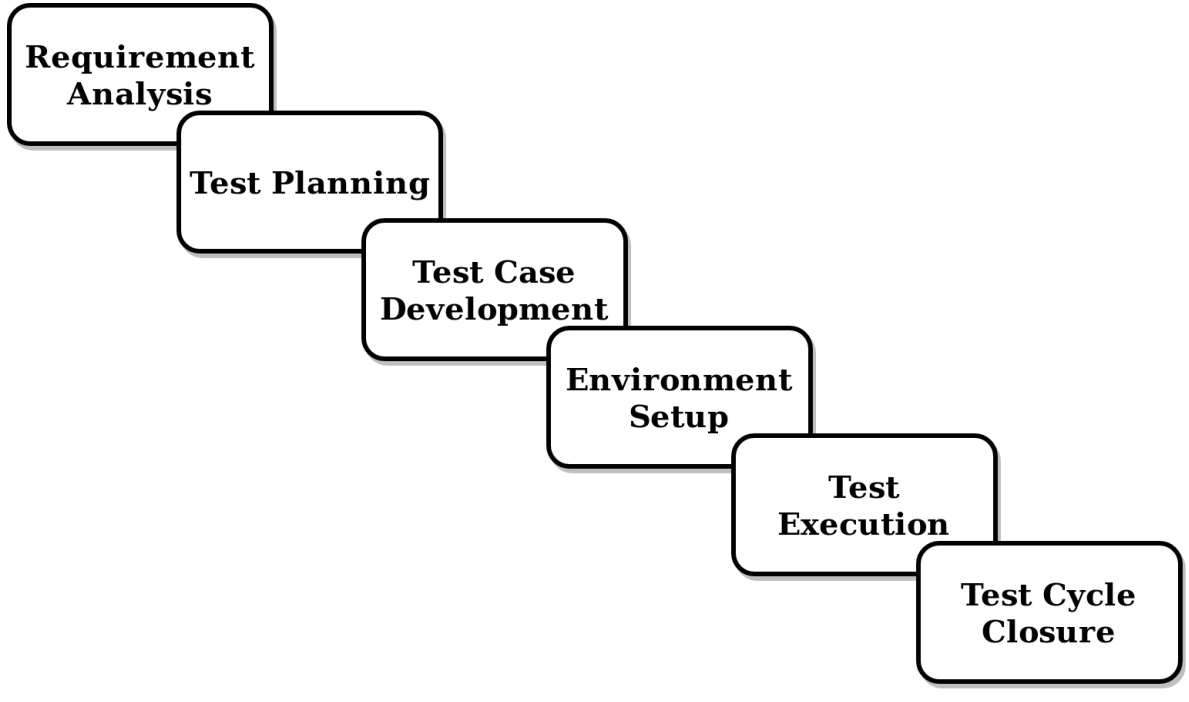
Systematic Analysis of testing-related publications concerning reprocucibility and comparability

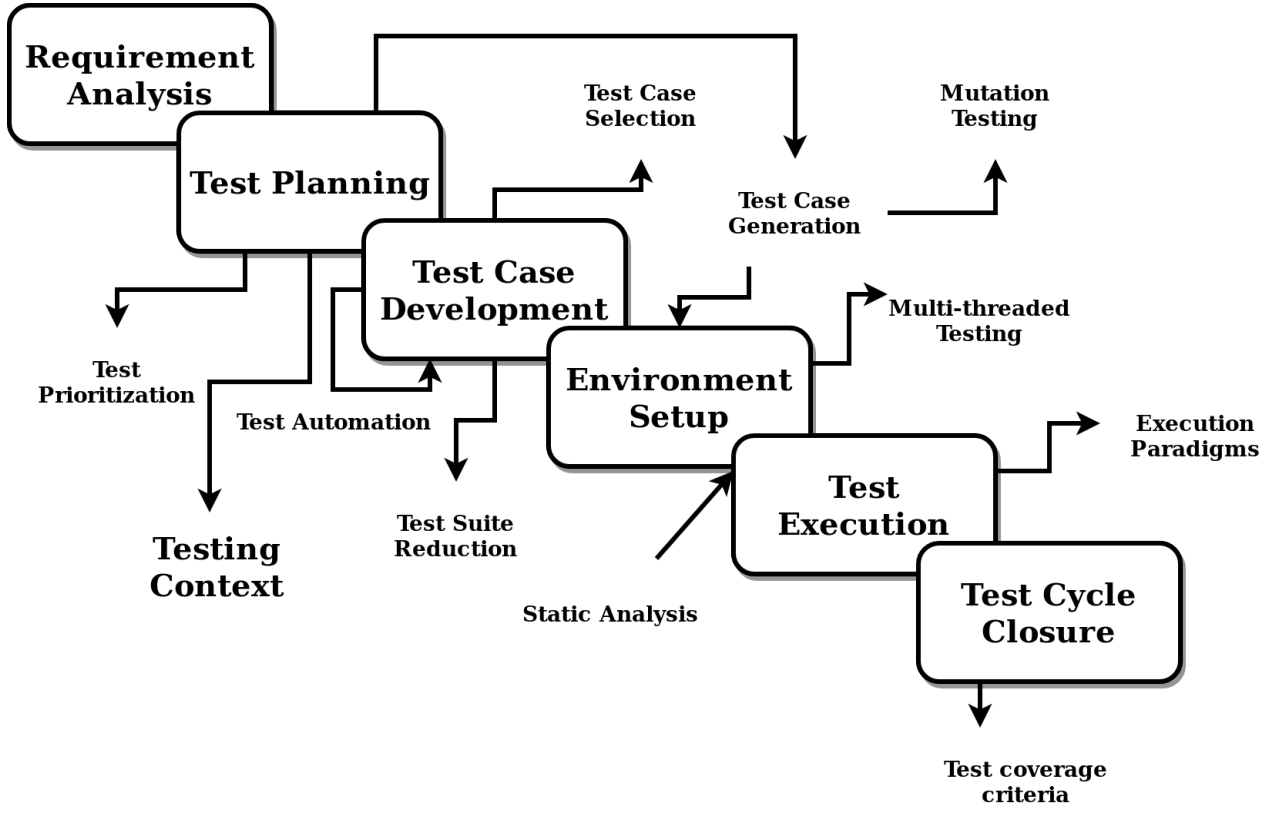
Bachelor's Thesis Defense by Artur Solomonik
Referees: Prof. Dr. Norbert Siegmund, Prof. Dr. Martin Potthast

Software Testing

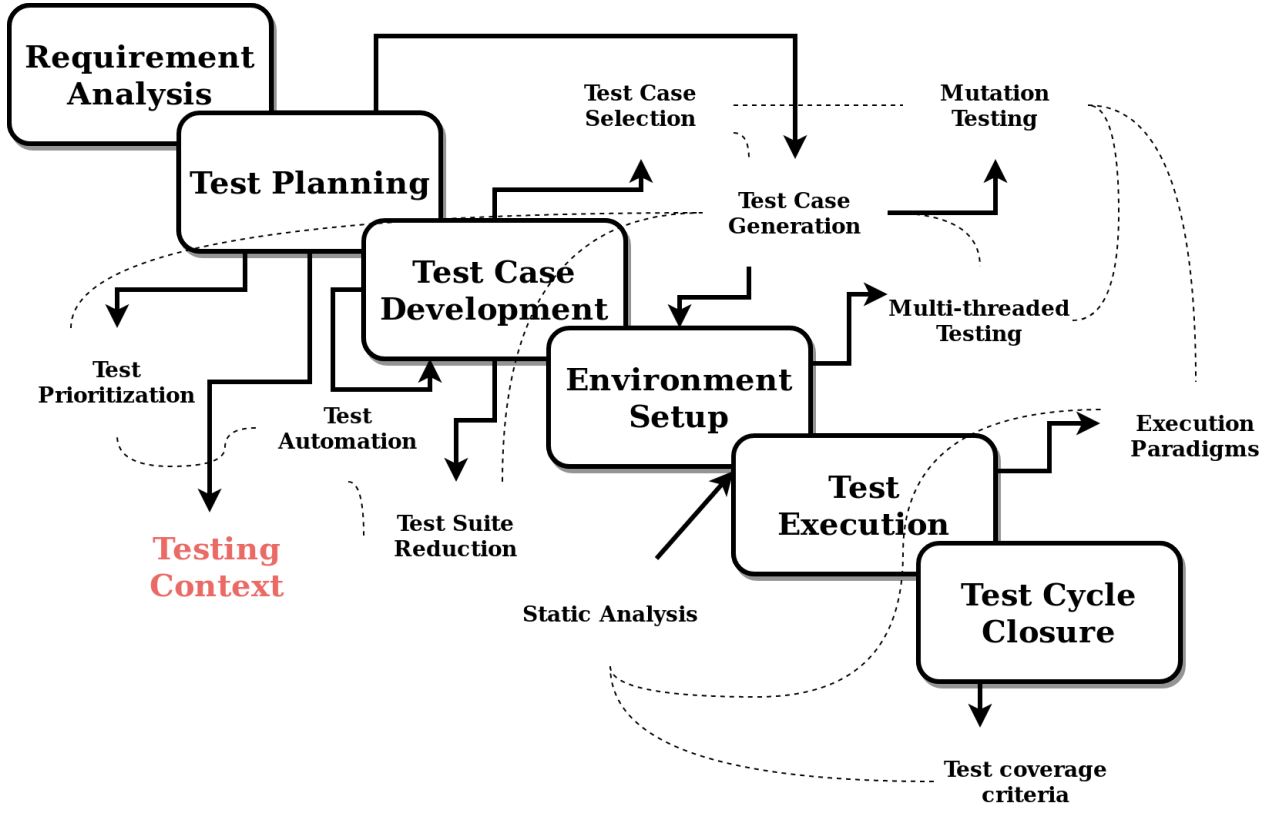
Software Testing Life Cycle



Software Testing Life Cycle



Software Testing Life Cycle



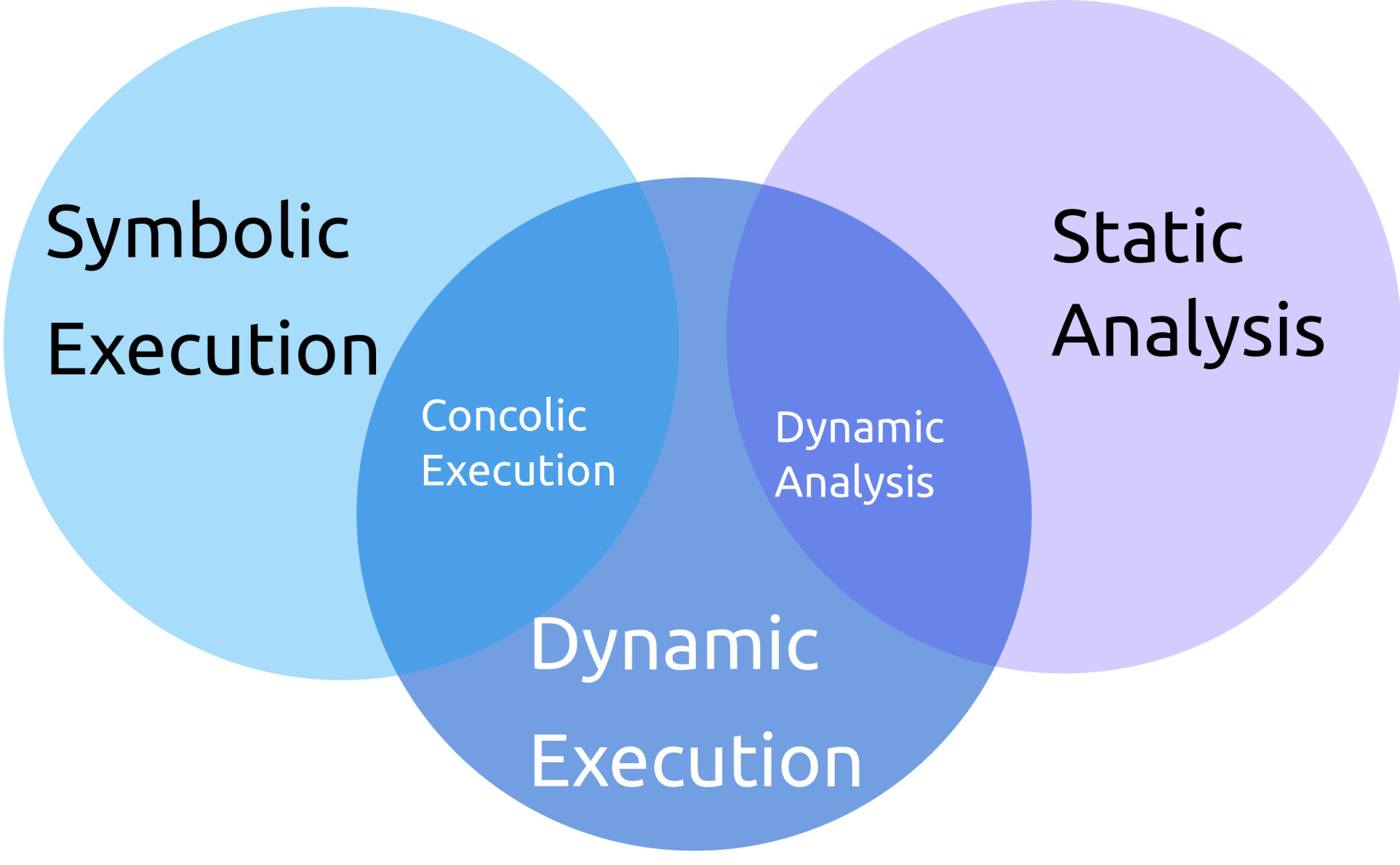
Software Testing Research

- Generating test suites
  - Exploration pinciples
  - Mutation testing
  - Executing generated test suites
  - Prioritization and Reduction of Test Cases
- Automating test case creation, selection and execution
- Finding new approaches on organizing testing processes
  - Testing Workflow
  - Decision Making Process
  - *When and What to Automate?*

Software Testing Research

- Testing Levels
    - Data-Flow Testing, Static Code Analysis | **Unit Testing**
    - Backbone-, Client-Server-, Bottom-Up | **Integration Testing**
    - GUI Testing, End-To-End Testing | **System Testing**
    - Reliability and Stability, Chaos Testing | **Acceptance Testing**
- Execution Paradigms

Test Execution Paradigms



How do we know the testing system is working?

