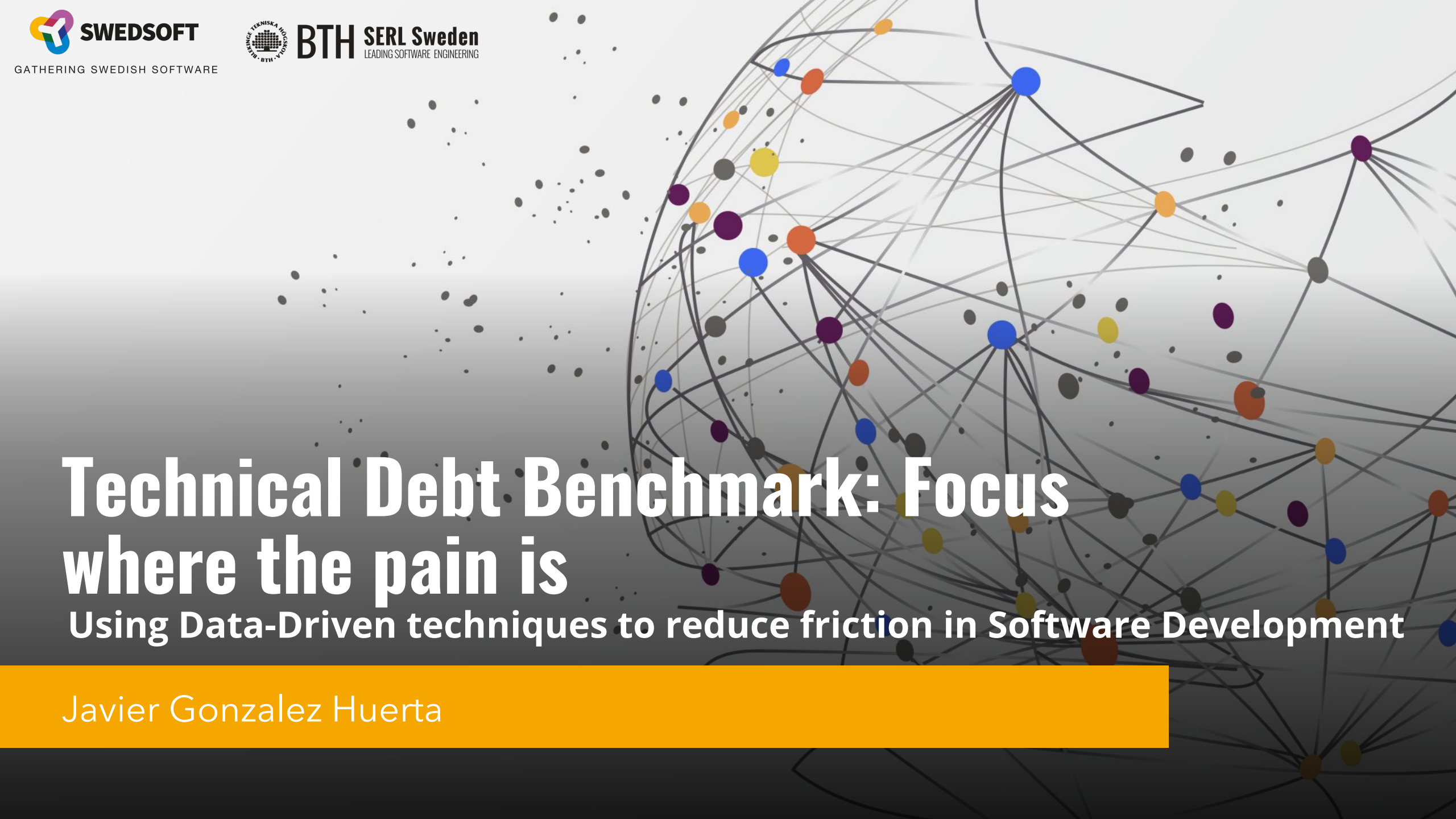
SWEDSOFT
GATHERING SWEDISH SOFTWARE

BTH SERL Sweden
LEADING SOFTWARE ENGINEERING



Technical Debt Benchmark: Focus where the pain is

Using Data-Driven techniques to reduce friction in Software Development

Javier Gonzalez Huerta

Who am I?

Javier Gonzalez-Huerta

Associate Professor in Software Engineering at Blekinge Institute of Technology
- Program Manager: Civilingenjör i Mjukvarutveckling

- >10 years in industry

- PhD in Software Development (Software Engineering)
- Now I'm a researcher / practitioner (not necessarily in that order)
- Collaborate (doing research) with several companies in Sweden:

Spotify®

ERICSSON

FORTNOX
Småföretagens bästa vän

VOLVO

But this is a team work...

Dr. Ehsan Zabardast

Senior Researcher at Blekinge Institute of Technology
Management Consultant at Reinsight

Dr. Binish Tanveer

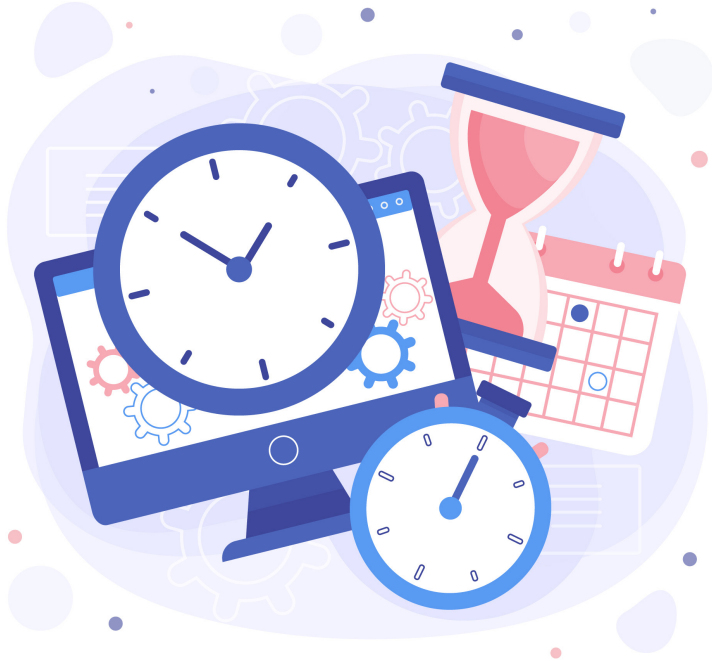
Associate Senior Lecturer at Blekinge Institute of Technology

Bhuwan Paudel

Ph.D. Candidate at Blekinge Institute of Technology

Technical Debt?
Focus where the pain is – Technical Debt Benchmark

Technical Debt (TD)



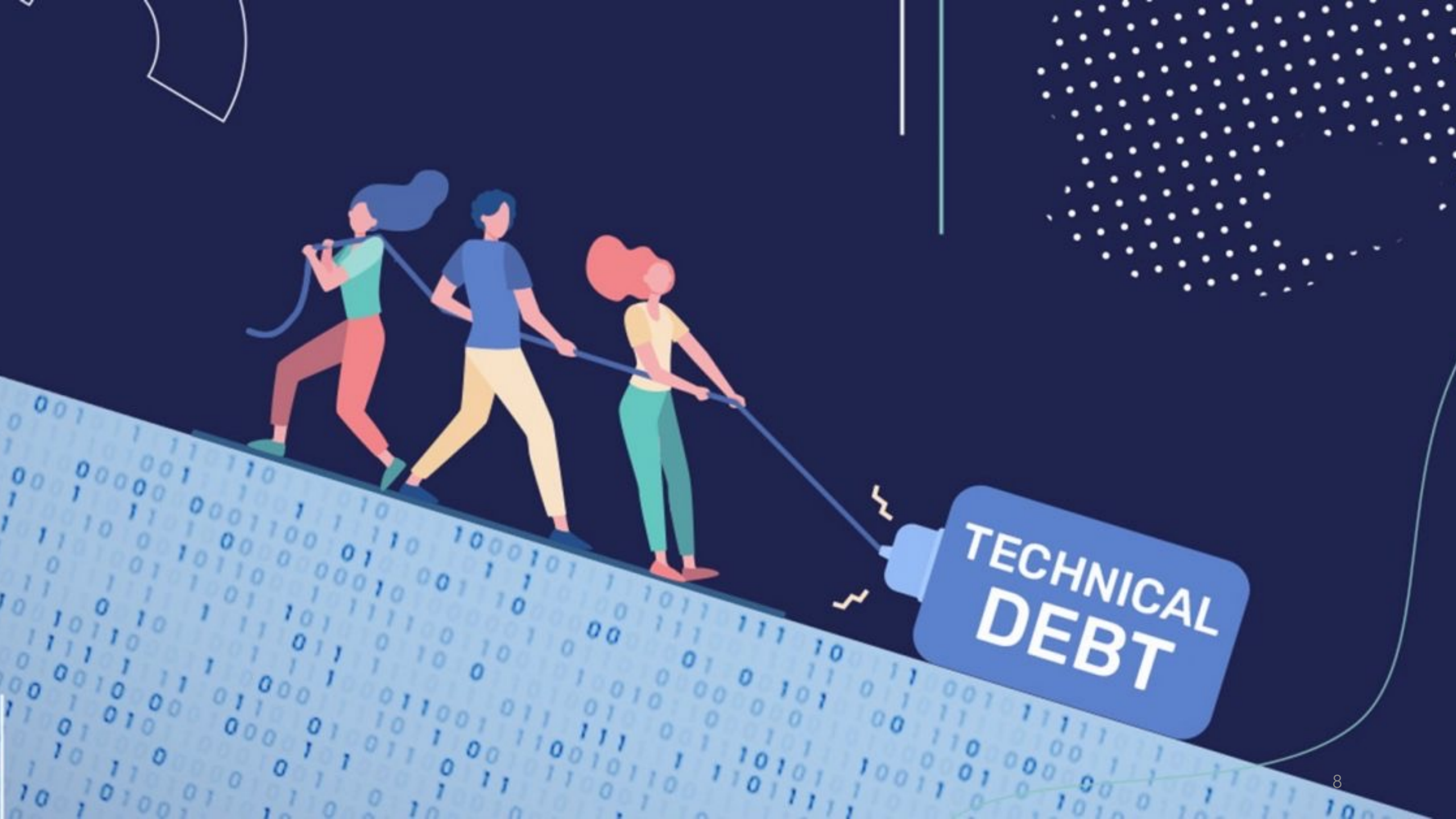
- Shortcomings of the TD metaphor?
  - What does it mean to say that we have accumulated 10 years of TD?
  - Everything counts equally?

Technical Debt (TD)

- Shortcomings of the TD metaphor
  - TD is more tailored for source code related artefacts
  - TD does not consider the TD propagation effects for example between Code and Tests
  - We don’t look at it from a system perspective
  - We tend to look at it in isolation

Limitations of the term TD

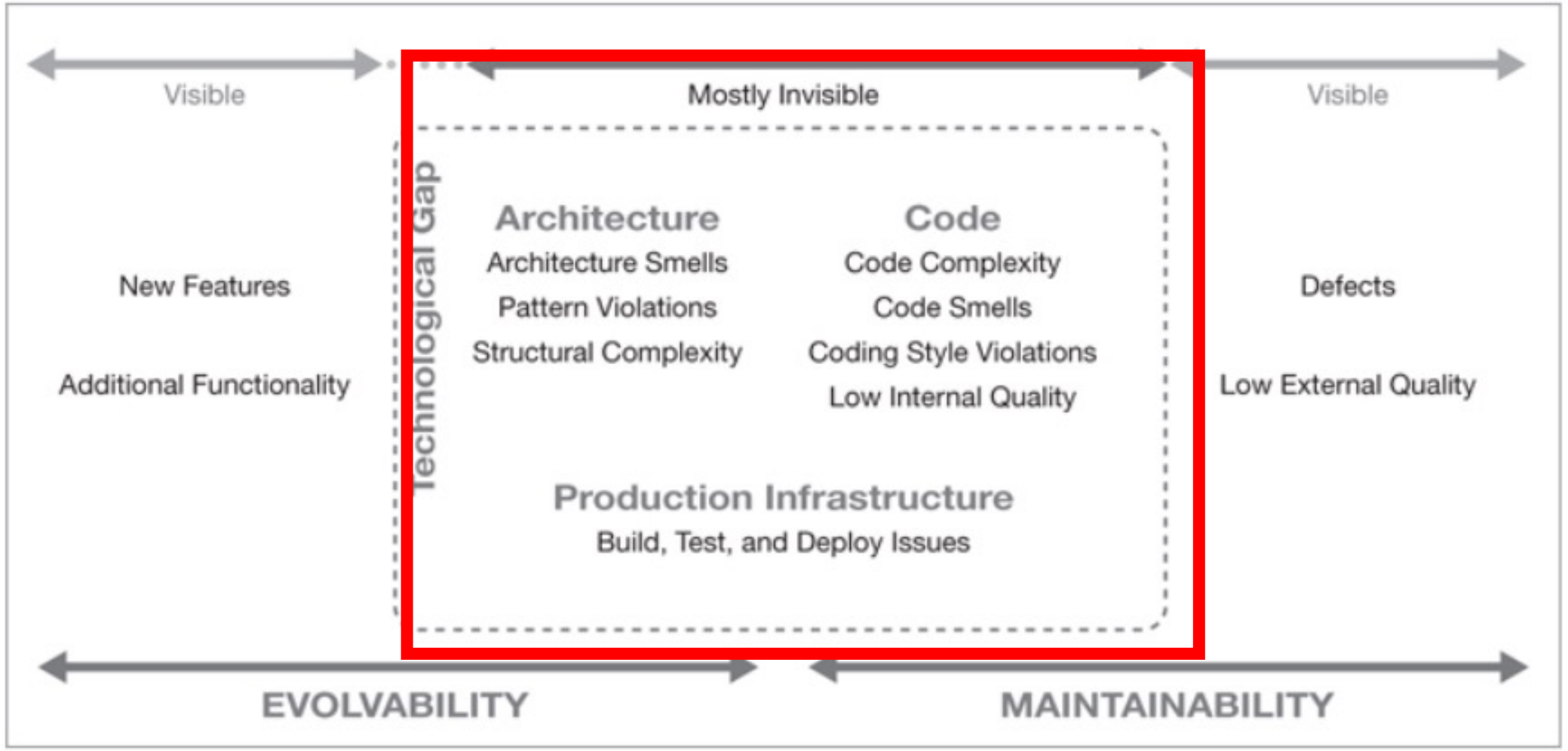
- Technical Debt seems a *catch-for-all* to put every negative consequence that happen to our software assets (code, tests, requirements…)
- **"Yes! we have a lot of this…"** is what we hear when we refer to it.


TECHNICAL
DEBT

TD Consequences

So what?

We need to remember…



Kruchten, P., Nord, R., Ozkaya, I.: Managing Technical Debt: Reducing Friction in Software Development. Pearson (2019).

From now on...

