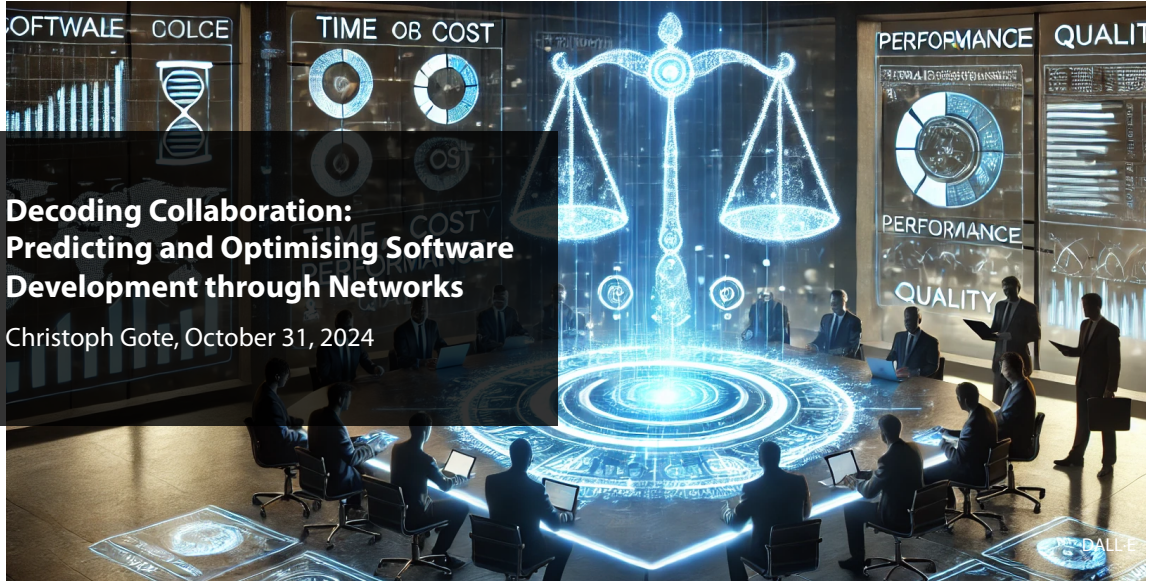
ETH zürich

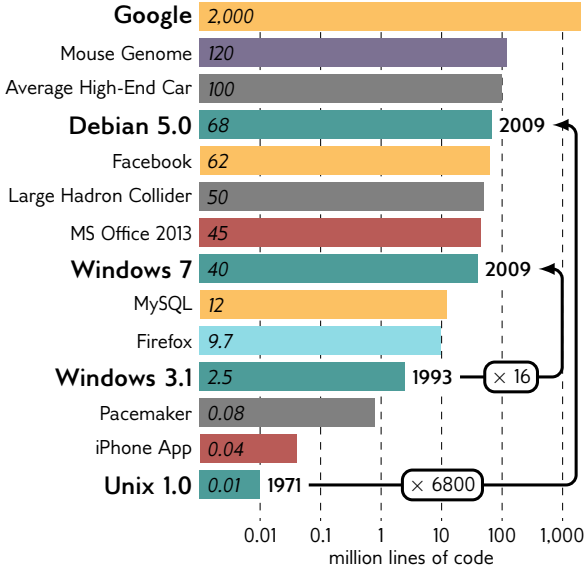
Chair of Systems Design
www.sg.ethz.ch



Decoding Collaboration: Predicting and Optimising Software Development through Networks

Christoph Gote, October 31, 2024

Introduction



Main question: How can

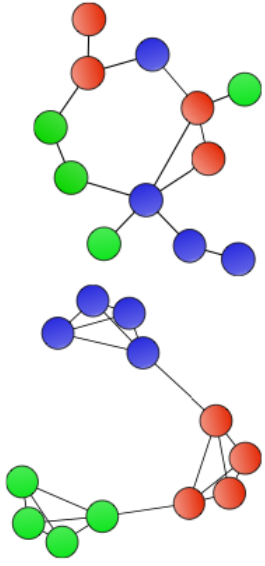
- data-driven modelling
- network science
- complex systems theory
- SG's interdisciplinary approach

benefit software development?

