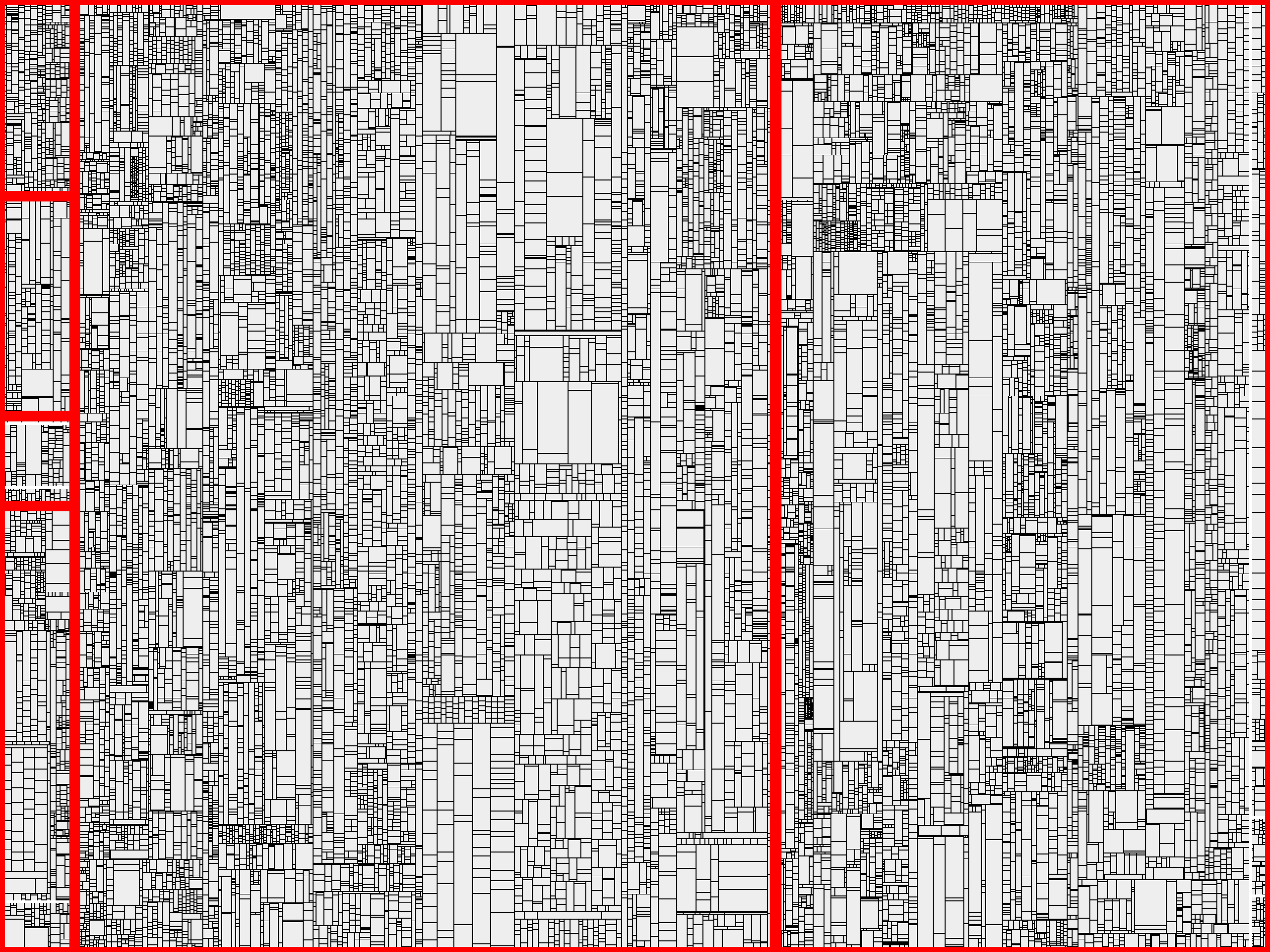
Ownership Olivier Dufour (#3)

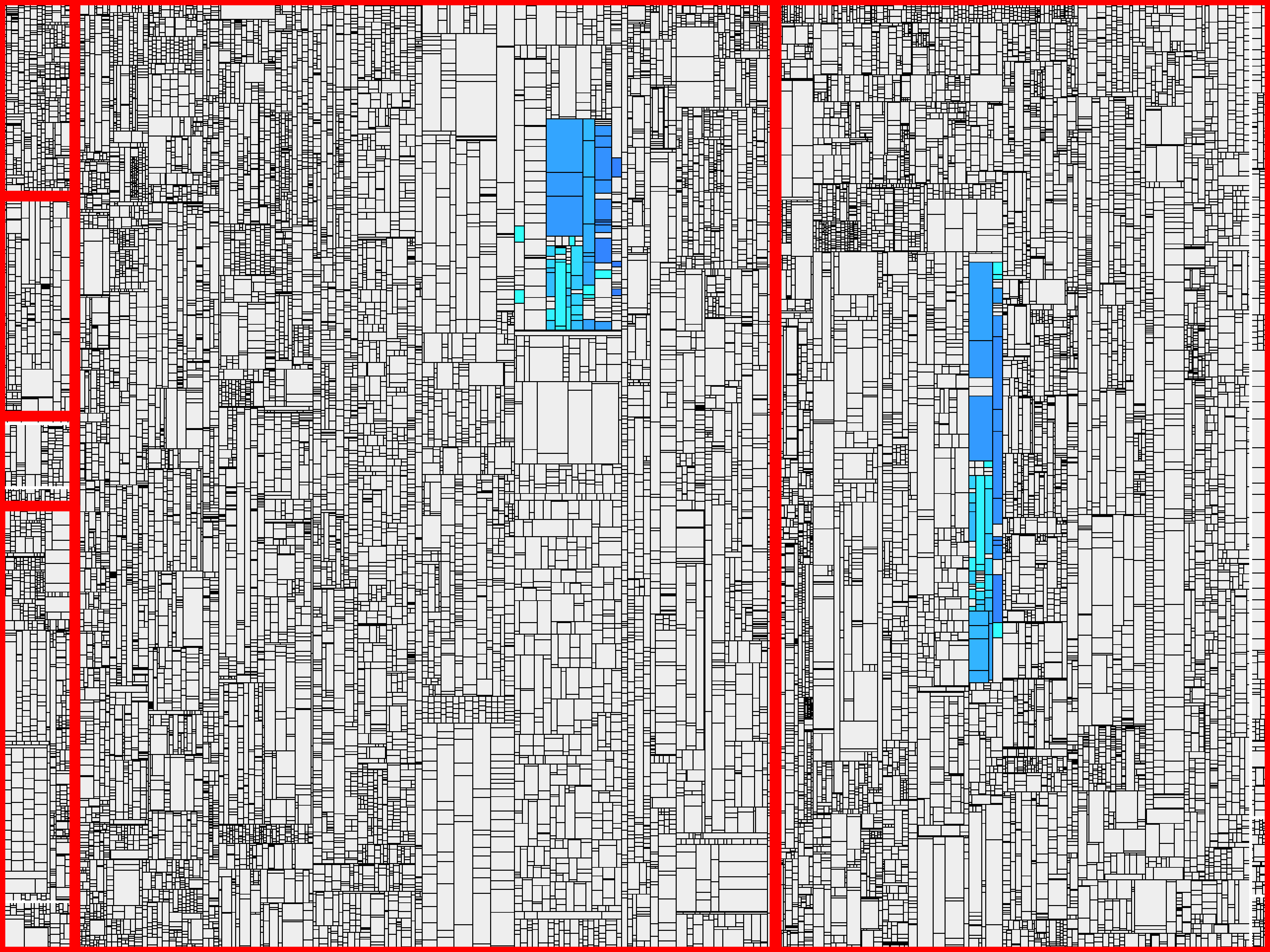


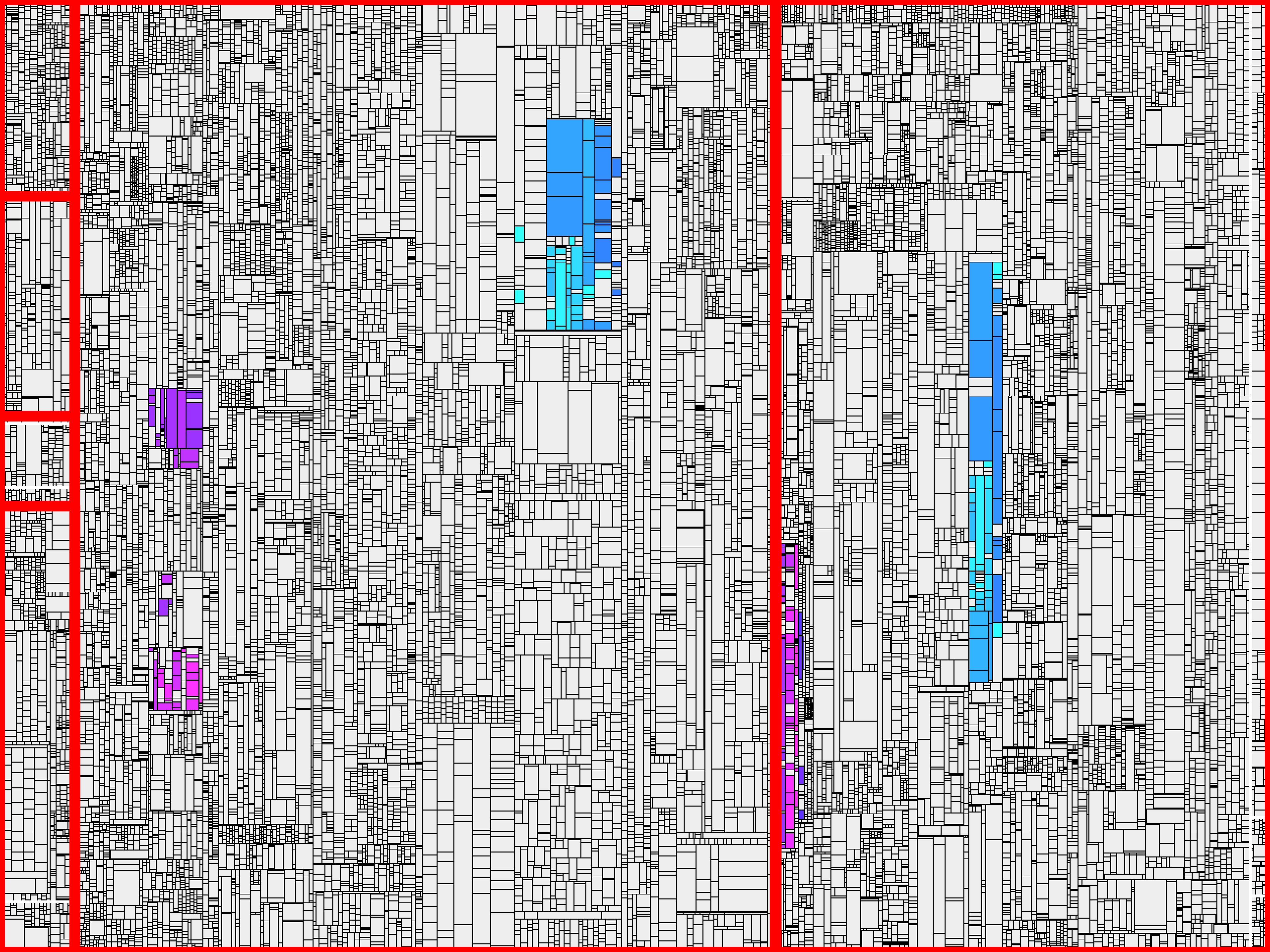
Ownership Robert Nordan (#4)

