Università degli Studi di Firenze

Dipartimento di Ingegneria dell'Informazione
Tesi di Laurea Magistrale in Ingegneria Informatica

Analysis of the Evolution of Code Technical Debt in Microservices Architectures

Candidato
Kevin Maggi

Relatori
Dott.Ric. Roberto Verdecchia
Prof. Enrico Vicario

Correlatore
Dott.Ric. Leonardo Scommegna

Anno Accademico 2022/2023

Concetti chiave: Microservices Architecture, Code Technical Debt, Software Evolution

Analysis of the Evolution of Code Technical Debt in Microservices Architectures

Methodology: Empirical quantitative study

Topics: TD: *short term expedients impacting maintainability*

MSA: *widespread scalable and flexible architecture*

Motivation: Lack of studies in literature

Background: Preliminary case study[1]

Aims: Insight on TD evolution and management in MSA

[1] R. Verdecchia, K. Maggi, L. Scommegna, and E. Vicario, "Tracing the Footsteps of Technical Debt in Microservices: A Preliminary Case Study," in International Workshop on Quality in Software Architecture (QUALIFIER), 2023.

Research Questions

RQ$_1$: *What is the evolution trend of Code Technical Debt in a microservice-based software-intensive system?*

H$_0^{1.1}$: *Technical Debt evolution does not change in time*

H$_0^{1.2}$: *Technical Debt evolution does not present periodic trend*

RQ$_2$: *Is there a relation between Code Technical Debt evolution and number of microservices?*

H$_0^{2}$: *Technical Debt evolution does not depend on number of microservices*

Research Questions

RQ$_1$: *What is the evolution trend of Code Technical Debt in a microservice-based software-intensive system?*

H$_0^{1.1}$: *Technical Debt evolution does not change in time*

H$_0^{1.2}$: *Technical Debt evolution does not present periodic trend*

RQ$_2$: *Is there a relation between Code Technical Debt evolution and number of microservices?*

H$_0^{2}$: *Technical Debt evolution does not depend on number of microservices*

Research Questions

RQ$_1$: *What is the evolution trend of Code Technical Debt in a microservice-based software-intensive system?*

H$_0^{1.1}$: *Technical Debt evolution does not change in time*

H$_0^{1.2}$: *Technical Debt evolution does not present periodic trend*

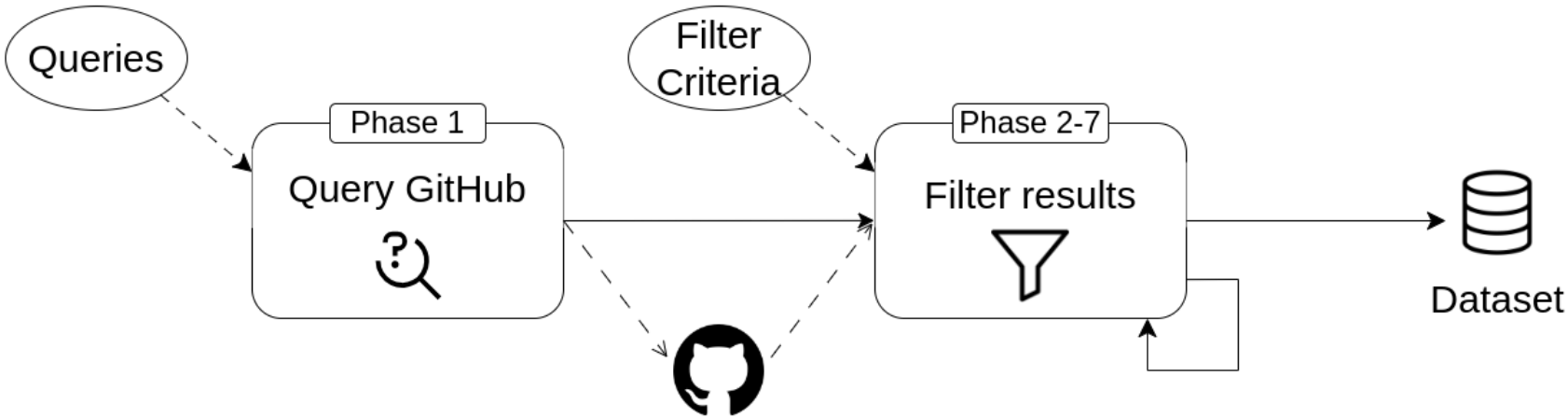
RQ$_2$: *Is there a relation between Code Technical Debt evolution and number of microservices?*