- Own plot; data source: informationisbeautiful.net

DALL·E

Modularity makes code easier to understand, maintain, and test.



Ask yourselves:
Which would you want to work in?
Which is more expandable?
Which is more maintainable?

Modularity
Degree of decoupling can be quantified via analysis of cluster structures in dependency networks.

- Zanetti and Schweitzer (2012): A Network Perspective on Software Modularity

Is this always the case? Should we minimise modularity?

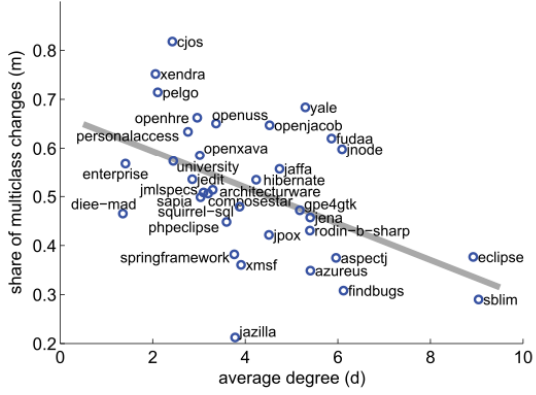
What is a dependency?

+ The result of code reuse
− A potential trigger for downstream co-changes

Dependencies

Dependencies can both enhance and hinder maintainability.

However, **most dependencies are never involved in propagated changes**.



- Geipel and Schweitzer (2009): Software Change Dynamics: Evidence From 35 Java Projects
- Geipel (2012): Modularity, Dependence and Change

Software evolves over time, affecting it's structure.

Over time, developers:

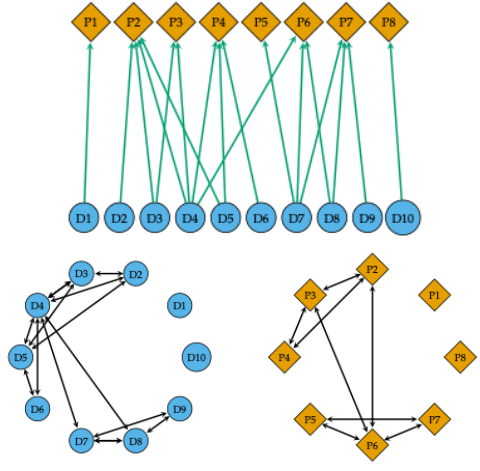
- add new features, fix bugs
- join or leave the project
- refactor code

Structure

A software's structure changes over time.

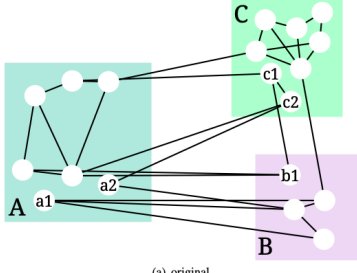
Network growth models

can help us understand and predict the evolution of software.



- Geipel, Tessone, and Schweitzer (2009): A Complementary View on the Growth of Directory Trees
- Schweitzer, Nanumyan, Tessone, and Xia (2014): How Do OSS Projects Change in Number and Size?

A software can be remodularised to restore modularity.



(a) original

(b) refactored

Software **modularity deteriorates over time** and needs to be restored for long-term sustainability.

Remodularisation

Through network and cluster analysis, of **dependencies and co-changes**, we can identify and automate remodularisation.

- Zanetti, Tessone, Scholtes, and Schweitzer (2014): Automated Software Remodularisation Based on Move Refactoring

DALL·E

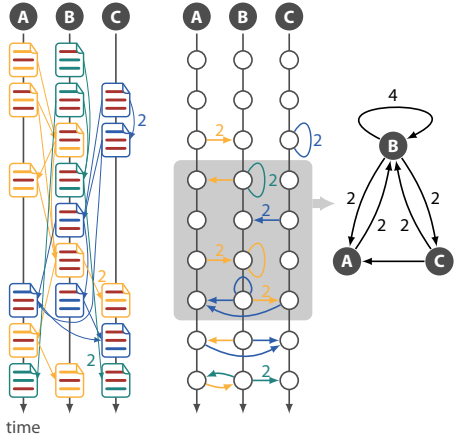
Software is created by teams of people!