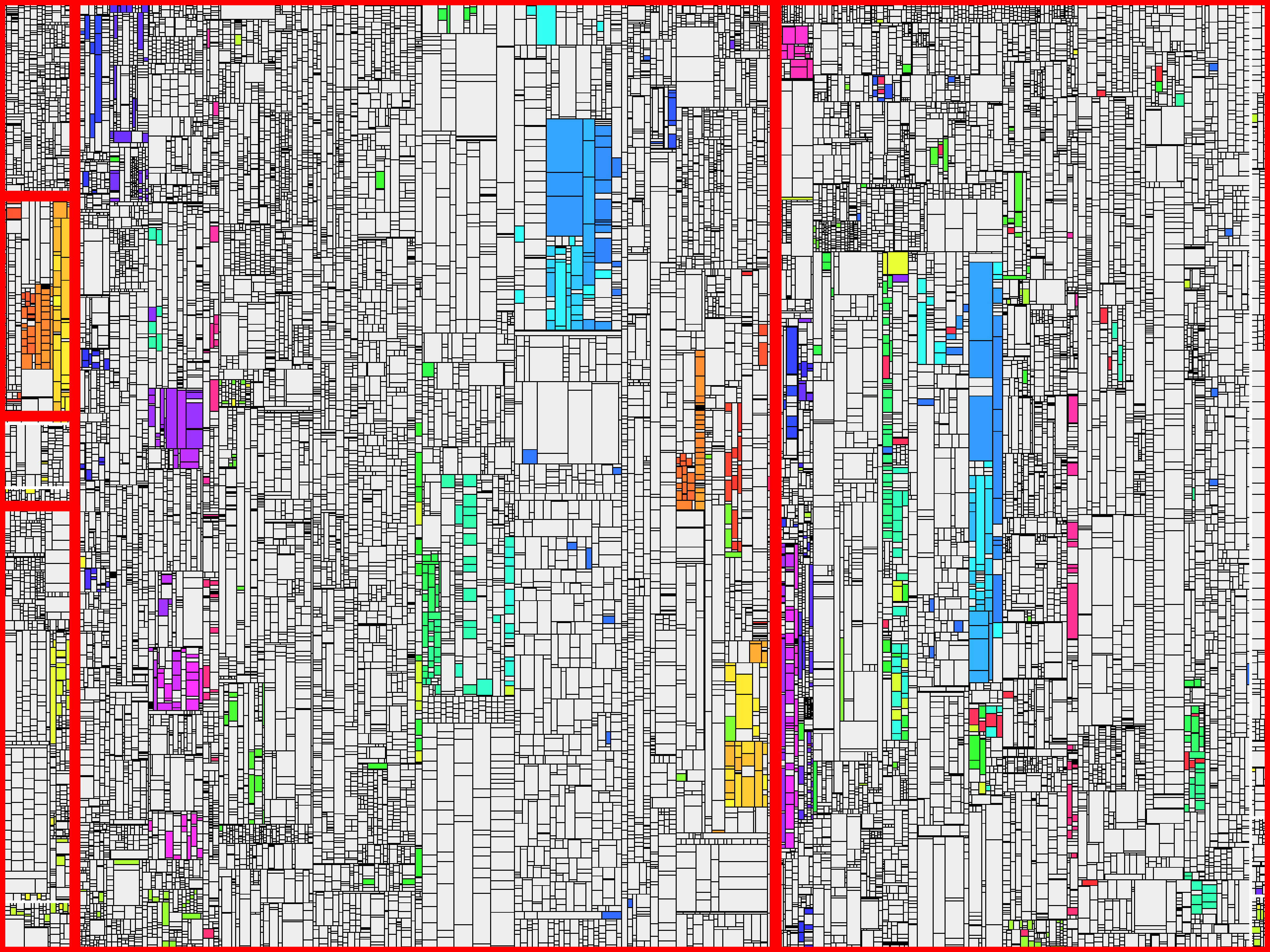




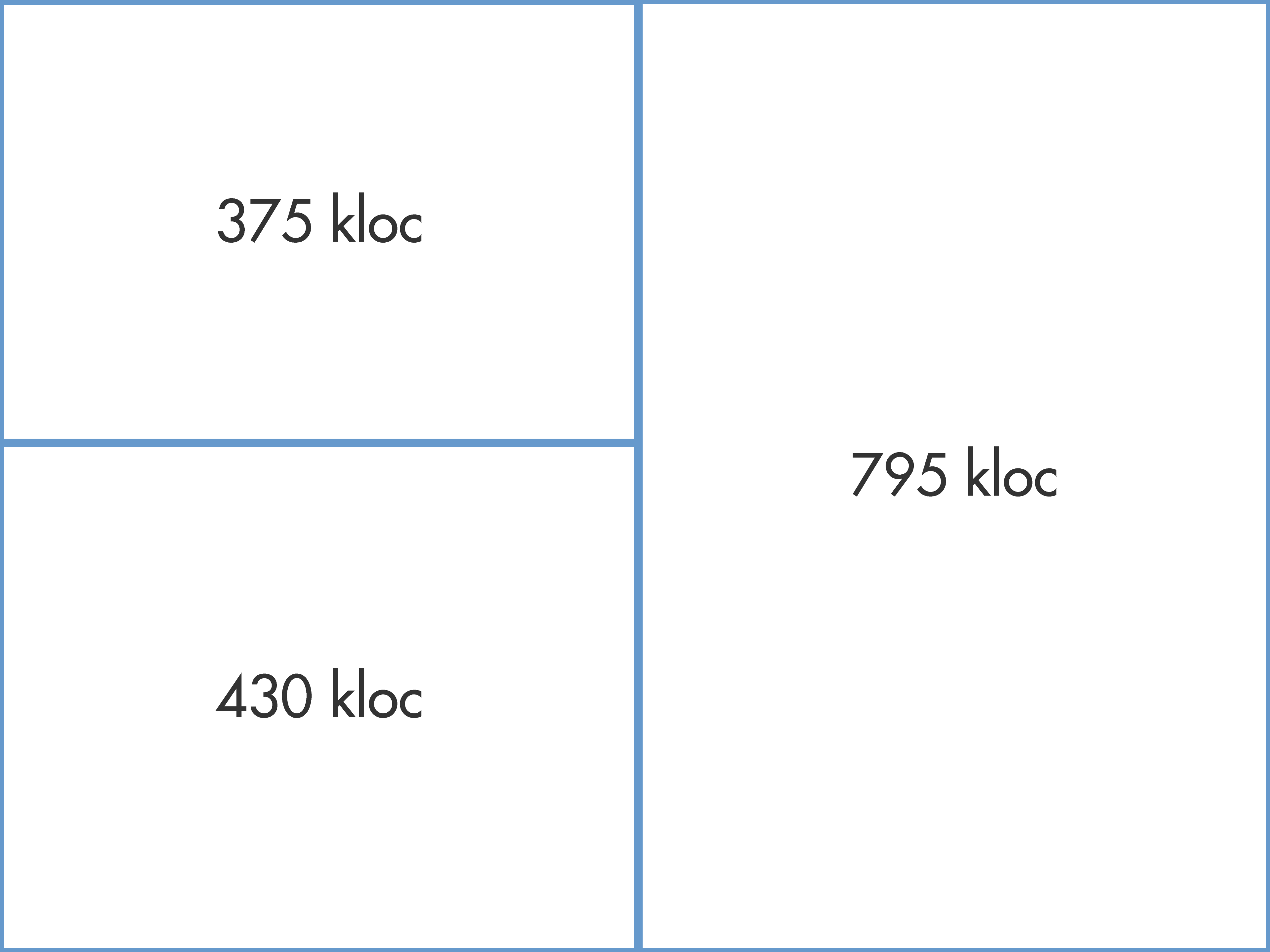








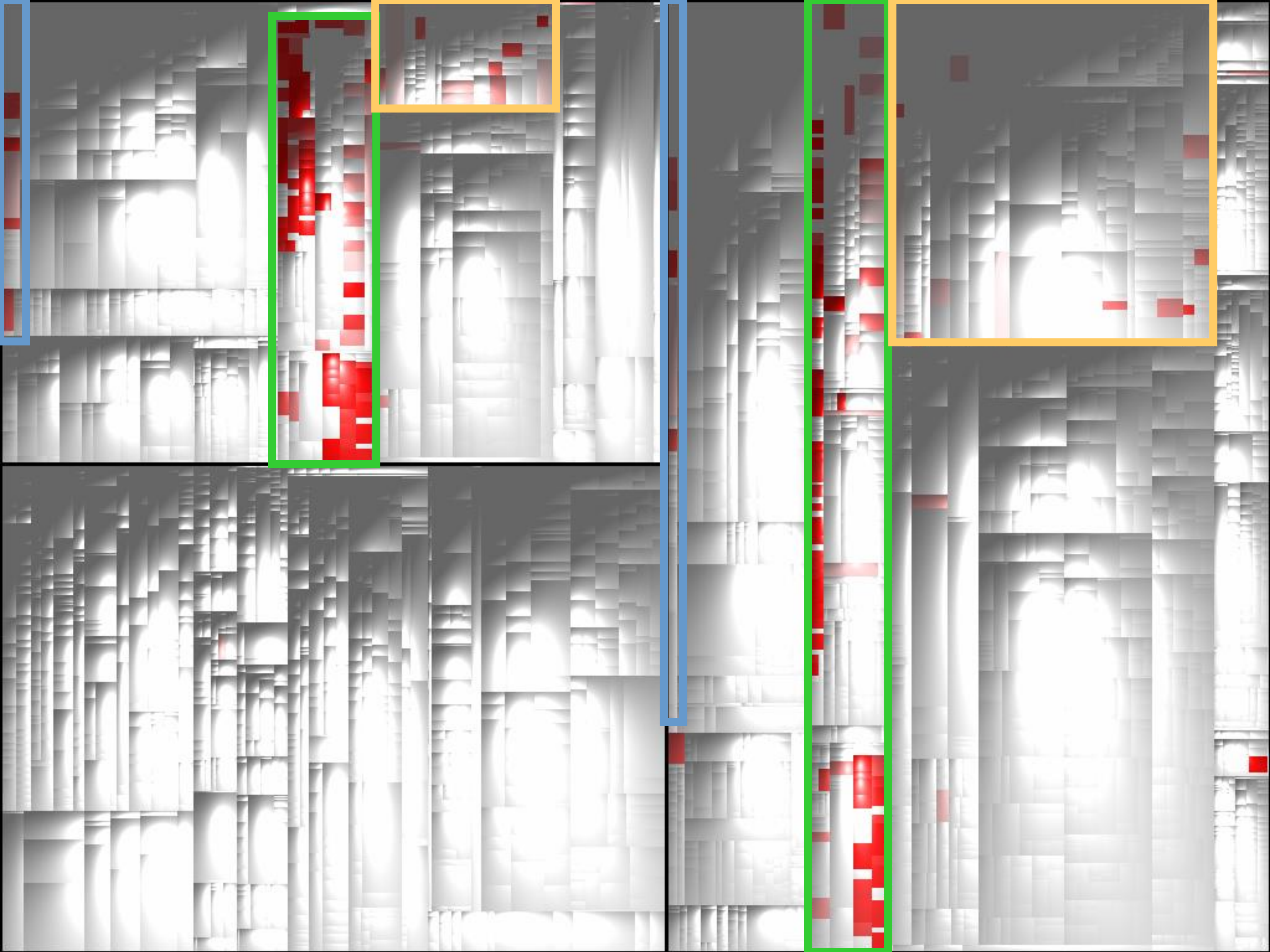
Mehrere Anwendungen in einer Abteilung



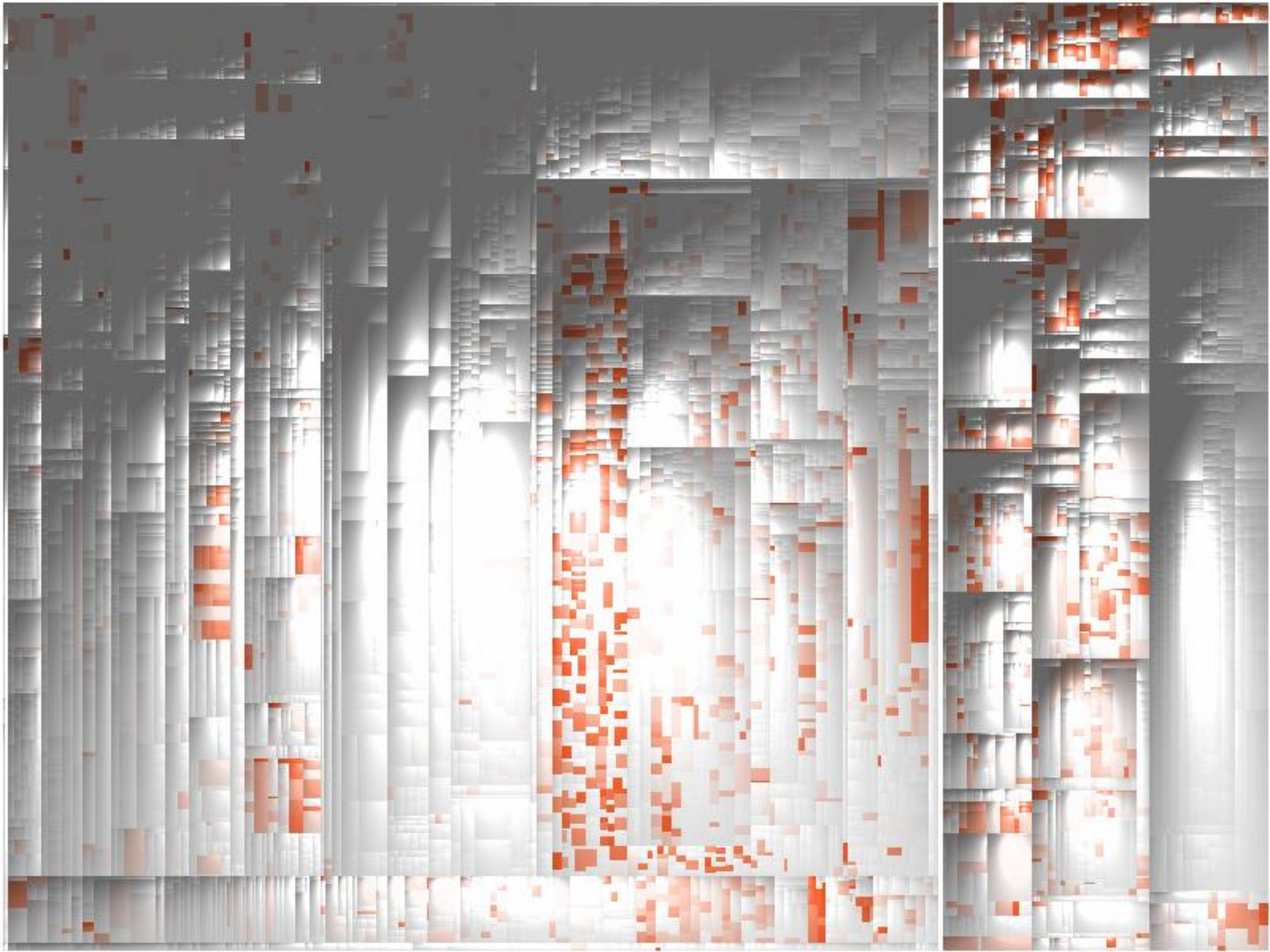
375 kloc

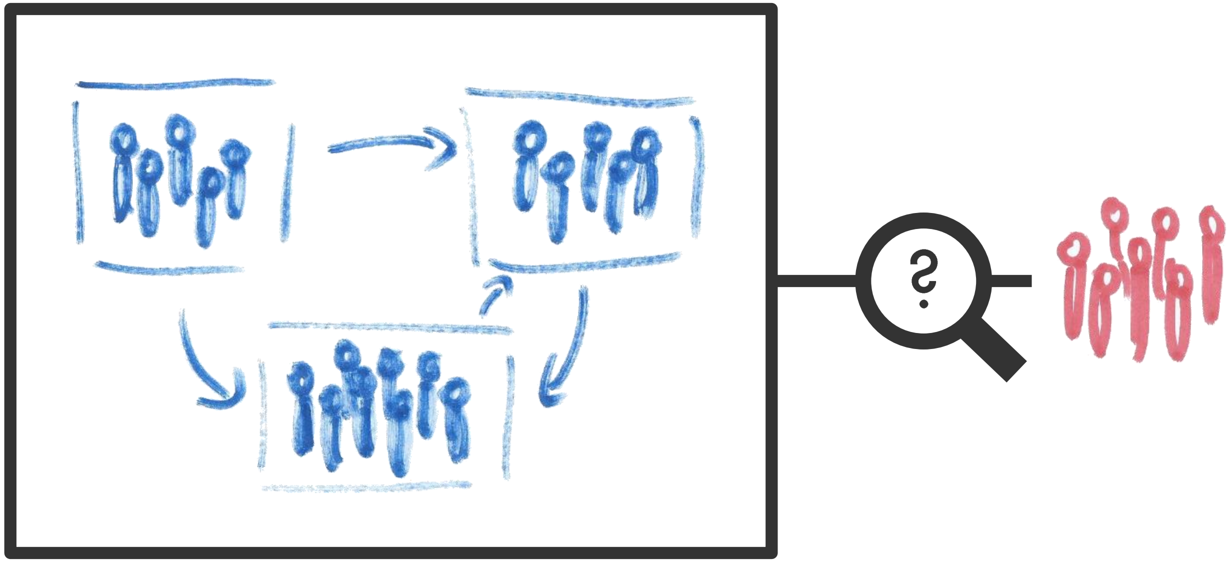
430 kloc

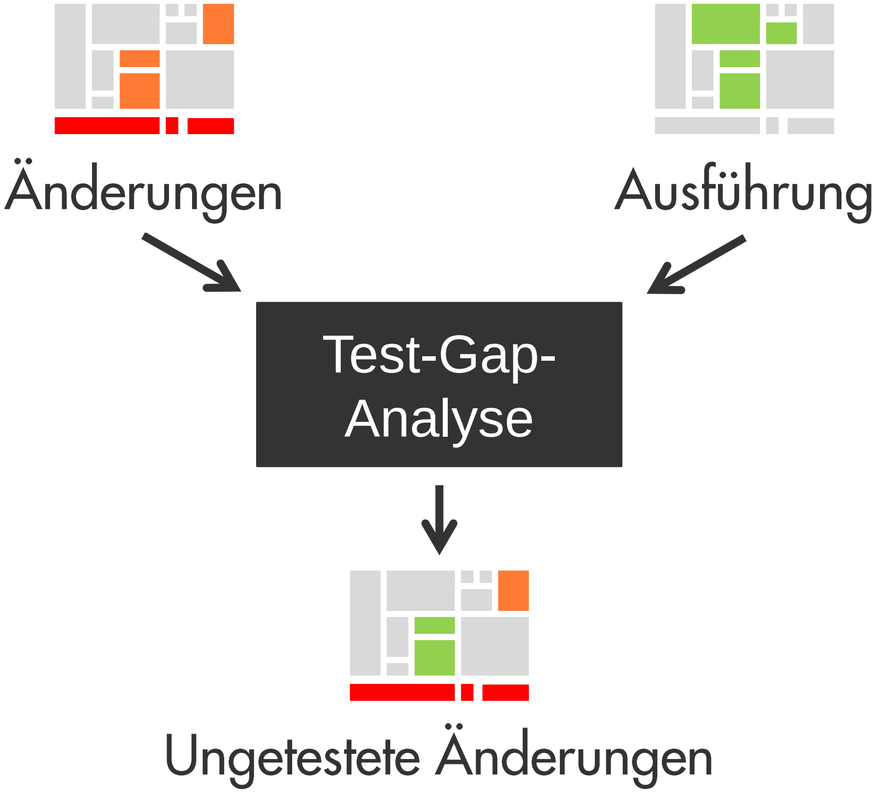
795 kloc