3. EXPERIMENTAL SETUP

In this study, we address four research questions.

- **RQ1:** What's the overall effectiveness (in terms of program coverage improvement and fault detection improvement) of *ISON*?
- **RQ2:** How does *ISON* perform in improving each test's effectiveness?
- **RQ3:** How does negation depth affect the effectiveness of *ISON*?
- **RQ4:** How does *ISON* perform comparing to traditional automated test generation approaches?

3.1 Tools

***ISON* implementation tool**. *ISON* has been implemented as a prototype for Java programs using JUnit tests. In implementation, we use *Soot program analysis framework* for isomorphism identification, *ASM bytecode manipulation engine*[10] for branch negation and output analysis. The source code of *ISON* is available on our homepage[11].

Mutant generation tool. We use *Major*[12] to generate mutants in the evaluation, because most mutation tools (e.g., Javalanche[13] and Pitest[14]) do not produce mutants with accessible source code whereas *Major* does. On the other side, *Major* is the only mutation tool whose mutants are proved to simulate real faults well [26].

Test generation tool. We compare our approach with state-of-the-art test generation tool *EvoSuite*[15] in our evaluation. *EvoSuite* is a search-based test generation tool which is capable to generate test oracles and deal with various cases that other test generation techniques/tools cannot handle [12]. Moreover, it has been reported to be one of the most practical and robust test generation tools [10]. We also tried to compare our technique with state-of-the-art symbolic execu-

mutants. These mutants were viewed as faults in this study.

For each subject, we constructed a common test set for its two versions to compare the outputs of the same test (following Section 2.3). In particular, the original tests of the new version were regarded as the base test set, from which we removed the tests that cannot run through on the old version. Then, we further removed the tests that produce non-deterministic outputs in two steps: (1) we manually removed the tests that will definitely produce non-deterministic outputs, e.g., tests that return the current time, and (2) we automatically checked the remaining tests by running each of them 5 times and comparing their outputs to ensure their determination, as previous work does [49].

Table 1 shows the basic information of the subjects (i.e., the versions under test). "LOC" shows the number of lines of executable source code calculated by *LocMetrics*[17]. "Tests" [illegible] actually used in the study. B_{all}/B_{cov} refers to the total number of branches and number of branches that are covered by the tests. M_{all}/M_{kill} refers to the total number of mutants (or faults) and the number of mutants that are killed by the tests. From the table, we use subjects of various sizes, whose LOC ranges from 258 to 23,293. Also, these subjects have various proportion of covered branches as well as killed mutants.

Table 1: Subjects, faults, and tests

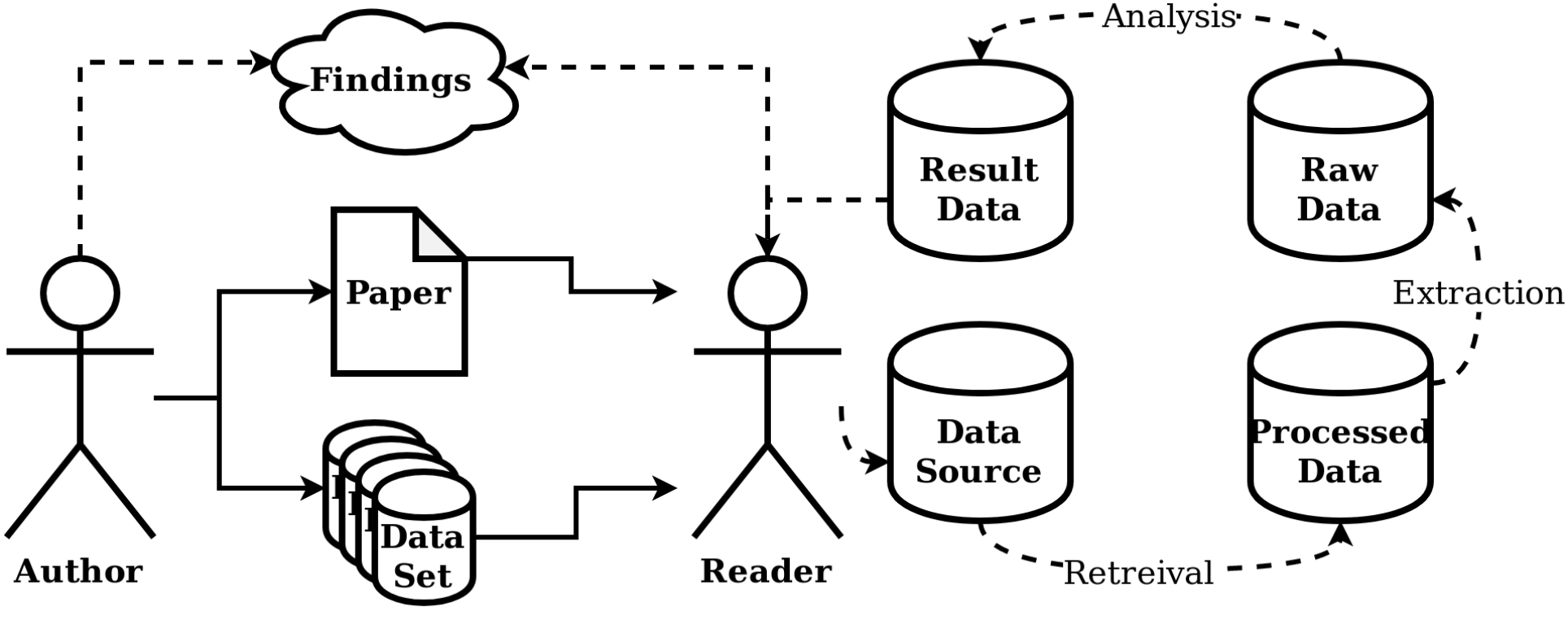
Subjects	LOC	Tests	B_{all}/B_{cov}	M_{all}/M_{kill}	Ver.
cors-filter	1,198	72	146/138	195/122	1.0.1
digester	23,293	184	1,432/885	200/84	3.2
evo-inflector	465	4	26/19	105/54	1.2.1
gelfj	1,416	27	216/107	200/41	1.1.1
gson-fire	895	31	194/114	200/92	1.3.1
hashids	258	11	74/46	200/135	1.0.1
jackson	654	43	154/23	200/48	1.5.1
java-jwt	422	55	104/93	158/89	2.1.0
jopt-simple	4,292	640	378/368	200/144	4.4
scrypt	467	20	116/48	200/93	1.4.0

Evaluating result data

- Present the result data set and identify significant values
- Connect hypotheses and results
- Compare related work and their findings
- Argument the improvement or benefits of the approach
- Apply suitable metrics

Reproducibility

Goal: Provide the reader with every information and resource necessary to recreate the findings presented in the paper



Reproducibility Attributes

- Reproduction score influenced by data set attributes
  - **Identification:** Explanation of where the data is and what it is called
  - **Description:** Level of the of the explanation regarding the element
  - **Availability:** Ease of accessing or obtaining the research elements
  - **Persistence:** Confidence in future state and availability of the elements
  - **Flexibility:** Adaptability of the elements to new environments

Reproducibility Attributes

- Reproduction score influenced by data set attributes
  - **Identification:** Explanation of where the data is and what it is called
  - **Description:** Level of the of the explanation regarding the element
  - **Availability:** Ease of accessing or obtaining the research elements
  - **Persistence:** Confidence in future state and availability of the elements
  - **Flexibility:** Adaptability of the elements to new environments

Reproducibility Attributes

- Reproduction score influenced by data set attributes
  - **Identification:** Explanation of where the data is and what it is called
  - **Description:** Level of the of the explanation regarding the element
  - **Availability:** Ease of accessing or obtaining the research elements
  - **Persistence:** Confidence in future state and availability of the elements
  - **Flexibility:** Adaptability of the elements to new environments
- Varying data sources - Attributes not applicable to anything

Comparability

Goal: Assess papers on whether empirical comparisons in the evaluation are appropriate or existent.

- Criteria for comprehensible evaluations
- Strategies of Comparison
- Connectivity to related work

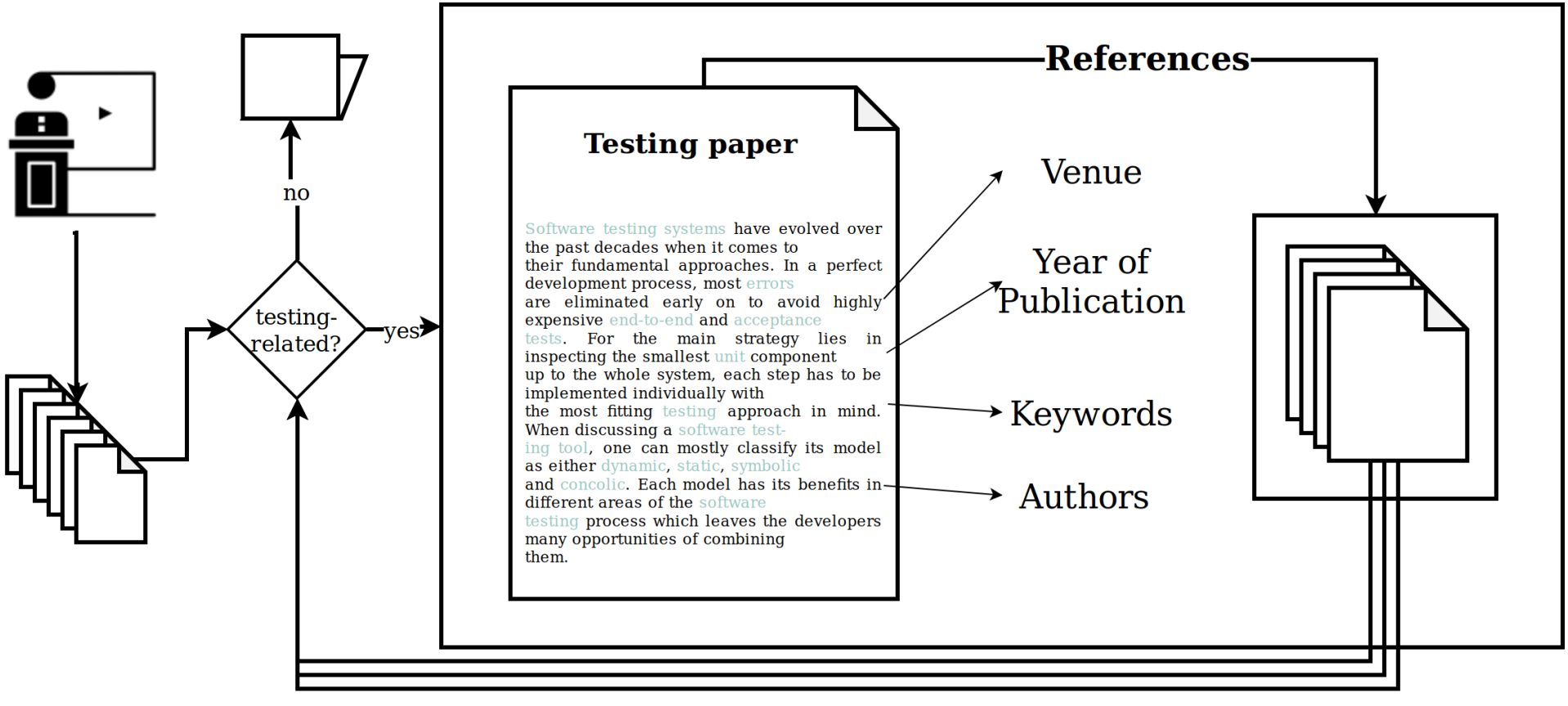
How can we understand the research strategies of software testing publications in terms of reproducibility and comparability?