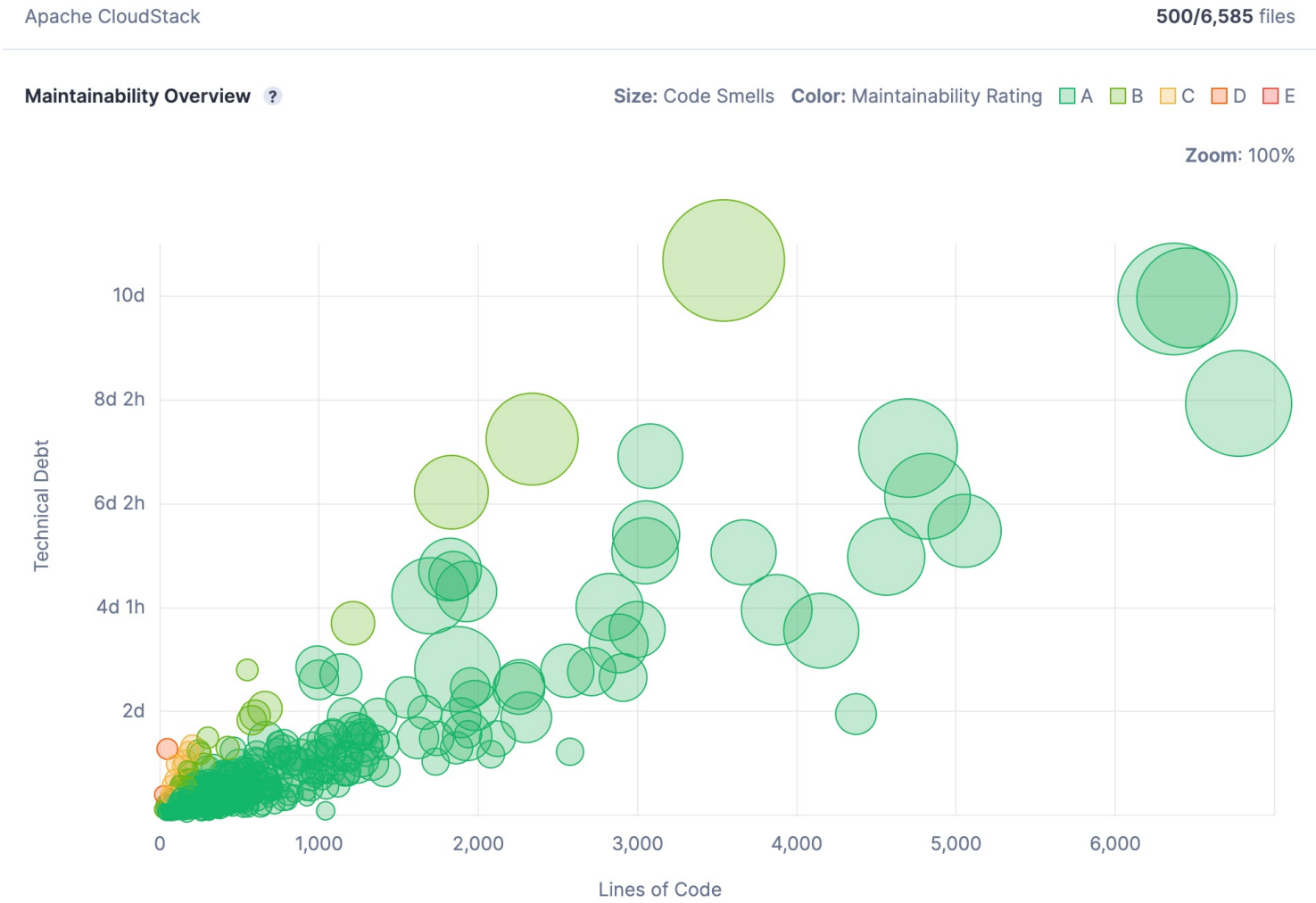
- I will illustrate the concepts with real examples from the collaborative research we perform with our industrial partners

Company A

Company B

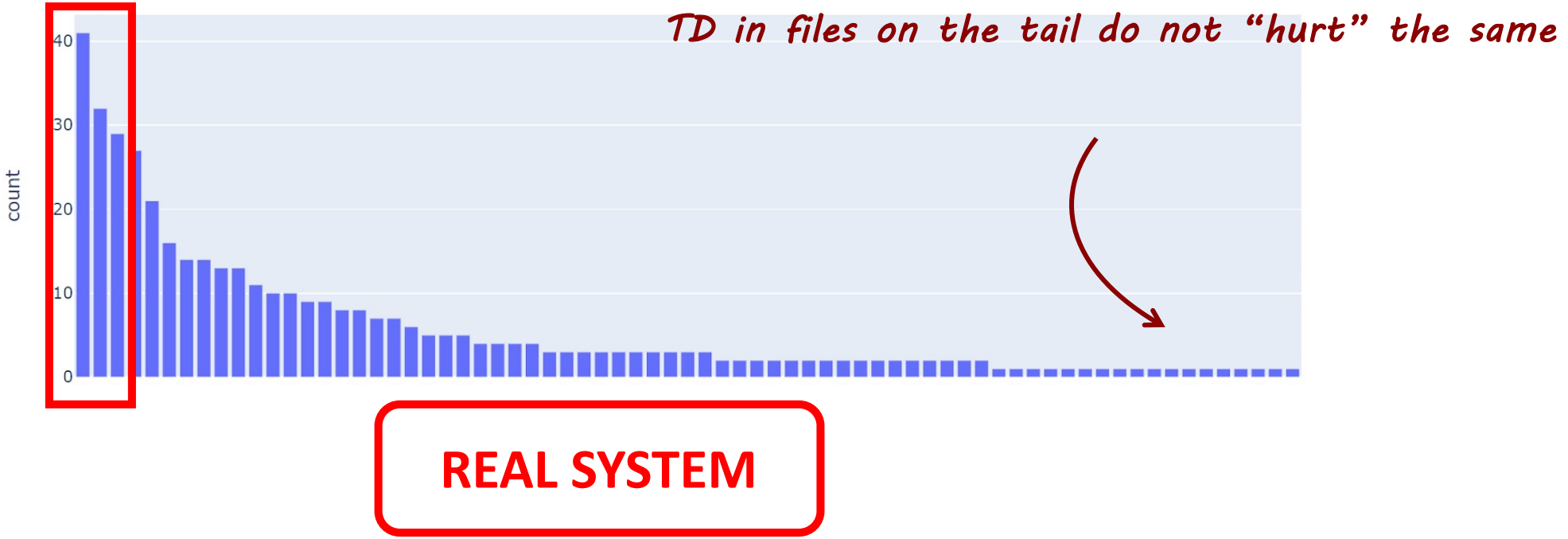
=

If we look at it the “normal way”



But.. Where is the actual pain?

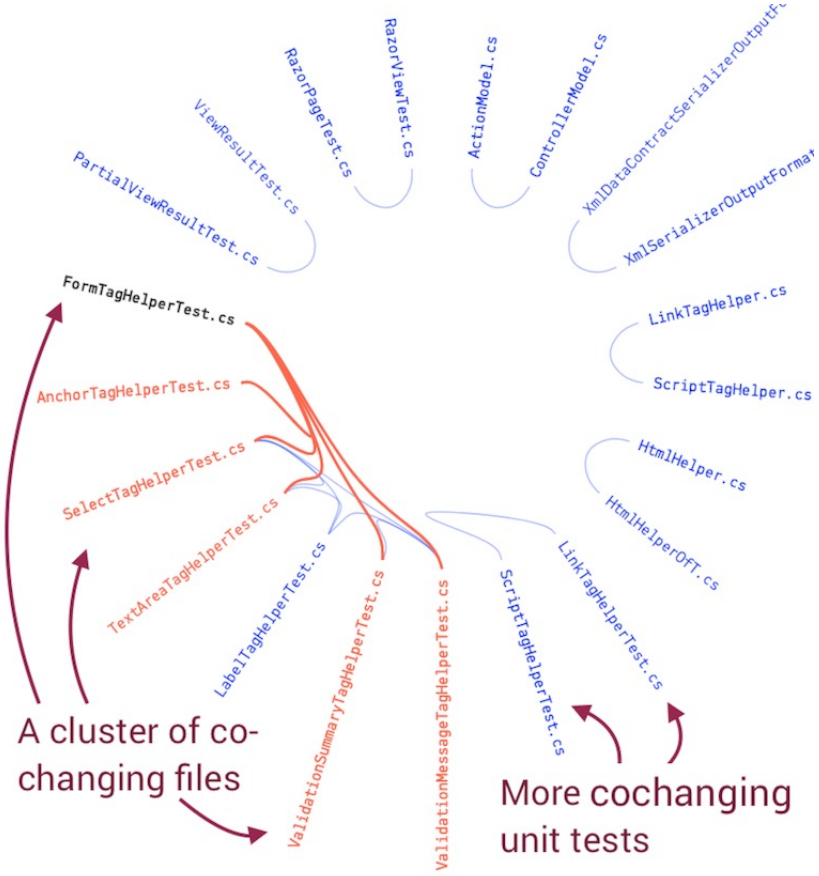
- Not all TD "hurts" the same
- For example: In any system, ~20% of the files take ~80% of the maintenance

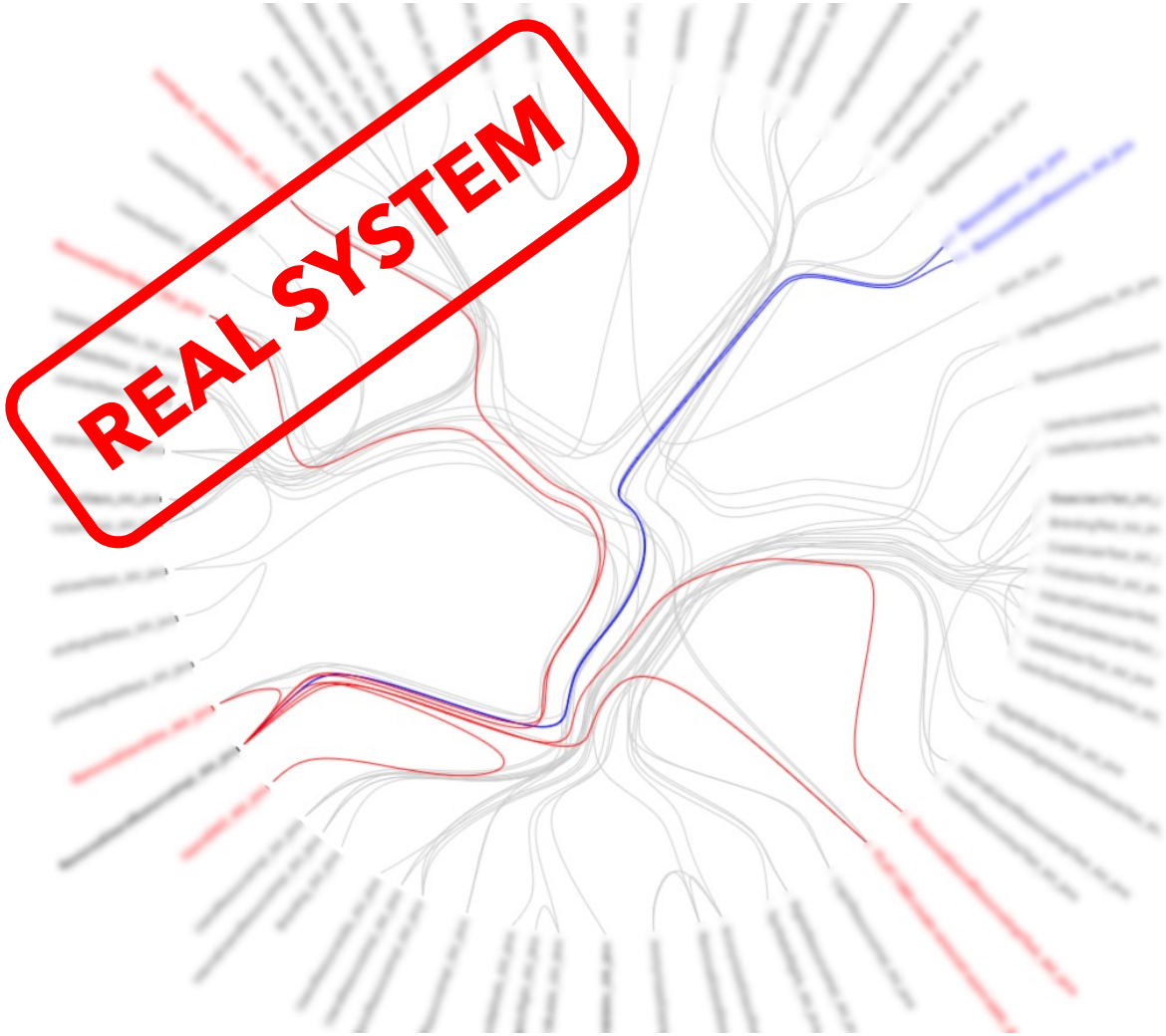


Company A

A. Tornhill, Your Code As A Crime Scene. The Pragmatic Bookshelf, 2015.

Making it more complex...





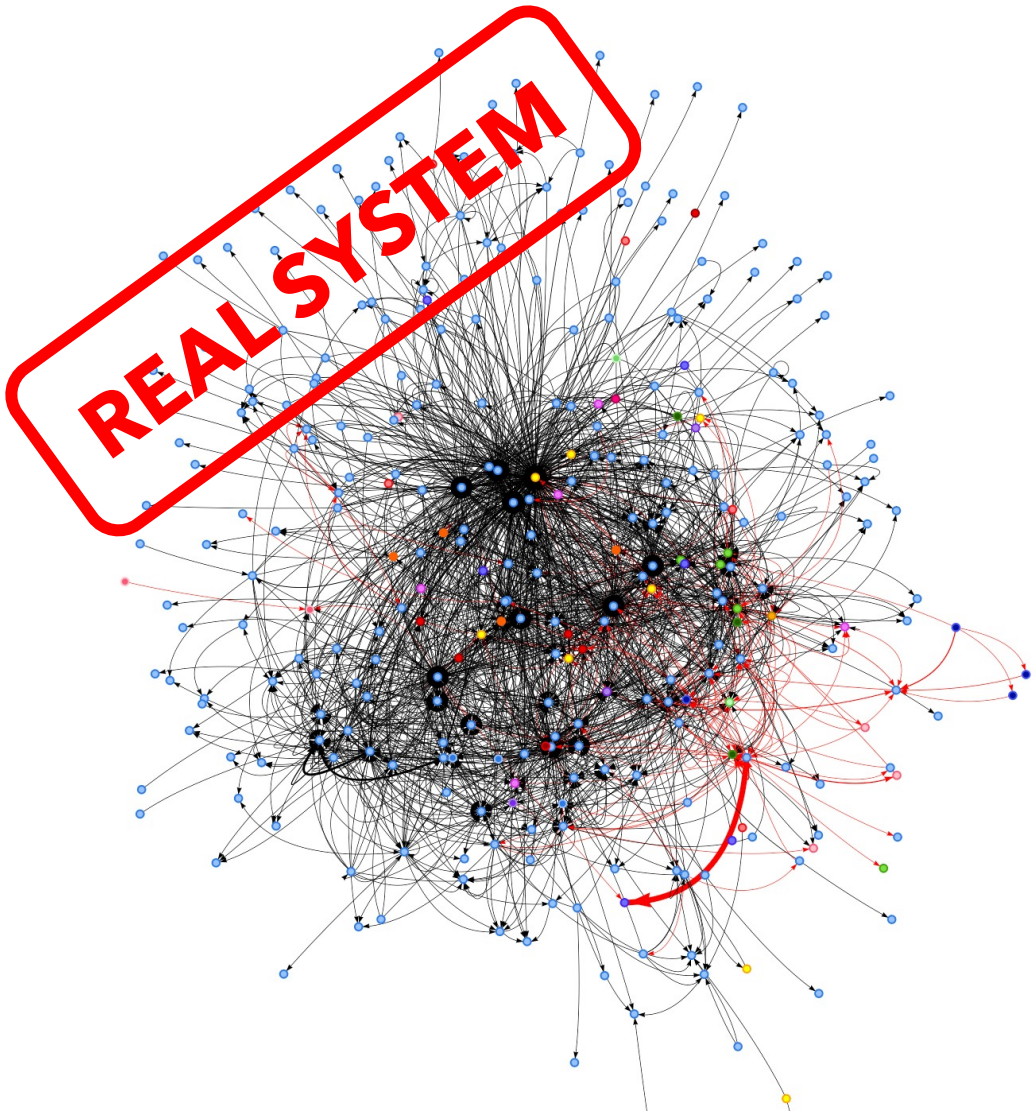
Company A

A. Tornhill, Your Code As A Crime Scene. The Pragmatic Bookshelf, 2015.

At large...