Architektur einer Produktlinie





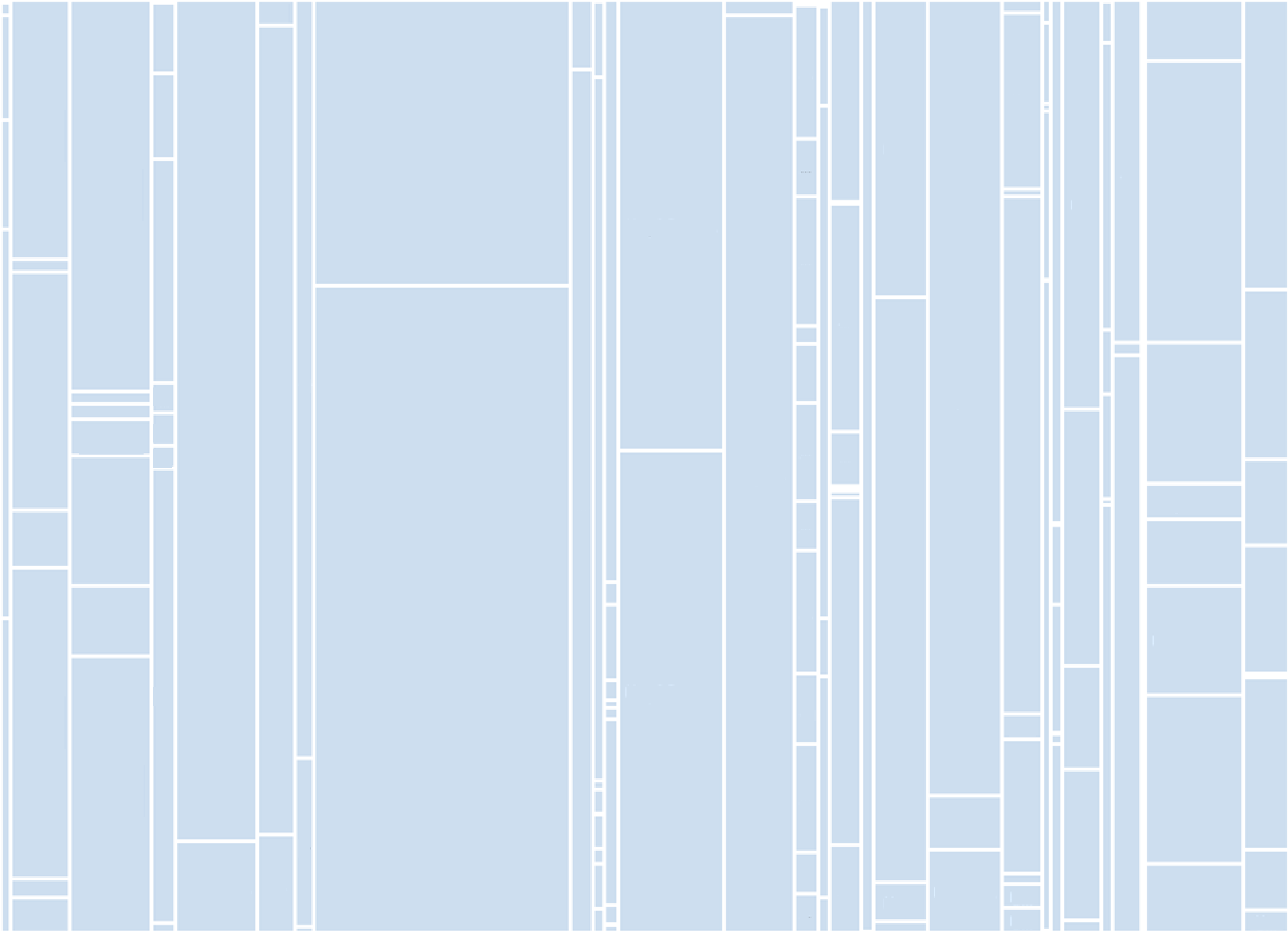


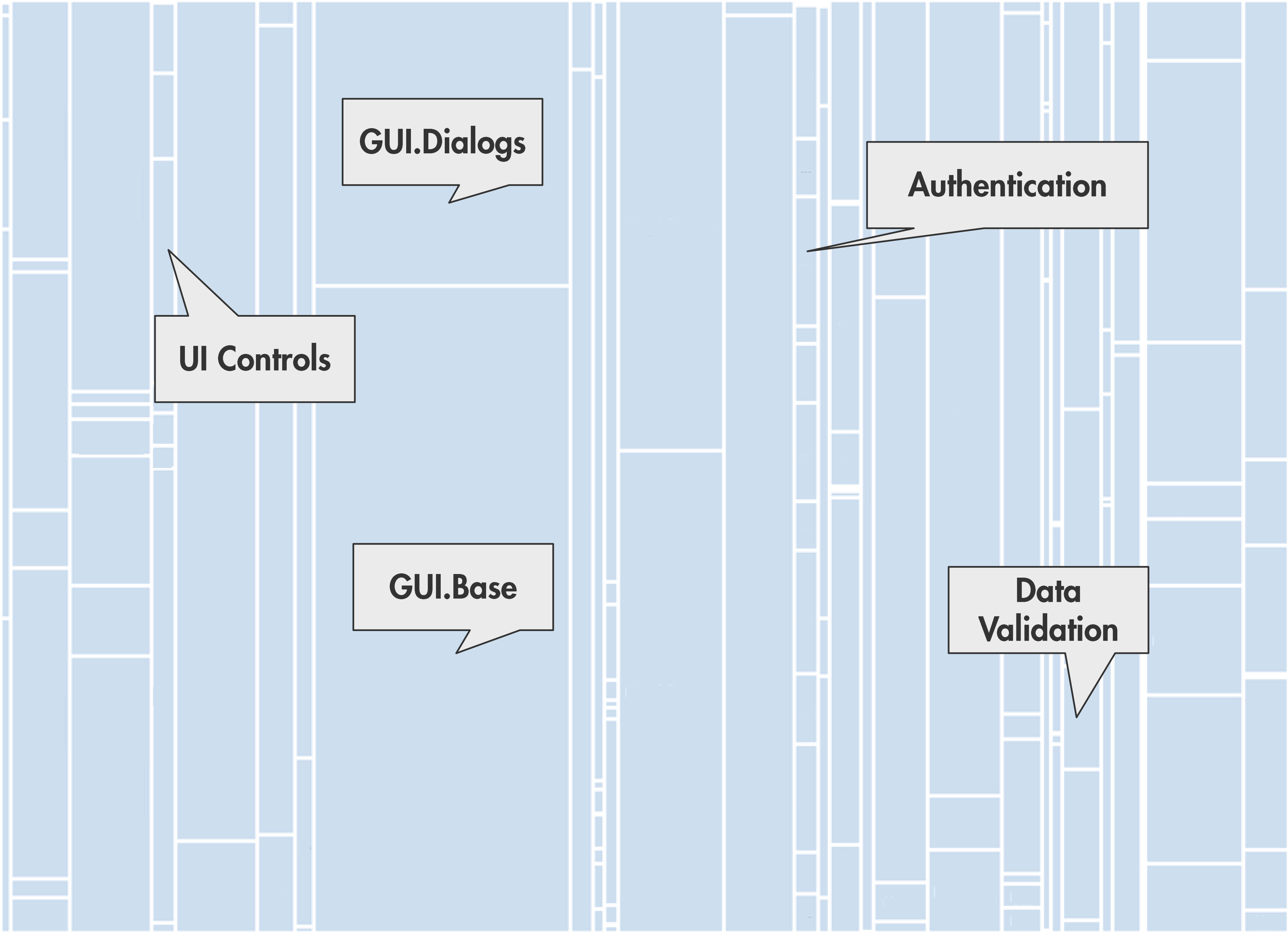
Änderungen

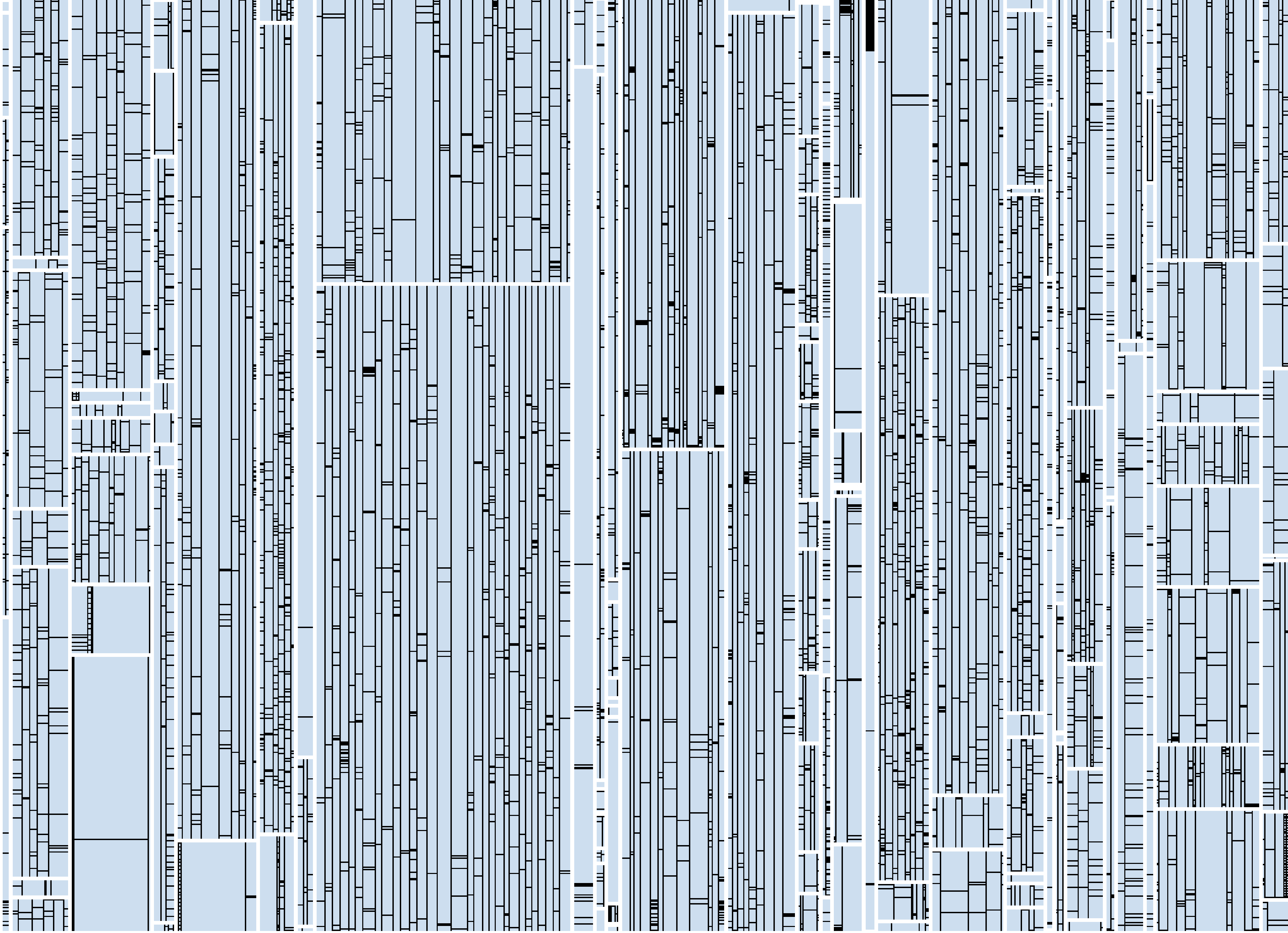
Ausführung

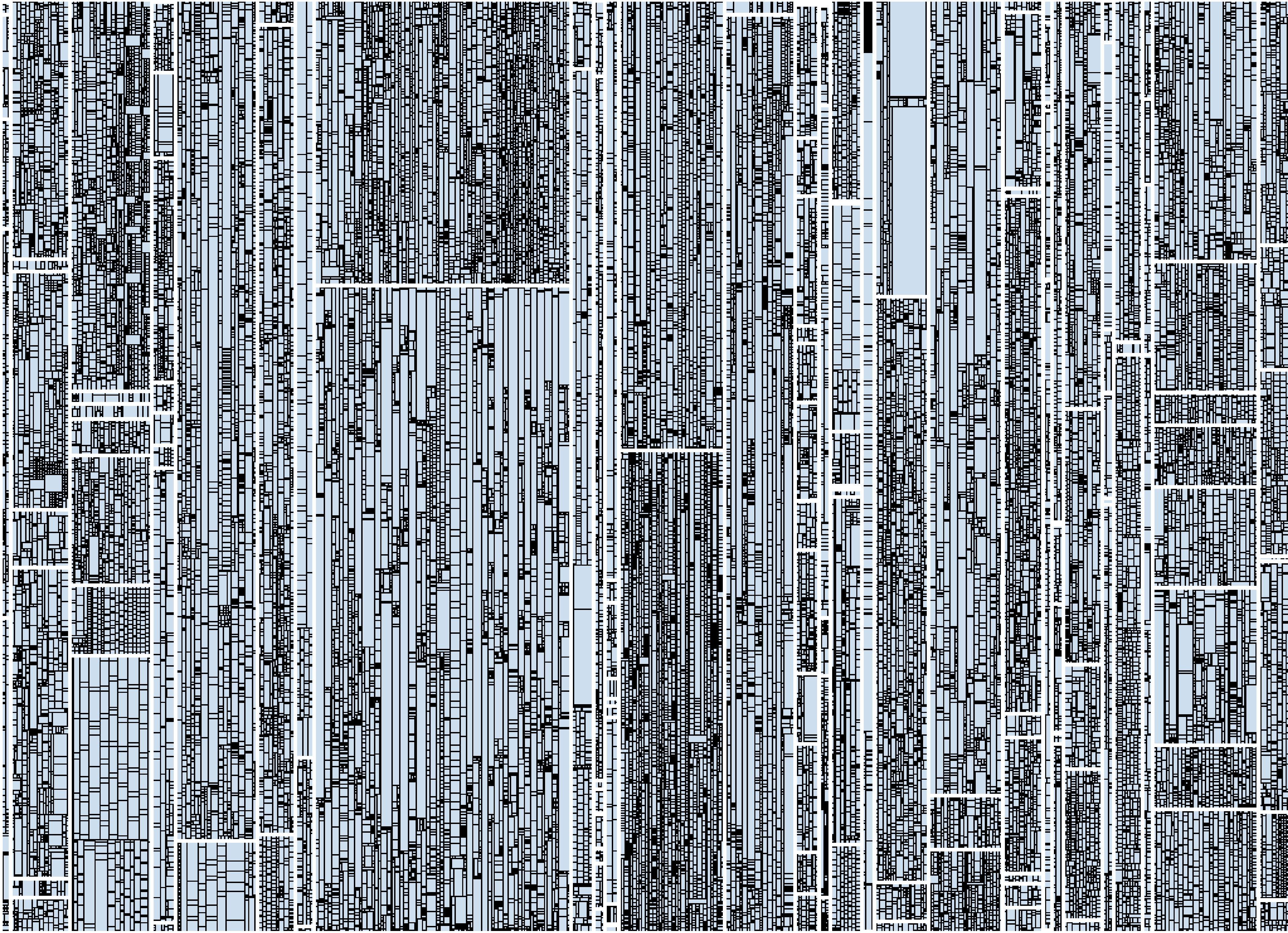
Test-Gap-Analyse

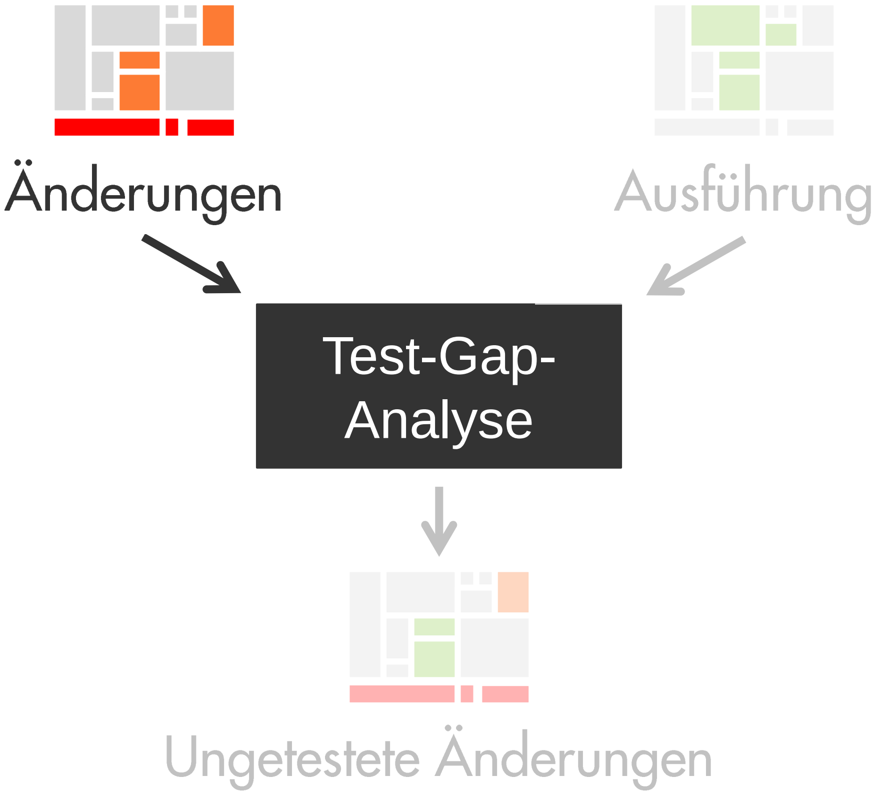
Ungetestete Änderungen




GUI.Dialogs
Authentication
UI Controls
GUI.Base
