ETH zürich

A Quantitative Study of the Ringelmann Effect in Software Development Teams

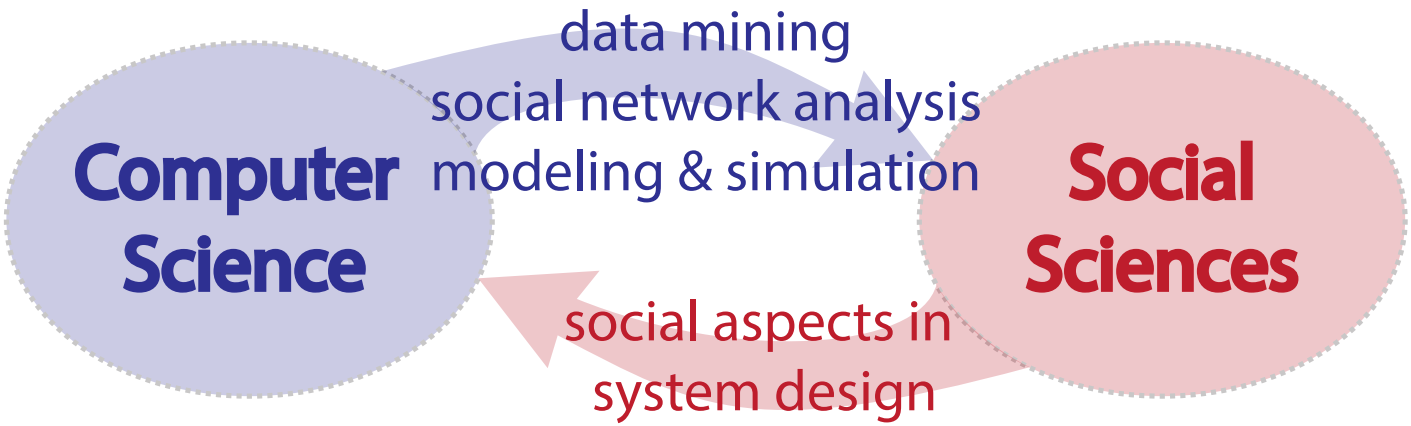
Ingo Scholtes
Chair of Systems Design, ETH Zürich
with Pavlin Mavrodiev and Frank Schweitzer

Computational Social Science

- **data mining & simulation**
  - extract knowledge from (noisy) data on social systems
  - micro- and macro-level simulations of social systems

⇒ new ways to test hypotheses drawn from sociological theory

Computational Social Science



- **data mining & simulation**
  - extract knowledge from (noisy) data on social systems
  - micro- and macro-level simulations of social systems

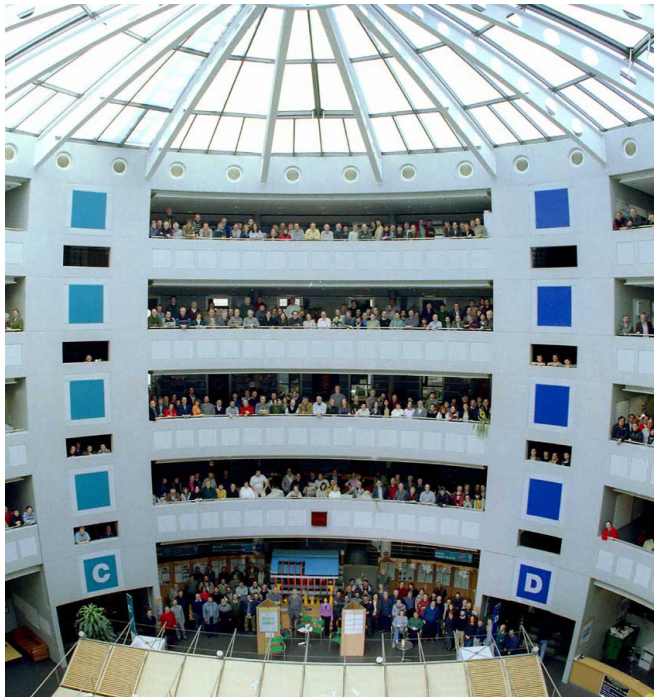
⇒ new ways to test hypotheses drawn from sociological theory

- **social aspects in system design**
  - regain control of socially-embedded (cyber-physical) systems
  - better understand & manage development processes

⇒ improve modeling, design and control of technical systems

Context: Collaborative Software Engineering

```
while len(stack)>0:

    (x,ts) = stack.pop()

    # Get indices of time range within which a time-respecting path via x
    # can possibly be continued
    min_ix = bisect_left(t.activities[x], ts)
    max_ix = bisect_left(t.activities[x], ts+delta)-1

    # For all time-stamps at which x is a source node ...
    for j in range(min_ix, max_ix+1):
        time = t.activities[x][j]

        # For all edges starting at node x at this time
        for e in t.sources[time][x]:

            # We found a new node on a time-respecting path
            new_node = (e[1], time+1)

            # This node can again continue time-respecting paths
            # The set will take care that no duplicates are recorded
            stack.add( new_node )

            # Check whether we found a time-respecting path shorter than t
            if D[name_map[v], name_map[e[1]]] > D[name_map[v], name_map[e[

                # In this case we update the distance matrix
                D[name_map[v], name_map[e[1]]] = D[name_map[v], name_map[e

                if collect_paths == True:
                    # Delete any previous shortest paths
                    Paths[v][e[1]] = []
```



technical aspects: *programming language, architecture, code complexity, support tools...*

social aspects: *development process, team composition, communication, coordination, ...*

Challenge: Understanding team productivity

- **software economics:** how productive are software development teams?

> **"Adding manpower to a late software project makes it later."**
>
> *Fred Brooks (1975)*

Fred Brooks

image: CC-BY-SA SD&M

Challenge: Understanding team productivity

- **software economics:** how productive are software development teams?

> **"Adding manpower to a late software project makes it later."**
>
> *Fred Brooks (1975)*

- **Ringelmann effect** in social psychology: **larger teams** are **less productive**

Maximilien Ringelmann (1861-1931)

image: CC-by-SA Bart Derksen

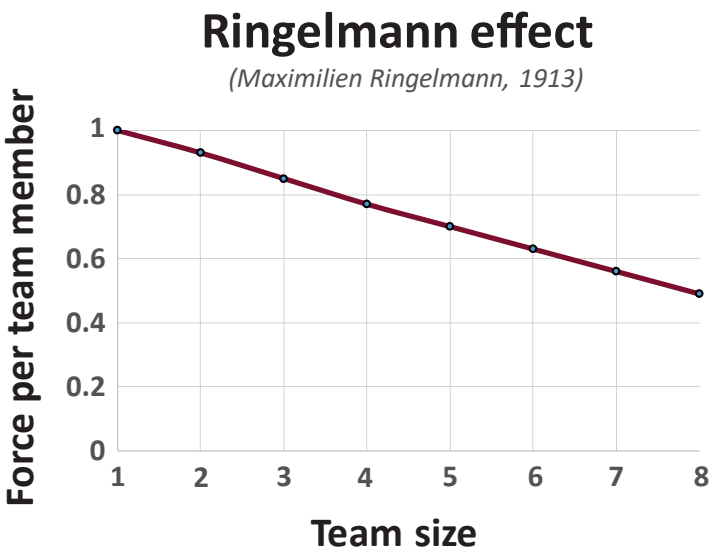
Challenge: Understanding team productivity

- **software economics:** how productive are software development teams?

> **"Adding manpower to a late software project makes it later."**
>
> *Fred Brooks (1975)*

- **Ringelmann effect** in social psychology: **larger teams** are **less productive**
  1. motivational factors: "social loafing"
  2. overhead of coordination

Maximilien Ringelmann
(1861-1931)



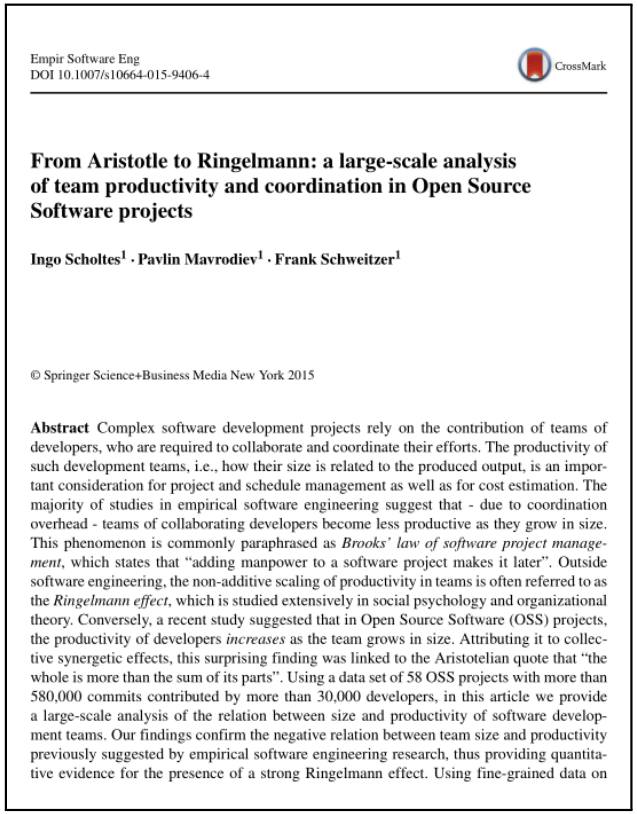
Challenge: Understanding team productivity

- **software economics:** how productive are software development teams?