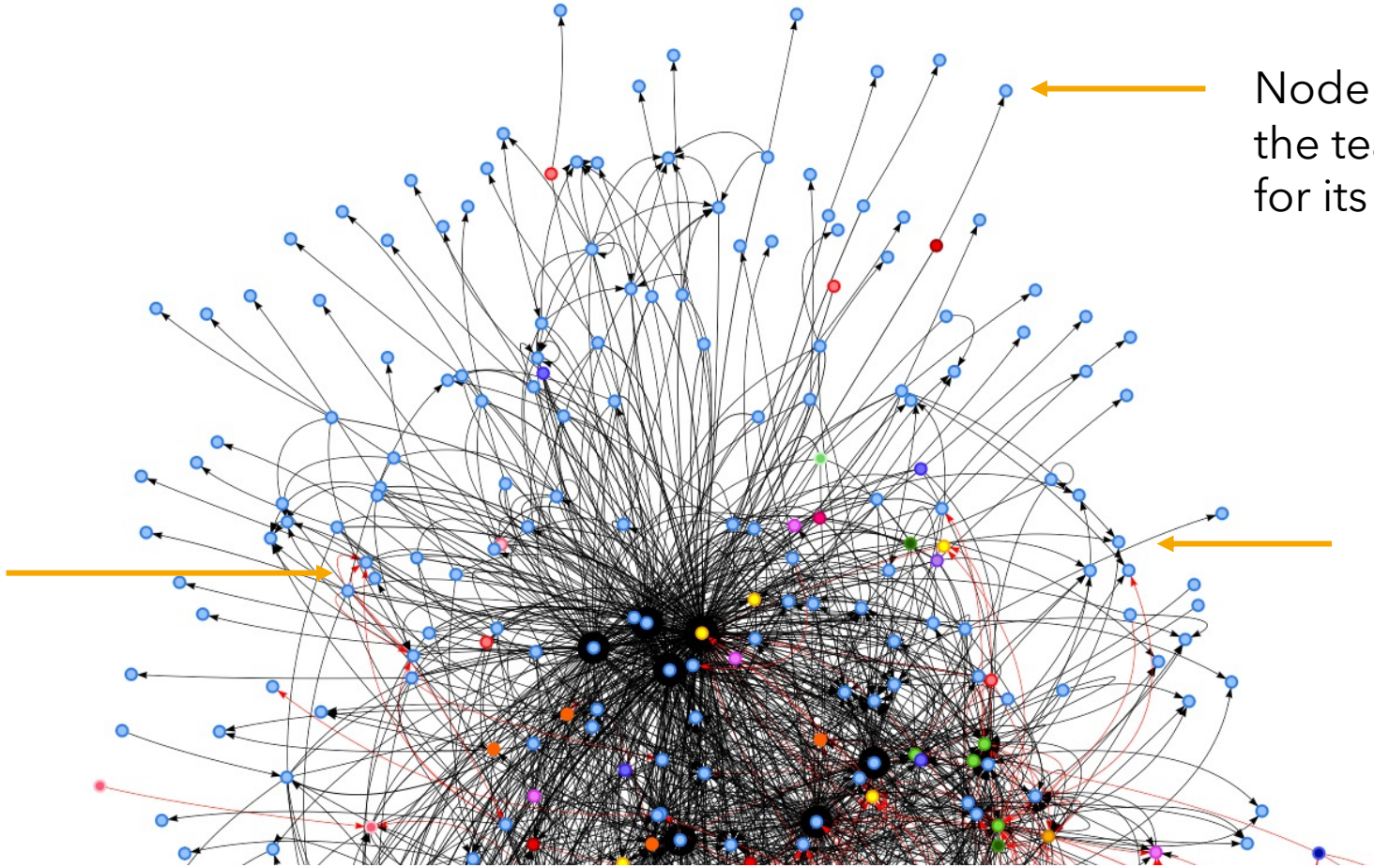
Organization and Architecture

- Looking at the whole system
  - Systems are more than one repository/component
  - Files are not the problem
- Change Cohesion at repository level



Company A

Organization and Architecture



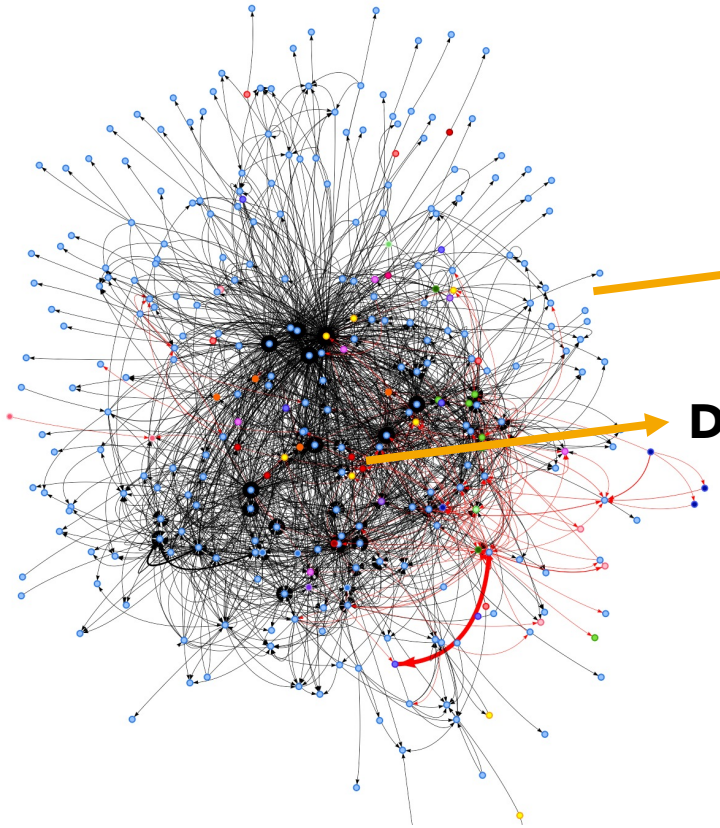
Node color denotes the team responsible for its quality

Red arrow: task dependencies (>5)

Black arrow: code or service dependencies (>5)

Company A

Software Component Dependencies



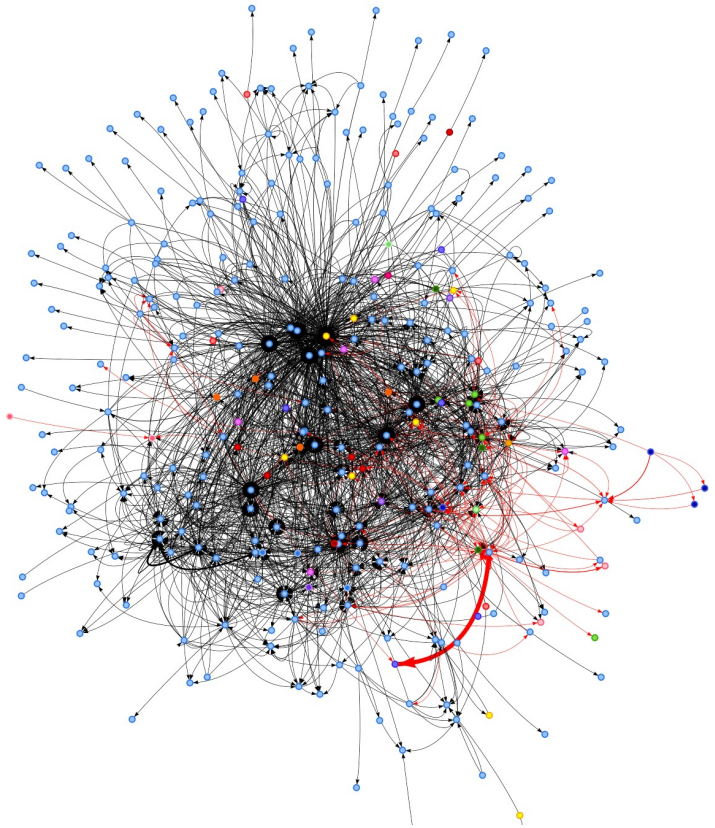
Isolated Repos

Borderline Repos

Dependency Magnets

Company A

Software Component Dependencies

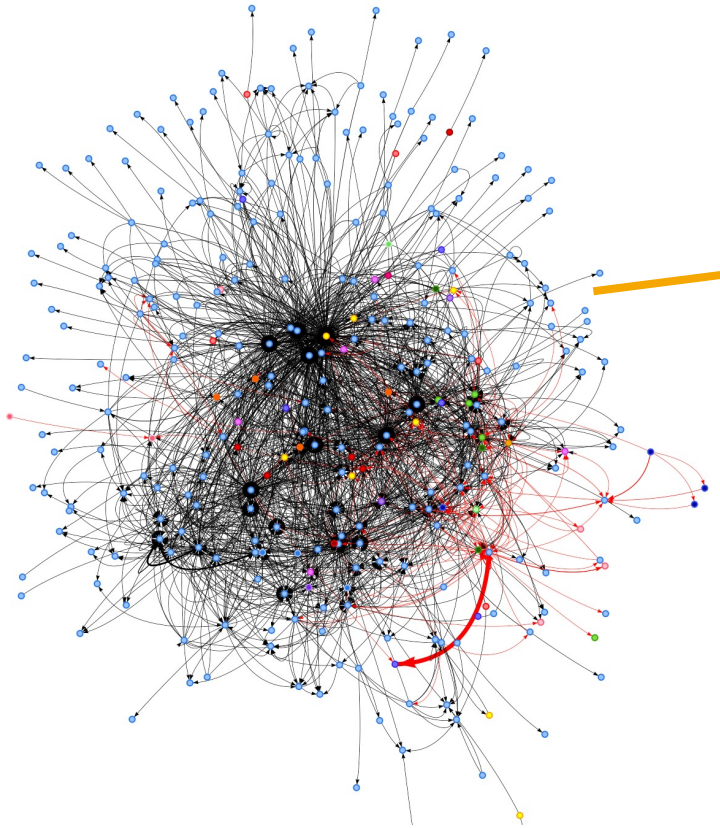


Isolated Repos

- Dependencies are mostly technical
- problems can be addressed easily
- Limited to no propagation of problems

Company A

Software Component Dependencies

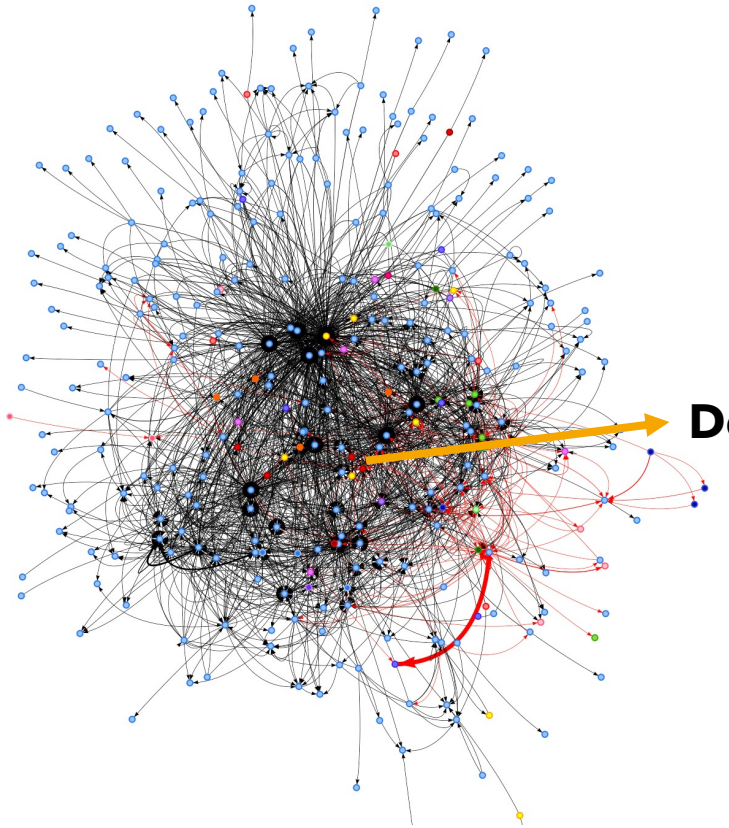


Borderline Repos

- Repos with smaller dependencies
- Mostly owned by the same team
- Small propagation of problems
- Problems can be fixed with small effort

Company A

Software Component Dependencies



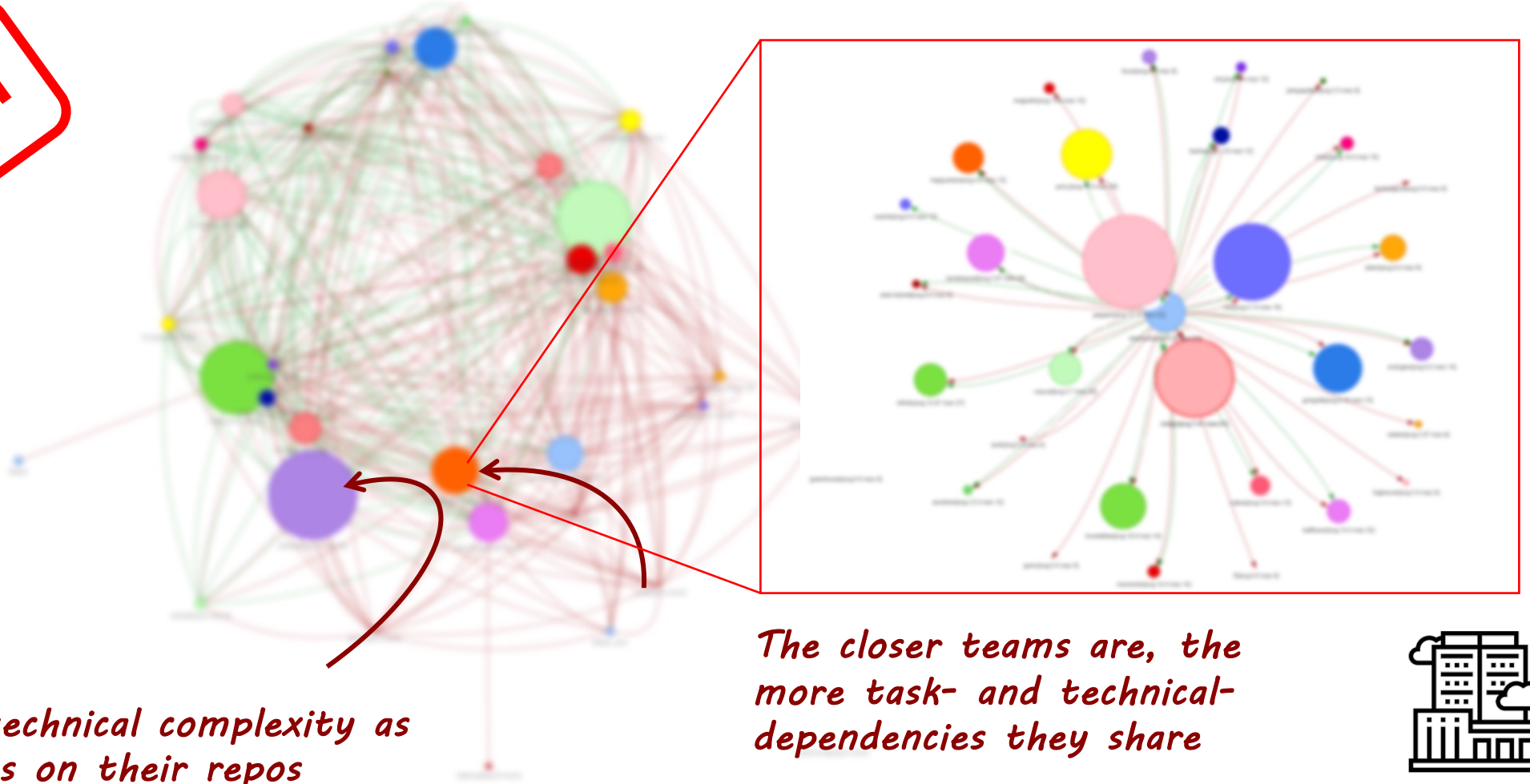
Dependency Magnets

- Regardless of the type
- High propagation of problems
- Problems need much more attention

Company A

Organisational Structures: looking at teams

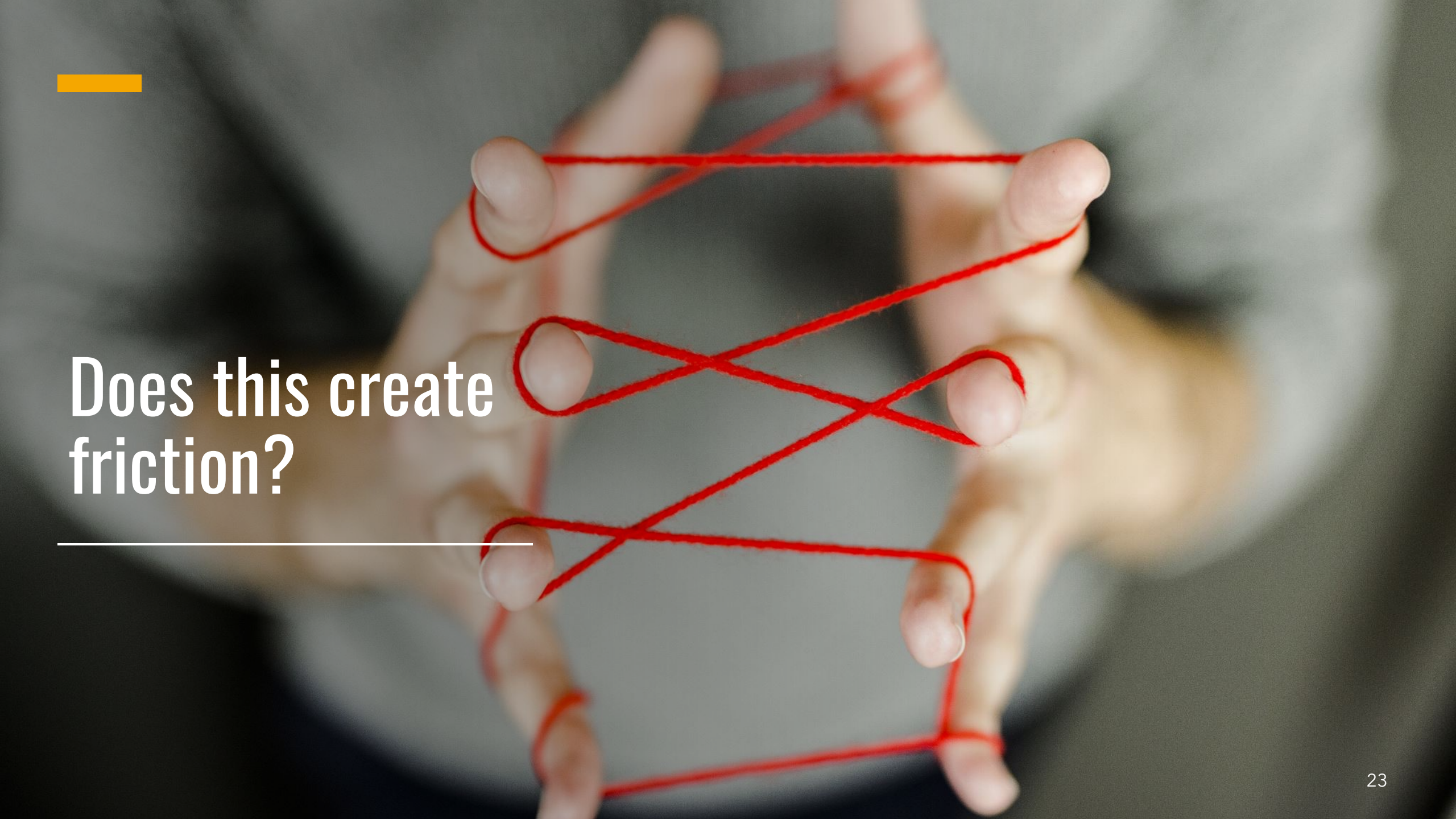
REAL SYSTEM



Nodes are teams
Size shows the technical complexity as # of dependencies on their repos