Data Validation





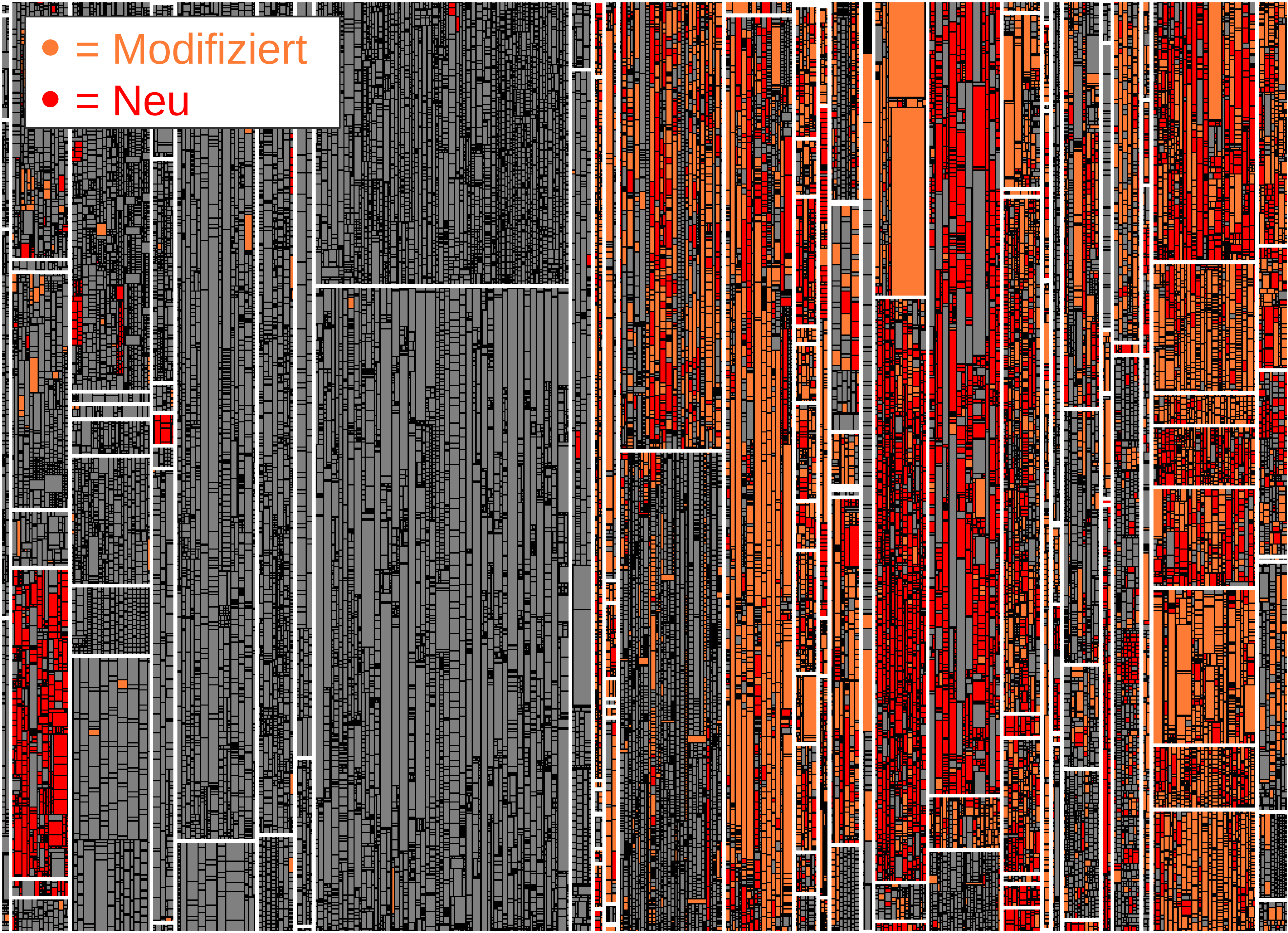


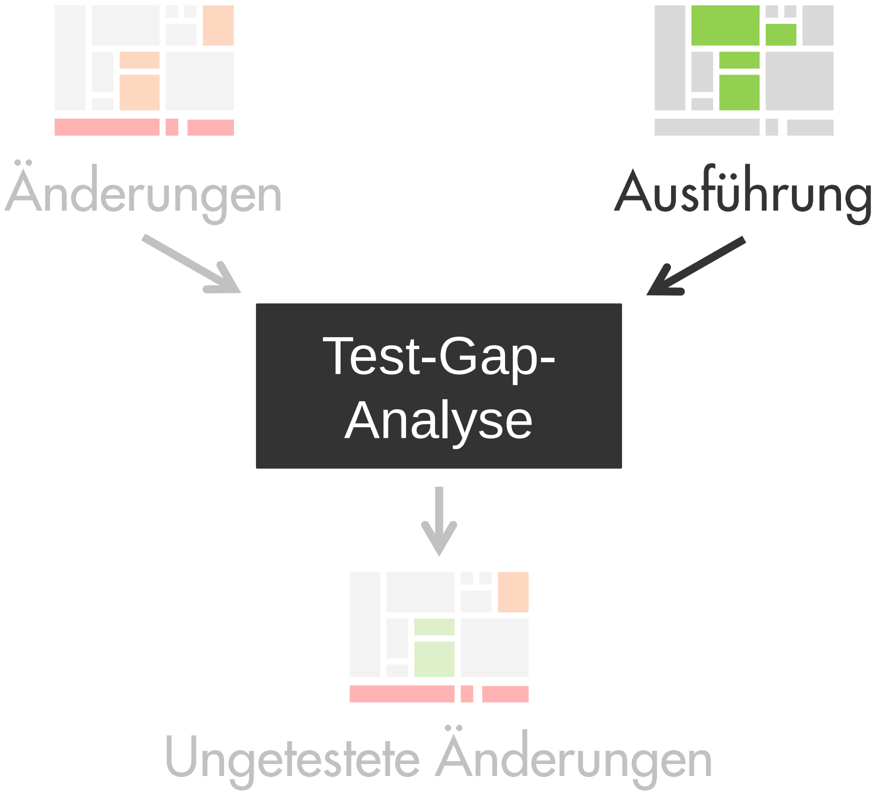
Änderungen

Ausführung

Test-Gap-Analyse

Ungetestete Änderungen


● = Modifiziert
● = Neu

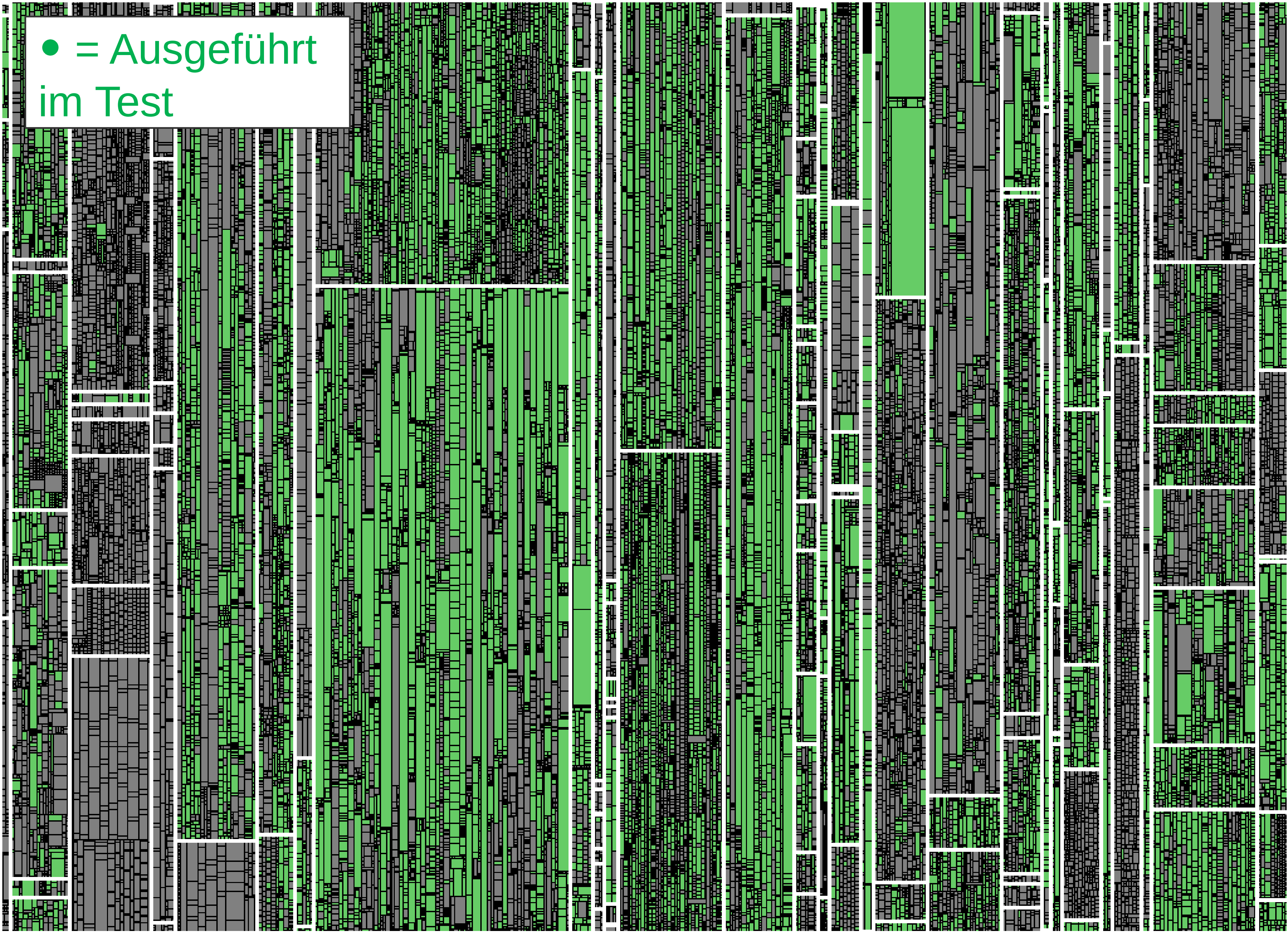


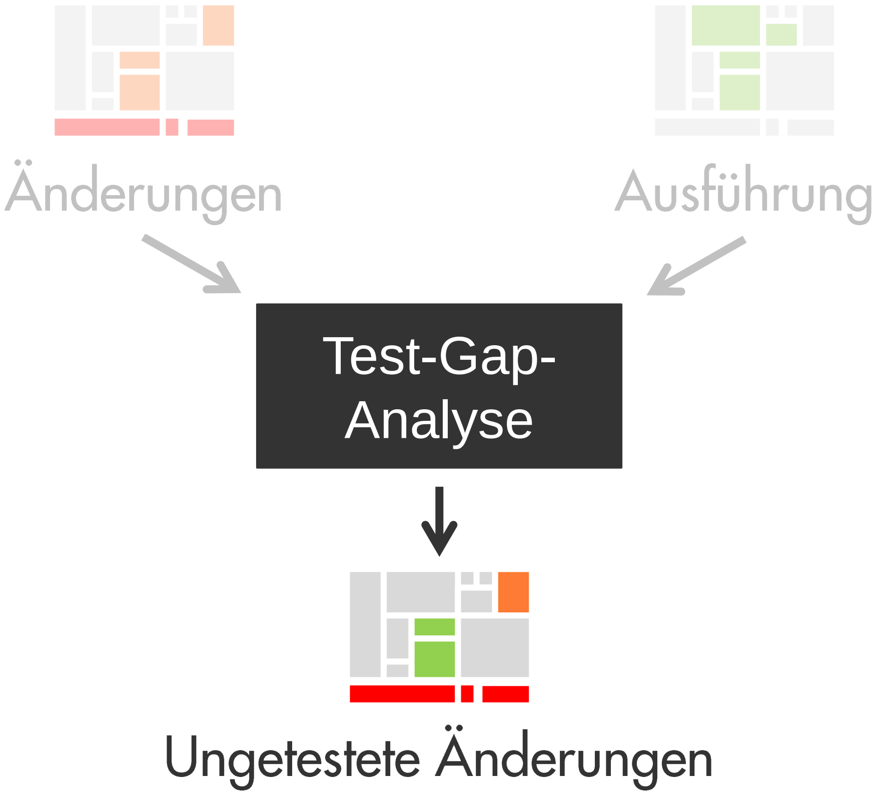
Änderungen

Ausführung

Test-Gap-Analyse

Ungetestete Änderungen


● = Ausgeführt