> **"Adding manpower to a late software project makes it later."**
>
> *Fred Brooks (1975)*

- **Ringelmann effect** in social psychology: **larger teams** are **less productive**
  1. motivational factors: "social loafing"
  2. overhead of coordination

- **research questions:**
  1. does the Ringelmann effect apply to Open Source communities?
  2. can we detect the underlying mechanisms?



Empir Software Eng
DOI 10.1007/s10664-015-9406-4

From Aristotle to Ringelmann: a large-scale analysis of team productivity and coordination in Open Source Software projects

Ingo Scholtes[1] · Pavlin Mavrodiev[1] · Frank Schweitzer[1]

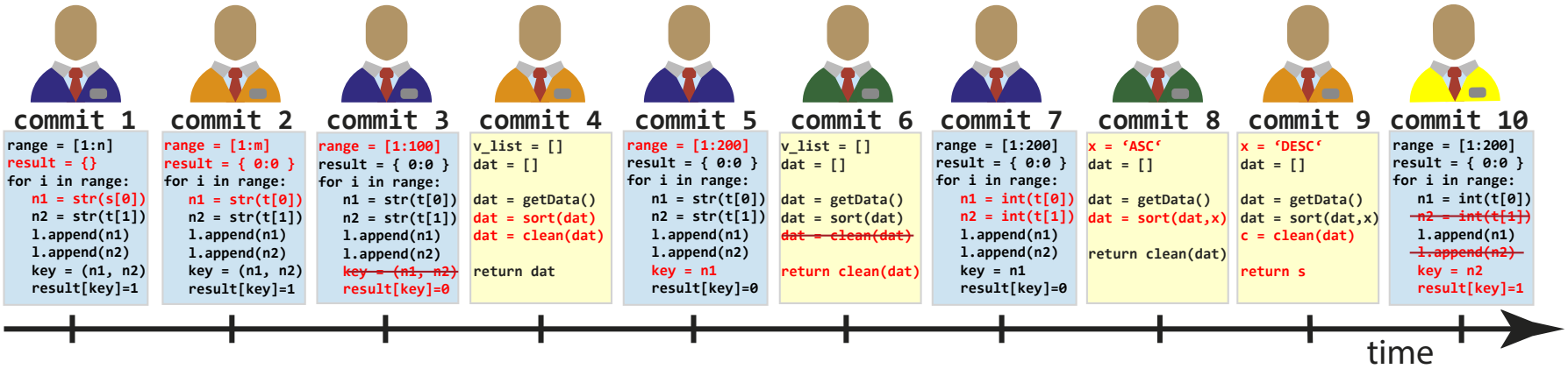
Abstract Complex software development projects rely on the contribution of teams of developers, who are required to collaborate and coordinate their efforts. The productivity of such development teams, i.e., how their size is related to the produced output, is an important consideration for project and schedule management as well as for cost estimation. The majority of studies in empirical software engineering suggest that - due to coordination overhead - teams of collaborating developers become less productive as they grow in size. This phenomenon is commonly paraphrased as *Brooks' law of software project management*, which states that "adding manpower to a software project makes it later". Outside software engineering, the non-additive scaling of productivity in teams is often referred to as the *Ringelmann effect*, which is studied extensively in social psychology and organizational theory. Conversely, a recent study suggested that in Open Source Software (OSS) projects, the productivity of developers *increases* as the team grows in size. Attributing it to collective synergetic effects, this surprising finding was linked to the Aristotelian quote that "the whole is more than the sum of its parts". Using a data set of 58 OSS projects with more than 580,000 commits contributed by more than 30,000 developers, in this article we provide a large-scale analysis of the relation between size and productivity of software development teams. Our findings confirm the negative relation between team size and productivity previously suggested by empirical software engineering research, thus providing quantitative evidence for the presence of a strong Ringelmann effect. Using fine-grained data on

Empirical Software Engineering, Vol. 21, No. 2, April 2016

Data Science approach: commit log analysis

data: GHTorrent **gitHub dump** (~1.5 TB)

identified **58 most active OpenSource projects**
- 580,000 commits
- 30,000 developers
- > 10 years of commit history

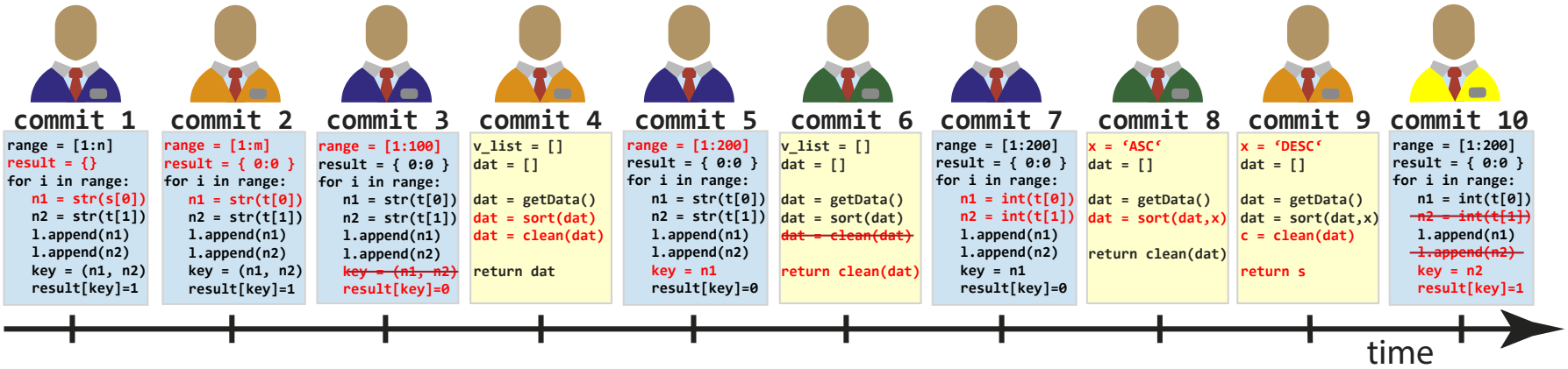


Data Science approach: commit log analysis

data: GHTorrent **gitHub dump** (~1.5 TB)

identified **58 most active OpenSource projects**
- 580,000 commits
- 30,000 developers
- > 10 years of commit history

Q1: What is the team?
>> active developers in reasonable time window



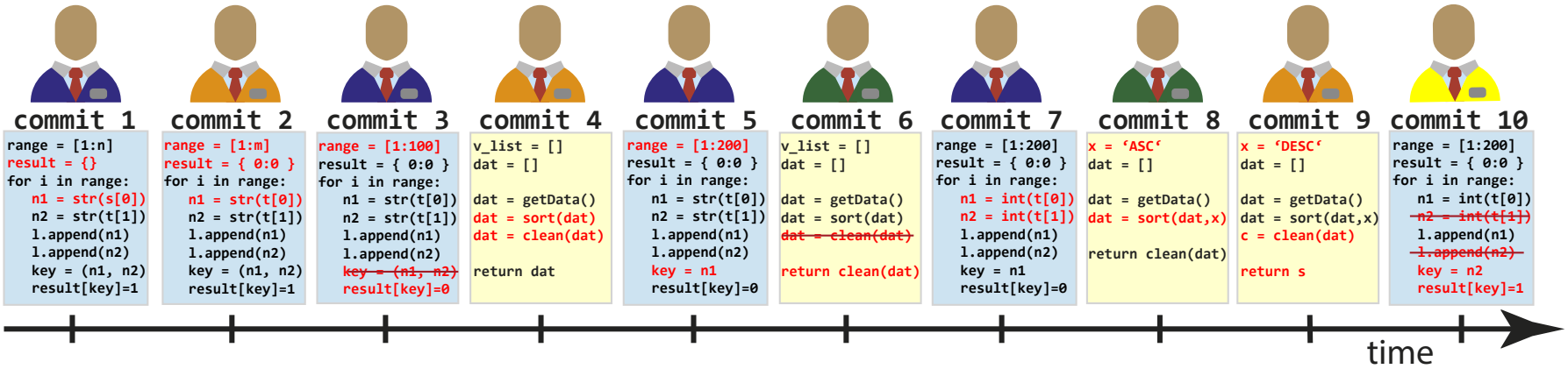
Data Science approach: commit log analysis

data: GHTorrent **gitHub dump** (~1.5 TB)

identified **58 most active OpenSource projects**
- 580,000 commits
- 30,000 developers
- > 10 years of commit history