The closer teams are, the more task- and technical-dependencies they share

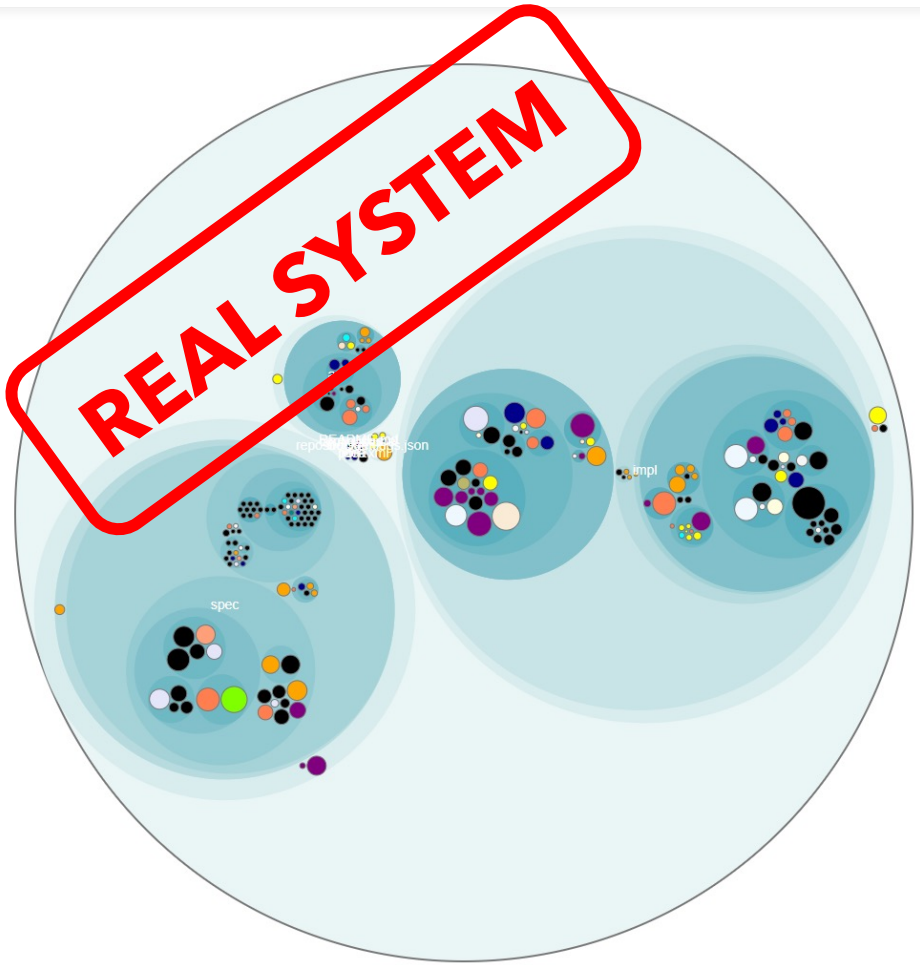
Company A



Does this create friction?

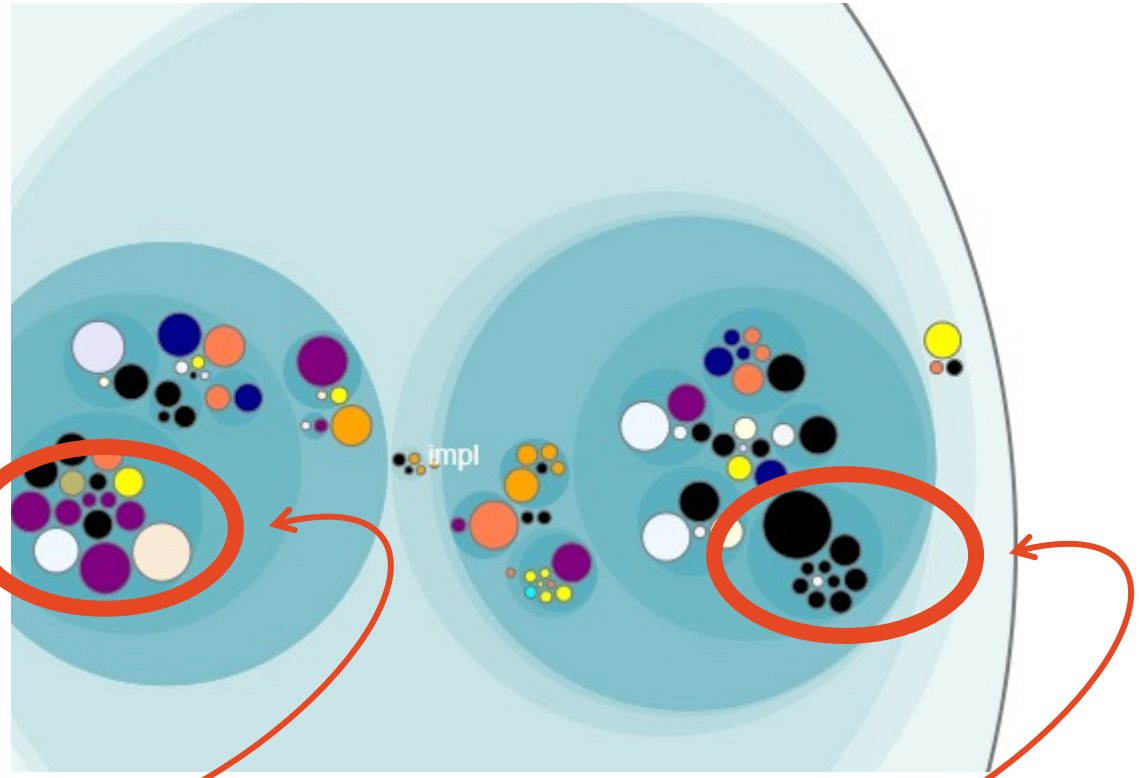
Ownership Misalignment

- **Ownership misalignment:** The team that owns a repo is not its *main* contributor



Ehsan Zabardast, Javier Gonzalez-Huerta, & Binish Tanveer. (2022). Ownership vs Contribution: Investigating the Alignment Between Ownership and Contribution. 19th IEEE International Conference on Software Architectures. https://doi.org/10.1109/ICSA-C54293.2022.00013

Responsibility diffusion & lack of ownership

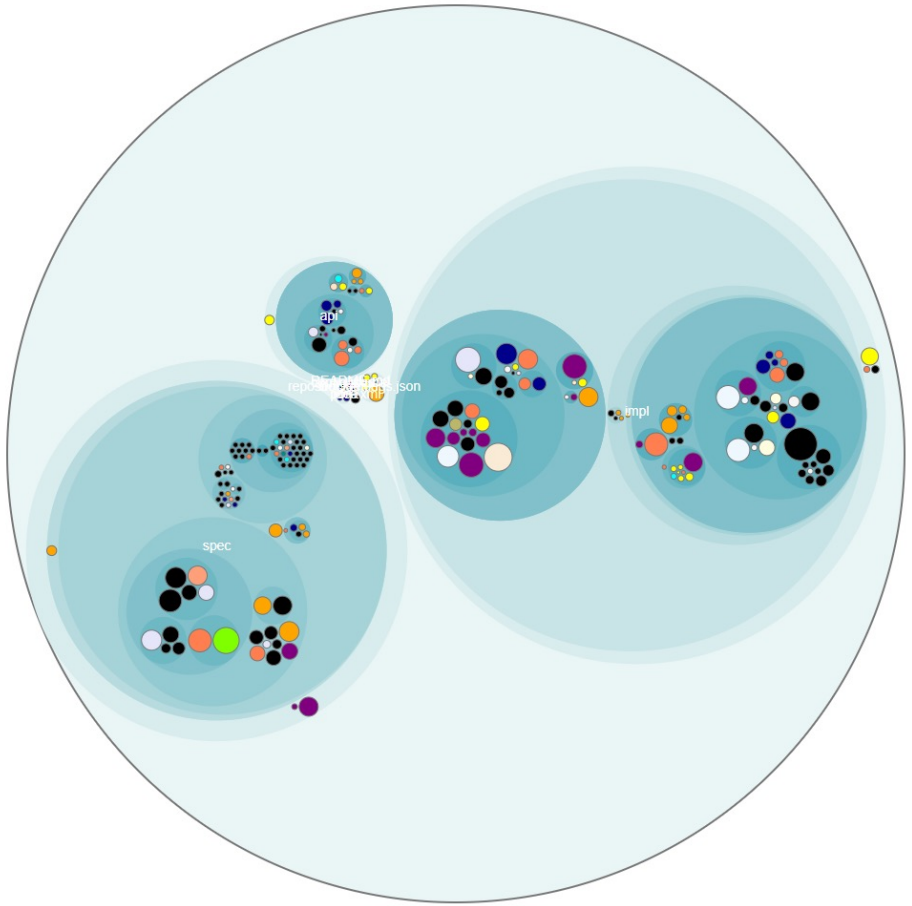


The colours of the circles denote the team of the main contributor

Black circles are code components whose main contributor has left the company

Responsibility Diffusion and Knowledge Vaporization

- A high number of teams/individuals contributing might cause *responsibility diffusion*
  - *"No one really owns the house, and no one cleans it"*
- We can also have knowledge vaporization, due to employees leaving the company



Barley, J. M., & Latanfi, B. (1968). Bystander Intervention in Emergencies: Diffusion of Responsibility. *Journal of Personality and Social Psychology*, *8*(4), 377–383.

Tornhill, A. (2015). *Your Code As A Crime Scene*. The Pragmatic Bookshelf. https://doi.org/10.1017/CBO9781107415324.004

Architecture and Organization Alignment: Scenarios

Scenario A: Technical Dependency between Repo A and Repo B - No Organizational Dependency (Shared Jira Tickets) **– No harm, beyond complexity**

Company A

Tanveer, B., Zabardast, E., & Gonzalez-Huerta, J. (2023). An approach to align socio-technical dependencies in large-scale software development. International Conference on Software Architecture.

Architecture and Organization Alignment

Scenario B: Technical Dependency between A and B + Organizational Dependency (e.g., Jira Tickets) – **Expected**

If the repos are owned by the different teams, we might incur more time for coordination and PR approval

Problematic case: If there are ownership misalignment problems, we (might) add overhead, since there might be a third team involved

Company A

Tanveer, B., Zabardast, E., & Gonzalez-Huerta, J. (2023). An approach to align socio-technical dependencies in large-scale software development. International Conference on Software Architecture.

Architecture and Organization Alignment

Scenario C: No Technical Dependency between Repo A and Repo B - But Organizational Dependency (Shared Jira Tickets) **– Not expected**

If the repos are owned by the different teams, we might incur more time for coordination and PR approval (**longer than in B since the coordination is unexpected)**

Problematic case: If there are ownership misalignment problems, we (might) add **even more overhead**, since there might be a third team involved

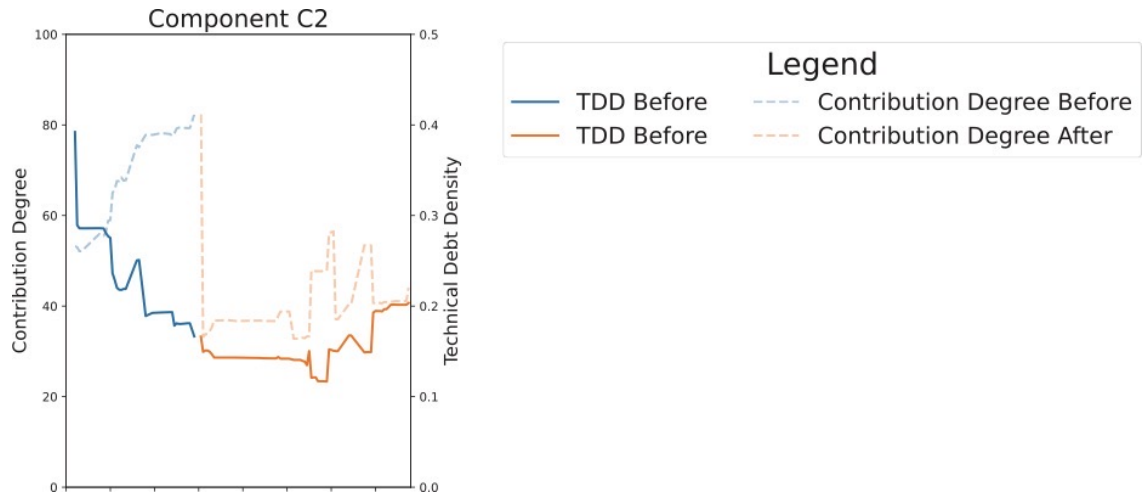
(c) Ehsan Zabardast & Javier Gonzalez Huerta

Company A

Tanveer, B., Zabardast, E., & Gonzalez-Huerta, J. (2023). An approach to align socio-technical dependencies in large-scale software development. International Conference on Software Architecture.

But the frictions are not limited to time

- These frictions can also manifest in how fast TD is accumulated
- We are going to talk about this during the last part of the presentation



Organizational factors and TD

Degree of Ownership and Evolution of TD

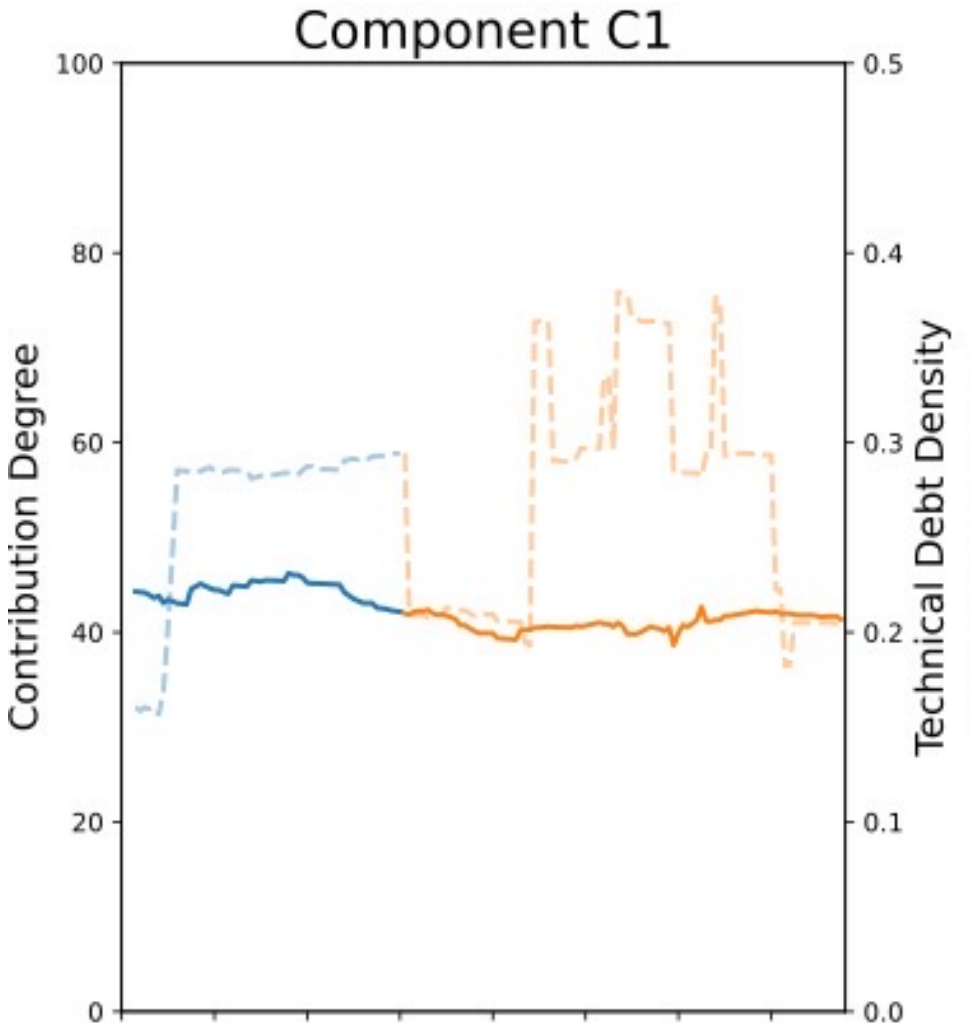
Objective

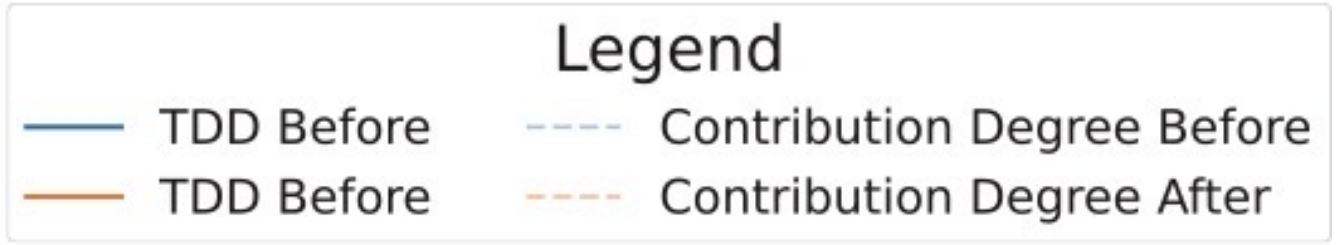
- We have conducted a study to understand if the degree of contribution can explain how fast or slow TD accumulates
- In this case, we have a formal ownership model:
  - There is a team that is responsible for the quality of each repo
- To see the degree of contribution, we analyse the authors of the code (commits), tickets (Jira), and pull requests (bitbucket).