H$_0^2$: *Technical Debt evolution does not depend on number of microservices*

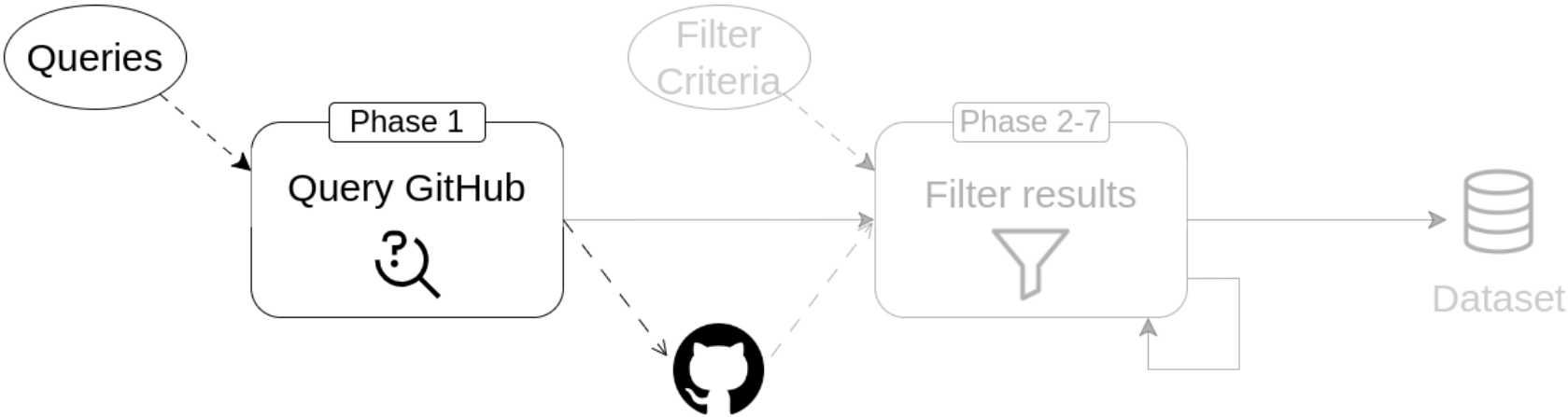
Research Process

Dataset Creation

Workflow



Querying


Queries
Phase 1
Query GitHub
Filter Criteria
Phase 2-7
Filter results
Dataset

Querying

$$language \in \{Java,\ Python,\ C\#,\ Go,\ TypeScript,\ JavaScript\}$$

and

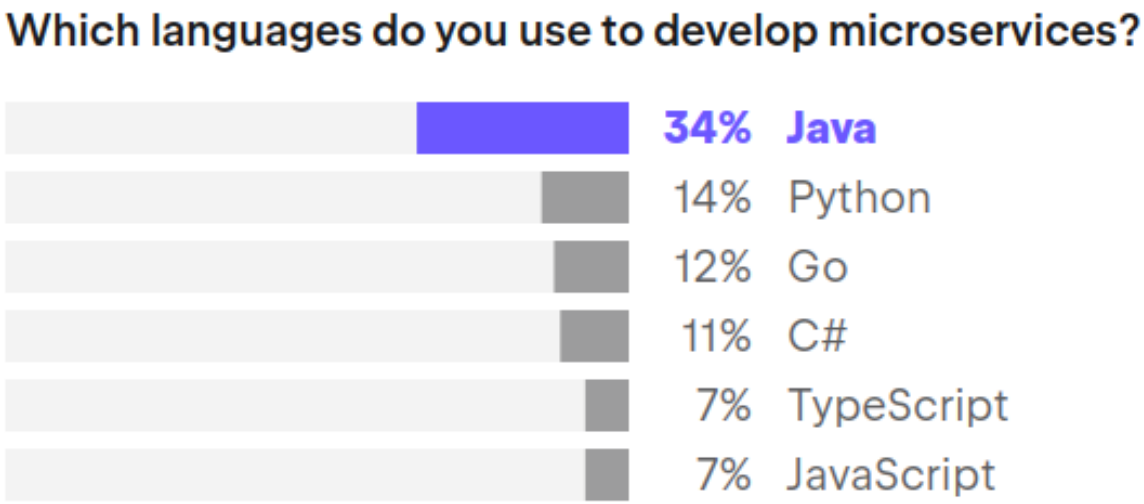
$$\left(\begin{array}{c} topic \in \{\text{microservice(s)},\ \text{microservice(s)-architecture}\} \\ \textbf{or} \\ keyword \in \{\text{microservice}\} \end{array}\right)$$