This raises many questions:

- How does team structure affect productivity?
- How does social structure evolve?
- Can we use social ties to predict software quality?
- What is the human risk in software development?

Require **framework to analyse social structures** in software development.

Developed two OSS tools: `git2net` and `gambit`.



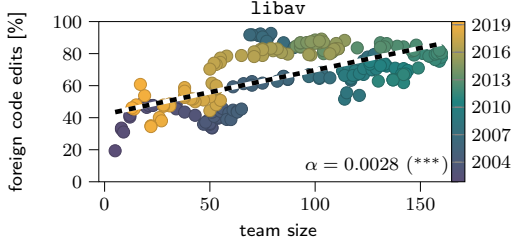
- Gote, Scholtes, and Schweitzer (2019): git2net – Mining Time-Stamped Co-Editing Networks from Large git Repositories
- Gote and Zingg (2021): gambit – An Open Source Name Disambiguation Tool for Version Control Systems

Coordination requirements tend to increase with team size.

Treating **OSS communities as social networks** allows us to measure collaboration dynamics and evolution.

We find that:

- As teams grow, **more foreign code is edited.**
- **Editing foreign code takes longer.**

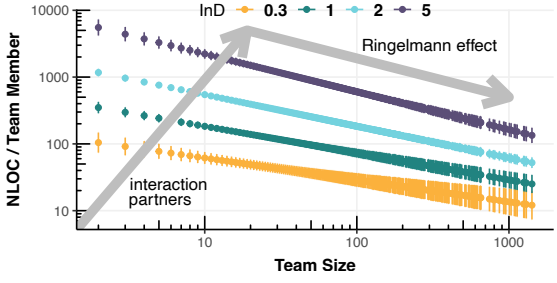


Coordination

Overhead increases with team size.

- Zanetti, Sarigöl, Scholtes, Tessone, and Schweitzer (2013): A Quantitative Study of Social Organisation in Open Source Software Communities
- Gote, Scholtes, and Schweitzer (2021): Analysing Time-Stamped Co-Editing Networks in Software Development Teams using git2net

“Social modularity” can mitigate the resulting productivity loss.



Generally, productivity decreases with team size.

Social modularity

Growth dynamics of coordination networks affect the magnitude of the productivity decrease.

Weak ties

Rarely activated coordination ties can be beneficial.

- Scholtes, Mavrodiev, and Schweitzer (2016): From Aristotle to Ringelmann: A Large-Scale Analysis of Team Productivity and Coordination in Open Source Software Projects
- Gote, Mavrodiev, Schweitzer, and Scholtes (2022): Big Data = Big Insights? Operationalising Brooks’ Law in a Massive GitHub Data Set


DALL·E
Individuals

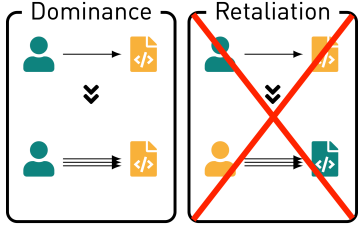
Individuals emerge as key players in many software projects.

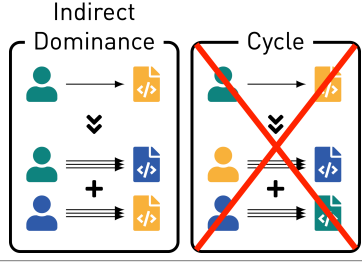
We **observe significant heterogeneity** in individual contributions:

- Team members have different levels of activity
- Team members take on different roles
- Team members act in different ways