Company A

Zabardast, E., Gonzalez-Huerta, J., Palma, F., & Chatzipetrou, P. (2023). The Impact of Ownership and Contribution Alignment on Code Technical Debt Accumulation. https://arxiv.org/abs/2304.02140

The effect of ownership over TD over time

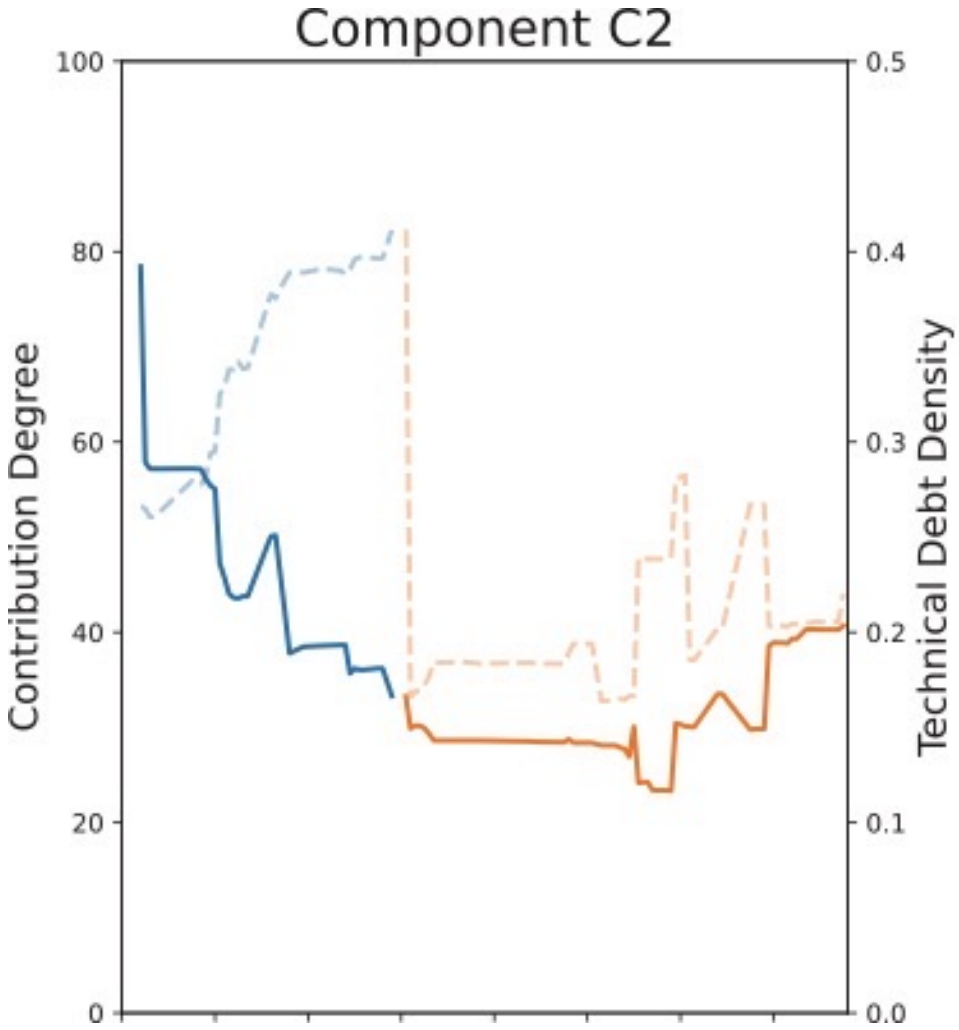


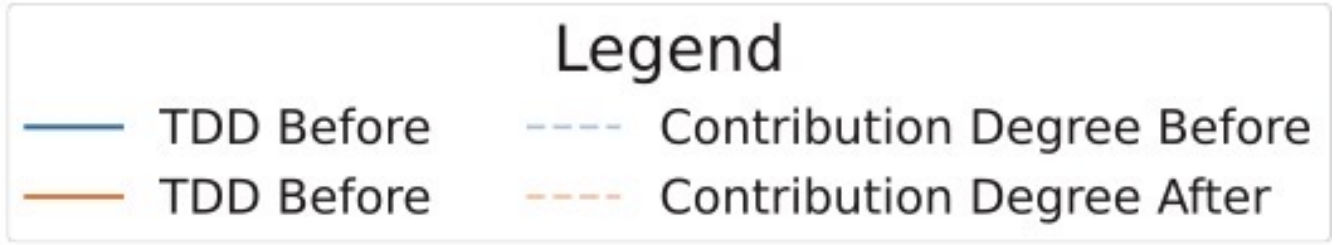


REAL SYSTEM

Company A

The effect of ownership over TD over time

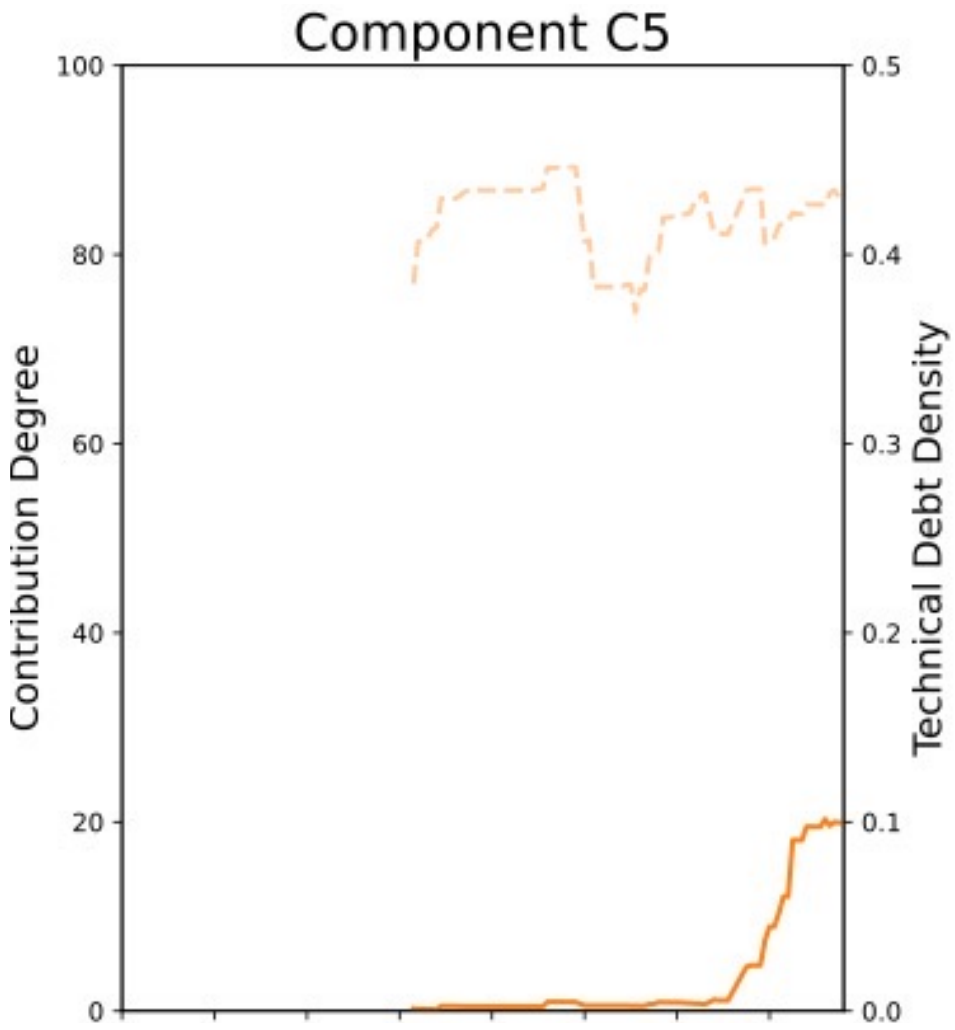


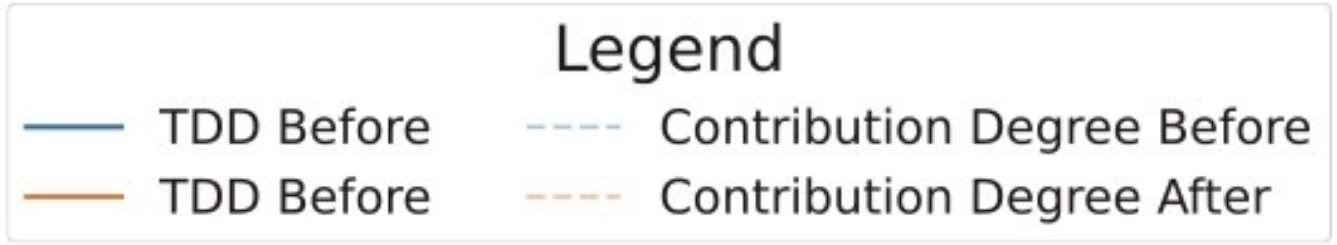


It seems when contribution goes below a certain level (here <40%), after a period, things tend to go out of control

Company A

The effect of ownership over TD over time





Another observation is that if you "own" everything, you feel more entitled to take risks. This has been sometimes referred as *Technical Credit*

Company A


Organizational factors and TD
Teams' behaviour when introducing Code Clones

Objective

- We have conducted a study to understand how different teams behave in different repos when it comes to producing clean code
- As a pilot, we focus on the introduction of code clones

- **Disclaimer:** The quality of the repos we have studied is very high, according to static analysis tools.

Company B

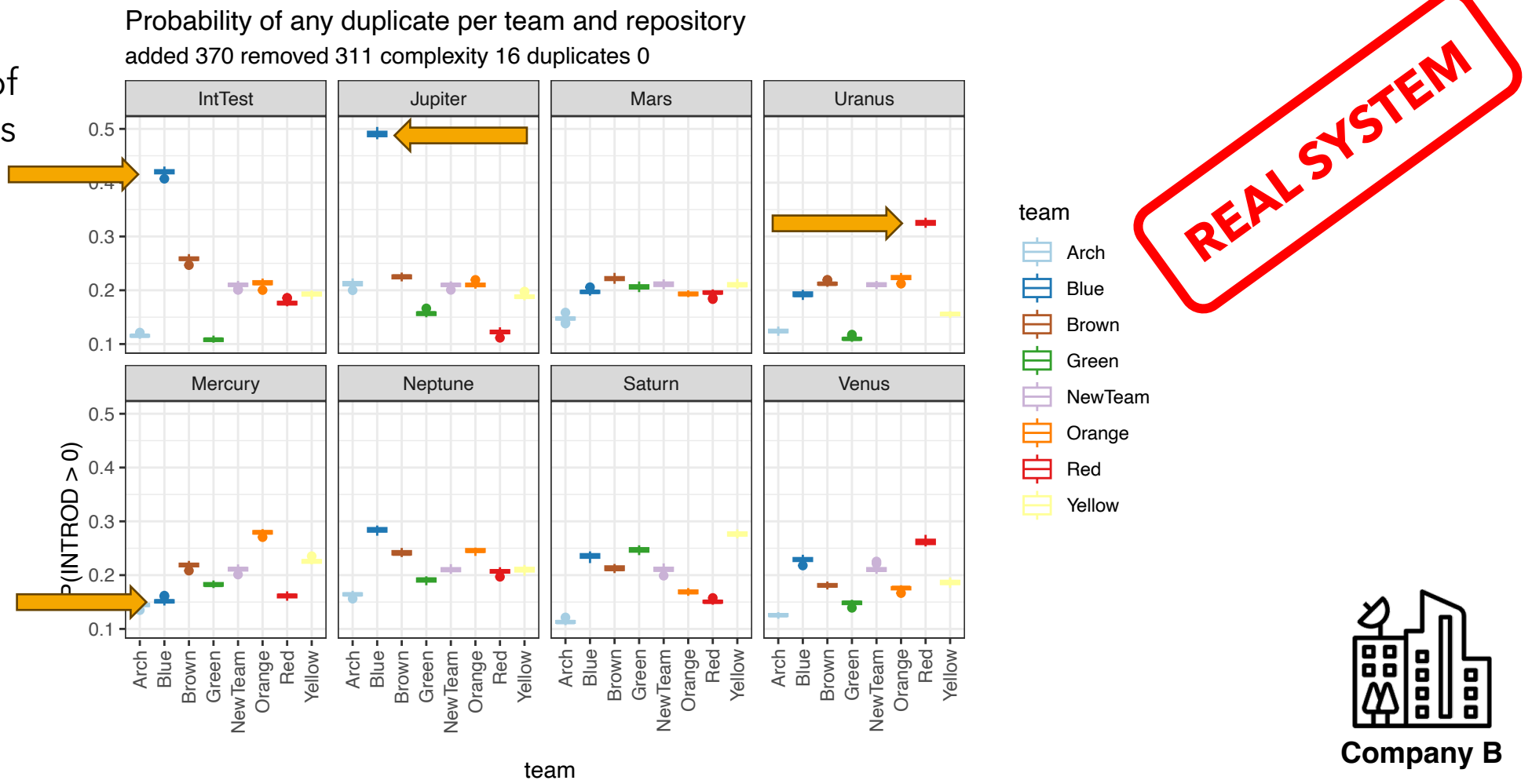
What have we done?

- We mined Git commits to see who committed where
  - In total, we looked at 8 repos during the WFH/Hybrid period 2020-2022
- We look at duplicates before and after each commit
- We calculate statistics to see which teams are contributing to each repo
- We can predict the probability of a team introducing clones

Company B

In a nutshell...