Paper Classification

Data Source

- Papers from 10 popular software engineering conferences (ASE, ICSE, ISSTA, ...)
- Additional publications from two journals (ESE, TOSEM)
- Frequently mentioned publications
- Papers from modification / refinement phases

Processed Data Set



testing papers and evaluation data

Datei Bearbeiten Ansicht Einfügen Format Daten In Arbeit... Hilfe 3 Freigeben Anmelden

100% Nur Kommentierzugriff

fx year

	A	D	E	F	G	H	I	J	K	L
1	year	title	paper	id	contribution	classification	tested_version	availability	source_code	benchmarks
2	2016	Battles with F	https://(	3	static analysis	static	-	open	https://github.com/suk	30 Tizen Web App
3	2016	FOREPOST:	https://(	4	performance testing	dynamic	-	open	http://www.cs.wm.edu	[155] [156] [157]
4	2016	SimCoTest: A	https://(	5	test generation	dynamic		open	https://github.com/ma	3 industrial SL/SF
5	2016	Automated Te	https://(	6	test generation	concolic	-	open	https://github.com/dar	-
6	2016	HoliCoW: Aut	https://(	7						
7	2015	Ekstazi: Light	https://(	8	test selection	dynamic		open	http://ekstazi.org/	32 Open-Source p
8	2015	TesMa and C.	https://(	9	test generation	concolic	-	open	https://github.com/kse	[5] [6]
9	2015	Dynamic Data	https://(	10	data flow testing	dynamic	-	closed	-	Table 3
10	2015	ZoomIn: Disc	https://(	11	test generation			closed	-	Table 1
11	2015	An Informatio	https://(	12	test prioritization	concrete	-	closed	-	[31] [32] [33] [34]
12	2015	Automated M	https://(	13	gui testing	concrete	-	closed	-	[38] [39] [40]
13	2015	Regular Prop	https://(	14	concolic execution	concolic	-	open	https://github.com/psy	Table 1
14	2015	Metamorphic	https://(	15	acceptance testing	concrete	-	closed	-	-
28	2014	Enhancing Sy	https://(	29	sysmbolic execution	symbolic		closed	-	[1], Debian binarie
29	2014	Interpolated M	https://(	30	test case derivation	static	-	open	http://se.fbk.eu/techno	[41]
30	2014	ConLock: A C	https://(	31	multithreaded testing	dynamic		closed	-	[2] [3] [4]
31	2014	Coverage Is M	https://(	32						
32	2013	JST: An Autor	https://(	33	test generation	symbolic		closed	-	unmentioned
33	2013	Automatic De	https://(	34	performance testing	dynamic		closed	-	one industrial syst
34	2013	Memoise: A T	https://(	35	symbolic execution	symbolic	-	open	https://github.com/guc	[42]
38	2016	Python Predic	https://(	39	symbolic execution	symbolic	-	open	https://sites.google.cc	python GitHub pro
39	2016	Extracting Ins	https://(	40	symbolic execution	concolic	-	open	https://github.com/nirh	-
40	2016	DiagDroid: Ar	https://(	41	performance testing	dynamic	-	open	http://www.cudroid.co	F-Droid Apps
41	2016	Analyzing the	https://(	42	mutation testing	static		closed	-	[12]

papers paper_evaluation benchmark paper_benchmark

testing papers and evaluation data

Datei Bearbeiten Ansicht Einfügen Format Daten In Arbeit... Hilfe

Freigeben Anmelden

100% Nur Kommentierzugriff

fx year

	A	D	E	F	G	H	I	J	K	L
1	year	title	paper	id	contribution	classification	tested_version	availability	source_code	benchmarks
2	2016	Battles with F	https://(	3	static analysis	static	-	open	https://github.com/suk	30 Tizen Web App
3	2016	FOREPOST:	https://(	4	performance testing	dynamic	-	open	http://www.cs.wm.edu	[155] [156] [157]
4	2016	SimCoTest: A	https://(	5	test generation	dynamic		open	https://github.com/ma	3 industrial SL/SF
5	2016	Automated Te	https://(	6	test generation	concolic	-	open	https://github.com/dar	-
6	2016	HoliCoW: Aut	https://(	7						
7	2015	Ekstazi: Light	https://(	8	test selection	dynamic		open	http://ekstazi.org/	32 Open-Source p
8	2015	TesMa and C.	https://(	9	test generation				https://github.com/kse	[5] [6]
9	2015	Dynamic Data	https://(	10	data flow testing				-	Table 3
10	2015	ZoomIn: Disc	https://(	11	test generation			closed	-	Table 1
11	2015	An Informatio	https://(	12	test prioritization			closed	-	[31] [32] [33] [34]
12	2015	Automated M	https://(	13	gui testing	concrete	-	closed	-	[38] [39] [40]
13	2015	Regular Prop	https://(	14	concolic execution				https://github.com/psy	Table 1
14	2015	Metamorphic	https://(	15	acceptance testing			closed	-	-
28	2014	Enhancing Sy	https://(	29	sysmbolic execution	symbolic		closed	-	[1], Debian binarie
29	2014	Interpolated N	https://(	30	test case derivation	static	-	open	http://se.fbk.eu/techno	[41]
30	2014	ConLock: A C	https://(	31	multithreaded testing	dynamic		closed	-	[2] [3] [4]
31	2014	Coverage Is N	https://(	32						
32	2013	JST: An Autor	https://(	33	test generation	symbolic		closed	-	unmentioned
33	2013	Automatic De	https://(	34	performance testing	dynamic		closed	-	one industrial syst
34	2013	Memoise: A T	https://(	35	symbolic execution	symbolic	-	open	https://github.com/guc	[42]
38	2016	Python Predic	https://(	39	symbolic execution	symbolic	-	open	https://sites.google.cc	python GitHub pro
39	2016	Extracting Ins	https://(	40	symbolic execution	concolic	-	open	https://github.com/nirh	-
40	2016	DiagDroid: Ar	https://(	41	performance testing	dynamic	-	open	http://www.cudroid.co	F-Droid Apps
41	2016	Analyzing the	https://(	42	mutation testing	static		closed	-	[12]

Raw Data Set

Spreadsheet with **8060** registered papers of which **360** are classified by **23** columns

205 documented benchmarks

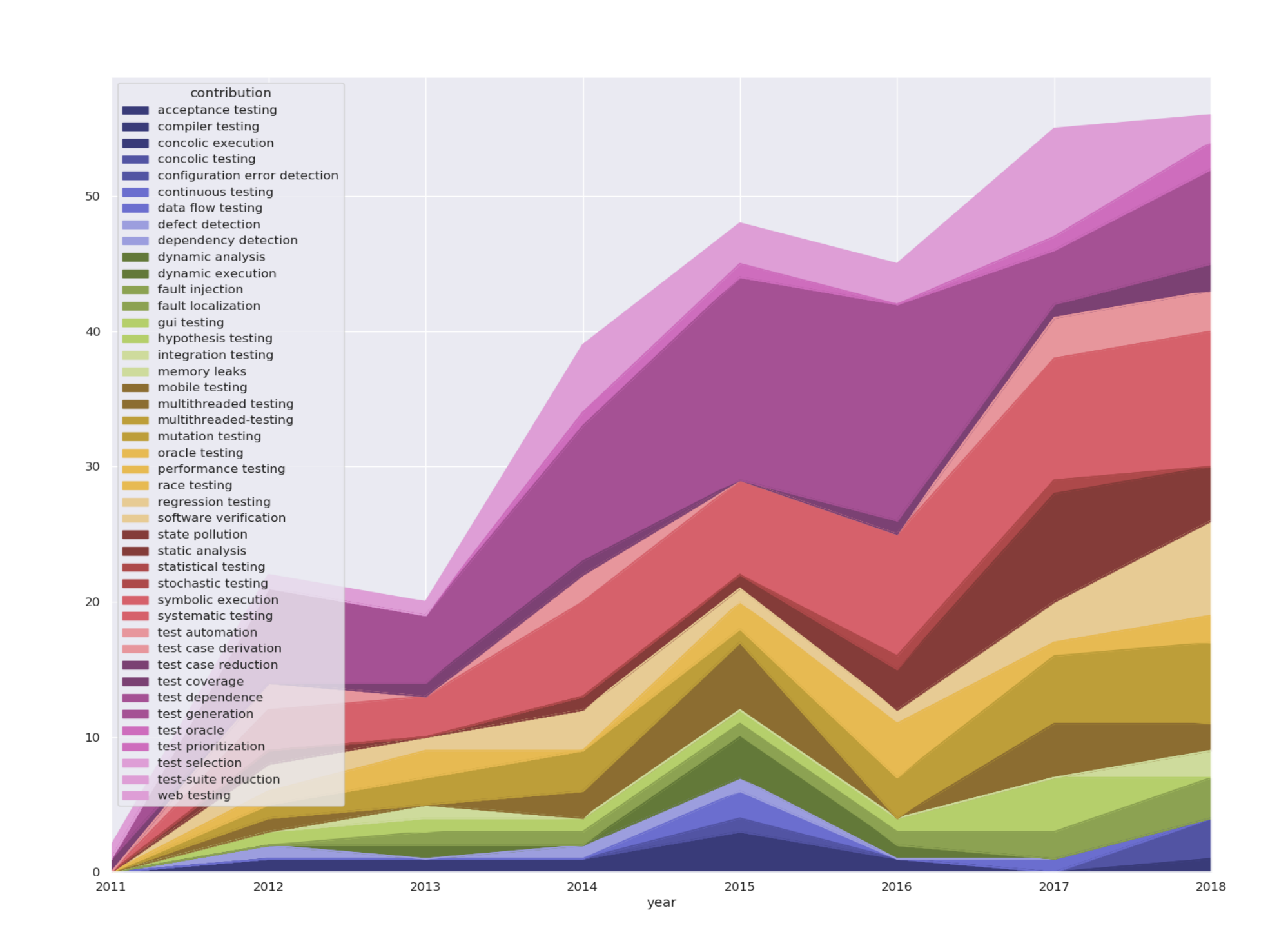
Over **15000** bibliographic and semantic connections between records

papers paper_evaluation benchmark paper_benchmark

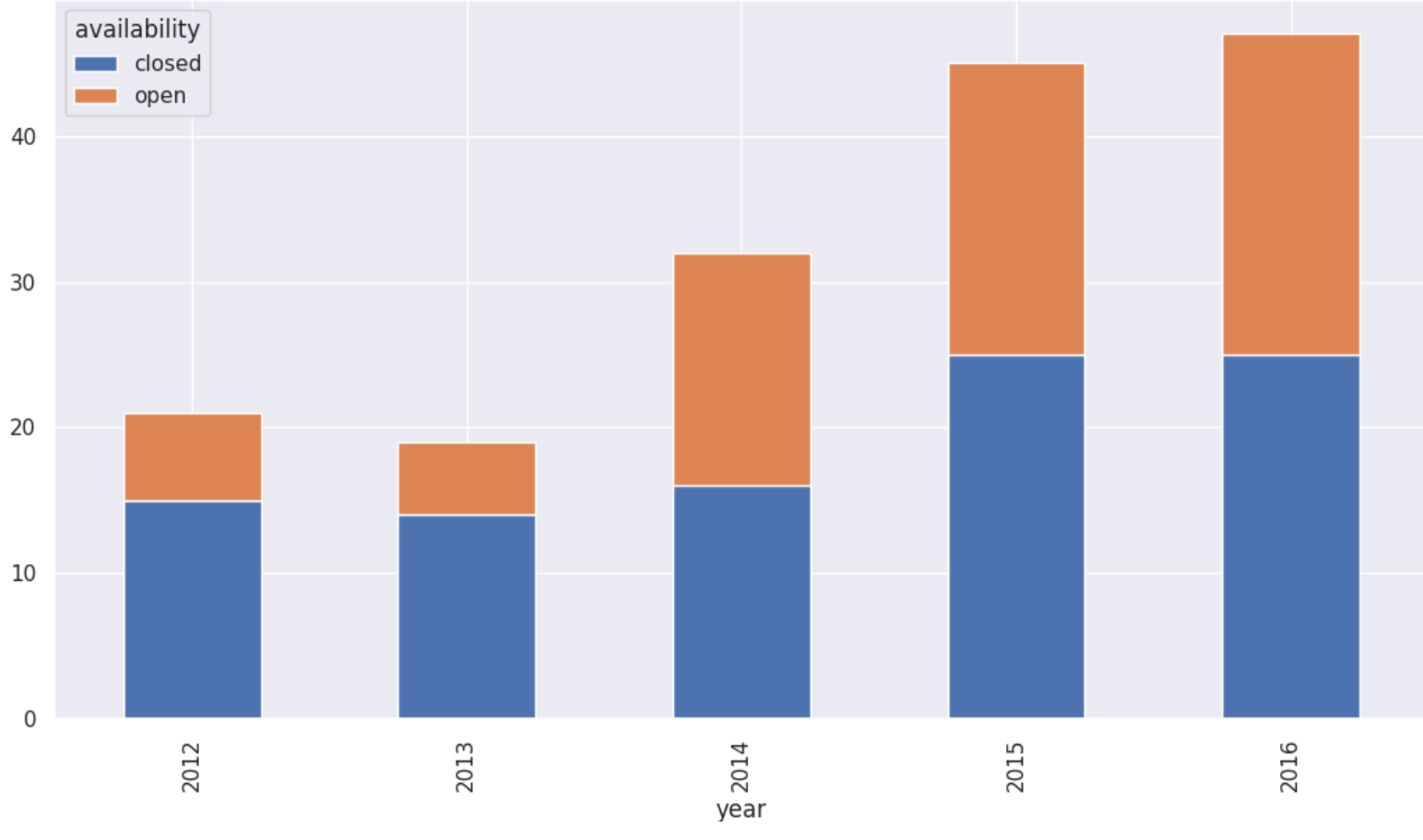
Classification	Parameters
Availability	[open/closed]
Data Set State	[vanilla/modified]
Selection Cause	[...]
Modification Cause	[...]
Sub-Check Systems	[single/multiple] [named/unnamed]