Querying

$$language \in \{Java,\ Python,\ C\#,\ Go,\ TypeScript,\ JavaScript\}$$

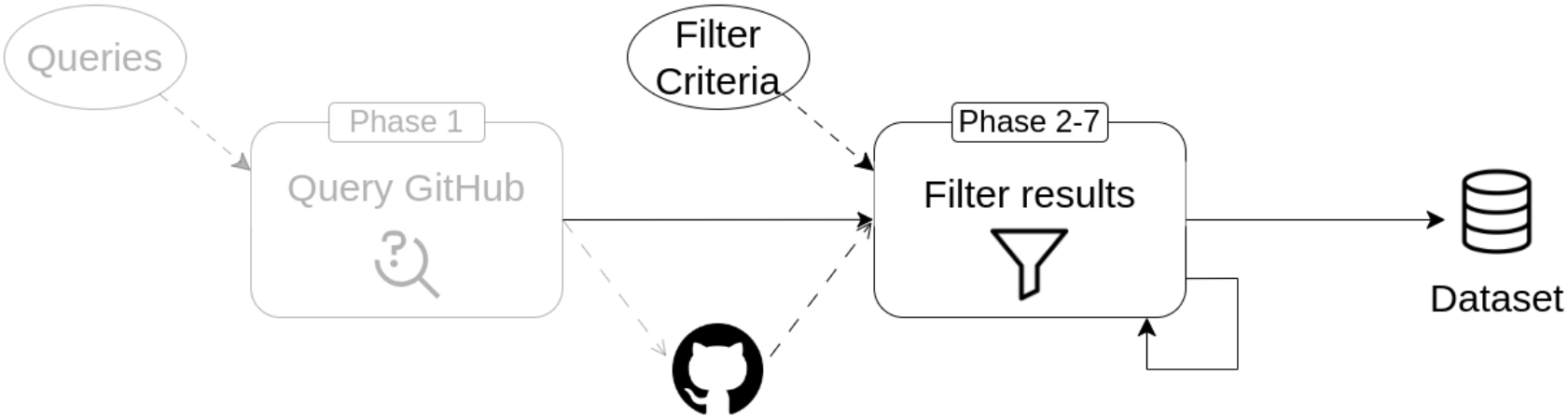
and

$$\left(\begin{array}{c} topic \in \{\text{microservice(s)},\ \text{microservice(s)-architecture}\} \\ \textbf{or} \\ keyword \in \{\text{microservice}\} \end{array} \right)$$



[2] Jetbrains survey (29,269 developers), "The state of developer ecosystem 2022", 2022

Filtering


Queries
Filter Criteria
Phase 1
Query GitHub
Phase 2-7
Filter results
Dataset

Filtering

→ **2491** results from query

- long-living
- industrial-like development
- use of *Docker*
- real-world MSA or industrial MSA demo

→ **46** meet requirements

Filtering

→ **2491** results from query

→ **46** meet requirements

- **interesting evolution** in microservices:
  - enough microservices
  - not too flat evolution
  - microservices since begin