Q1: What is the team?
>> active developers in reasonable time window

Q2: What is productivity?
>> Levenshtein distance between commits

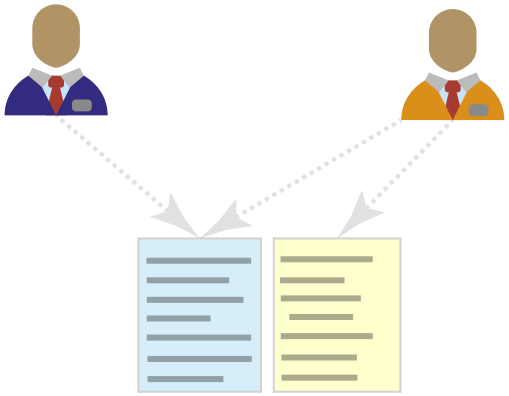


Data Science approach: commit log analysis

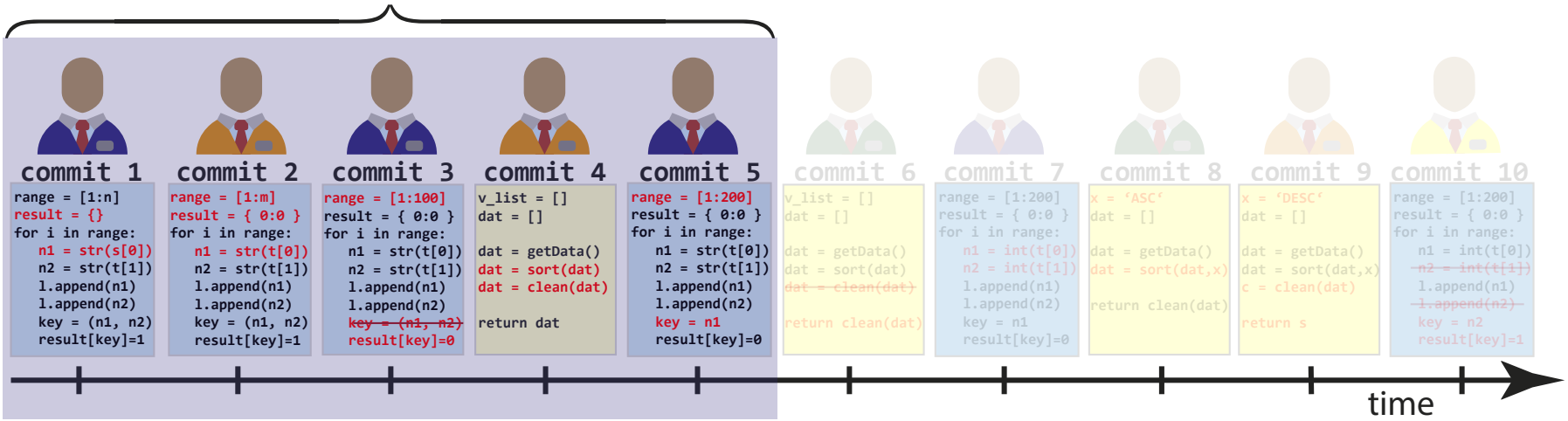
code contributions

	75 characters
	89 characters

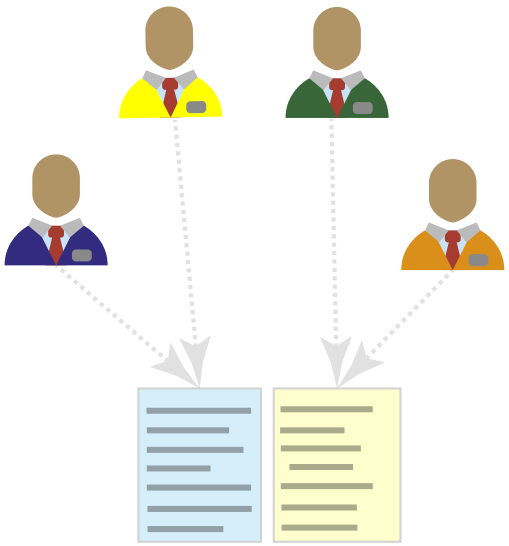
team size: 2
mean contribution: 82



time window 1



Data Science approach: commit log analysis



code contributions

	45 characters
	17 characters
	29 characters
	16 characters

team size: 4

mean contribution: 26.75

time window 2

commit 1
```
range = [1:n]
result = {}
for i in range:
   n1 = str(s[0])
   n2 = str(t[1])
   l.append(n1)
   l.append(n2)
   key = (n1, n2)
   result[key]=1
```

commit 2
```
range = [1:m]
result = { 0:0 }
for i in range:
   n1 = str(t[0])
   n2 = str(t[1])
   l.append(n1)
   l.append(n2)
   key = (n1, n2)
   result[key]=1
```

commit 3
```
range = [1:100]
result = { 0:0 }
for i in range:
   n1 = str(t[0])
   n2 = str(t[1])
   l.append(n1)
   l.append(n2)
   key = (n1, n2)
   result[key]=0
```

commit 4
```
v_list = []
dat = []

dat = getData()
dat = sort(dat)
dat = clean(dat)

return dat
```

commit 5
```
range = [1:200]
result = { 0:0 }
for i in range:
   n1 = str(t[0])
   n2 = str(t[1])
   l.append(n1)
   l.append(n2)
   key = n1
   result[key]=0
```

commit 6
```
v_list = []
dat = []

dat = getData()
dat = sort(dat)
dat = clean(dat)

return clean(dat)
```

commit 7
```
range = [1:200]
result = { 0:0 }
for i in range:
   n1 = int(t[0])
   n2 = int(t[1])
   l.append(n1)
   l.append(n2)
   key = n1
   result[key]=0
```

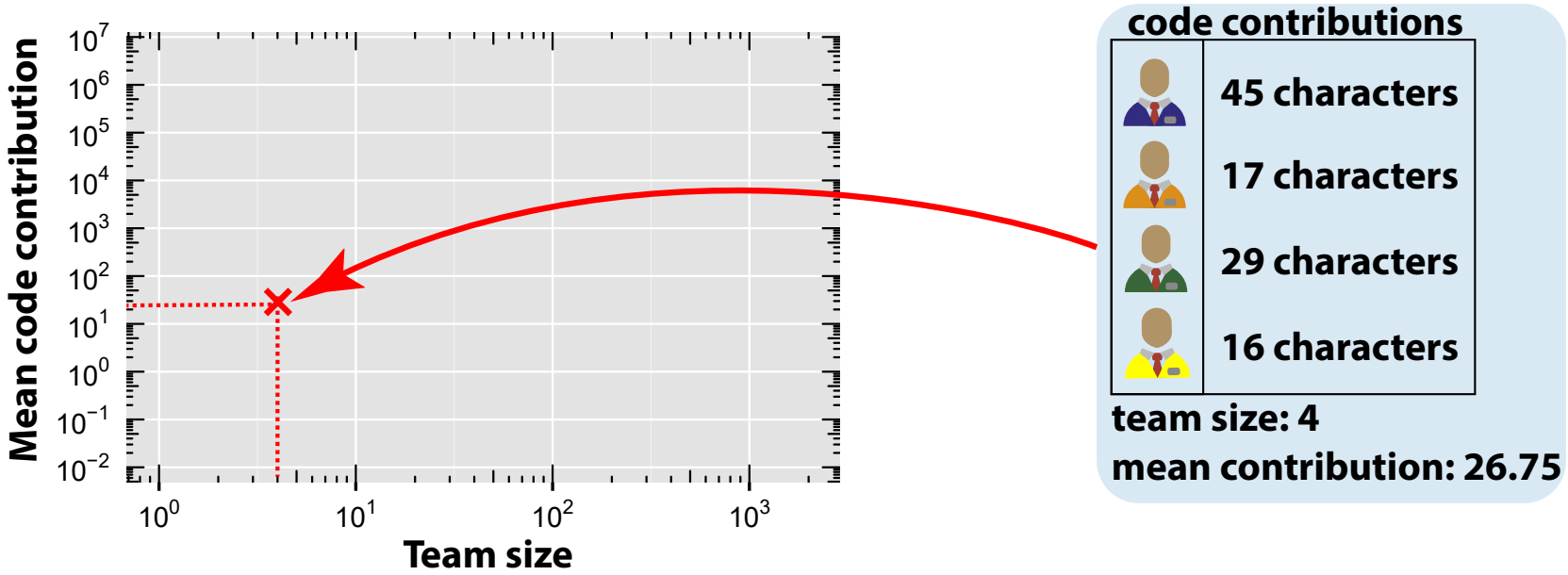
commit 8
```
x = 'ASC'
dat = []

dat = getData()
dat = sort(dat,x)

return clean(dat)
```