Hierarchies

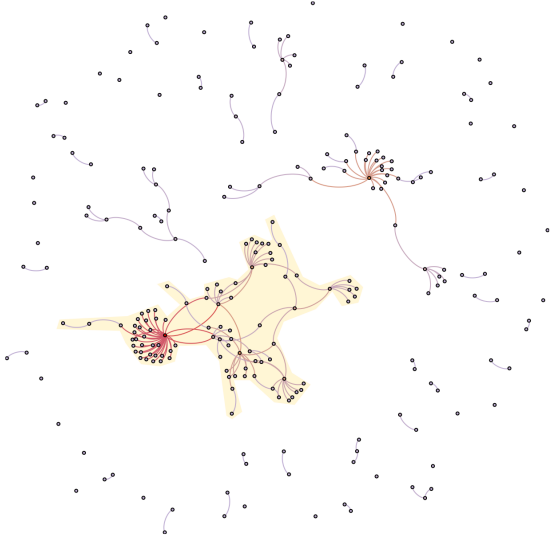
Formation of social hierarchies with the presence of dominance patterns and the absence of retaliation in code editing behaviour.



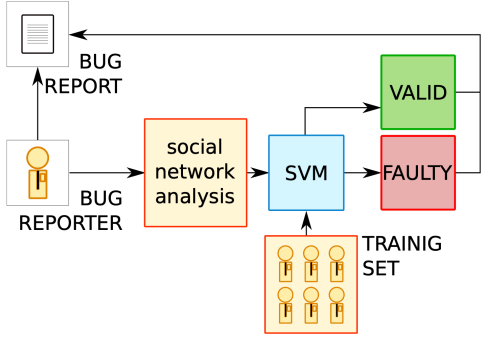


- Geipel, Press, and Schweitzer (2014): Communication in Innovation Communities: An Analysis of 100 Open Source Software Projects
- Brandenberger, Gote, and Schweitzer (in preparation): The Changing Nature of Social Hierarchies

This leads to a highly informative core-periphery structure.

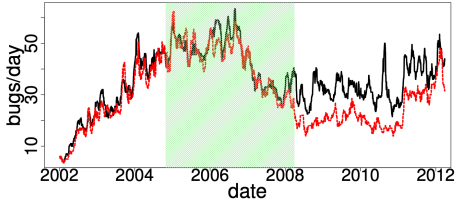


(c) Firefox (Oct. 2003) 241 nodes, 184 links



- Zanetti, Scholtes, Tessone, and Schweitzer (2013): Categorising Bugs with Social Networks: A Case Study on Four Open Source Software Communities

Leaves projects vulnerable to the departure of central contributors.



Case study: Gentoo OSS community

- Central contributor between 2005 and 2008
- Highest bug fixing rate in project lifetime
- Lowest time to resolve new bugs

After departure:

- Significant reorganisation required
- Bug fixing rate drops
- Time to resolve new bugs increases
- Negative emotions within community
- Risk for turnover cascades

- Zanetti, Scholtes, Tessone, and Schweitzer (2013): The Rise and Fall of a Central Contributor: Dynamics of Social Organisation and Performance in the Gentoo Community
- Garcia, Zanetti, and Schweitzer (2013): The Role of Emotions in Contributors Activity: A Case Study of the Gentoo Community

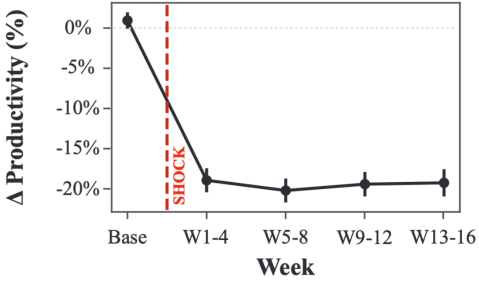
Teams with higher productivity are affected more strongly!

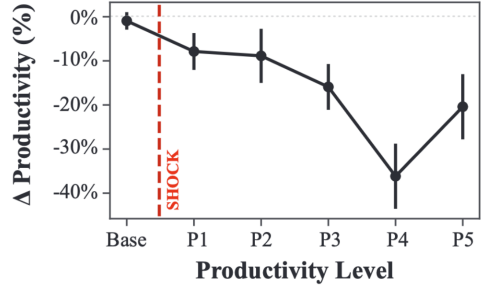
We observe this **sustained impact on productivity** in many OSS projects.