→ **15** selected

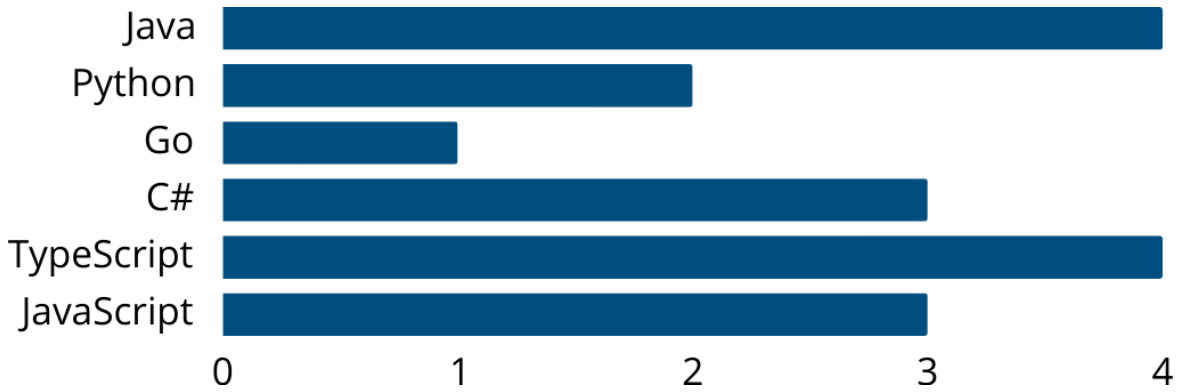
Filtering

→ **2491** results from query

→ **46** meet requirements

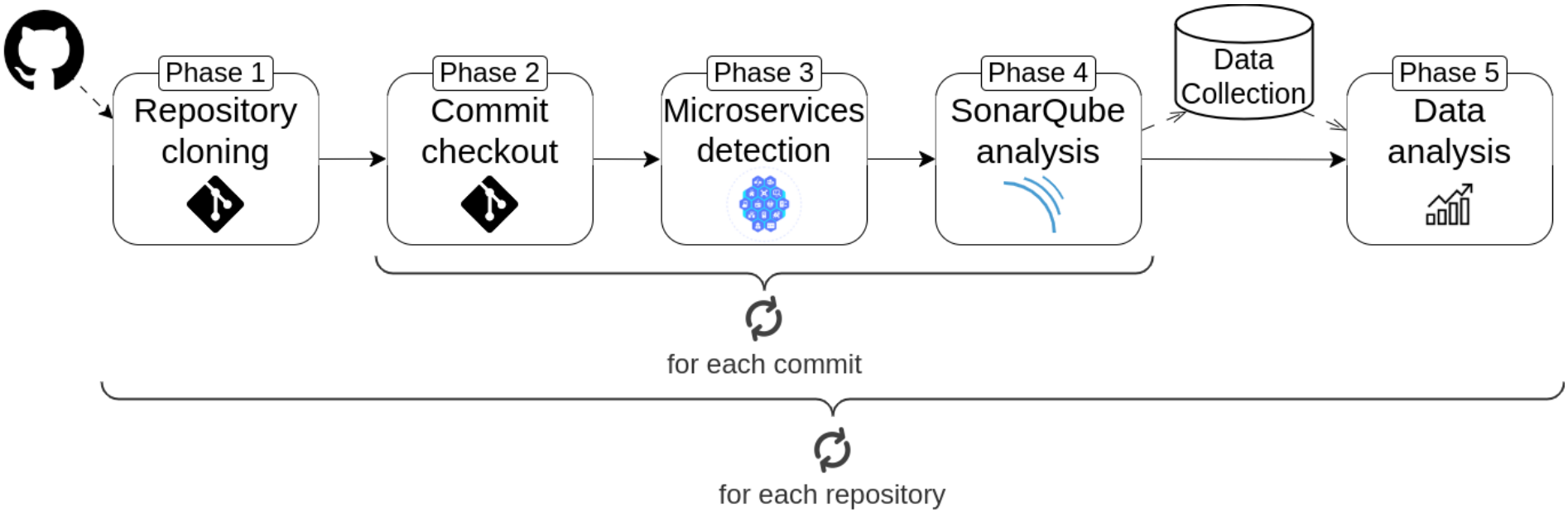
→ **15** selected

AVG # ms: 6,8

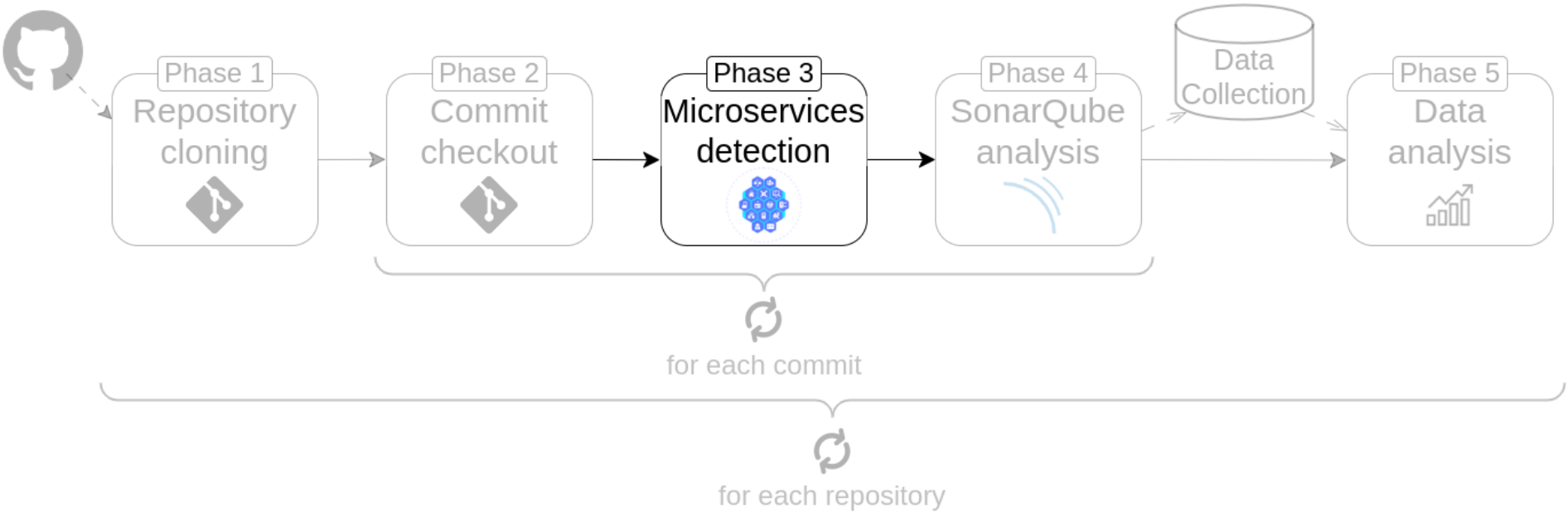


Dataset Analysis

Workflow



Microservices detection



Microservices detection

Problem: enumeration of microservices in a MSA

State of Art: unsuitable or inaccurate methods in literature

Solution: **improving state of art** with a new method

Approach: adoption of a *lightweight static black-box* approach based on parsing of Docker configuration files

Effectiveness: high accuracy from preliminary evaluation

Microservices detection

Problem: enumeration of microservices in a MSA

State of Art: unsuitable or inaccurate methods in literature

Solution: **improving state of art** with a new method

Approach: adoption of a *lightweight static black-box* approach based on parsing of Docker configuration files

Effectiveness: high accuracy from preliminary evaluation

Microservices detection

Problem: enumeration of microservices in a MSA