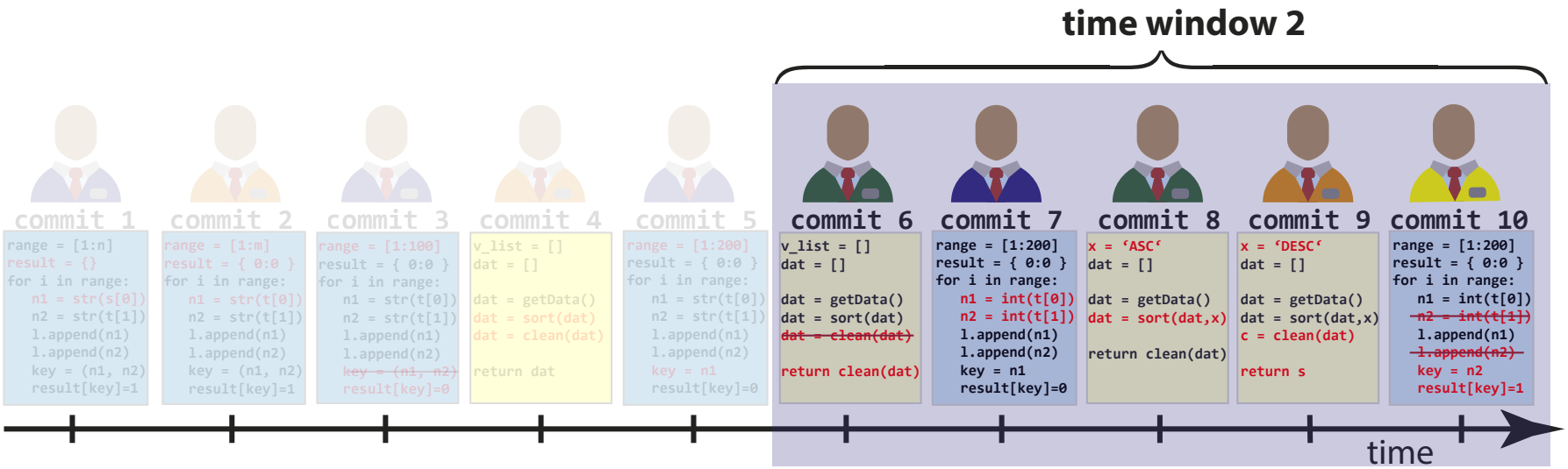
commit 9
```
x = 'DESC'
dat = []

dat = getData()
dat = sort(dat,x)
c = clean(dat)

return s
```

commit 10
```
range = [1:200]
result = { 0:0 }
for i in range:
   n1 = int(t[0])
   n2 = int(t[1])
   l.append(n1)
   l.append(n2)
   key = n2
   result[key]=1
```