Classification	Parameters
Contribution	[...]
Choice of Metric	[functionality/performance/both]
Metrics	[ ] Metrics

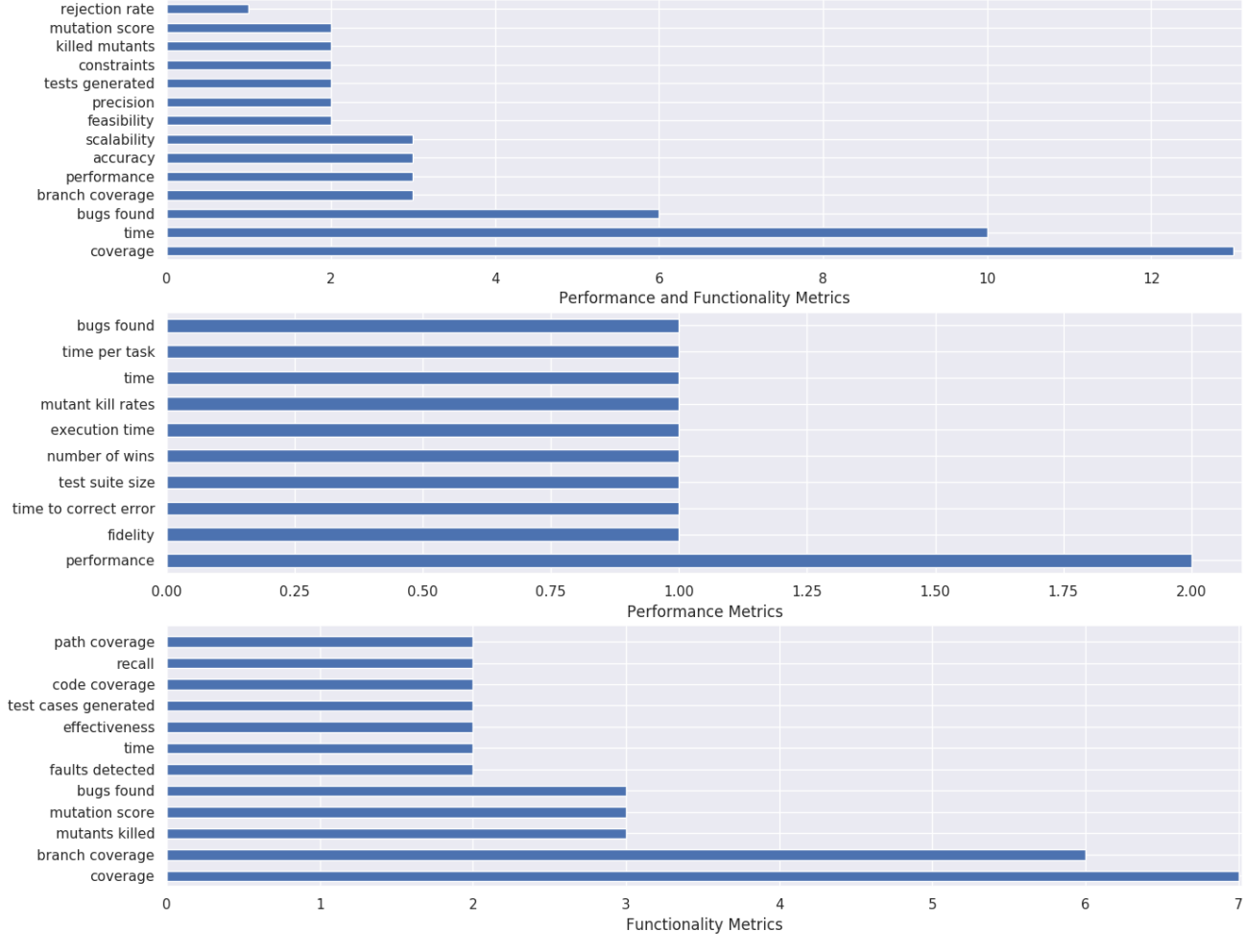
Classification	Parameters
Error Creation	[generation/real world/both]
Error Annotation	[TRUE/FALSE]
Comparison	[TRUE/FALSE] [former/foreign/parallel] [exclusive/inclusive]


contribution
acceptance testing
compiler testing
concolic execution
concolic testing
configuration error detection
continuous testing
data flow testing
defect detection
dependency detection
dynamic analysis
dynamic execution
fault injection
fault localization
gui testing
hypothesis testing
integration testing
memory leaks
mobile testing
multithreaded testing
multithreaded-testing
mutation testing
oracle testing
performance testing
race testing
regression testing
software verification
state pollution
static analysis
statistical testing
stochastic testing
symbolic execution
systematic testing
test automation
test case derivation
test case reduction
test coverage
test dependence
test generation
test oracle
test prioritization
test selection
test-suite reduction
web testing
0
10
20
30
40
50
2011
2012
2013
2014
2015
2016
2017
2018
year