im Test

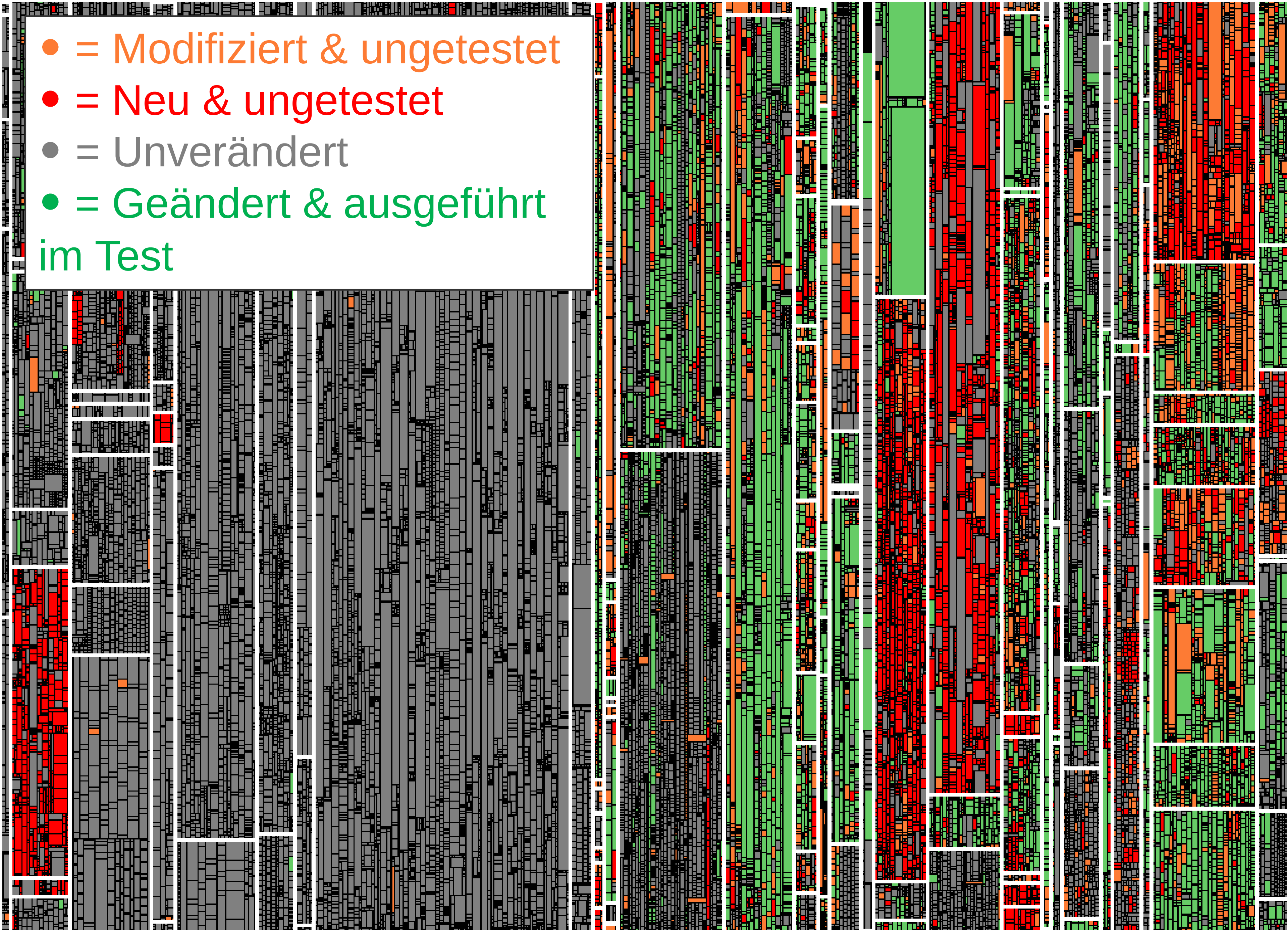


Änderungen

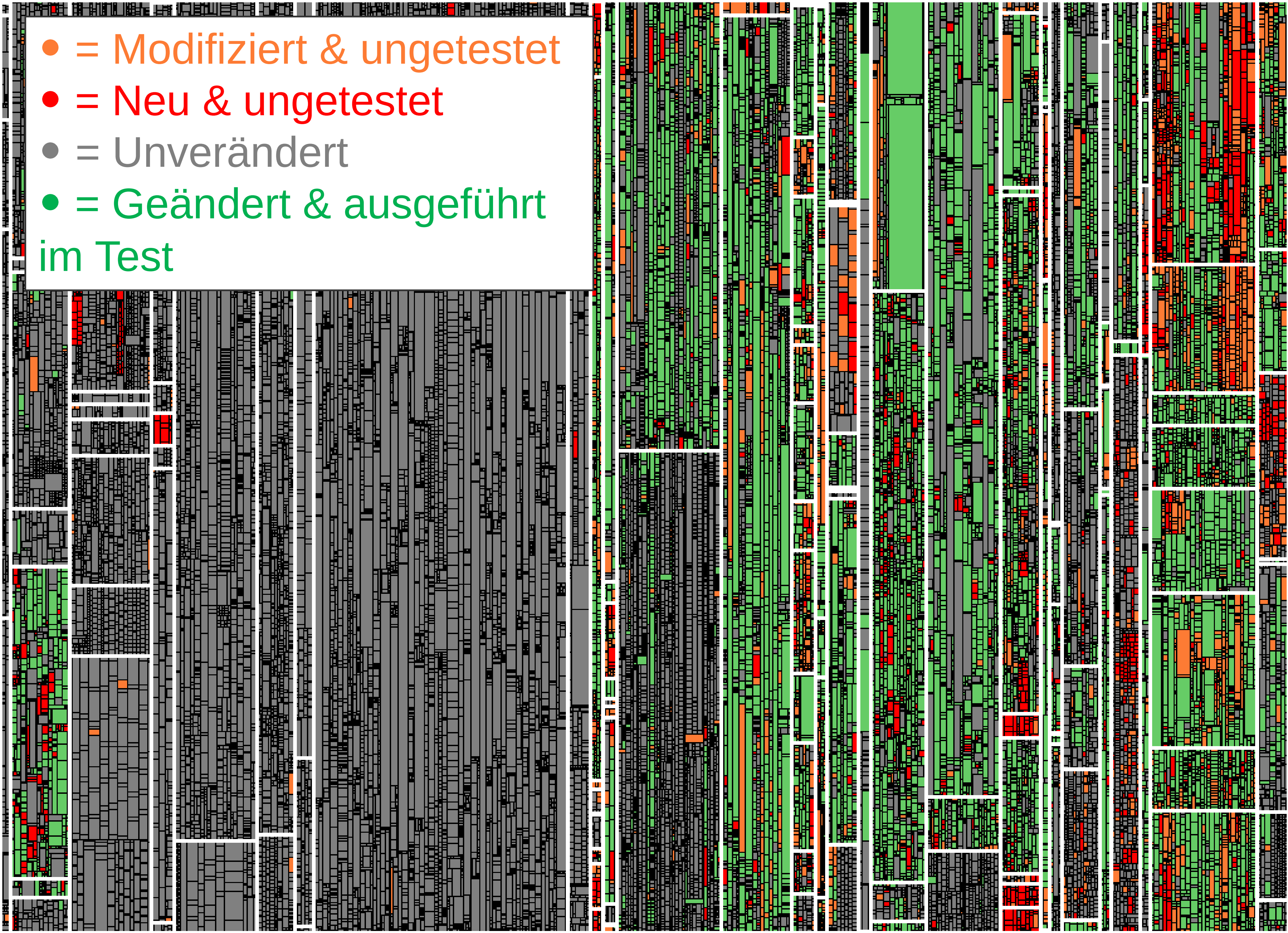
Ausführung

Test-Gap-Analyse

Ungetestete Änderungen

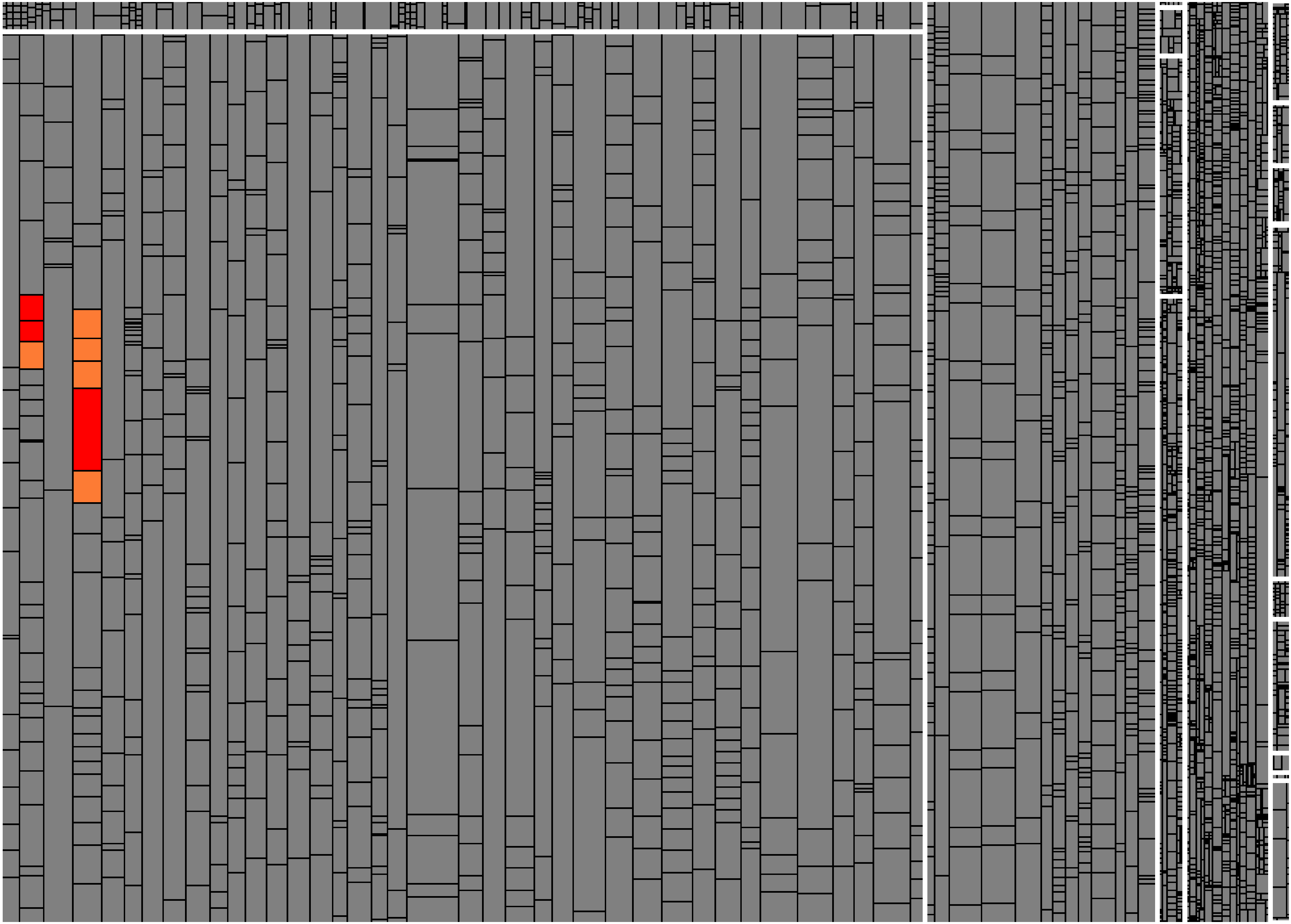


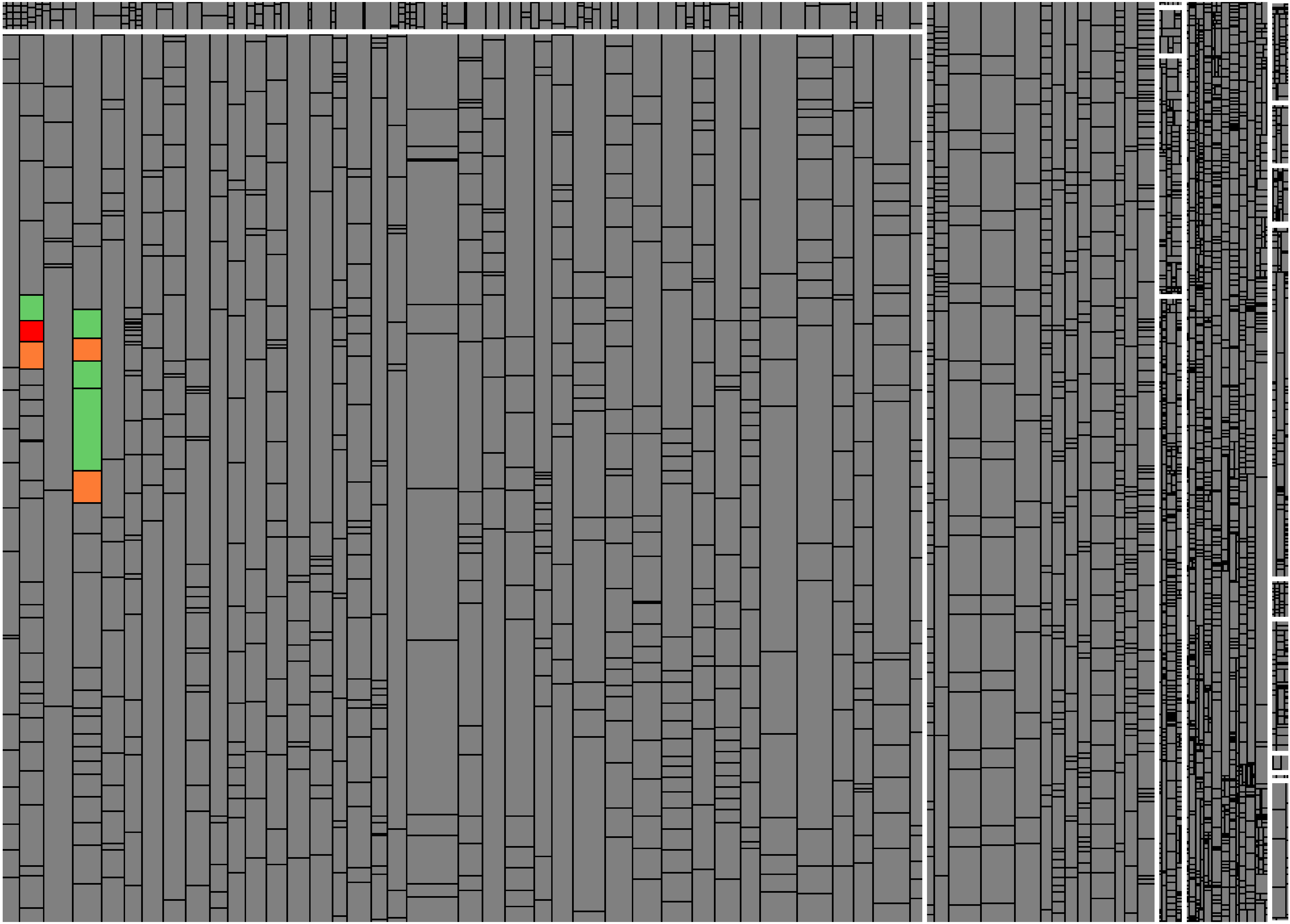
● = Modifiziert & ungetestet
● = Neu & ungetestet
● = Unverändert
● = Geändert & ausgeführt
im Test