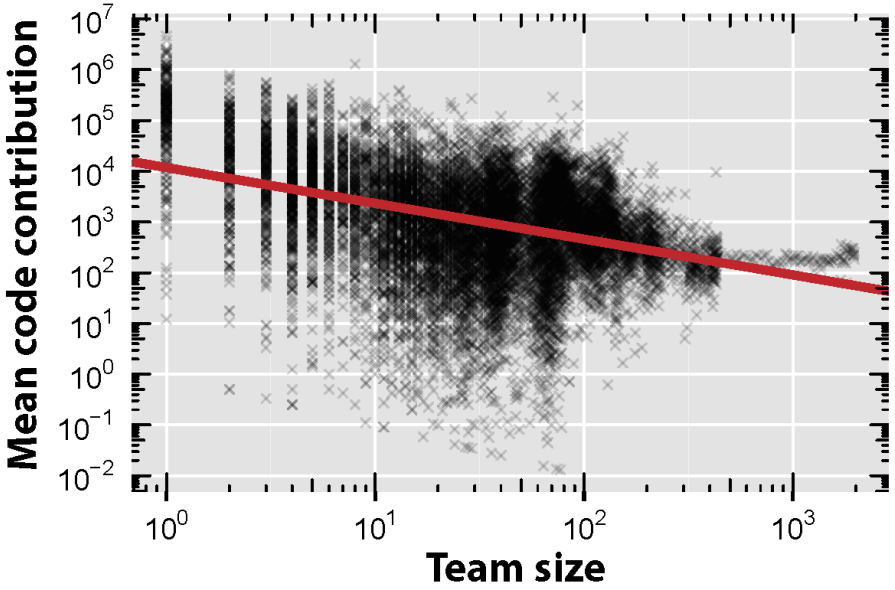
time

Data Science approach: commit log analysis





Data Science approach: commit log analysis

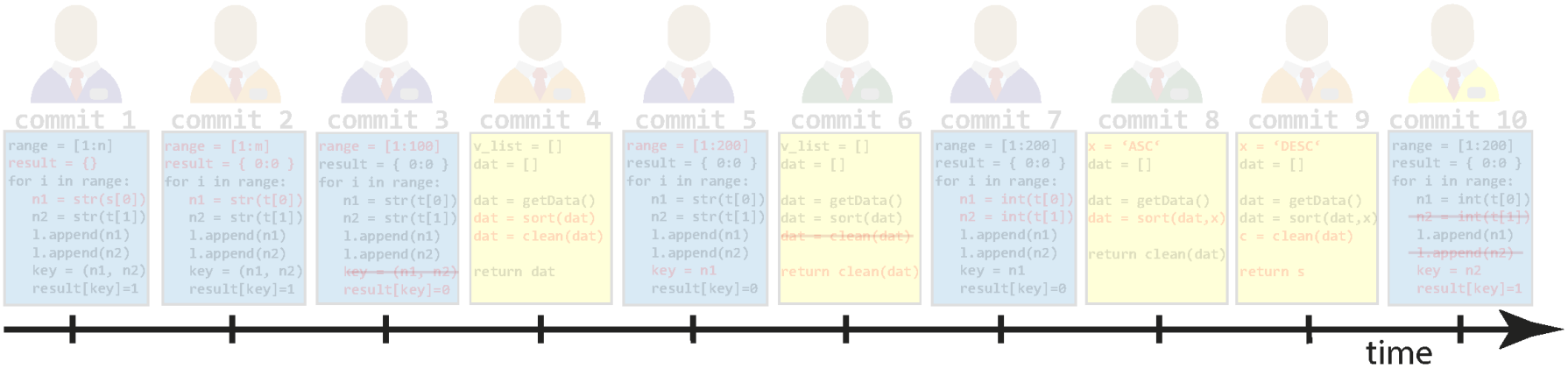


analysis across 58 projects

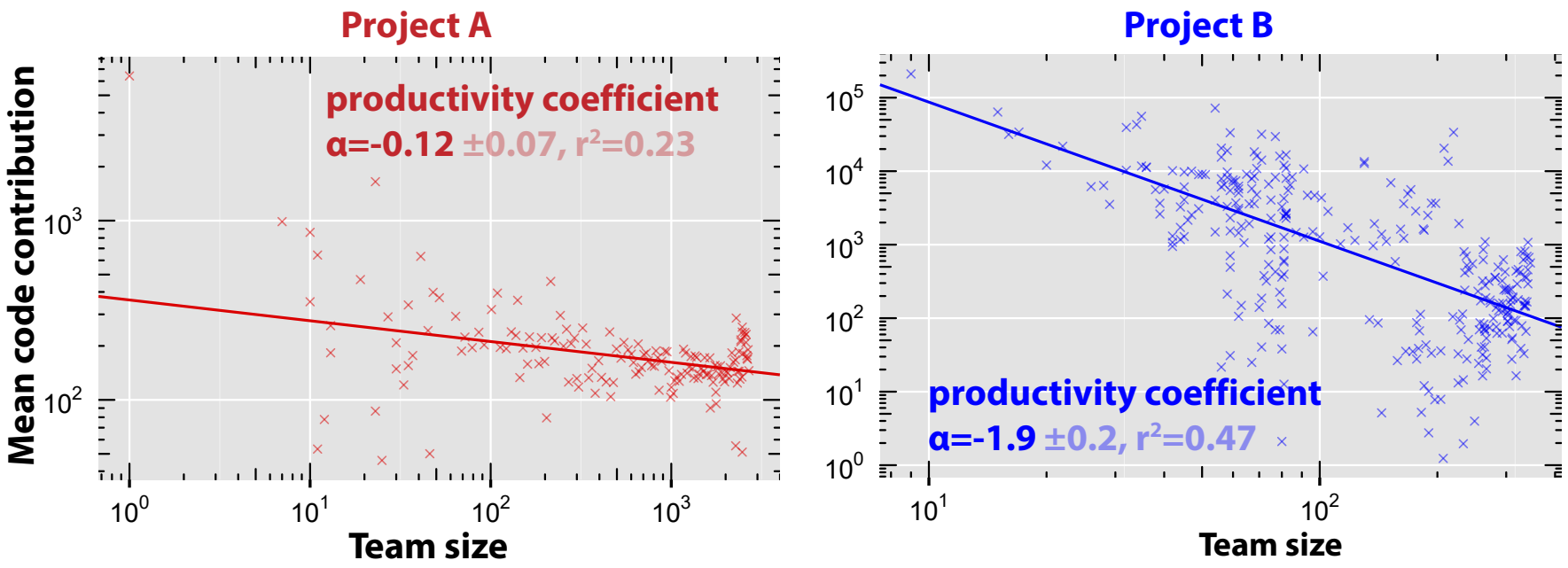
robust log-linear regression yields **negative relation**

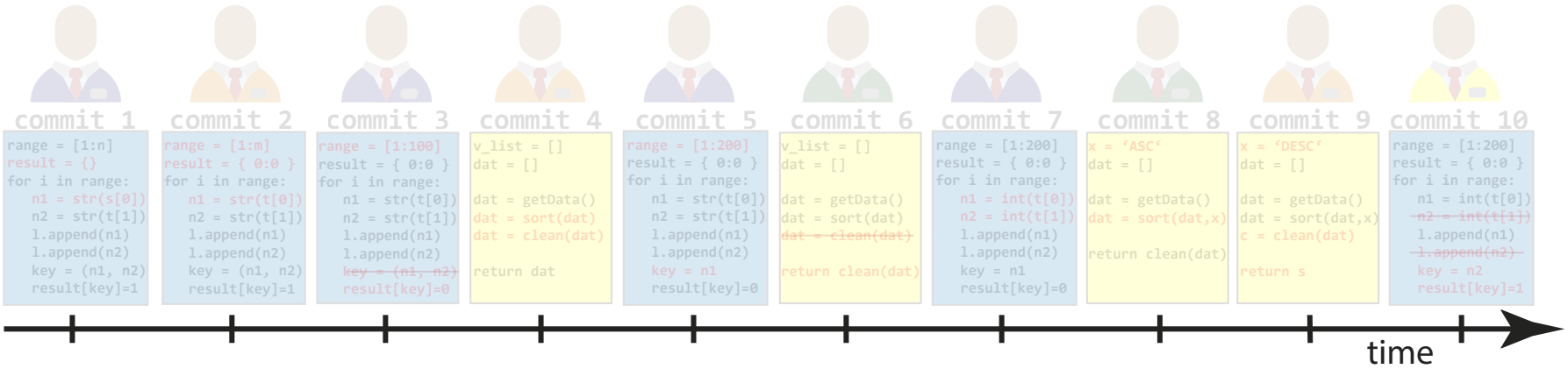
α = -0.86 ± 0.02, r^2=0.25

we **confirm** and **quantify** the **Ringelmann effect** in Open Source communities!