Teams organised to **maximise productivity are most suseptible**.

Tradeoff

Productivity and resilience cannot be maximised simultaneously.



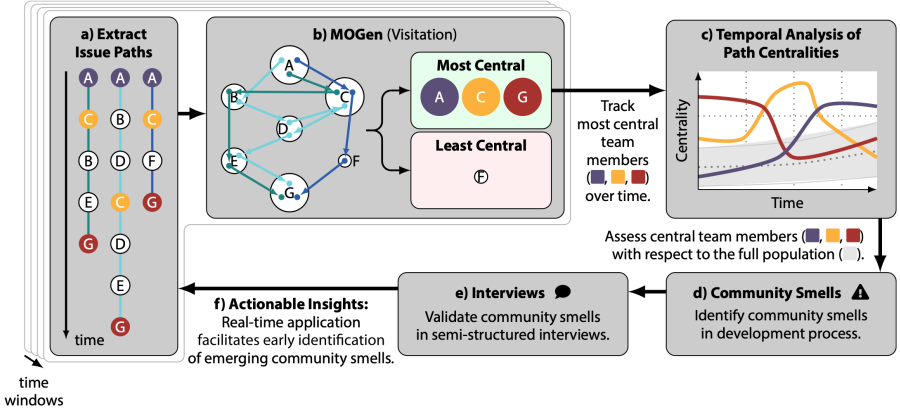


- Russo, Gote, Zingg, Casiraghi, Verginer, and Schweitzer (2024): Shock! Quantifying the Impact of Core Developers' Dropout on the Productivity of OSS Projects


USER INTRFACES
USER INTRFACES
BYSTEM ANCRETCTURE
DALL·E
BACKEND
CLOUD
BACKEND SYSTEMS
CYSTEM ANPHAECTURE
CLS AFBTCCTURE
Application

We can help teams understand and track risks...

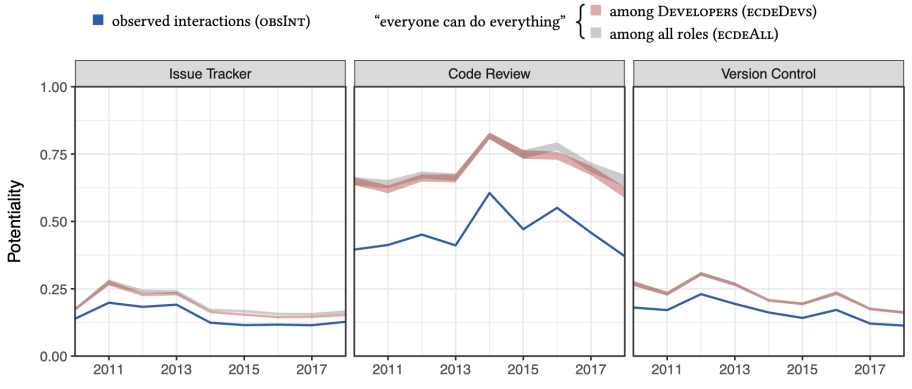
Collaboration with IT security company **genua GmbH**: Validated approach to **track team productivity** and **locate community smells** in real-world application.



- Gote, Perri, Zingg, Casiraghi, Arzig, von Gernler, Schweitzer, and Scholtes (2023): Locating Community Smells in Software Development Processes Using Higher-Order Network Centralities

... and find optimal tradeoff between productivity and resilience.

Developed approach to **detect current** and **assess and compare target team structures** based on their resilience.



Key takeaways from SG's software development research:

Code:

- Minimising co-changes makes projects maintainable and expandable.
- Remodularisation can counter effects of growth and aging.

Teams:

- Coordination overhead increases with team size.
- "Social modularity" can mitigate productivity loss.

Individuals:

- Central contributors can greatly benefit projects.
- But their departure can have severe consequences.

Application:

- Tracking team structures can help teams find optimal tradeoff between productivity and resilience.