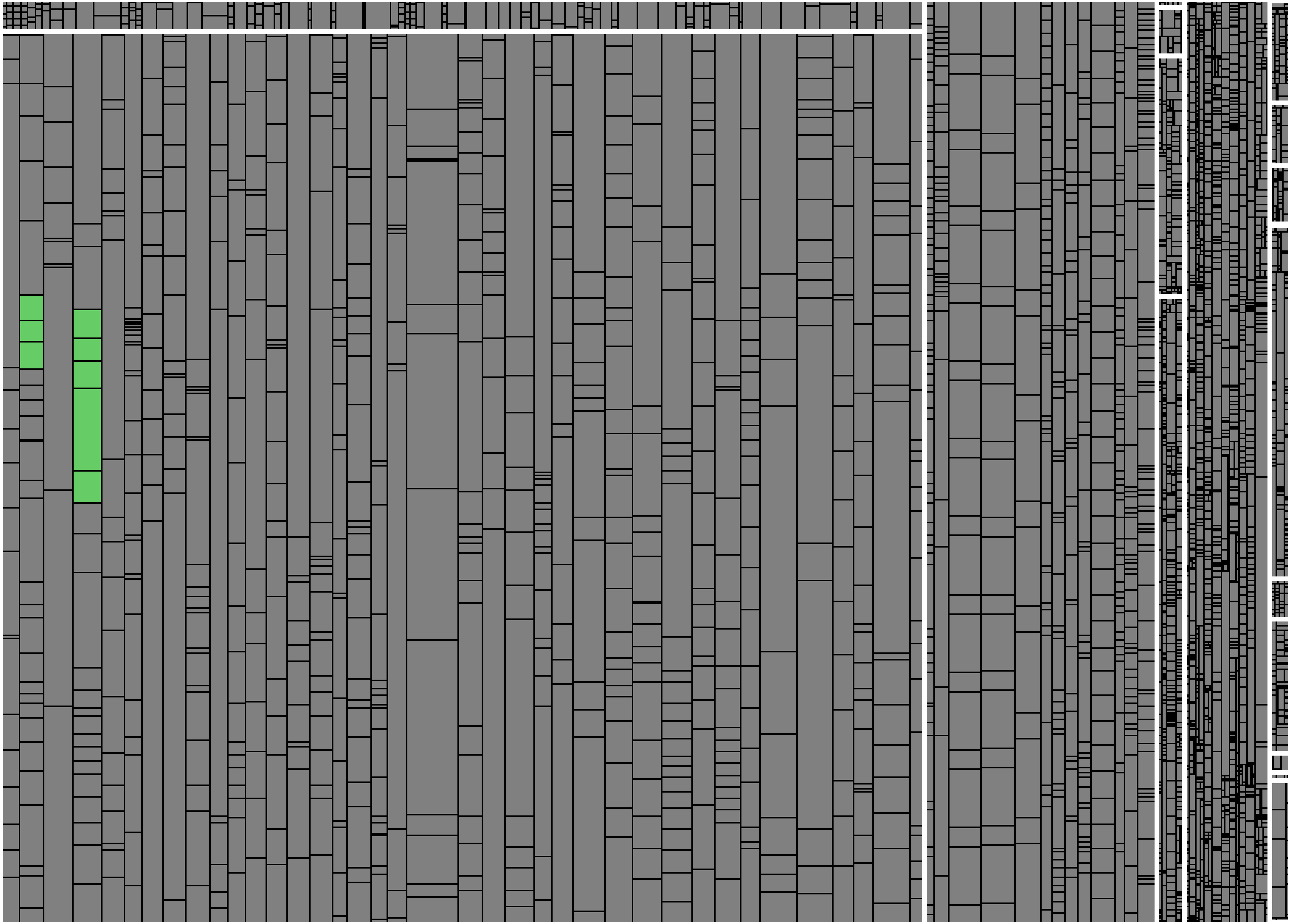

● = Modifiziert & ungetestet
● = Neu & ungetestet
● = Unverändert
● = Geändert & ausgeführt im Test