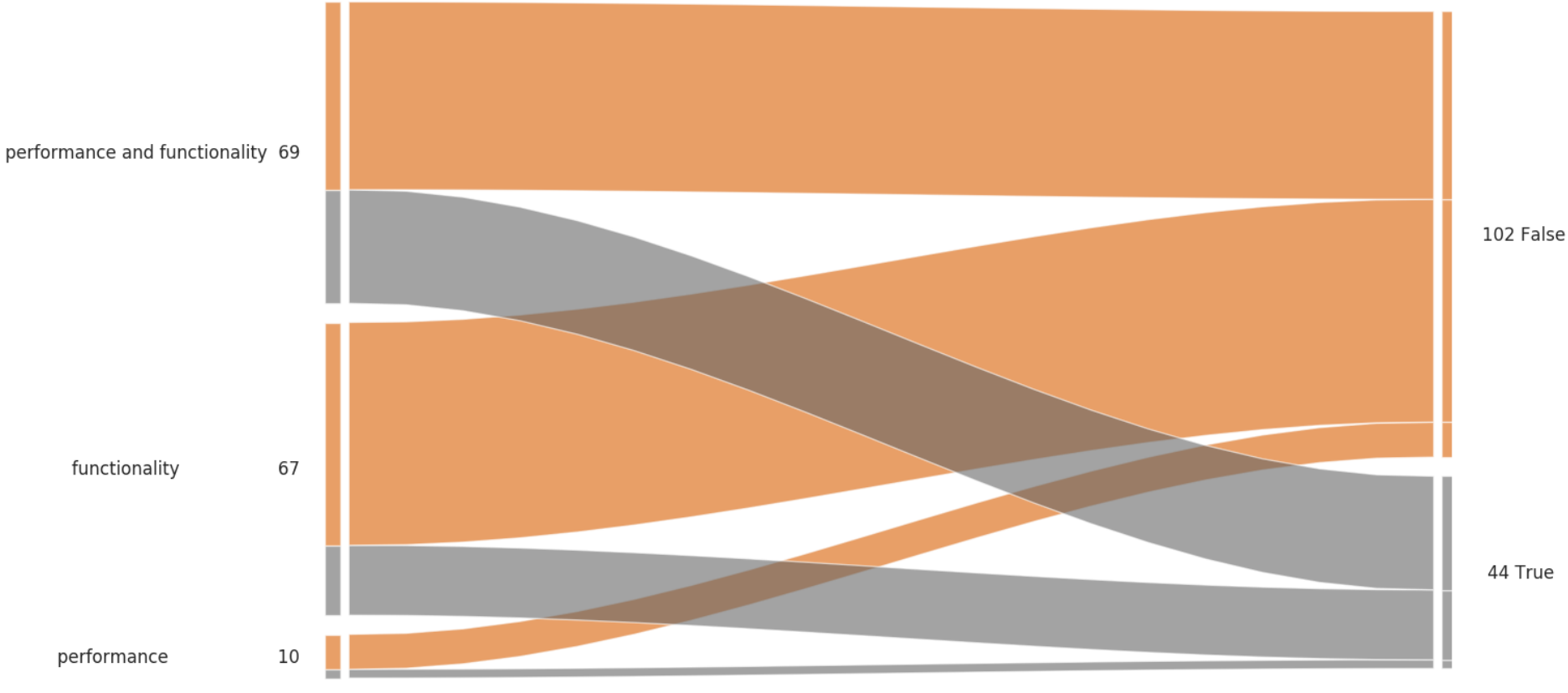
Open Source vs. Closed Source



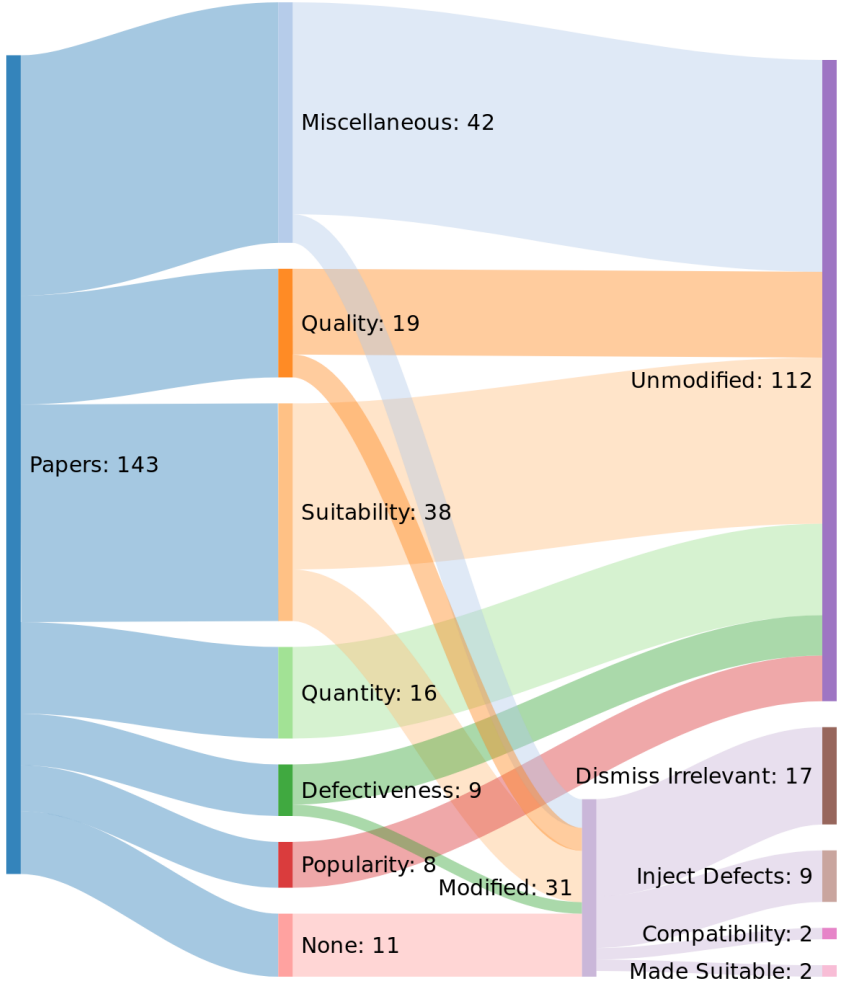
Software Testing Evaluation Metrics

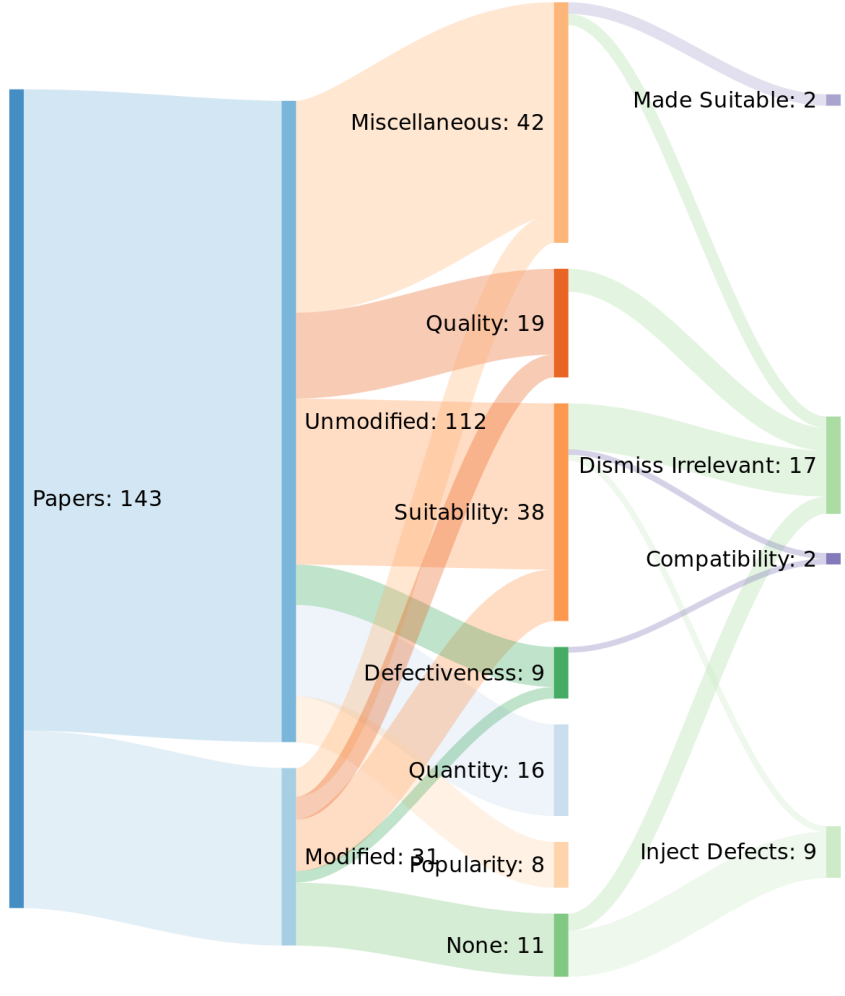


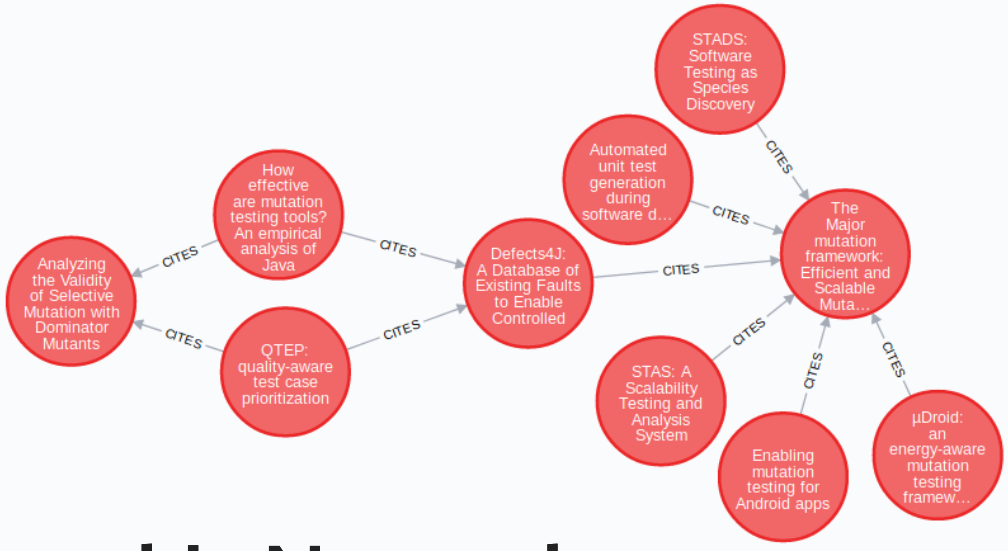
Choice of Metric and Error Annotation



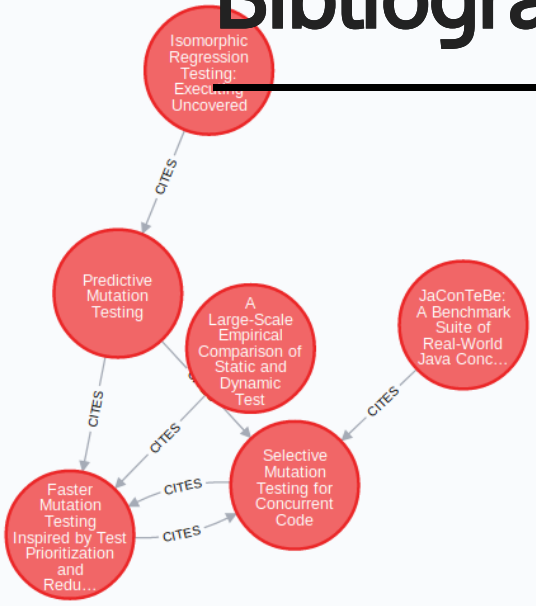
Selection and modification causes of benchmarks

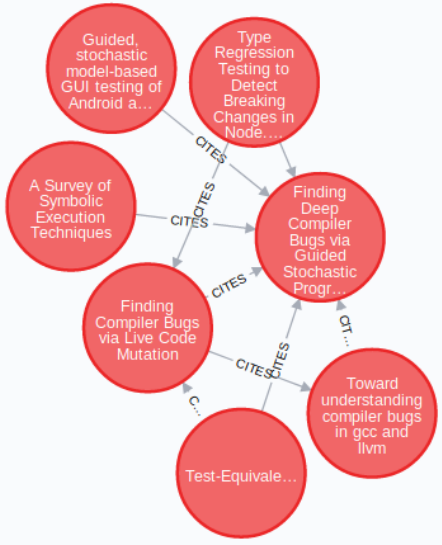






Bibliographic Networks





Goal: Visualizing great amounts of bibliographic data, increasing the interactivity with a set of publications and creating dynamic, time-based insight on the netvork evolution.

Current implementations of paper networks

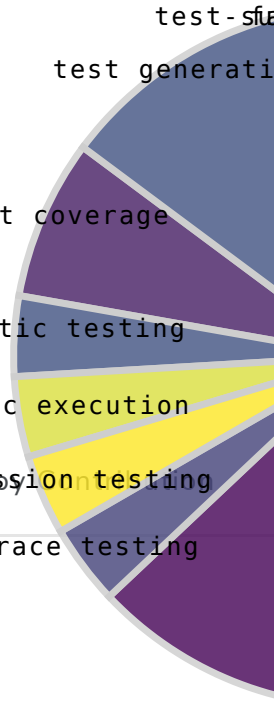
- Visualize the connection and influence between authors
- Giving insight rather than specific values
- Connected over citations, bibliographic coupling, co-citations or co-authorship relations
- Color- and size-coding node information
- Geographic hierarchies

Additions and Improvements

- Benchmarks and software systems as their own entities in a network
- More insight on reproducibility
- Multidimensional graph data visualization without clutter
- Tailouring the visualization to a certain aspect of a publication (e.g. the evaluation)

Visualizing bibliographic networks

TeLO-s



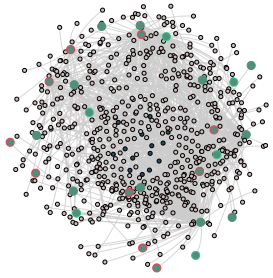

```
MATCH n = ({contribution: 'mutation testing'})-->() return n
```

✔ SEND | 🖫 SAVE QUERY

Use Cypher Query | Sort By Venue | Temporary Highlighting | Permanent Highlighting | Color Nodes by

TESTING
LITERATU
OVERVIEW
SYSTEM

This



TeLO-s

MATCH n = ({contribution: 'mutation testing'})-->() return n

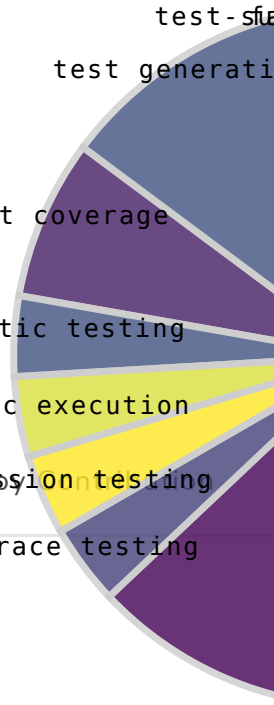
✔ SEND

SAVE QUERY

Use Cypher Query Sort By Venue Temporary Highlighting Color Nodes by

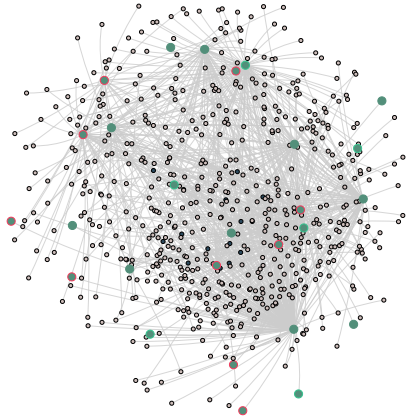
TeLO-S

D3 visualization of testing publications in a node-link force-directed graph



TESTING
LITERATU
OVERVIEW
SYSTEM

This



Cypher Query Input and Configuration

Selecting sepecific nodes from the Neo4J graph data base and manipulating the layout and color-coding

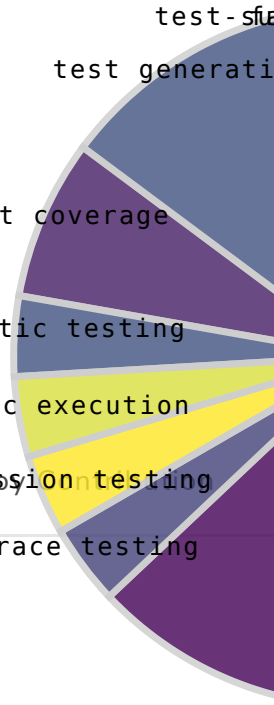
MATCH n = ({contribution: 'mutation testing'})-->() return n

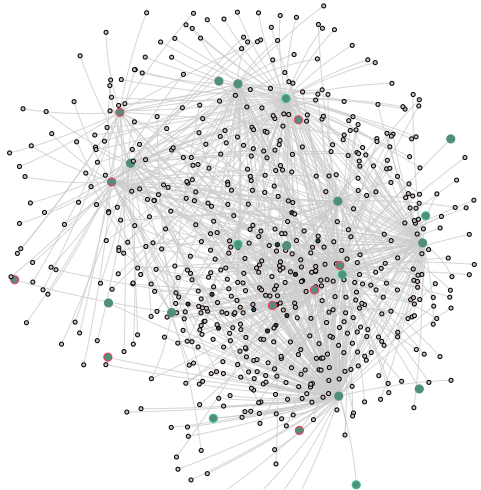
✔ SEND | SAVE QUERY

Use Cypher Query | Sort By Venue | Temporary Highlighting | Color Nodes by

TeLO-S

D3 visualization of testing publications in a node-link force-directed graph





TESTING
LITERATU
OVERVIEW
SYSTEM

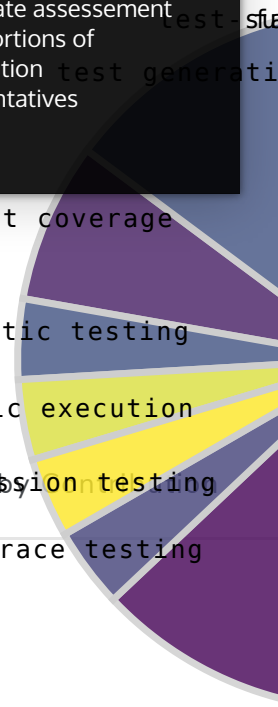
This

Cypher Query Input and Configuration

Selecting sepecific nodes from the Neo4J graph data base and manipulating the layout and color-coding

Contribution Plot

Immediate assessement of proportions of contribution representatives



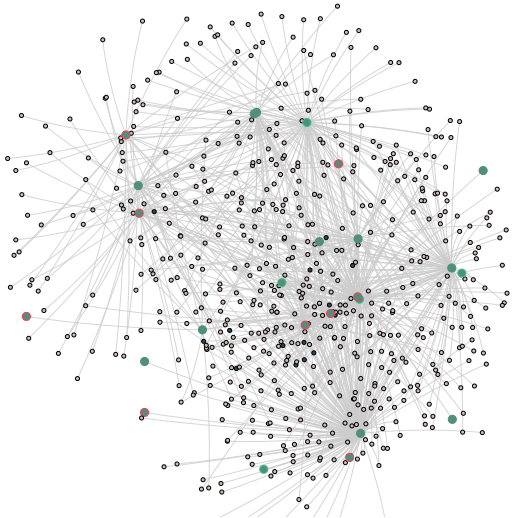

```
MATCH n = ({contribution: 'mutation testing'})-->() return n
```

✔ SEND | 🖫 SAVE QUERY

Use Cypher Query | Sort By Venue | Temporary Highlighting | Color Nodes by

TeLO-S

D3 visualization of testing publications in a node-link force-directed graph



TESTING
LITERATU
OVERVIEW
SYSTEM

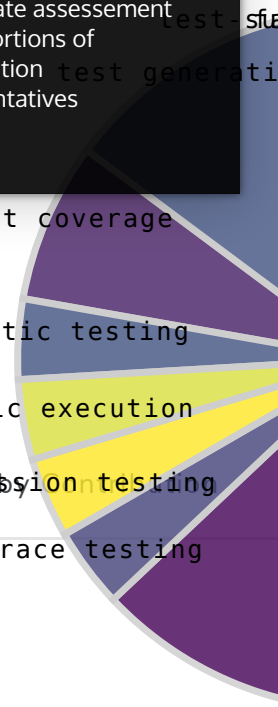
This

Cypher Query Input and Configuration

Selecting sepecific nodes from the Neo4J graph data base and manipulating the layout and color-coding

Contribution Plot

Immediate assessement of proportions of contribution representatives




```
MATCH n = ({contribution: 'mutation testing'})-->() return n
```

TeLO-S

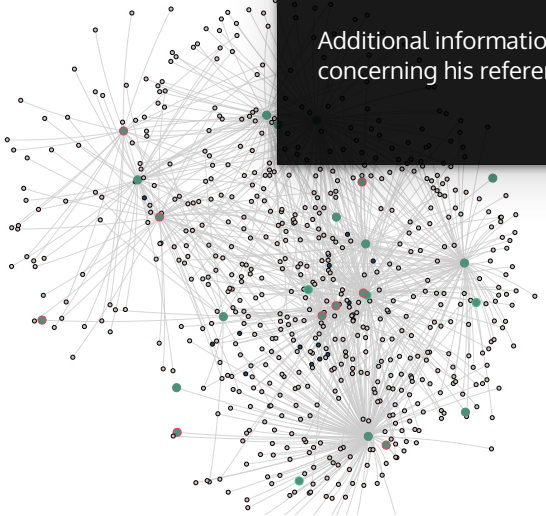
D3 visualization of testing publications in a node-link force-directed graph