High probability of
Introducing clones

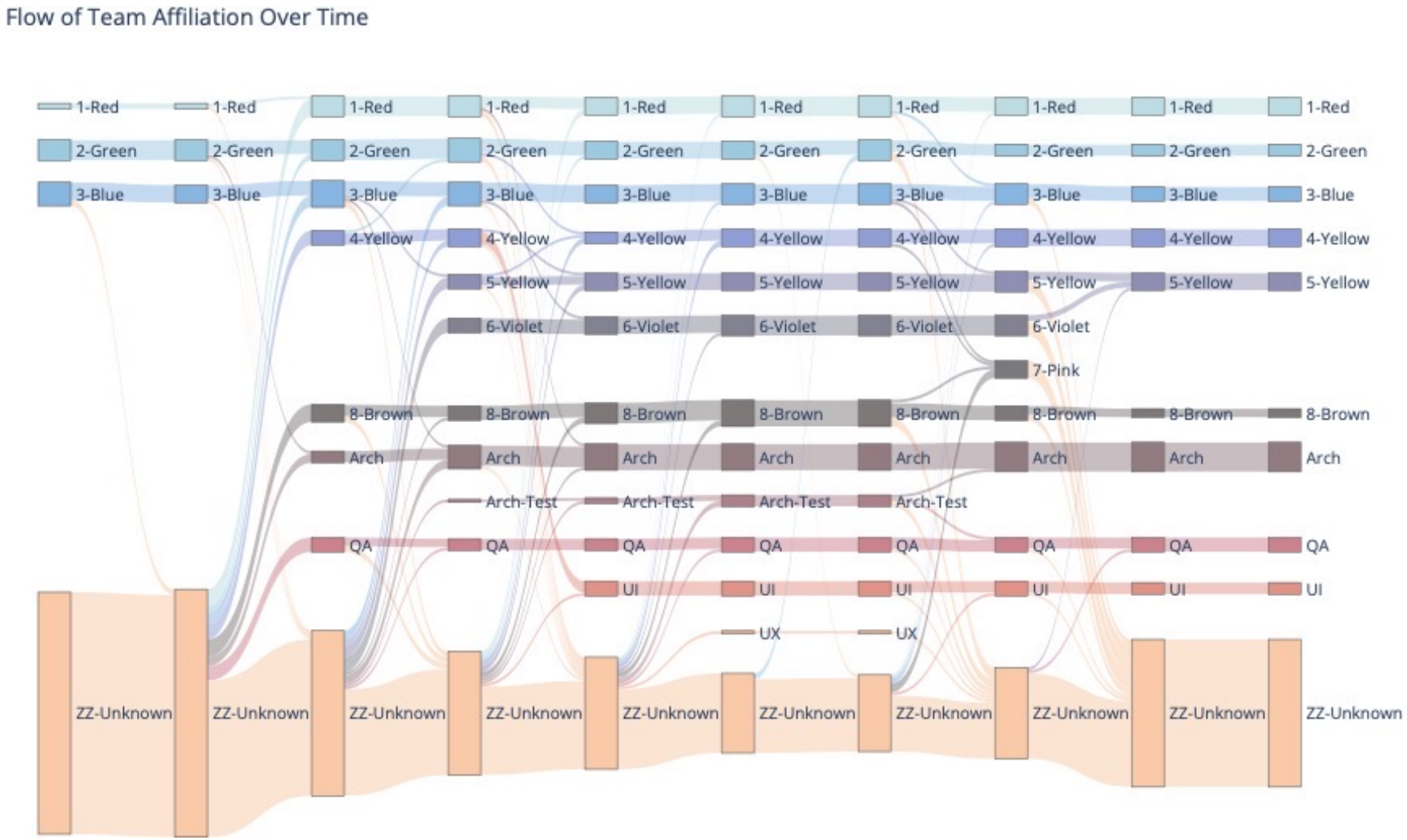


REAL SYSTEM

Company B

Company B

Evolution of Teams Over Time

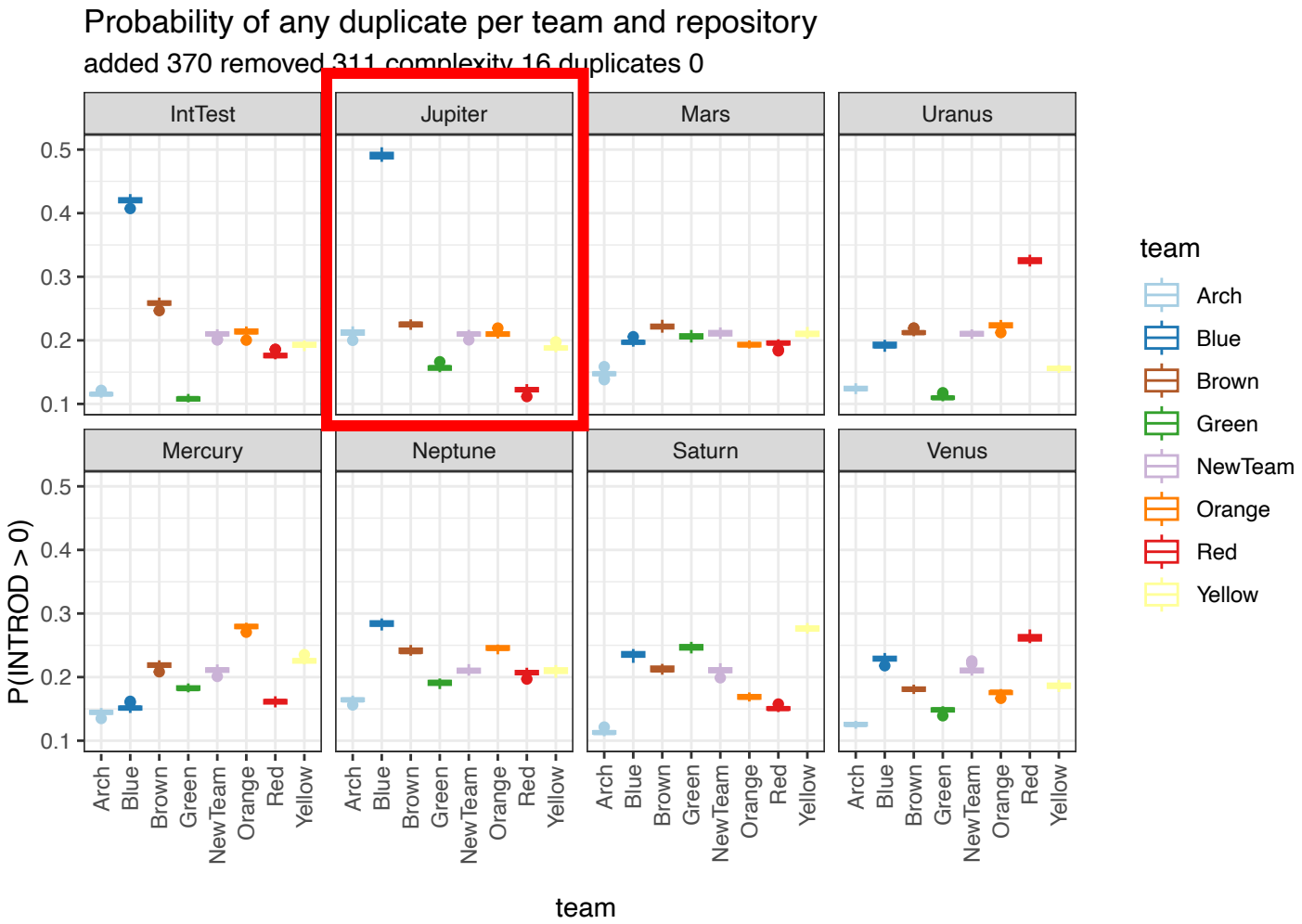


Company B

If we zoom in…

In a nutshell…

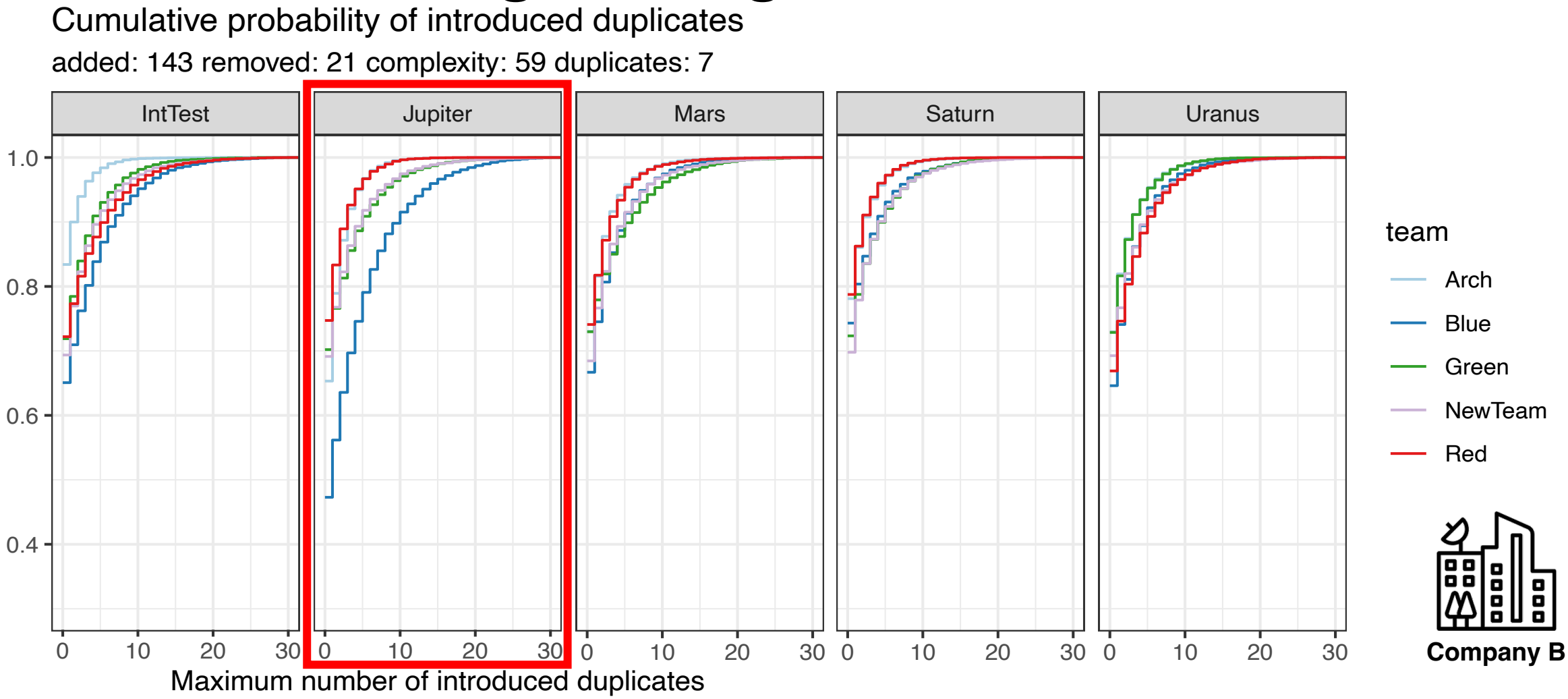
High probability of
Introducing clones



Company B

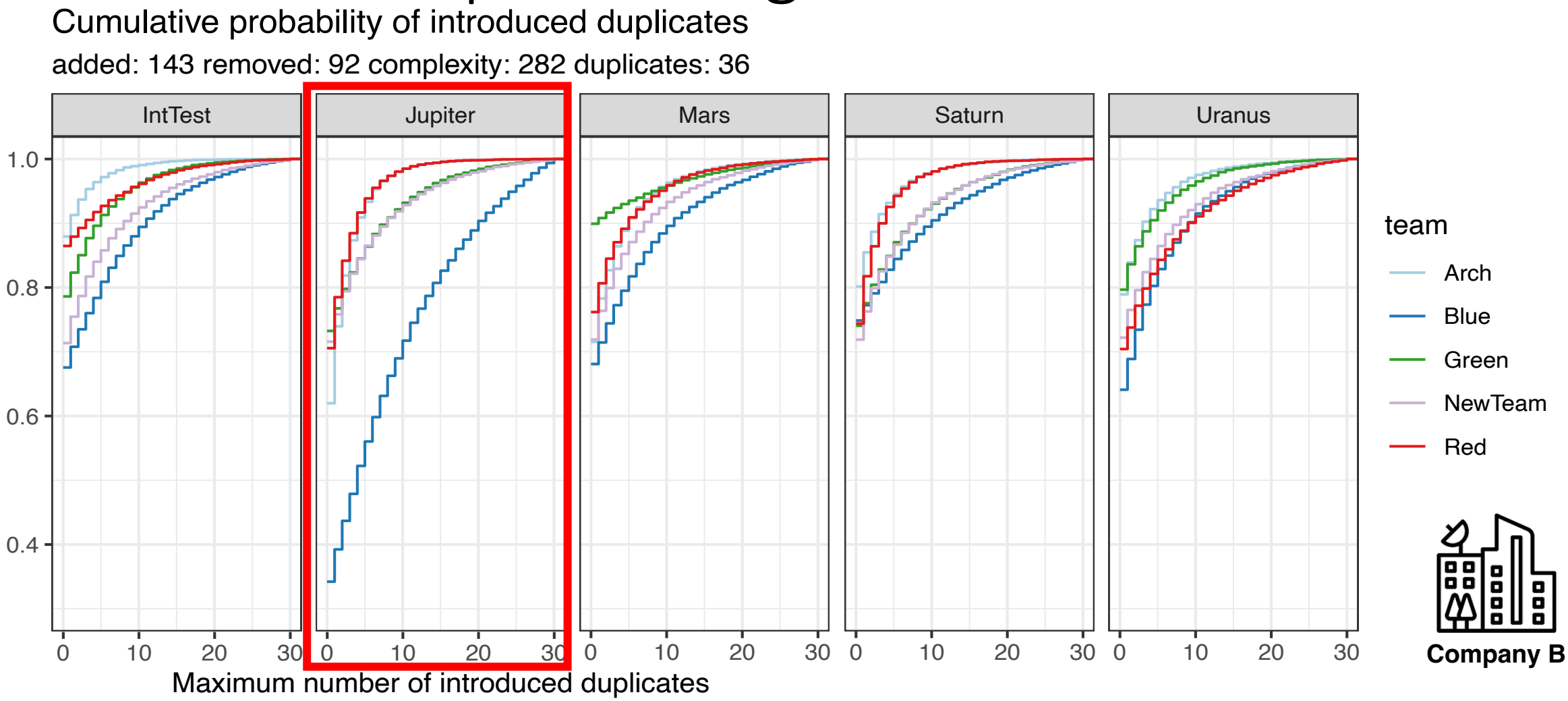
Company B

Results: average change



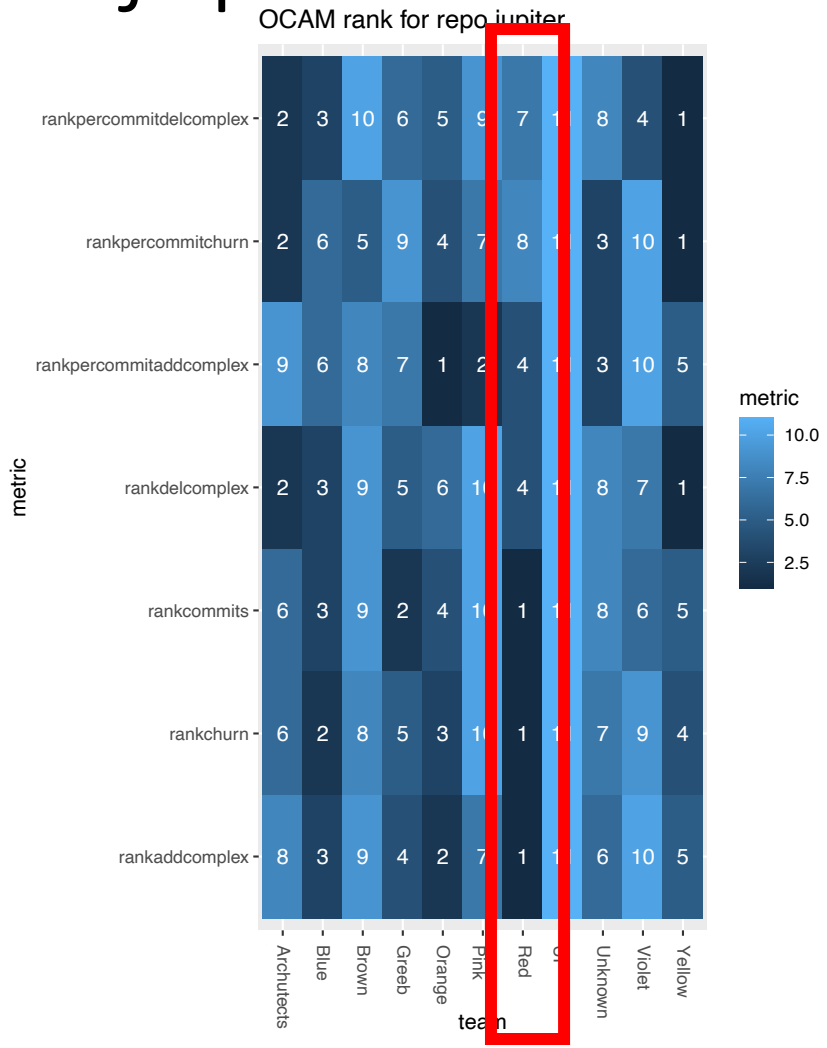
Company B

Results: complex change



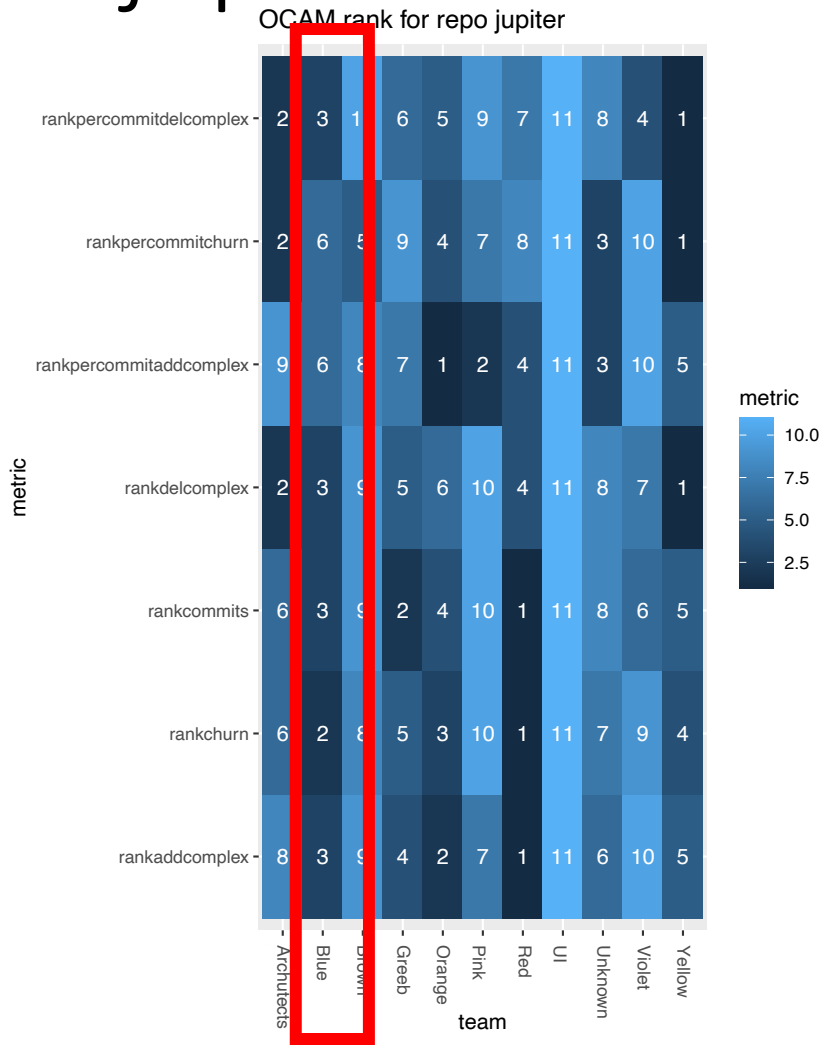
Company B

Contributions to jupiter



Company B

Contributions to jupiter

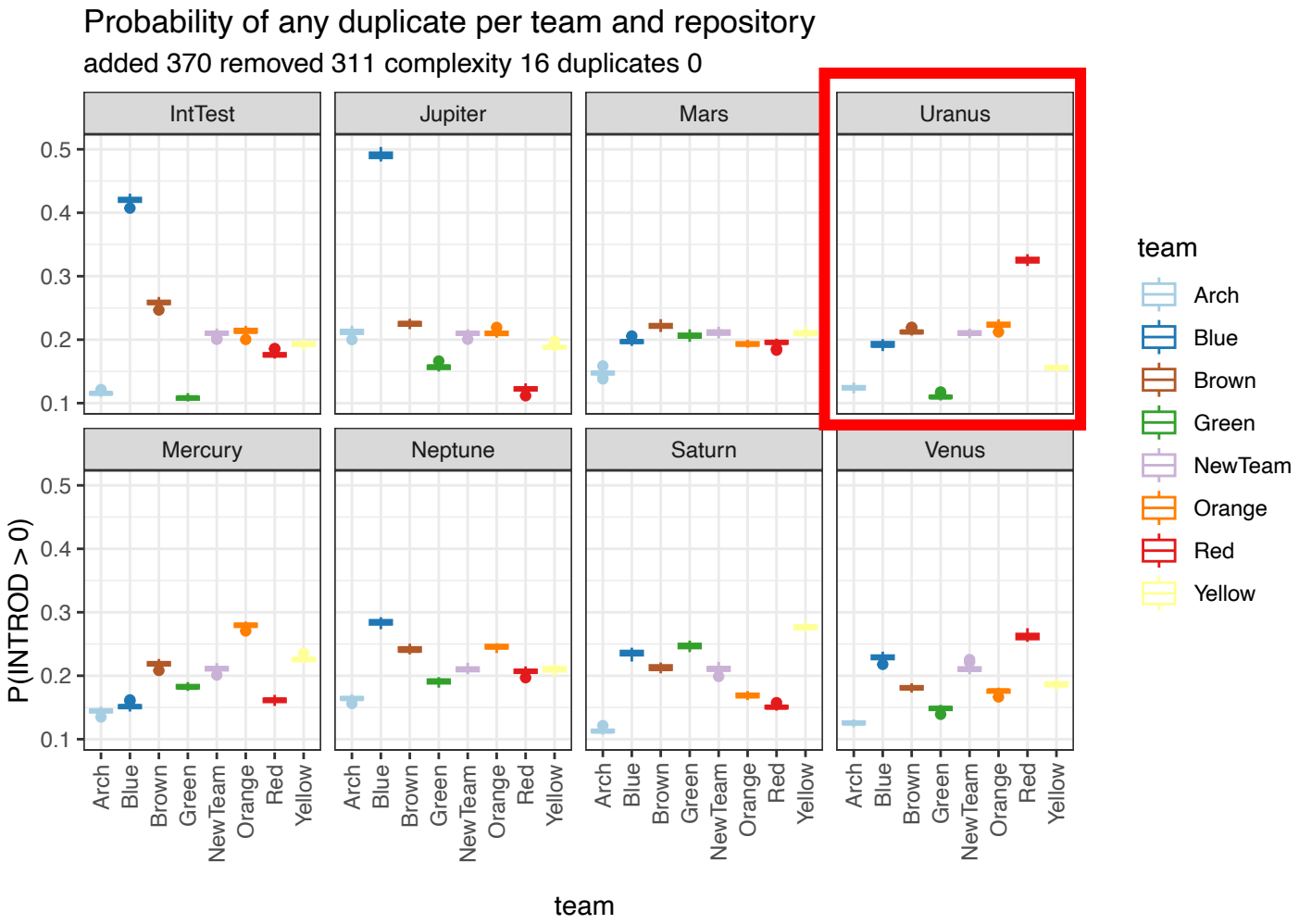


Company B

Another Repo…

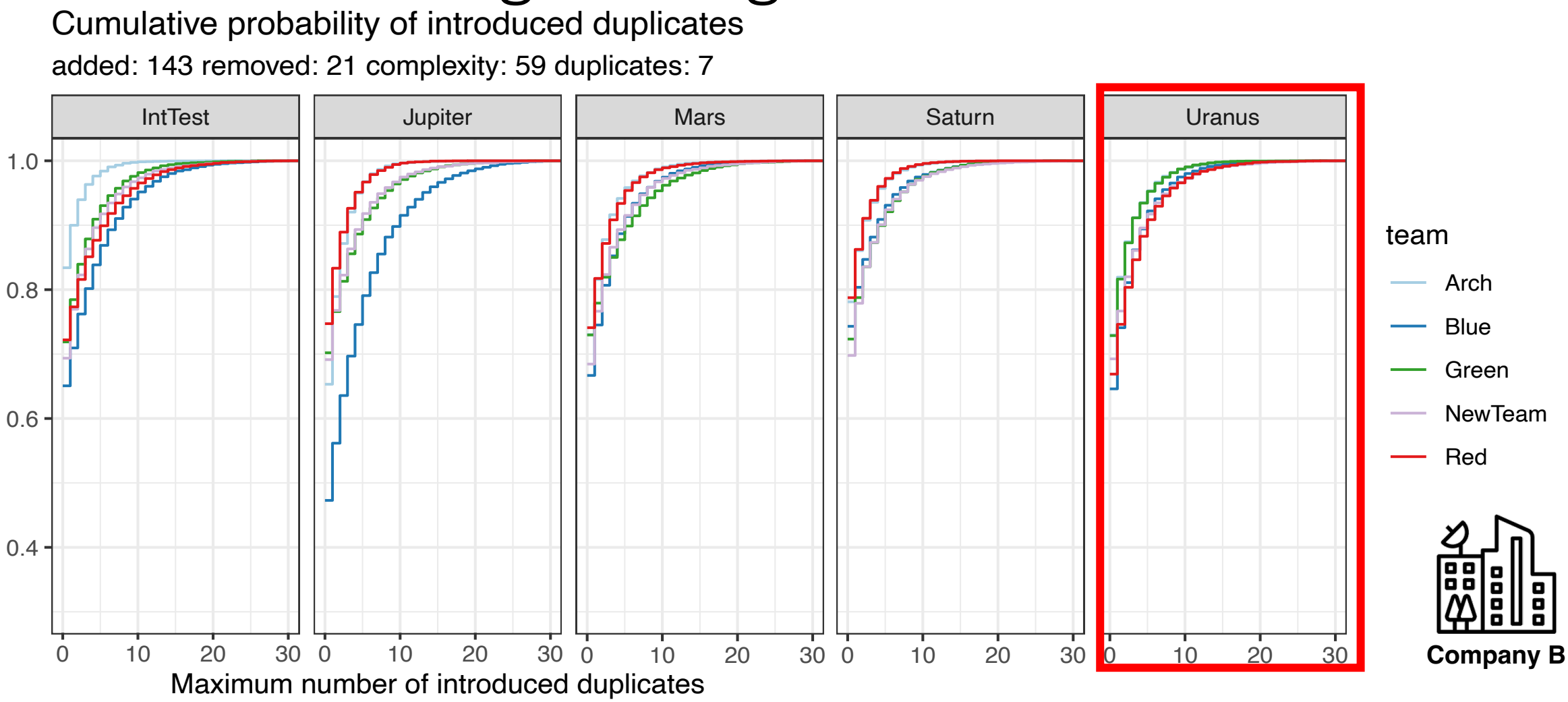
In a nutshell…

High probability of
Introducing clones

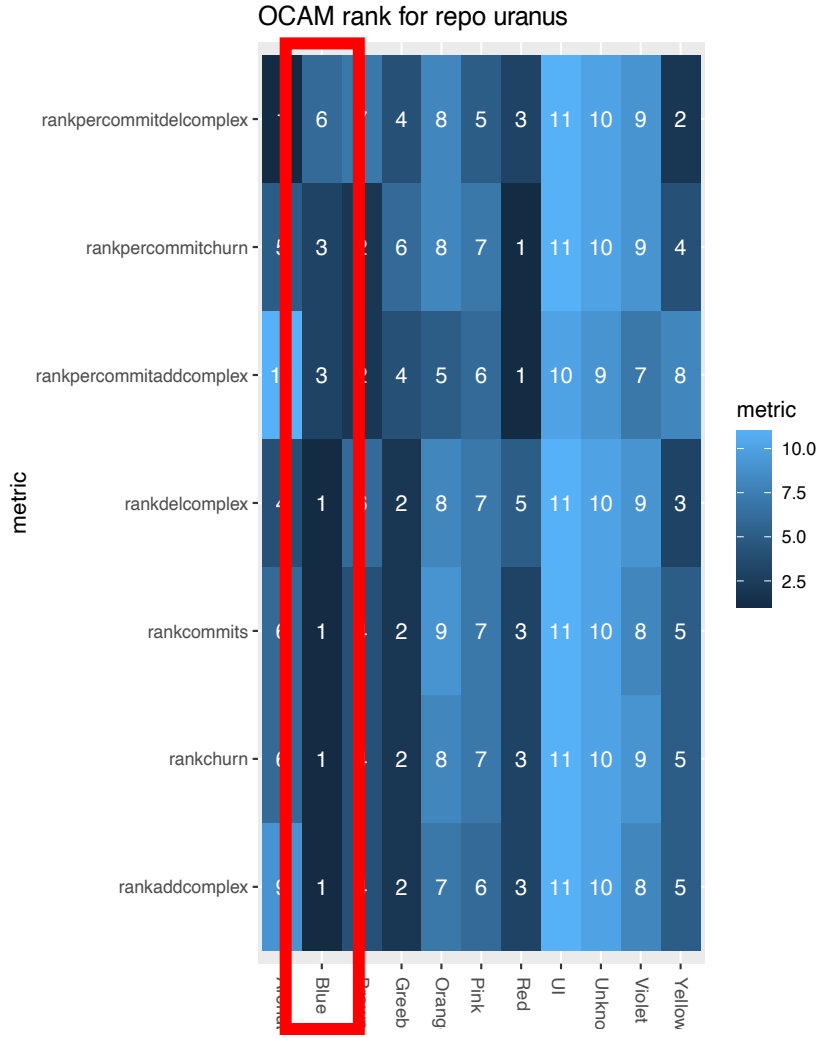