~~100% Change Coverage → 0 Fehler~~

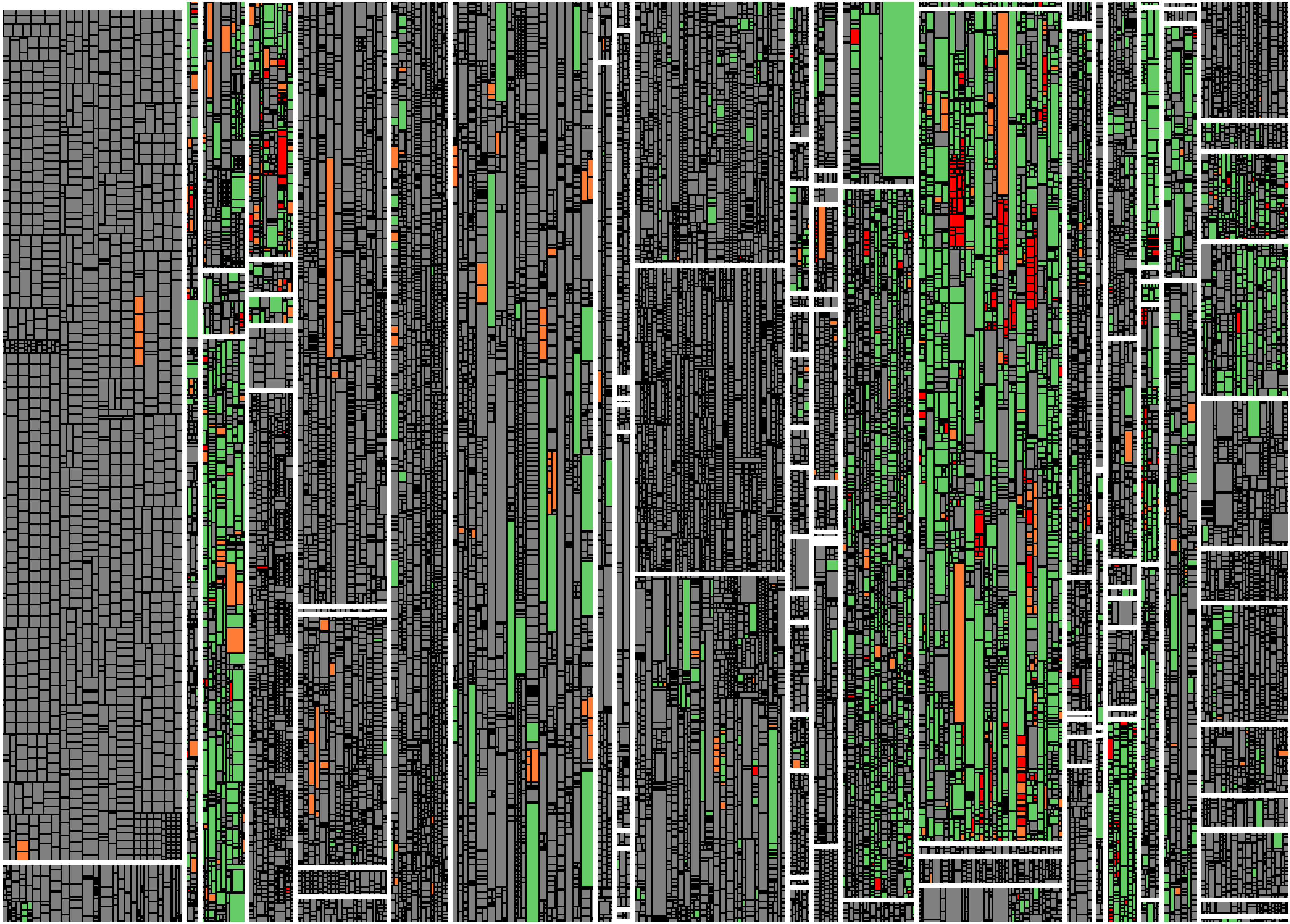
Fehlernachtest
Bei Hotfix

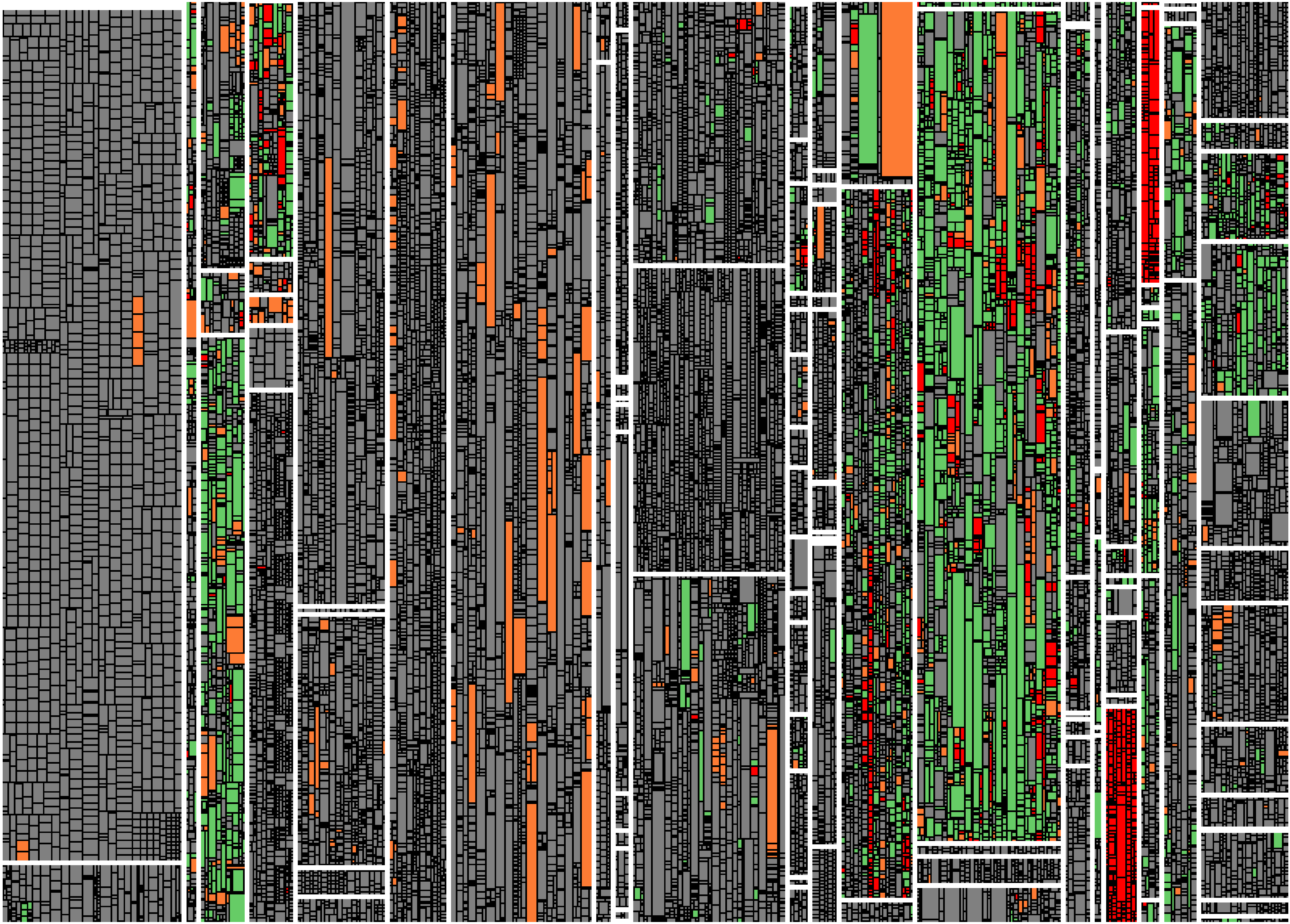


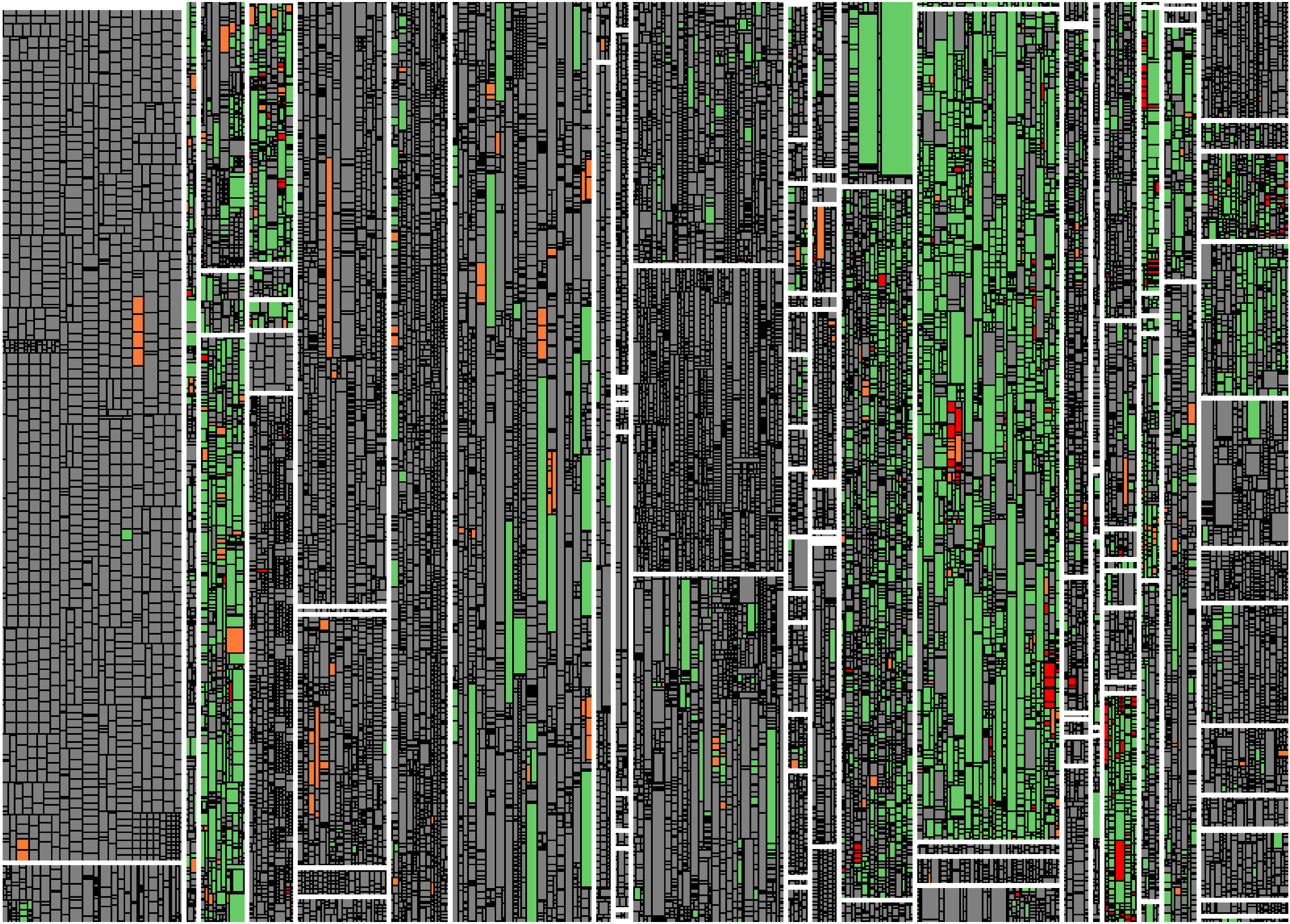


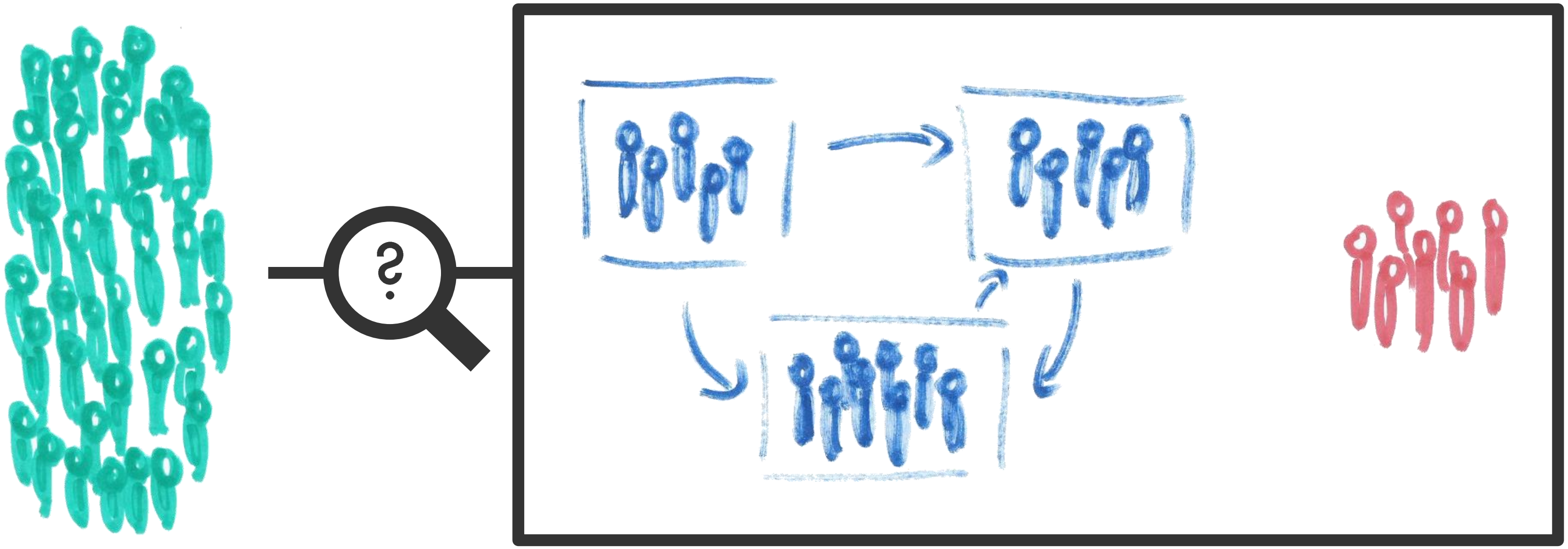


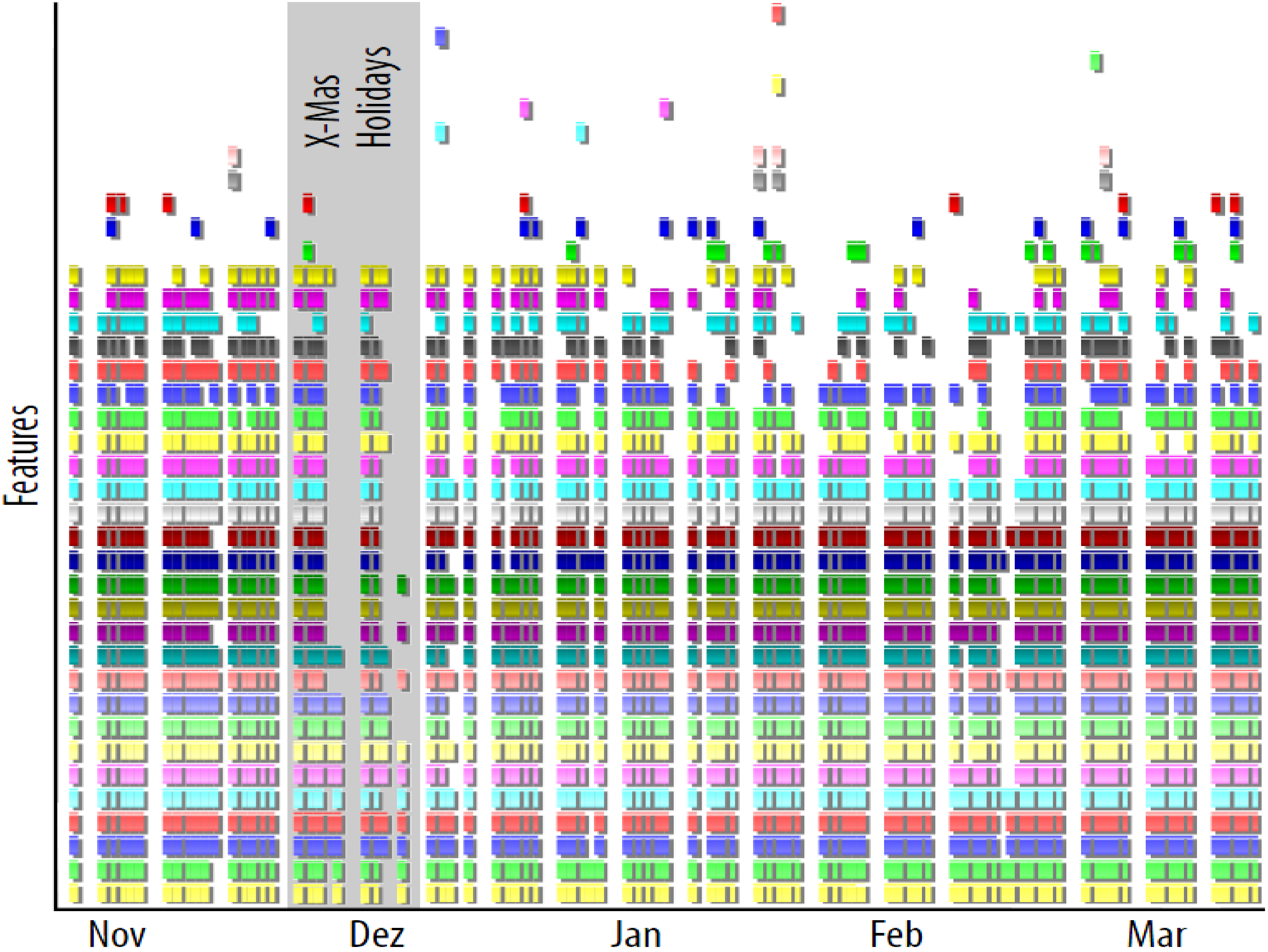
Release-Test





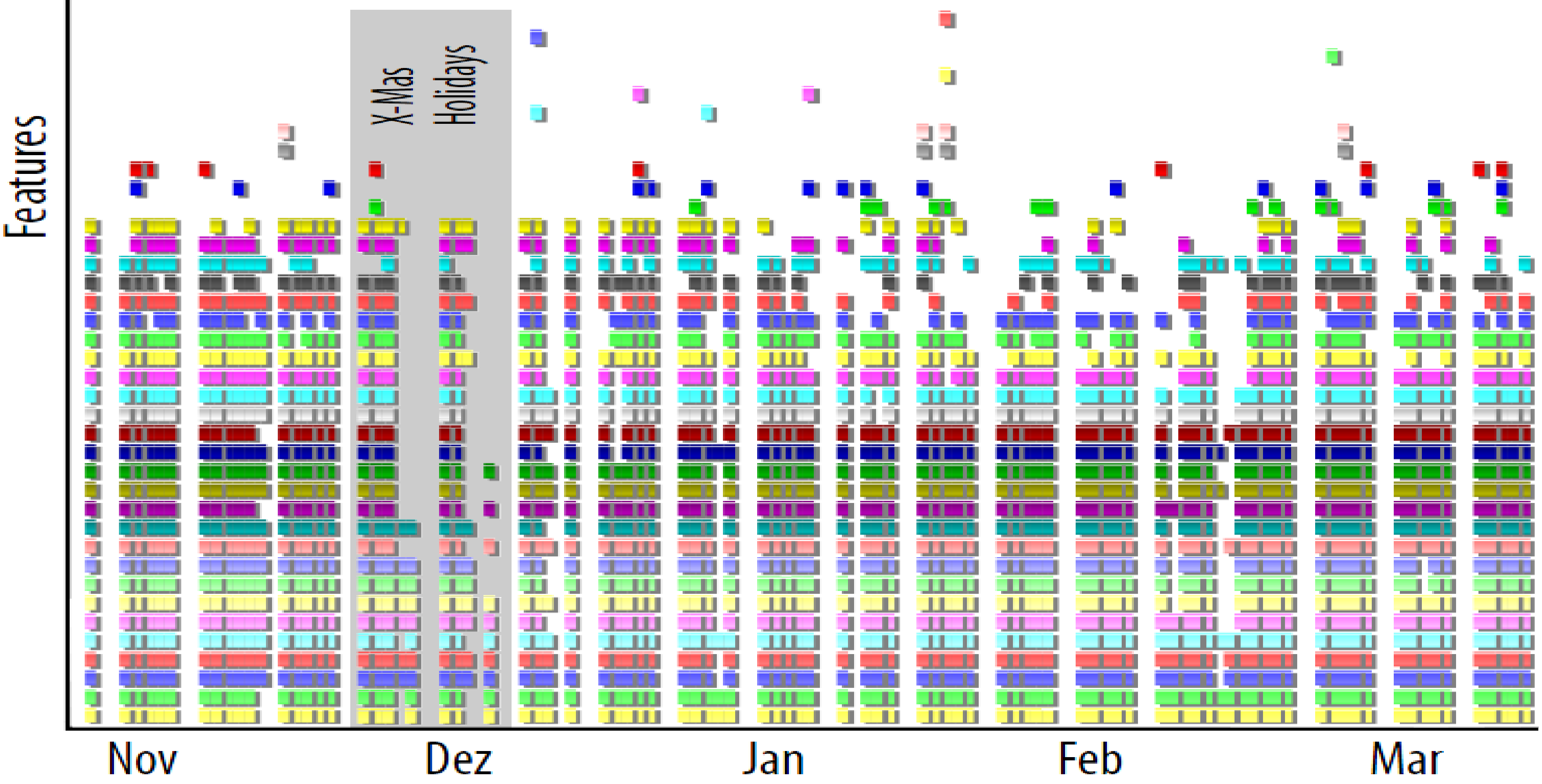


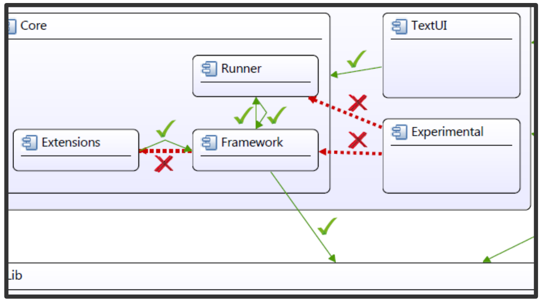


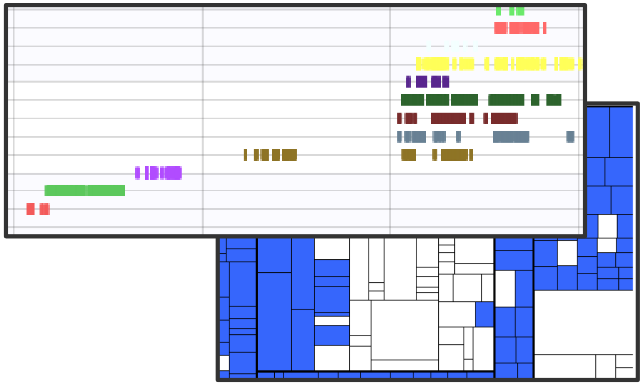

X-Mas
Holidays
Features
Nov
Dez
Jan
Feb
Mar

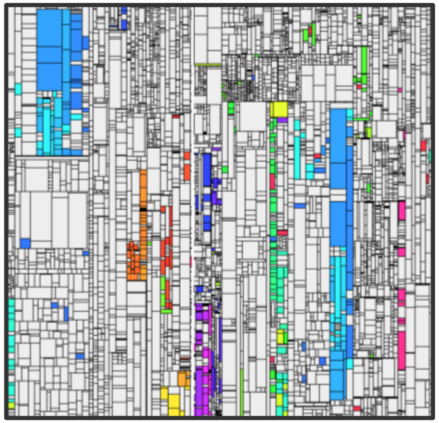
28% Features ungenutzt (15/53)