State of Art: unsuitable or inaccurate methods in literature

Solution: **improving state of art** with a new method

Approach: adoption of a *lightweight static black-box* approach based on parsing of Docker configuration files

Effectiveness: high accuracy from preliminary evaluation

Microservices detection

Problem: enumeration of microservices in a MSA

State of Art: unsuitable or inaccurate methods in literature

Solution: **improving state of art** with a new method

Approach: adoption of a *lightweight static black-box* approach based on parsing of Docker configuration files

Effectiveness: high accuracy from preliminary evaluation

Microservices detection

Problem: enumeration of microservices in a MSA

State of Art: unsuitable or inaccurate methods in literature

Solution: **improving state of art** with a new method

Approach: adoption of a *lightweight static black-box* approach based on parsing of Docker configuration files

Effectiveness: high accuracy from preliminary evaluation

Microservices detection

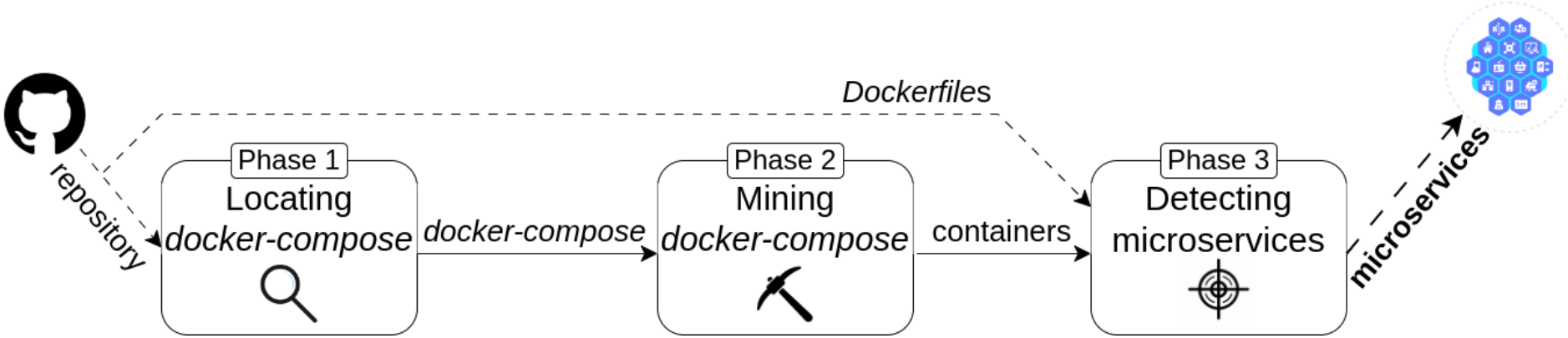
Problem: enumeration of microservices in a MSA