Company B

Results: average change

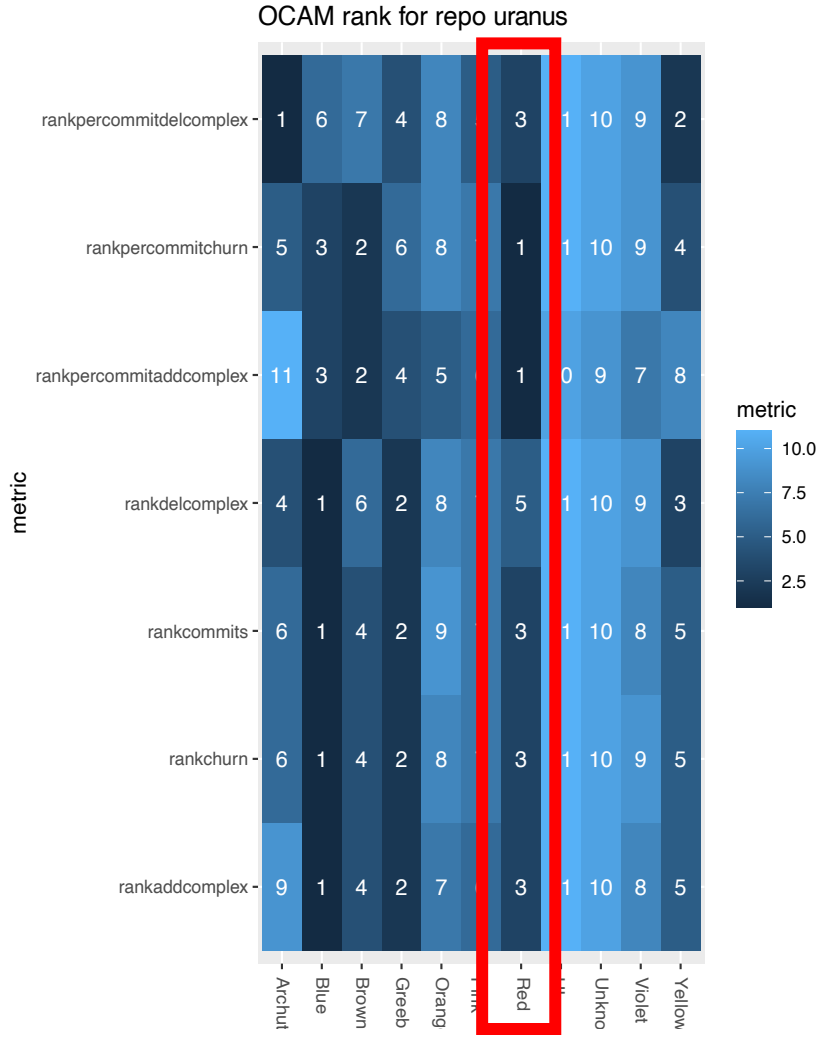


Contributions to Uranus



Company A

Contributions to Uranus

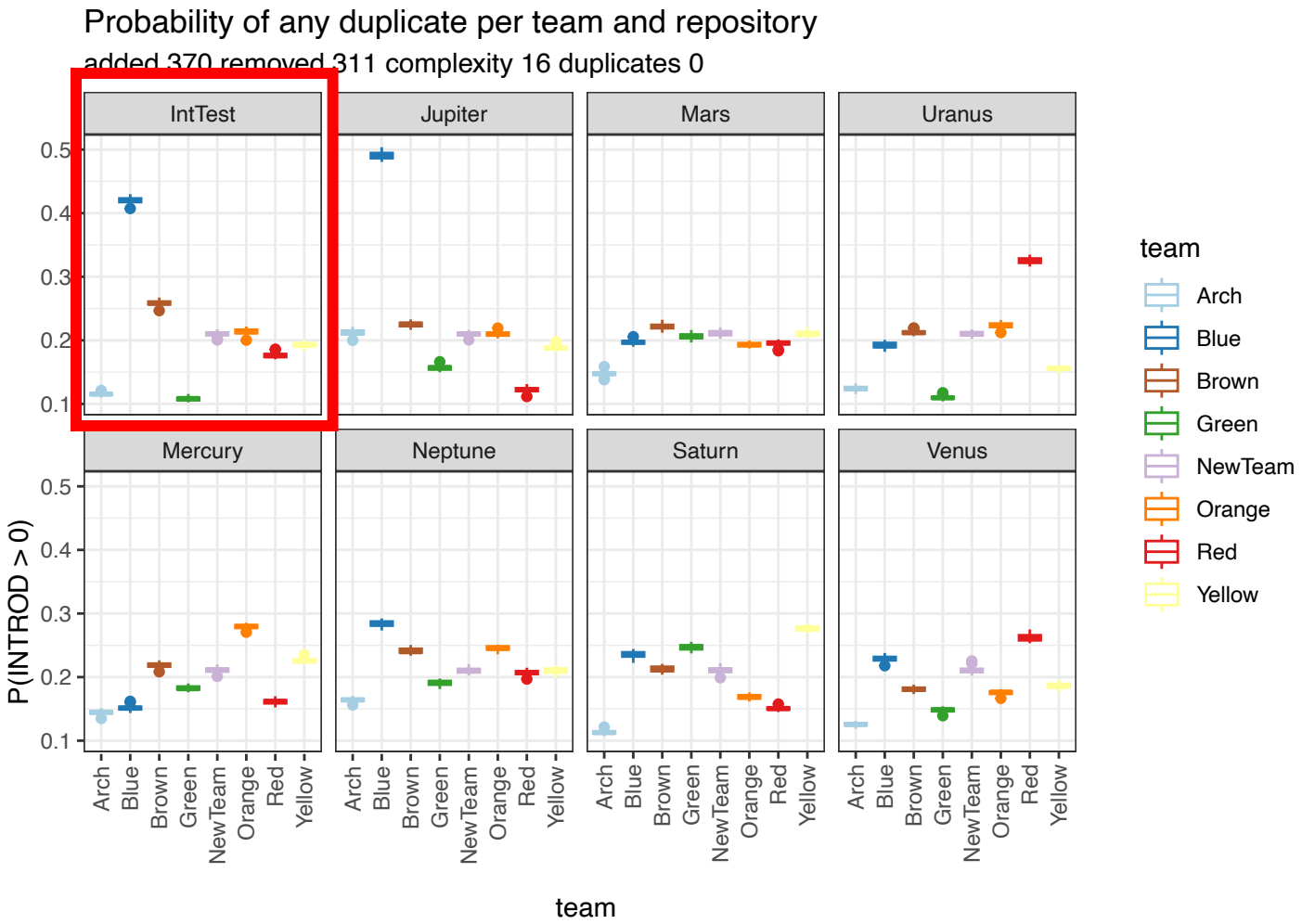


Company B

Integration Tests

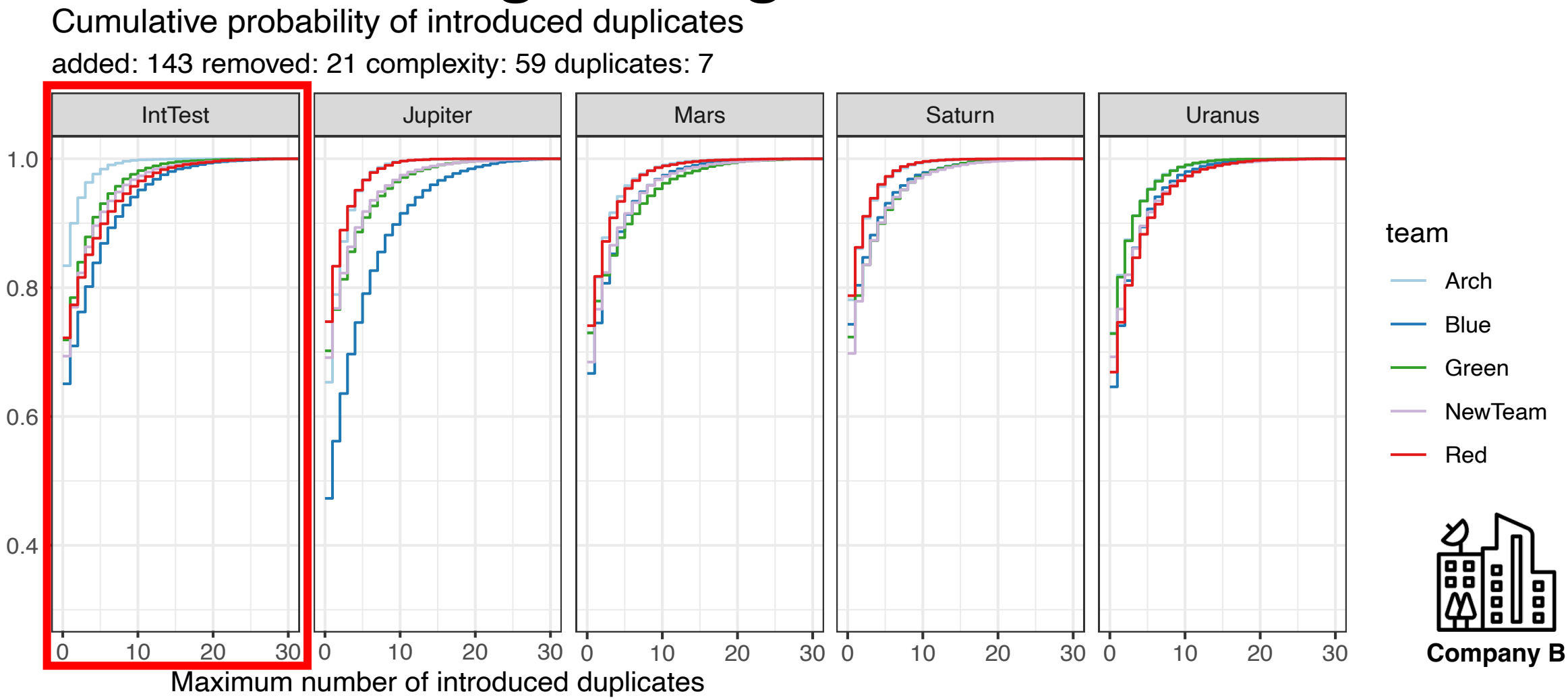
In a nutshell…

High probability of
Introducing clones

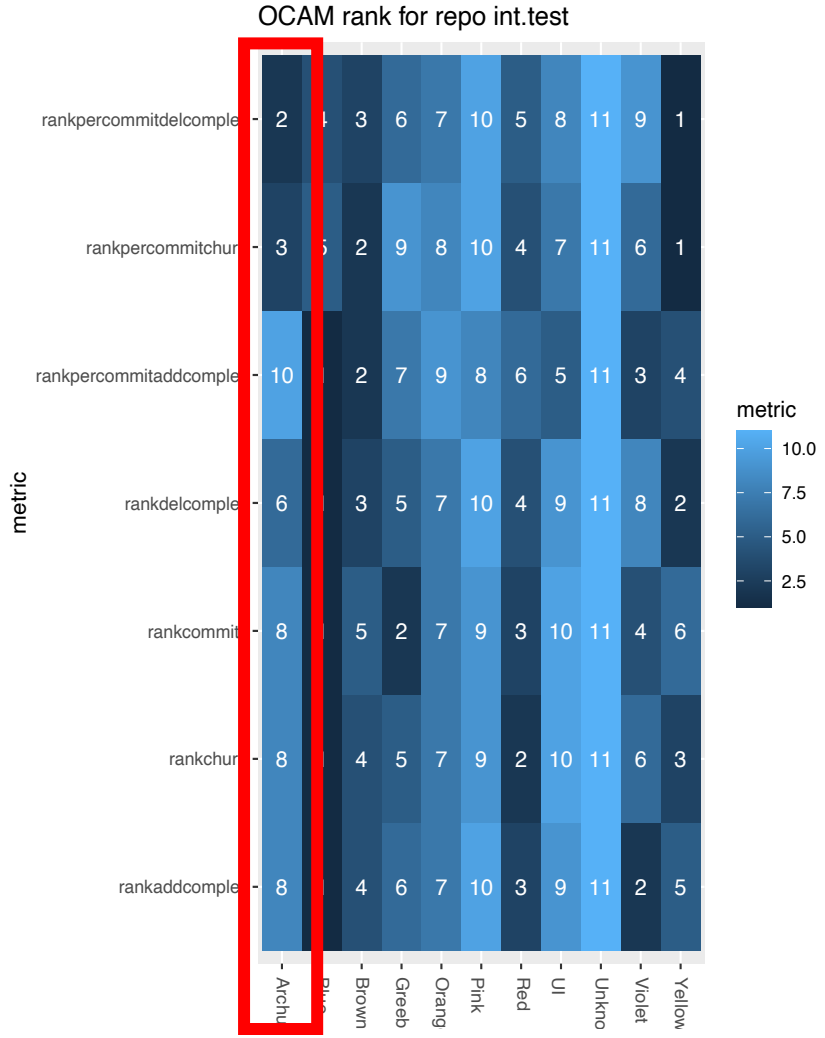


Company B

Results: average change

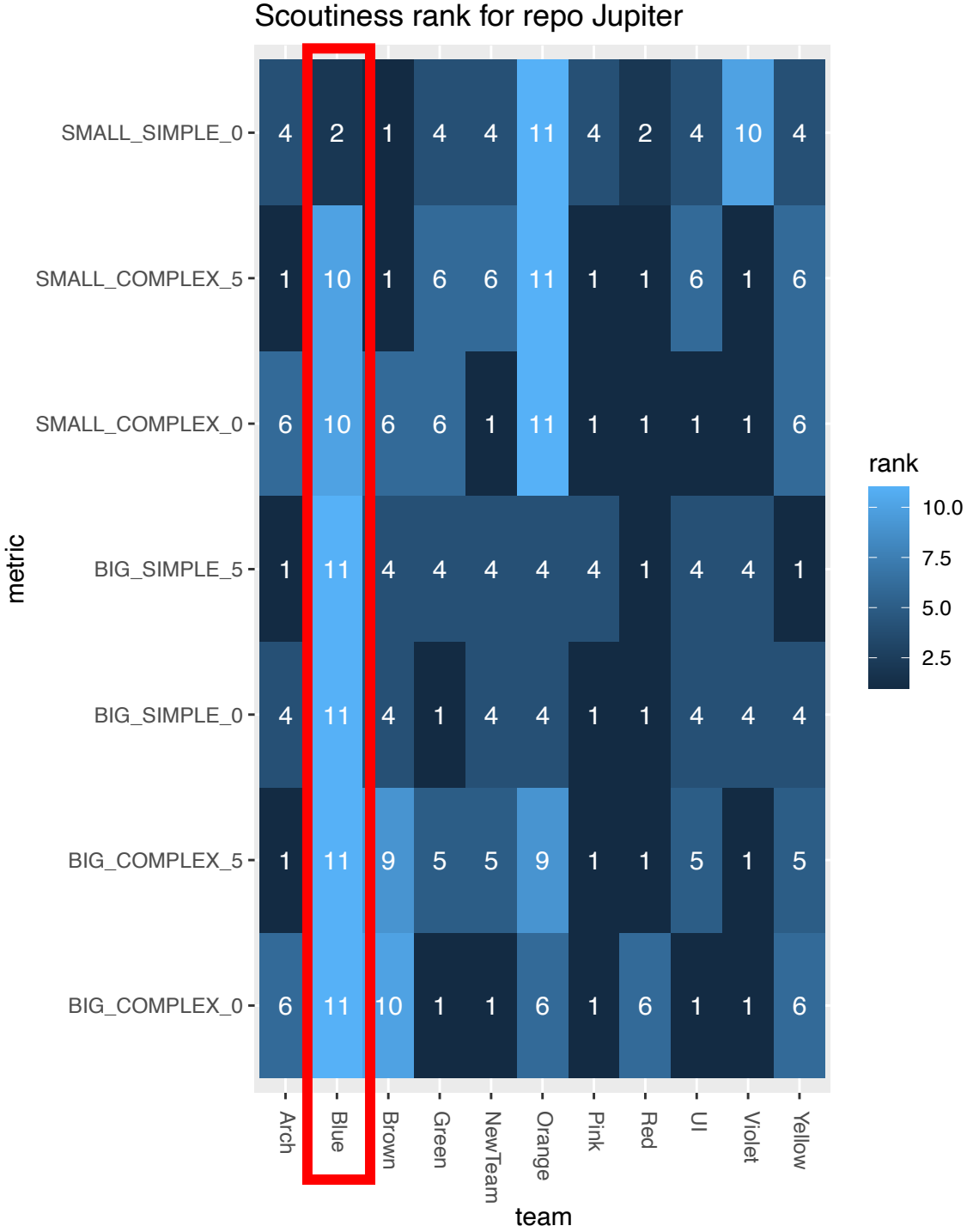


Contributions to Uranus



Company B

”Scoutiness” ranking



Company B

So what?

Lessons Learned

Aligning Architectural and Organisational Structures

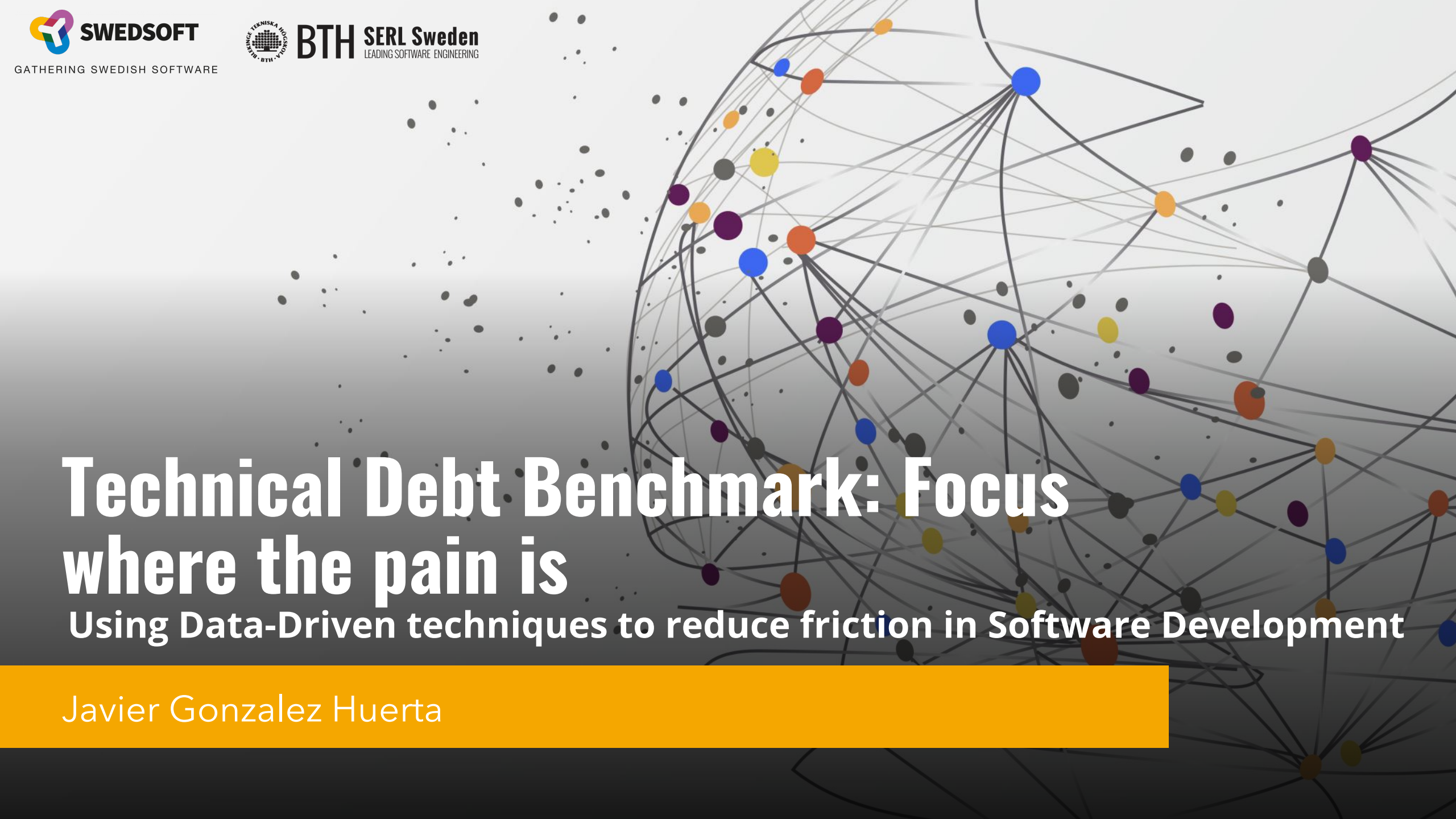
- **Align formal and actual ownership:** make the team who *knows best* responsible for the quality of that component.
- **Raise awareness of dependencies among teams:** Frequently, teams do not know the teams with which they have strong dependencies (Task & Technical)
- **Solutions are often not technical:** sometimes is not about refactoring the architecture, is about changing teams and responsibilities

Use Your Data!

- Data coming from the tools used in the daily work can assist in decisions to handle TD, focusing where the pain is
- Focus on what causes harm to the process
- Reducing communication overhead and coordination will allow teams focus on what matters
- If teams have a good control on what they are responsible of, will help them prioritise TD repayment

SWEDSOFT
GATHERING SWEDISH SOFTWARE

BTH SERL Sweden
LEADING SOFTWARE ENGINEERING



Technical Debt Benchmark: Focus where the pain is

Using Data-Driven techniques to reduce friction in Software Development

Javier Gonzalez Huerta