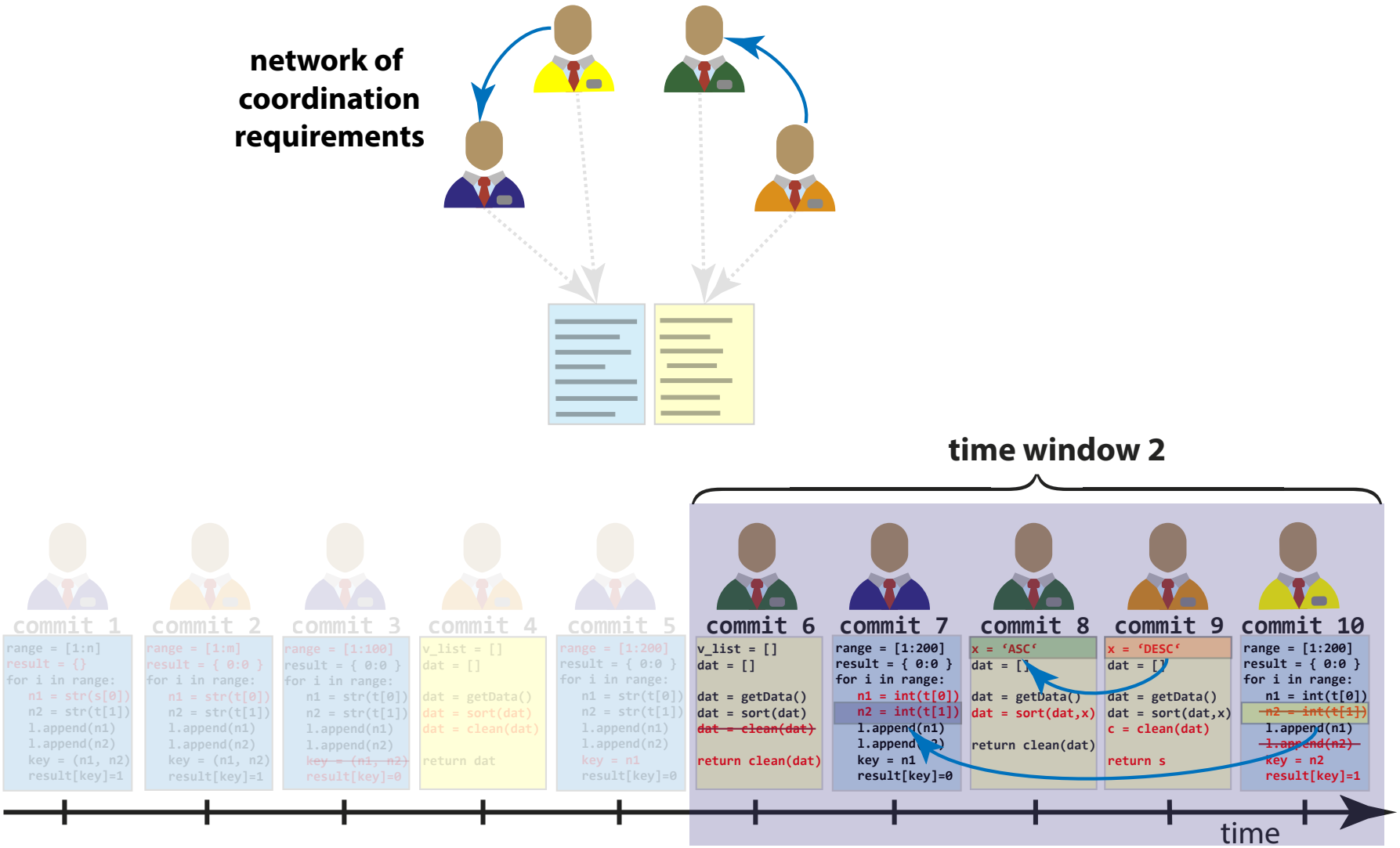


Data Science approach: commit log analysis

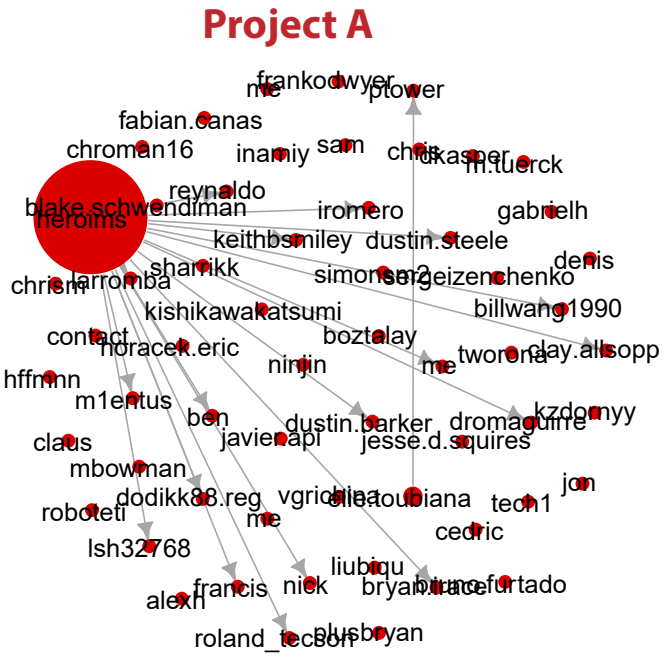


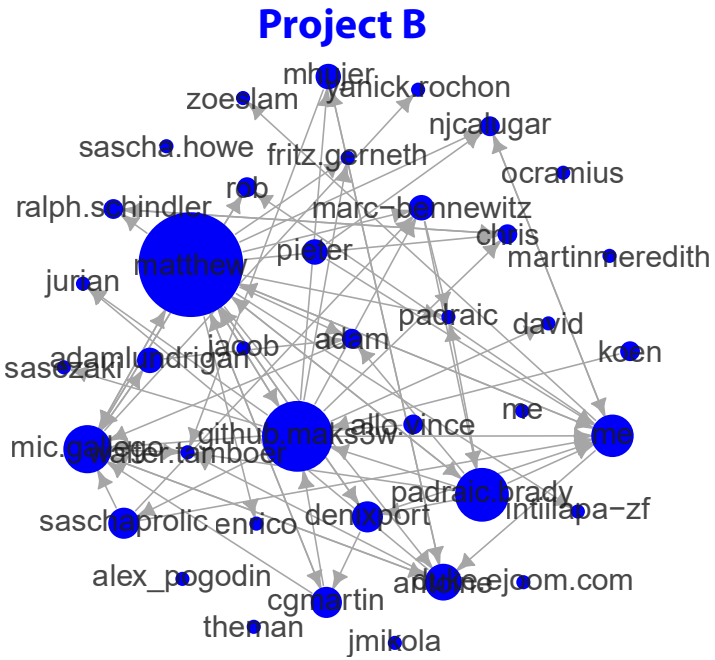


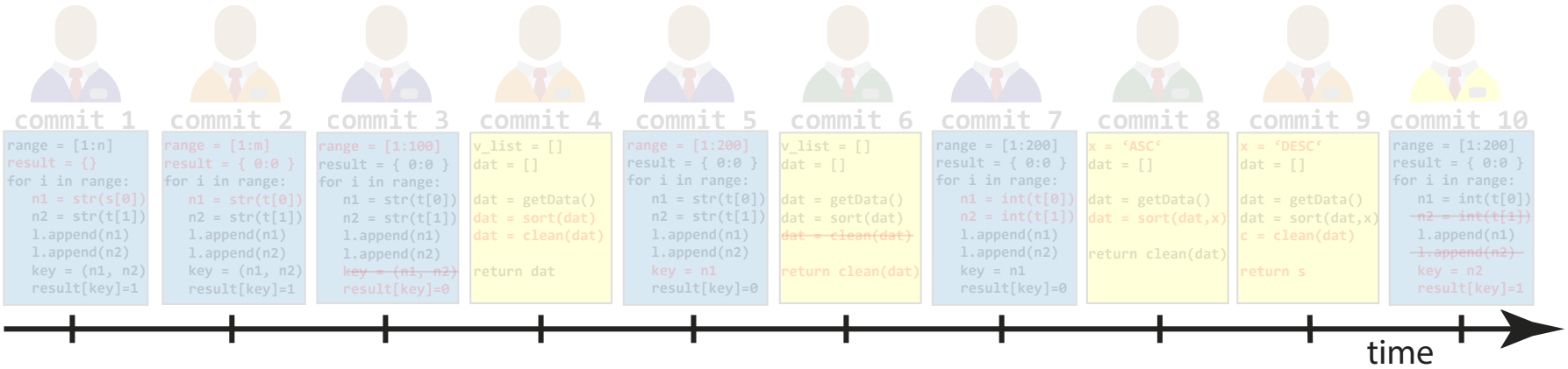
Coordination networks and productivity



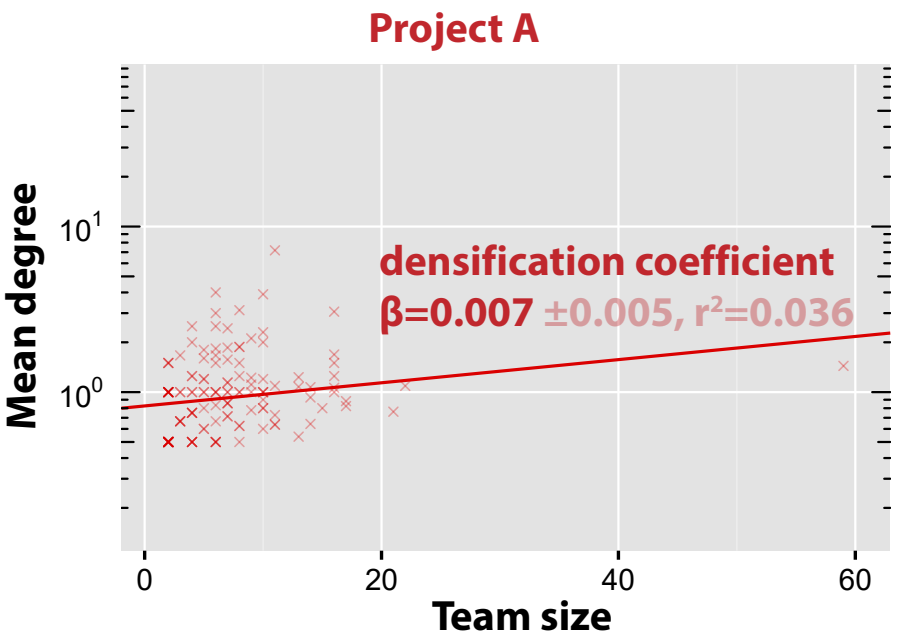
Coordination networks and productivity

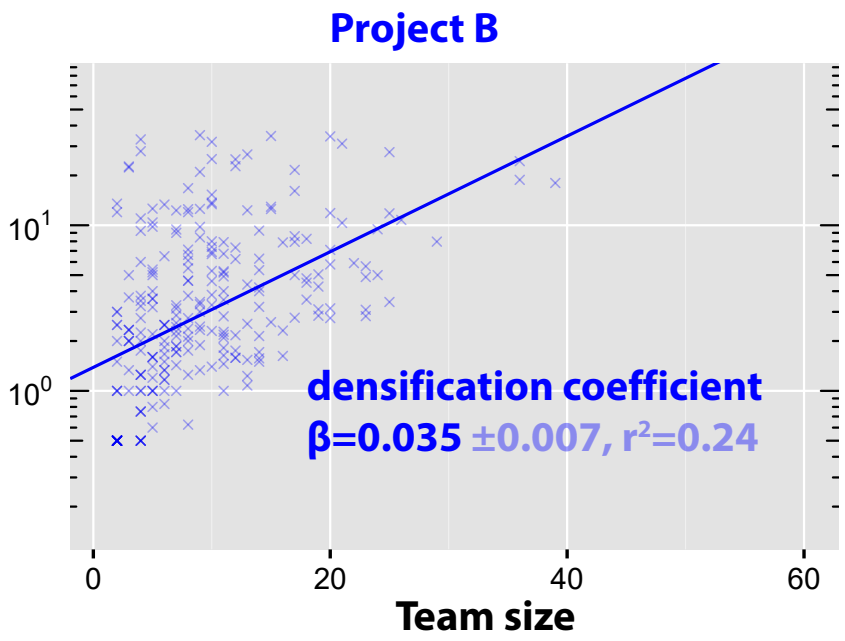


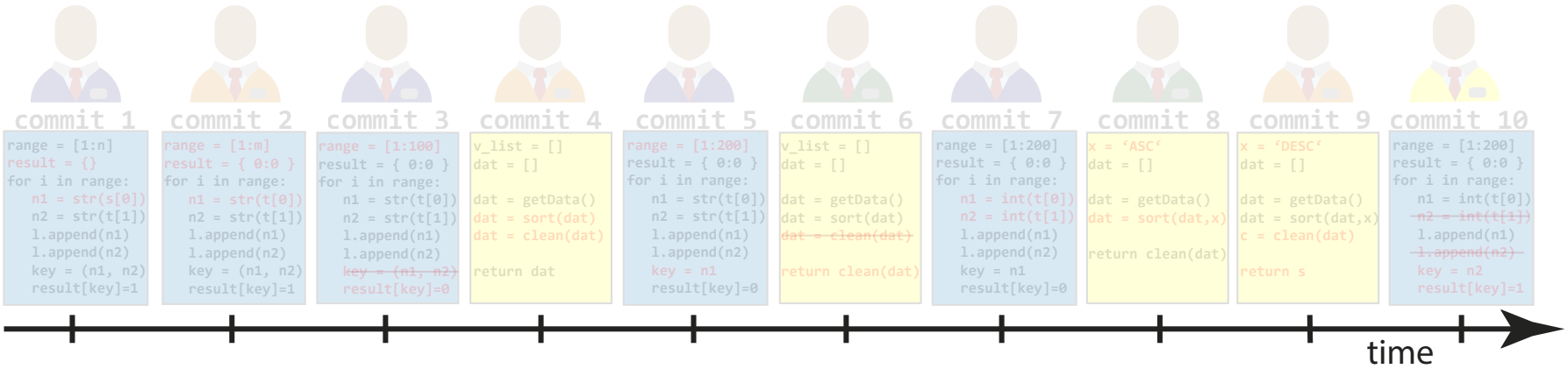




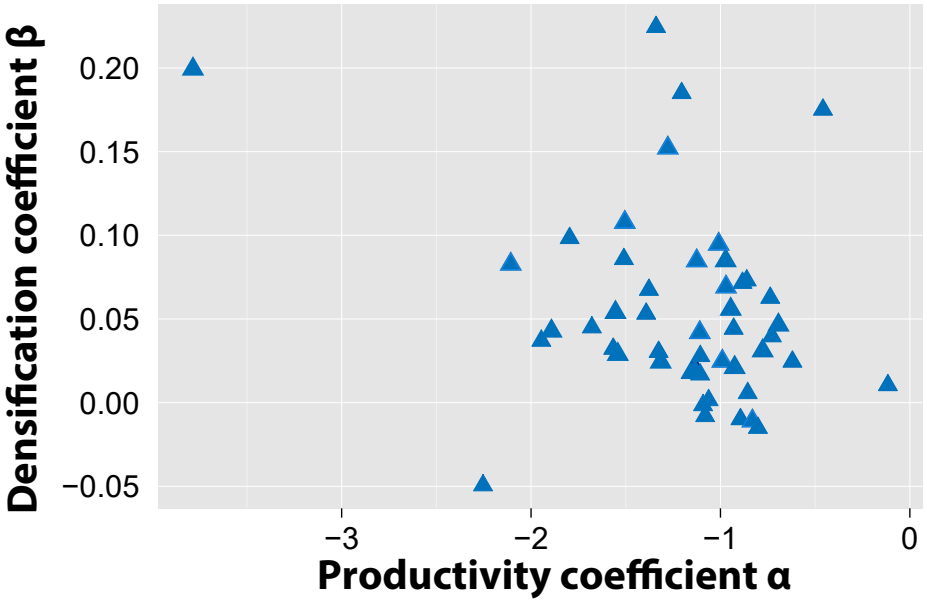
Coordination networks and productivity







Coordination networks and productivity

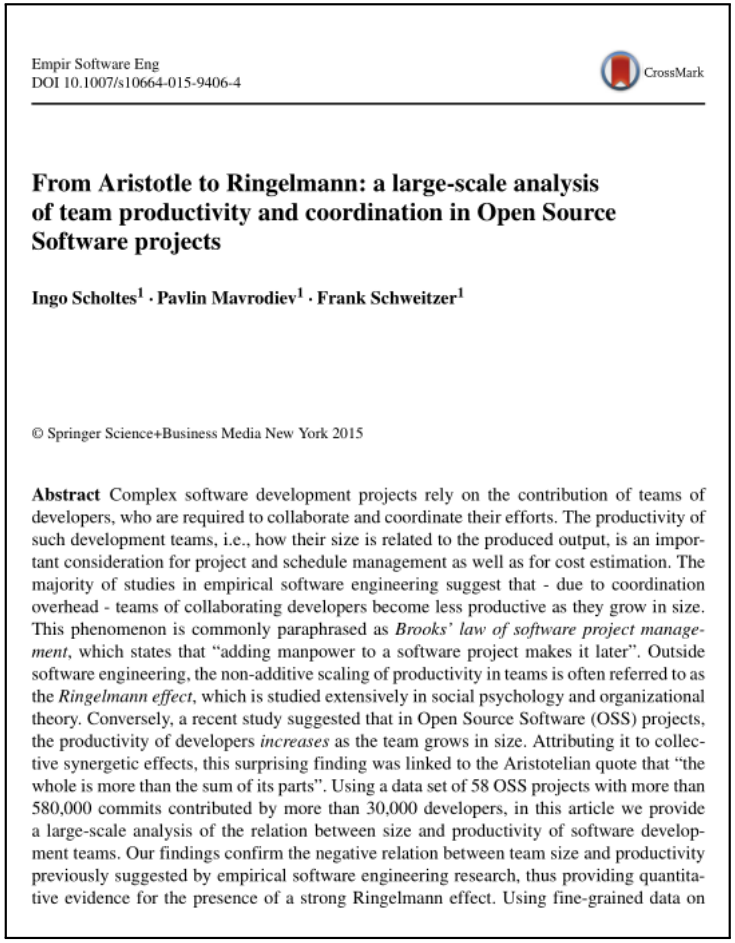


key finding:

statistically significant relation between **densification** of coordination networks and **decrease** in productivity as teams grow in size

Take-away messages

1. we provide quantitative evidence for the **Ringelmann effect** in Open Source software development teams



Empir Software Eng
DOI 10.1007/s10664-015-9406-4

From Aristotle to Ringelmann: a large-scale analysis of team productivity and coordination in Open Source Software projects

Ingo Scholtes[1] · Pavlin Mavrodiev[1] · Frank Schweitzer[1]

Abstract Complex software development projects rely on the contribution of teams of developers, who are required to collaborate and coordinate their efforts. The productivity of such development teams, i.e., how their size is related to the produced output, is an important consideration for project and schedule management as well as for cost estimation. The majority of studies in empirical software engineering suggest that - due to coordination overhead - teams of collaborating developers become less productive as they grow in size. This phenomenon is commonly paraphrased as *Brooks' law of software project management*, which states that "adding manpower to a software project makes it later". Outside software engineering, the non-additive scaling of productivity in teams is often referred to as the *Ringelmann effect*, which is studied extensively in social psychology and organizational theory. Conversely, a recent study suggested that in Open Source Software (OSS) projects, the productivity of developers *increases* as the team grows in size. Attributing it to collective synergetic effects, this surprising finding was linked to the Aristotelian quote that "the whole is more than the sum of its parts". Using a data set of 58 OSS projects with more than 580,000 commits contributed by more than 30,000 developers, in this article we provide a large-scale analysis of the relation between size and productivity of software development teams. Our findings confirm the negative relation between team size and productivity previously suggested by empirical software engineering research, thus providing quantitative evidence for the presence of a strong Ringelmann effect. Using fine-grained data on