State of Art: unsuitable or inaccurate methods in literature

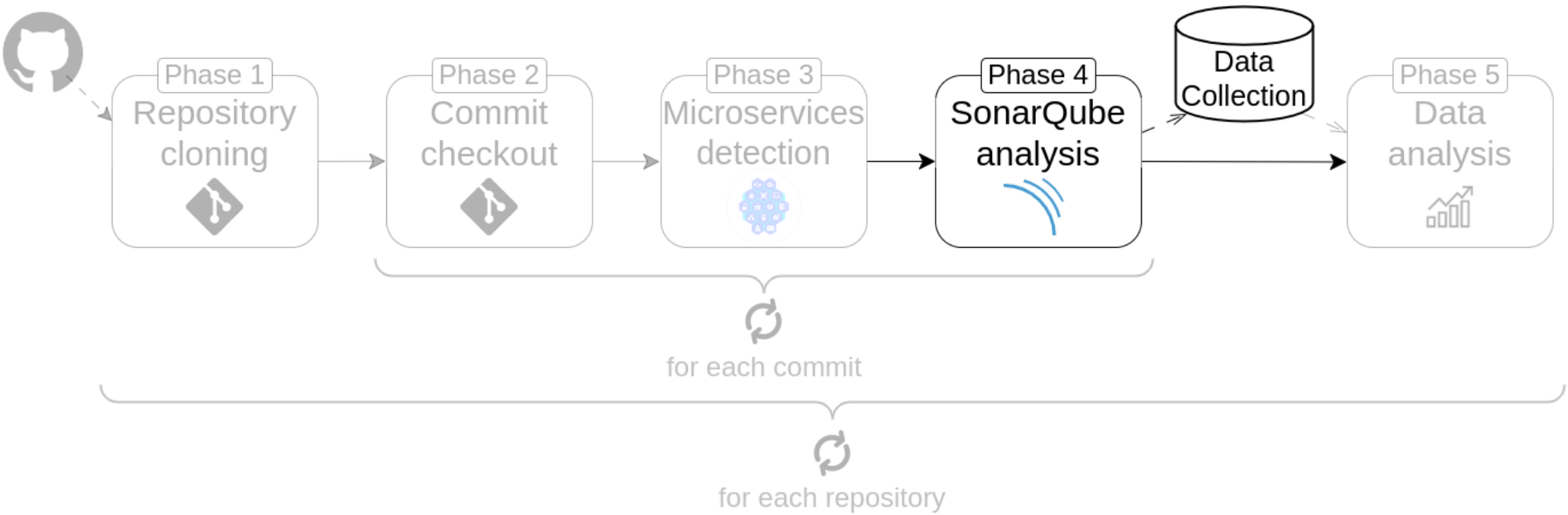
Solution: **improving state of art** with a new method

Approach: adoption of a *lightweight static black-box* approach based on parsing of Docker configuration files

Effectiveness: high accuracy from preliminary evaluation



Code Quality Analysis



Code Quality Analysis

1. **Compilation** (only Java and C#)

   → Forced to ignore non-blocking error: $< 1\%$ commits missed

2. **SonarScanner Analysis**

3. **SonarQube server results**

   → Technical Debt expressed with SQALE index[3]

[3] J.-L. Letouzey, "The SQALE method for evaluating Technical Debt," in 2012 Third International Workshop on Managing Technical Debt (MTD), 2012.

Code Quality Analysis

1. **Compilation** (only Java and C#)

   → Forced to ignore non-blocking error: $< 1\%$ commits missed

2. **SonarScanner Analysis**

3. **SonarQube server results**

   → Technical Debt expressed with SQALE index[3]

[3] J.-L. Letouzey, "The SQALE method for evaluating Technical Debt," in 2012 Third International Workshop on Managing Technical Debt (MTD), 2012.