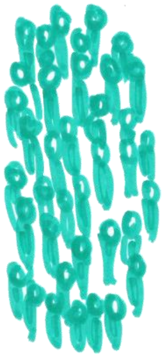
- 11 unerwartet

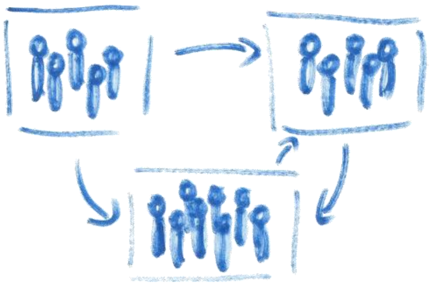


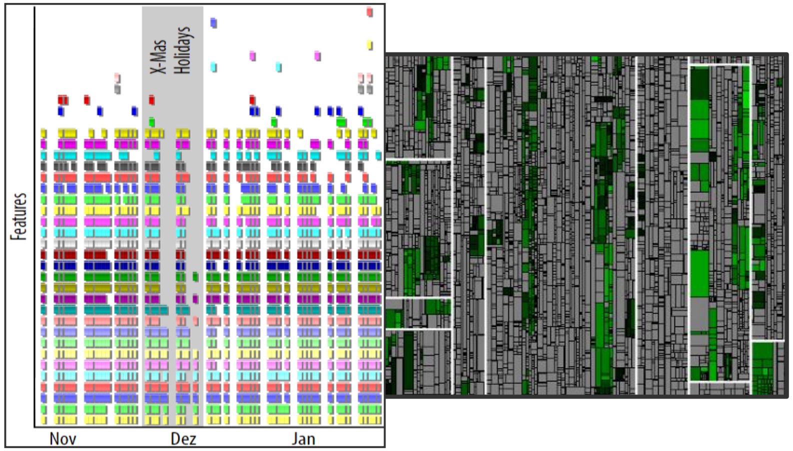

Core
TextUI
Runner
Extensions
Framework
Experimental
.ib

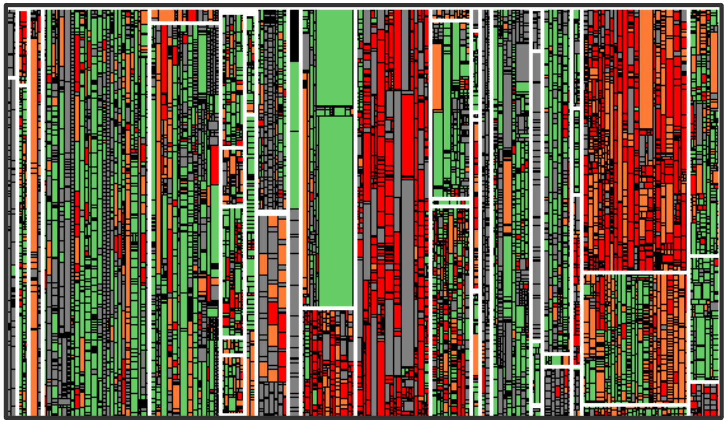








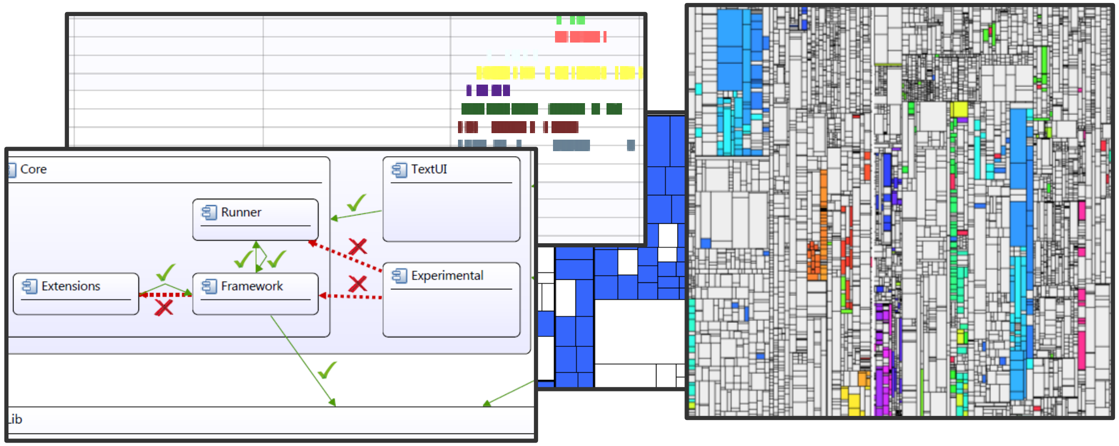

Features
X-Mas
Holidays
Nov
Dez
Jan



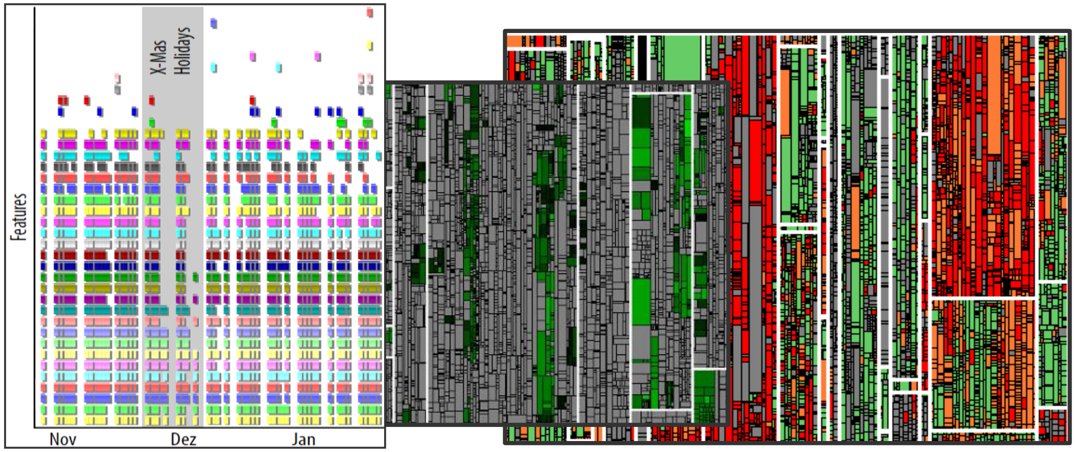
www.teamscale.io

Teamscale

Statische Analysen



Dynamische Analysen



Fazit

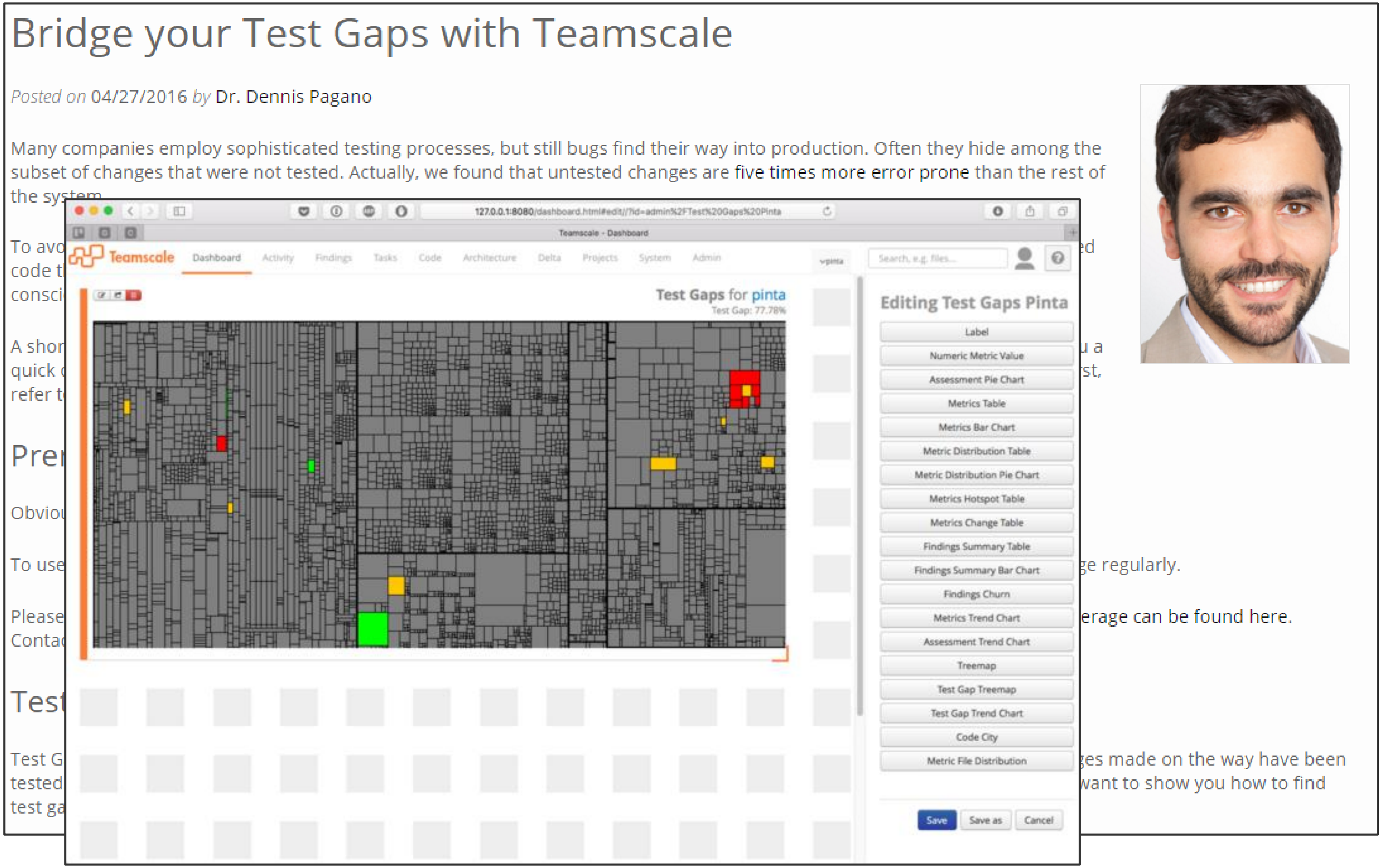
Es treten oft Probleme auf, wenn Architektur und Organisation nicht gut zusammenpassen.

Es gibt nützliche dynamische und statische Analysen, die Symptome ermitteln können, damit Ursachen abgestellt werden können.

Sie sind zu einem großen Grad automatisiert, erfordern aber Erfahrung in Konfiguration und Interpretation.

Wir unterstützen gerne.

https://www.cqse.eu/en/blog



Bridge your Test Gaps with Teamscale

Posted on 04/27/2016 *by* Dr. Dennis Pagano

Many companies employ sophisticated testing processes, but still bugs find their way into production. Often they hide among the subset of changes that were not tested. Actually, we found that untested changes are five times more error prone than the rest of the system.

To avo ... ed
code t
consci

A shor ... u a
quick ... st,
refer t

Pre

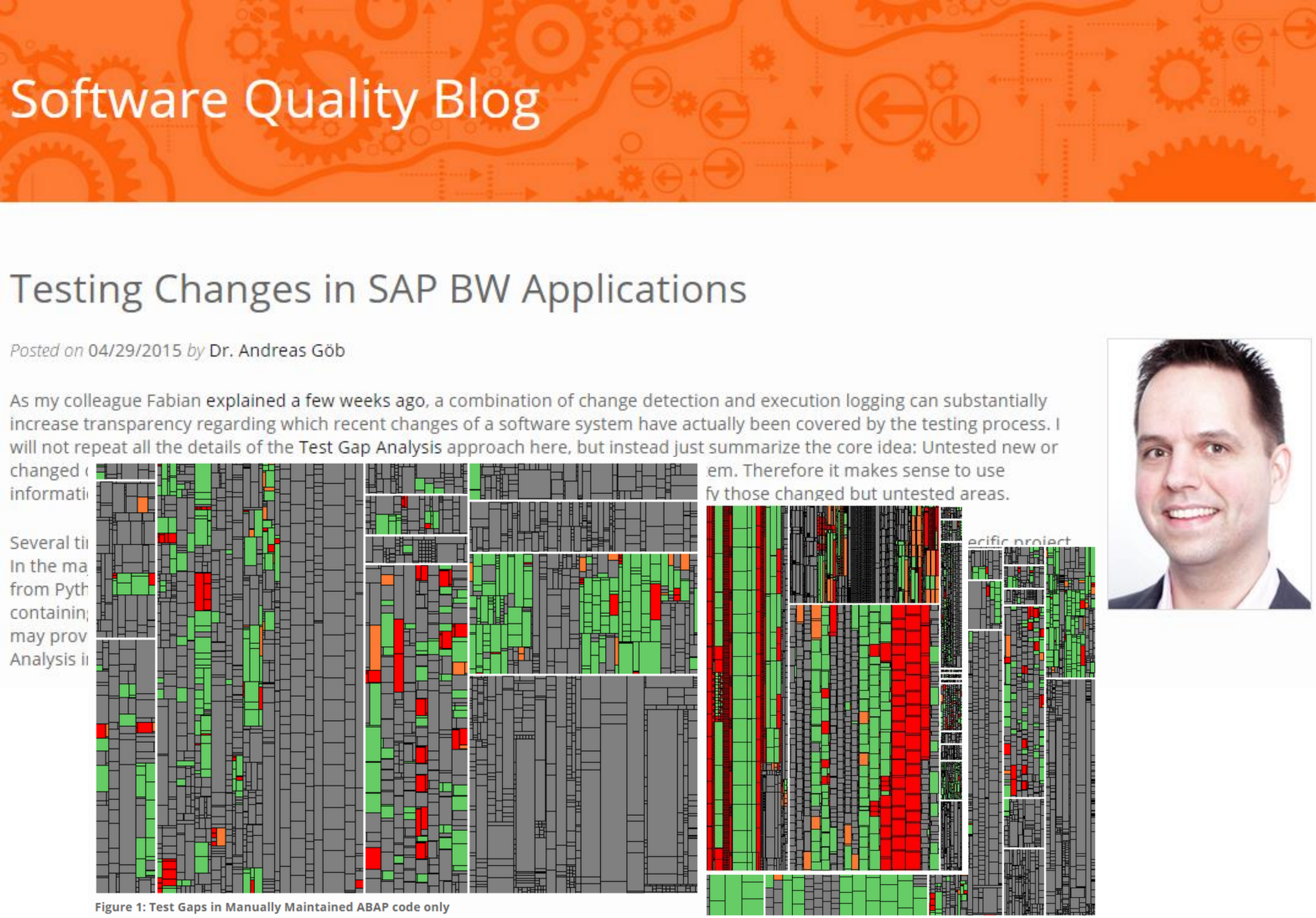
Obviou

To use ... ge regularly.

Please ... erage can be found here.
Conta

Test

Test G ... ges made on the way have been
tested ... want to show you how to find
test ga



Software Quality Blog

Testing Changes in SAP BW Applications

Posted on 04/29/2015 by Dr. Andreas Göb

As my colleague Fabian explained a few weeks ago, a combination of change detection and execution logging can substantially increase transparency regarding which recent changes of a software system have actually been covered by the testing process. I will not repeat all the details of the Test Gap Analysis approach here, but instead just summarize the core idea: Untested new or changed ‹ em. Therefore it makes sense to use informati fy those changed but untested areas.

Several ti ecific project

In the ma

from Pyth

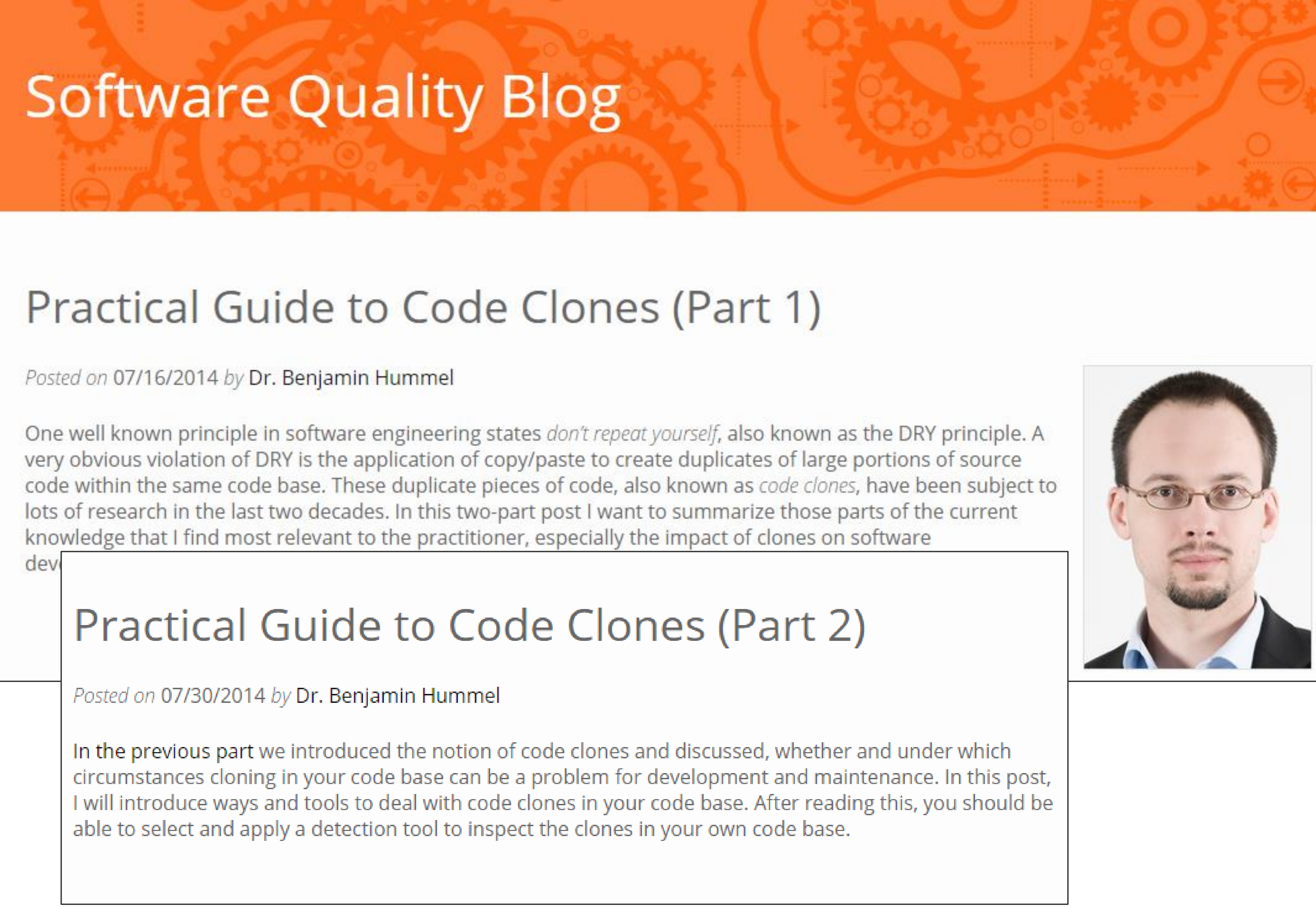
containin

may prov

Analysis i

Figure 1: Test Gaps in Manually Maintained ABAP code only

https://www.cqse.eu/en/blog/testing-changes-in-sap-bw/



Software Quality Blog

Practical Guide to Code Clones (Part 1)

Posted on 07/16/2014 *by* Dr. Benjamin Hummel

One well known principle in software engineering states *don't repeat yourself*, also known as the DRY principle. A very obvious violation of DRY is the application of copy/paste to create duplicates of large portions of source code within the same code base. These duplicate pieces of code, also known as *code clones*, have been subject to lots of research in the last two decades. In this two-part post I want to summarize those parts of the current knowledge that I find most relevant to the practitioner, especially the impact of clones on software dev

Practical Guide to Code Clones (Part 2)

Posted on 07/30/2014 *by* Dr. Benjamin Hummel

In the previous part we introduced the notion of code clones and discussed, whether and under which circumstances cloning in your code base can be a problem for development and maintenance. In this post, I will introduce ways and tools to deal with code clones in your code base. After reading this, you should be able to select and apply a detection tool to inspect the clones in your own code base.

https://www.cqse.eu/en/blog/practical-guide-to-code-clones-part1/

Kontakt

Dr. Elmar Jürgens · juergens@cqse.eu · +49 179 675 3863

@ElmarJuergens
www.cqse.eu/de/ressourcen/blog

CQSE GmbH
Lichtenbergstraße 8
85748 Garching bei München

CQSE
Continuous Quality in Software Engineering