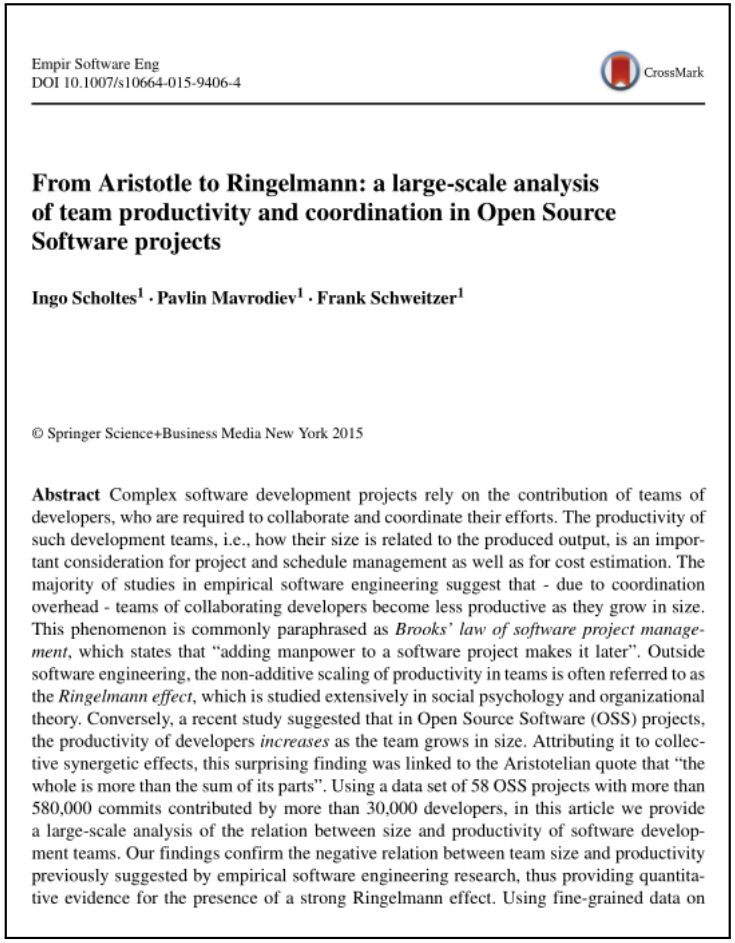
Empirical Software Engineering,

Vol. 21, No. 2, April 2016

Take-away messages

1. we provide quantitative evidence for the **Ringelmann effect** in Open Source software development teams

2. we uncover **coordination overhead** as one mechanism behind the observed scaling behavior of **team productivity**



Empir Software Eng
DOI 10.1007/s10664-015-9406-4

From Aristotle to Ringelmann: a large-scale analysis of team productivity and coordination in Open Source Software projects

Ingo Scholtes[1] · Pavlin Mavrodiev[1] · Frank Schweitzer[1]

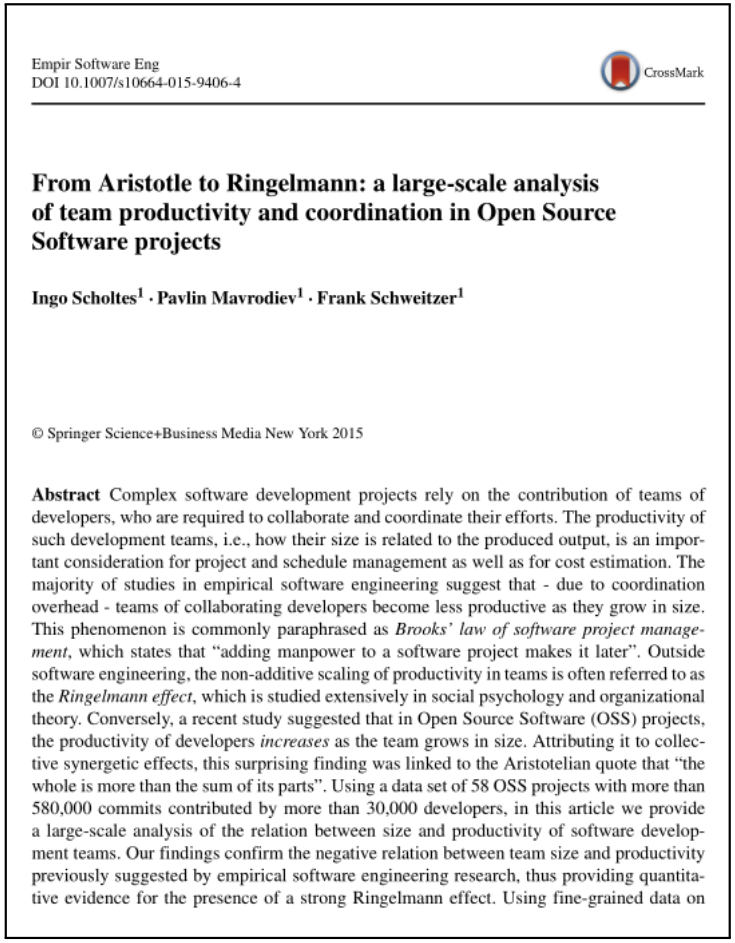
Abstract Complex software development projects rely on the contribution of teams of developers, who are required to collaborate and coordinate their efforts. The productivity of such development teams, i.e., how their size is related to the produced output, is an important consideration for project and schedule management as well as for cost estimation. The majority of studies in empirical software engineering suggest that - due to coordination overhead - teams of collaborating developers become less productive as they grow in size. This phenomenon is commonly paraphrased as *Brooks' law of software project management*, which states that "adding manpower to a software project makes it later". Outside software engineering, the non-additive scaling of productivity in teams is often referred to as the *Ringelmann effect*, which is studied extensively in social psychology and organizational theory. Conversely, a recent study suggested that in Open Source Software (OSS) projects, the productivity of developers *increases* as the team grows in size. Attributing it to collective synergetic effects, this surprising finding was linked to the Aristotelian quote that "the whole is more than the sum of its parts". Using a data set of 58 OSS projects with more than 580,000 commits contributed by more than 30,000 developers, in this article we provide a large-scale analysis of the relation between size and productivity of software development teams. Our findings confirm the negative relation between team size and productivity previously suggested by empirical software engineering research, thus providing quantitative evidence for the presence of a strong Ringelmann effect. Using fine-grained data on

Empirical Software Engineering,

Vol. 21, No. 2, April 2016

Take-away messages

1. we provide quantitative evidence for the **Ringelmann effect** in Open Source software development teams
2. we uncover **coordination overhead** as one mechanism behind the observed scaling behavior of **team productivity**
3. we show that social science theories provide **actionable insights** into software engineering processes



Empir Software Eng
DOI 10.1007/s10664-015-9406-4

From Aristotle to Ringelmann: a large-scale analysis of team productivity and coordination in Open Source Software projects

Ingo Scholtes[1] · Pavlin Mavrodiev[1] · Frank Schweitzer[1]

Abstract Complex software development projects rely on the contribution of teams of developers, who are required to collaborate and coordinate their efforts. The productivity of such development teams, i.e., how their size is related to the produced output, is an important consideration for project and schedule management as well as for cost estimation. The majority of studies in empirical software engineering suggest that - due to coordination overhead - teams of collaborating developers become less productive as they grow in size. This phenomenon is commonly paraphrased as *Brooks' law of software project management*, which states that "adding manpower to a software project makes it later". Outside software engineering, the non-additive scaling of productivity in teams is often referred to as the *Ringelmann effect*, which is studied extensively in social psychology and organizational theory. Conversely, a recent study suggested that in Open Source Software (OSS) projects, the productivity of developers *increases* as the team grows in size. Attributing it to collective synergetic effects, this surprising finding was linked to the Aristotelian quote that "the whole is more than the sum of its parts". Using a data set of 58 OSS projects with more than 580,000 commits contributed by more than 30,000 developers, in this article we provide a large-scale analysis of the relation between size and productivity of software development teams. Our findings confirm the negative relation between team size and productivity previously suggested by empirical software engineering research, thus providing quantitative evidence for the presence of a strong Ringelmann effect. Using fine-grained data on

Empirical Software Engineering,
Vol. 21, No. 2, April 2016

Thank You!

I Scholtes, M Strohmaier, F Schweitzer: **Is Computer Science becoming a Social Science?**, Under review, October 2016

I Scholtes, P Mavrodiev, F Schweitzer: **From Aristotle to Ringelmann: A large-scale analysis of productivity and coordination in Open Source Software projects**, Empirical Software Engineering, Vol. 21, No. 2, April 2016

I Scholtes: **Understanding Complex Systems: When Big Data meets Network Science**, it - Information Technology, Vol. 57, No. 4, pp. 252-256, July 31 2015

MS Zanetti, I Scholtes, CJ Tessone, F Schweitzer: **The Rise and Fall of a Central Contributor: Centralization and Performance in the Gentoo Community**, Cooperative and Human Aspects in Software Engineering, San Francisco, CA, USA, May 2013

MS Zanetti, I Scholtes, CJ Tessone, F Schweitzer: **Categorizing Bugs with Social Networks: A Case Study on Four Open Source Software Communities**, 35th International Conference on Software Engineering (ICSE '13), pp. 1032-1041, San Francisco, CA, USA, May 2013

@ingo_s

http://www.ingoscholtes.net

ischoltes@ethz.ch