✔ SEND | 🖫 SAVE QUERY

Use Cypher Query | Sort By Venue | Temporary Highlighting | Color Nodes by

Node analysis

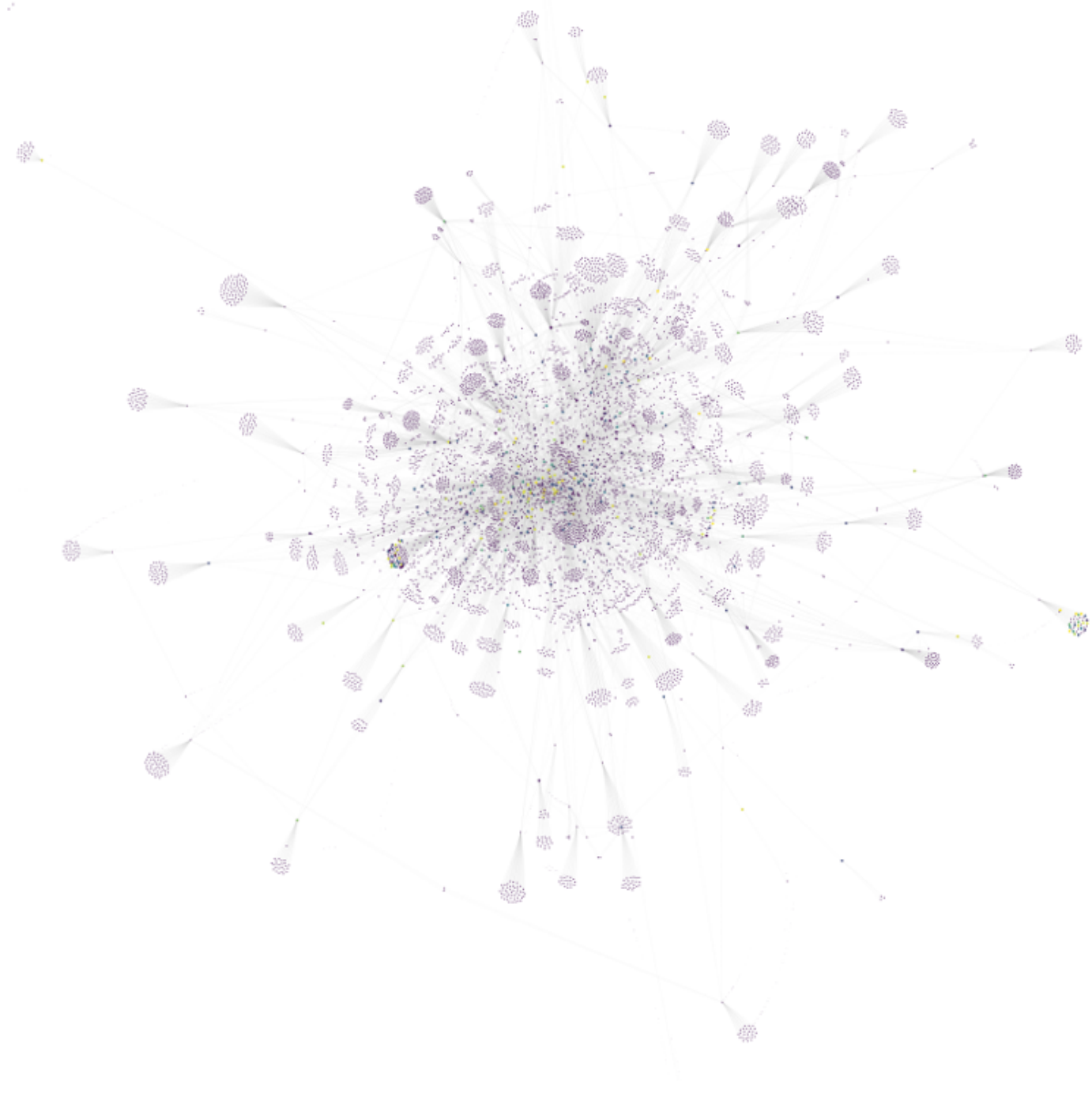
Additional information on a selected node concerning his references

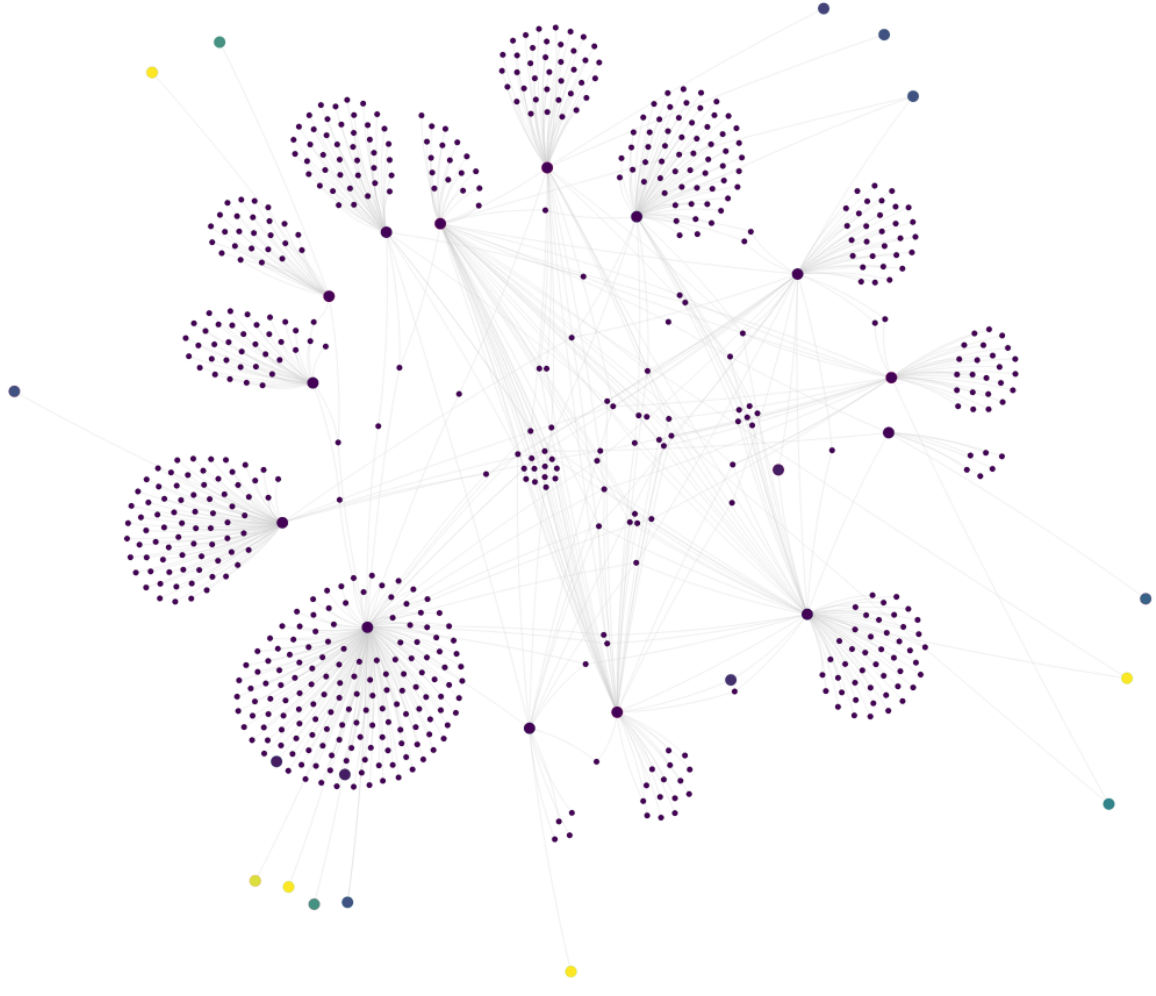


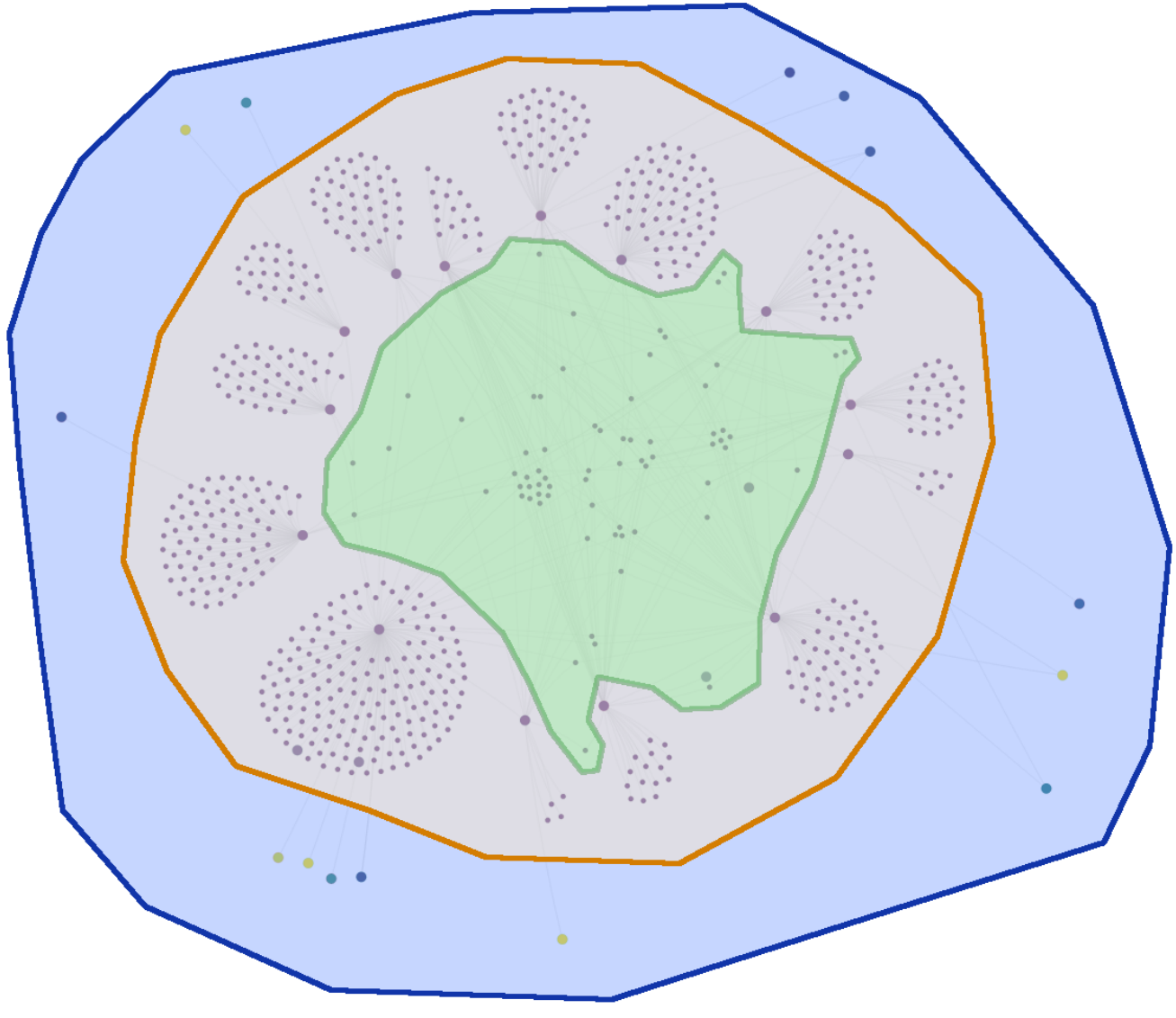
TESTING
LITERATU
OVERVIEW
SYSTEM

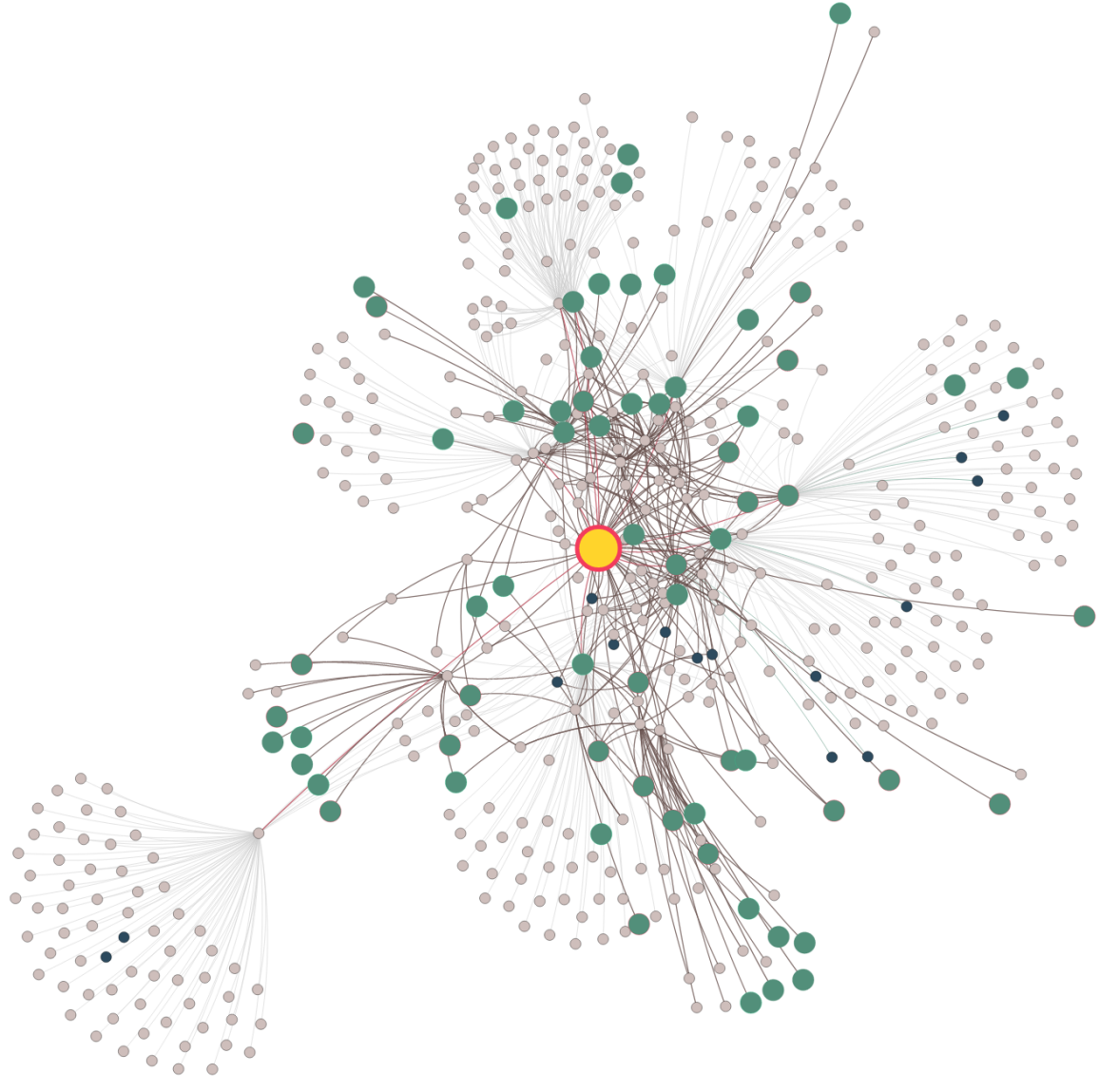
This

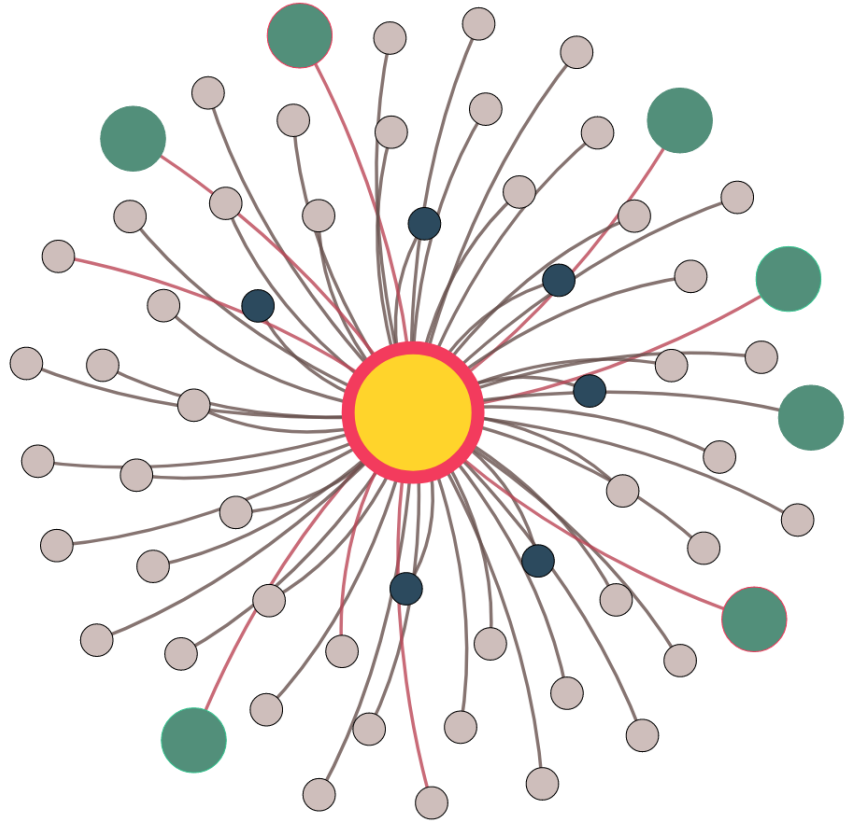
Findings

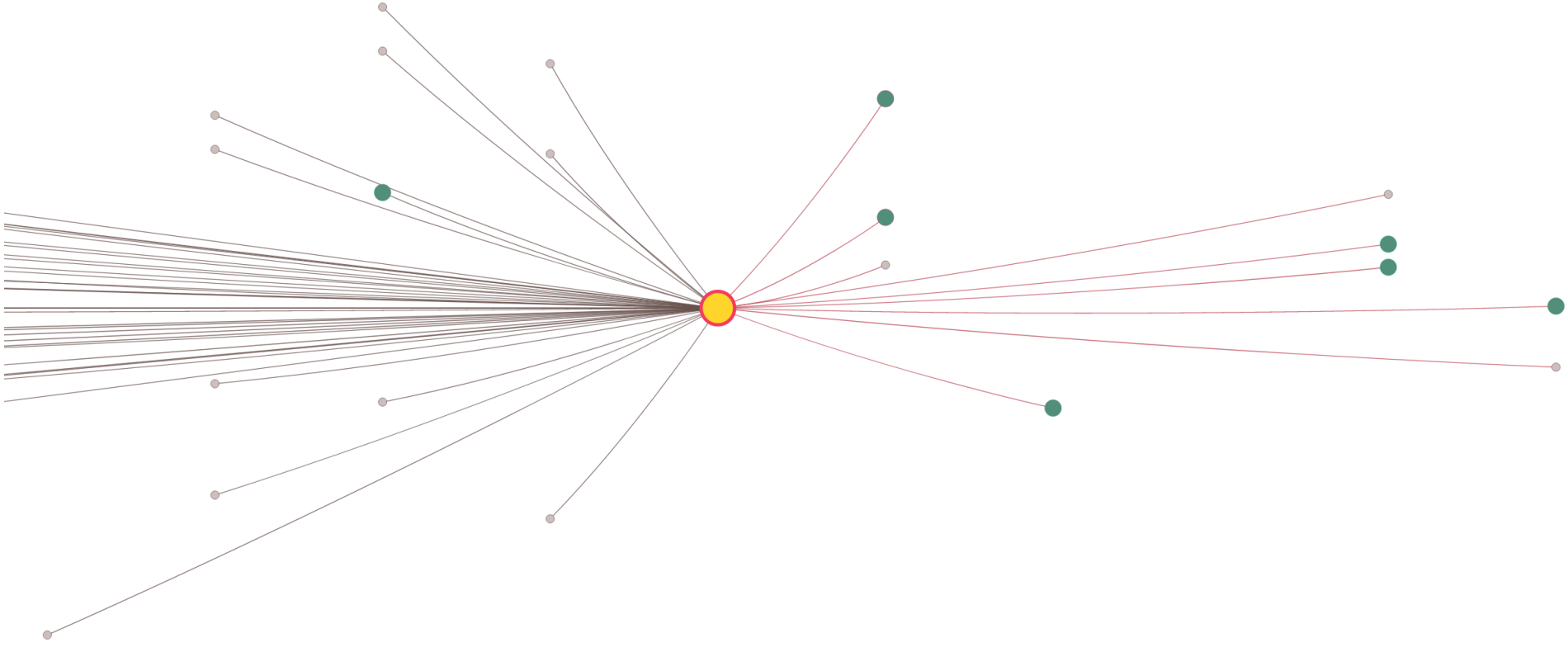


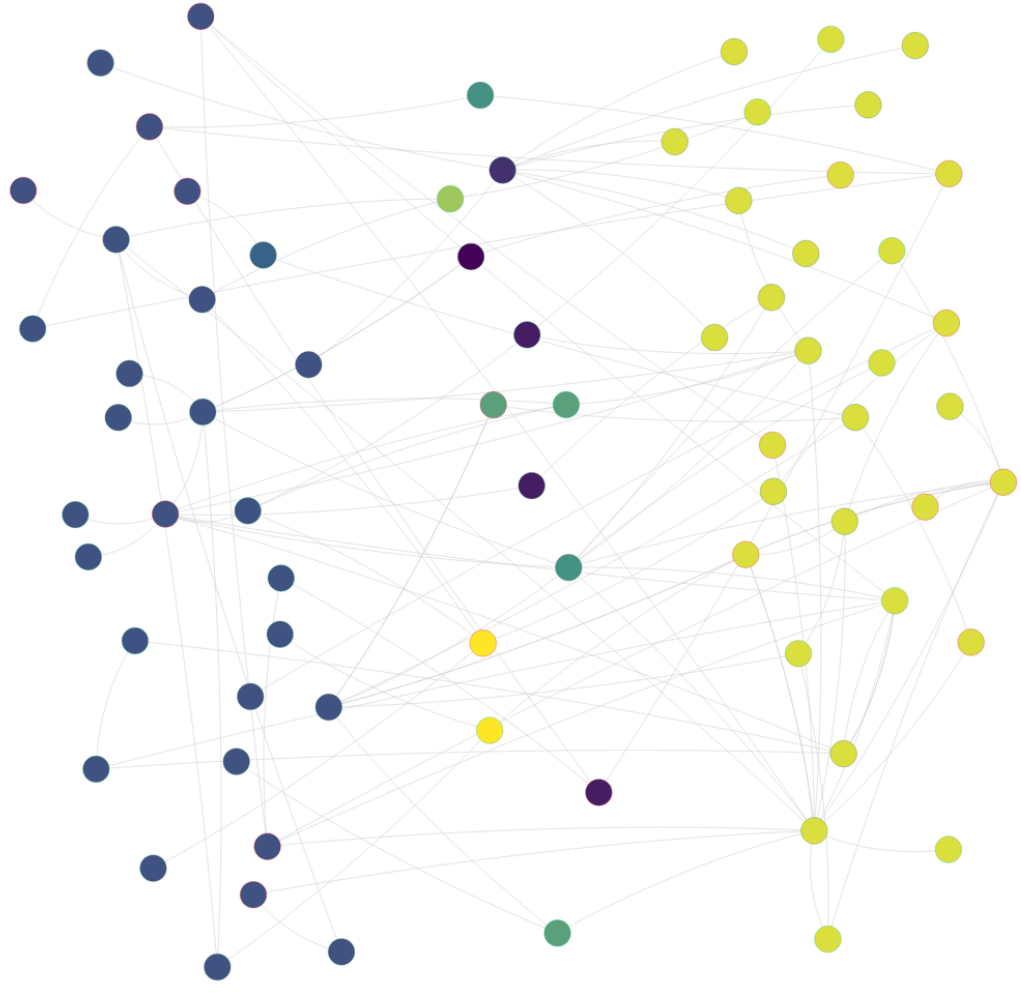


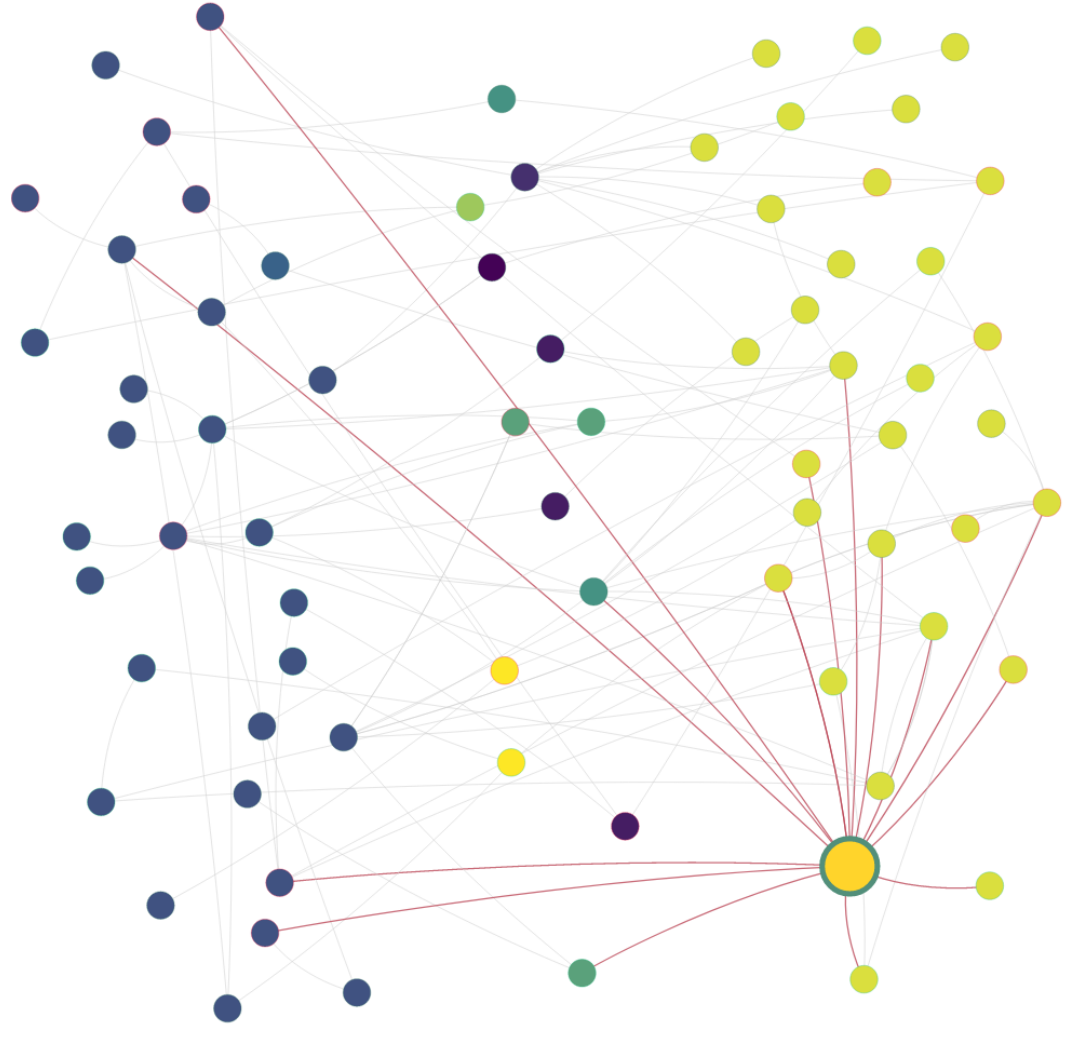






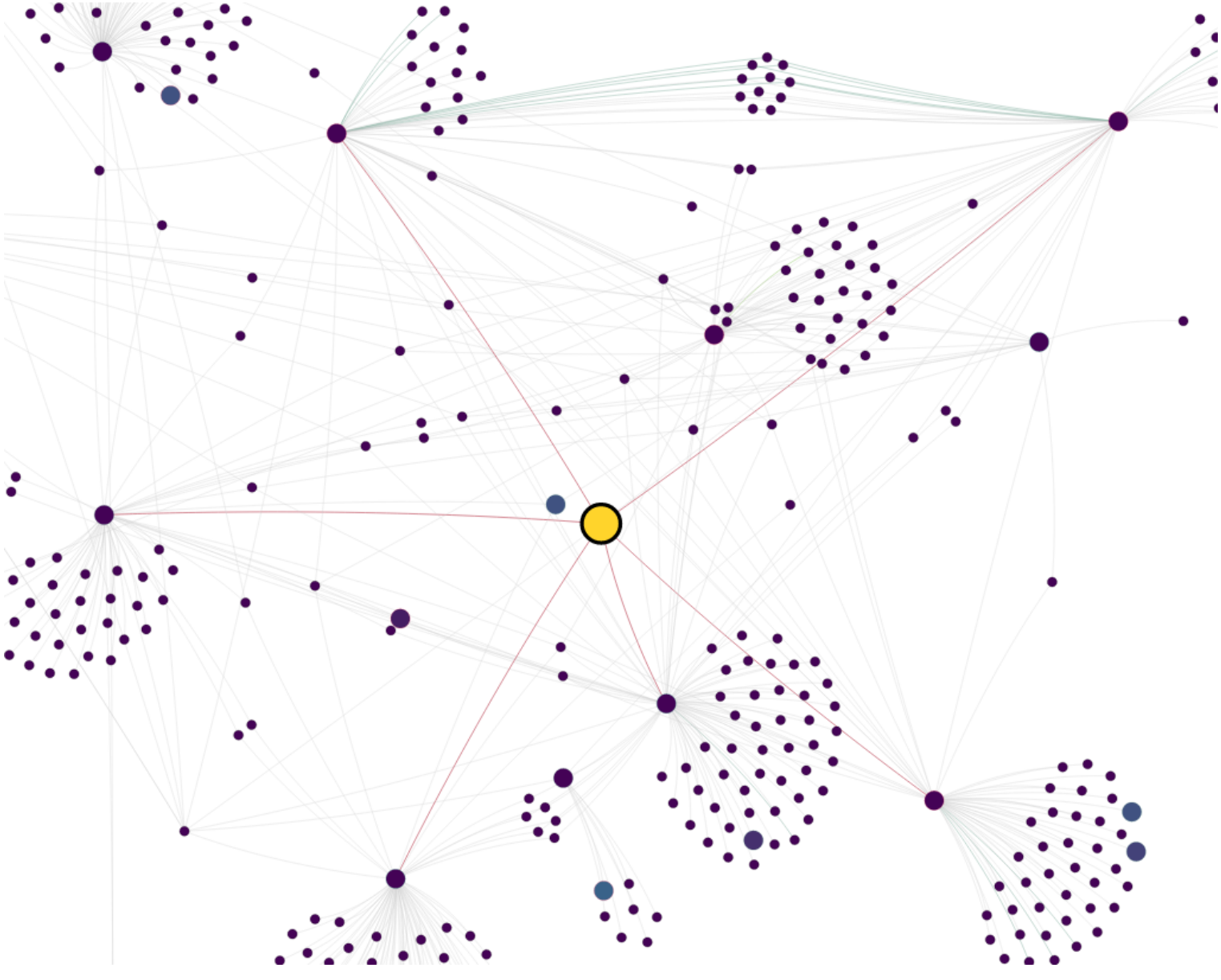






Patterns

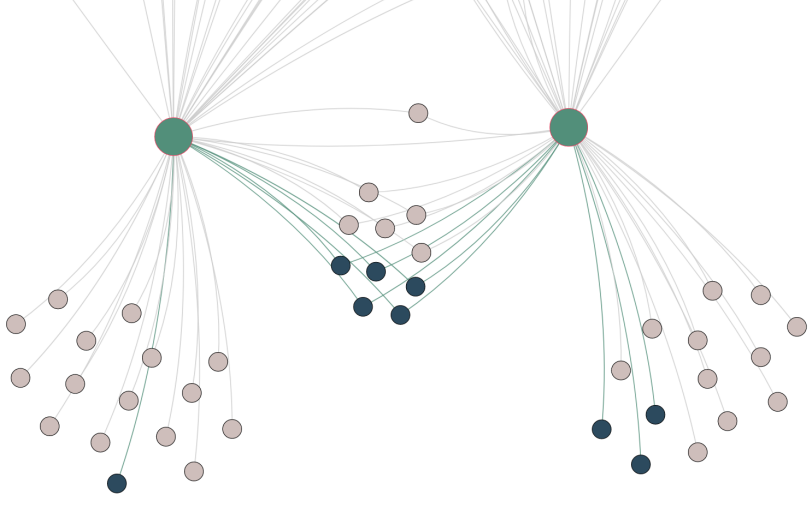
Vanishing Point Pattern

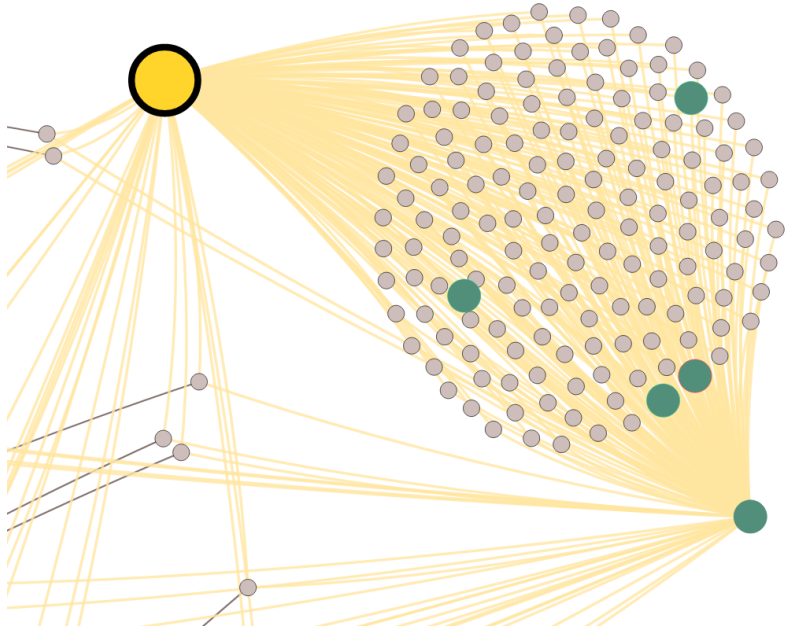


Outsider Pattern

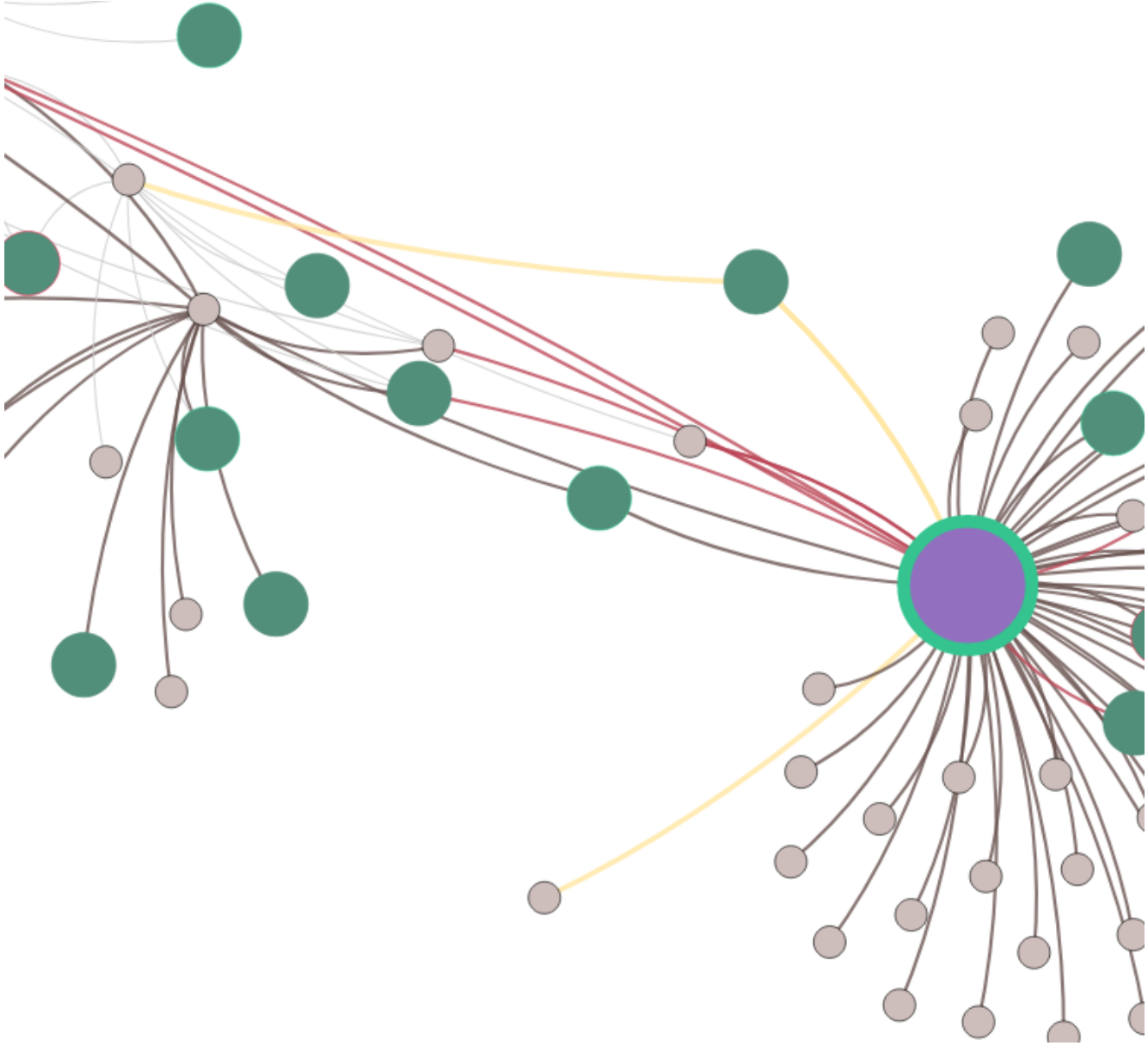
- Loose nodes in a subgraph without any connection to other queried nodes
- Nodes might imply a connection to other unqueried research fields
- Misclassifications or special cases

Familiar Foreigner Pattern





Chain Pattern



Conclusion

- Most evaluations conducted similarly
- Choice of benchmark varies significantly
- Availability as a major reproducibility issue
- Solution: Dedicated sub-check systems (possibly provided by conferences)
- Mutation scores and coverage metrics widely used
- Findings of closely related papers rarely mentioned
- Bibliographic networks benefit from sub-check system nodes and different relation types
- Comparability improves continuous improvement of research
- Comparing evaluations unfortunately very uncommon, yet beneficial

Future Work

- Adding referencing patterns to the visualization
- Classifiers for testing paper classification
- Multiple refinement cycles of the data set using relevant citations
- Implementation of author nodes, citation scores and bibliographic coupling
- Hierarchical edge bundling regarding relevancy, geography or popularity
- Generalization for other research topics aside from software testing

Thank you for your attention.