Code Quality Analysis

1. **Compilation** (only Java and C#)

   → Forced to ignore non-blocking error: $< 1\%$ commits missed

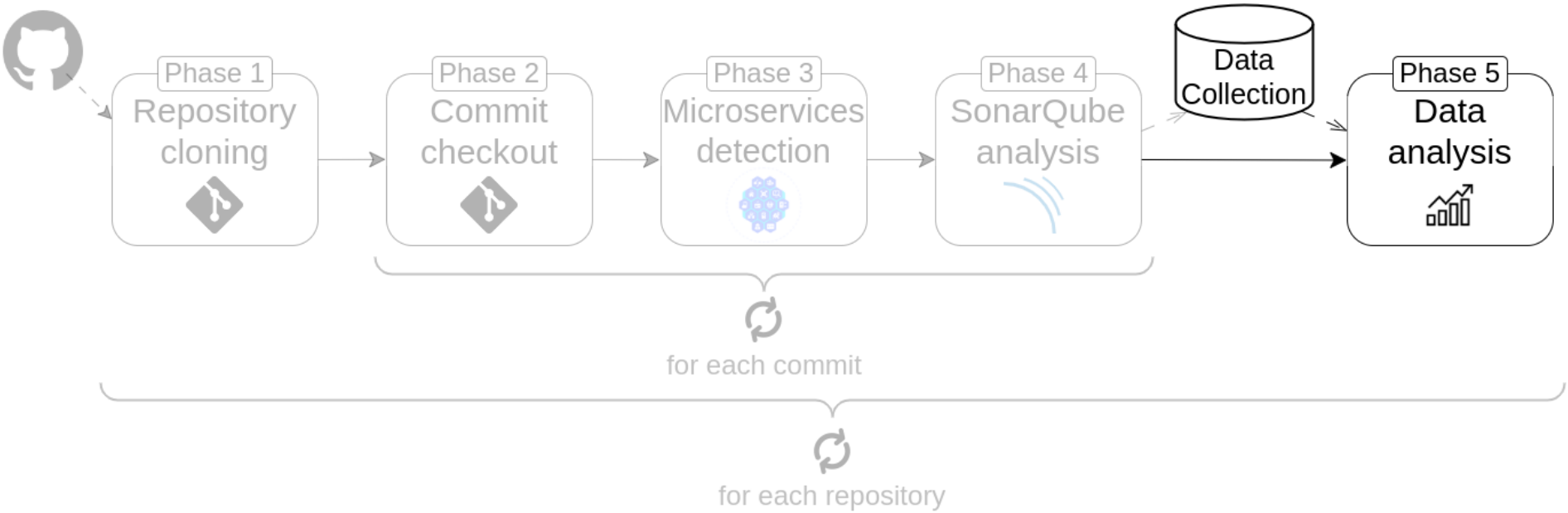
2. **SonarScanner Analysis**

3. **SonarQube server results**

   → Technical Debt expressed with SQALE index[3]

[3] J.-L. Letouzey, "The SQALE method for evaluating Technical Debt," in 2012 Third International Workshop on Managing Technical Debt (MTD), 2012.

Data Analysis



Data Analysis

RQ_1:

- *Mann-Kendall* **test for trend**
  - *LOESS regression* for **graphical means analysis of trend**
- **manual inspection** of top TD hotspots
- *Ollech&Webel* combined **test for seasonality**
  - *STL* **decomposition** of TD evolution

RQ_2:

- *Cross*-**Correlation between TD and microservices**
  - *Granger Causality* **test for causal relationship**
- *Cross*-**Correlation between TD growth rate and microservices**

Data Analysis

RQ_1:

- *Mann-Kendall* **test for trend**
  - *LOESS regression* for **graphical means analysis of trend**
- **manual inspection** of top TD hotspots
- *Ollech&Webel* combined **test for seasonality**
  - *STL* **decomposition** of TD evolution

RQ_2:

- *Cross*-**Correlation between TD and microservices**
  - *Granger Causality* **test for causal relationship**
- *Cross*-**Correlation between TD growth rate and microservices**

Data Analysis

RQ_1:

- *Mann-Kendall* **test for trend**
  - *LOESS regression* for **graphical means analysis of trend**
- **manual inspection** of top TD hotspots
- *Ollech&Webel* combined **test for seasonality**
  - *STL* **decomposition** of TD evolution

RQ_2:

- *Cross*-**Correlation between TD and microservices**
  - *Granger Causality* **test for causal relationship**
- *Cross*-**Correlation between TD growth rate and microservices**

RQ_1: trend

General (very) strong trend to grow

# systems	Trend
8	very strong growing
3	strong growing
1	slight growing
1	slight shrinking

$H_0^{1.1}$: *Technical Debt evolution does not change in time*

$\rightarrow$ **REJECTED**

RQ_1: trend

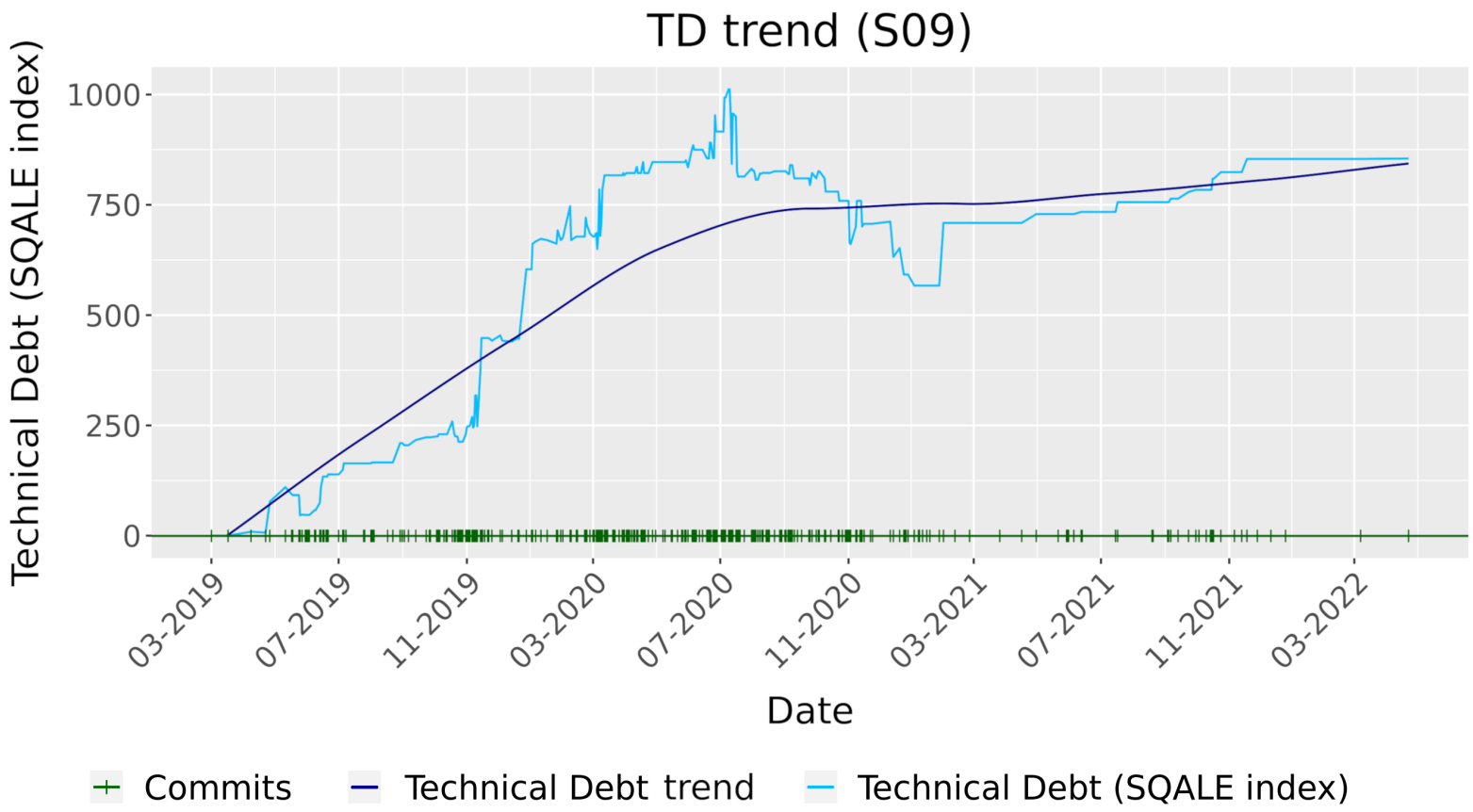
General (very) strong trend to grow

# systems	Trend
8	very strong growing
3	strong growing
1	slight growing
1	slight shrinking

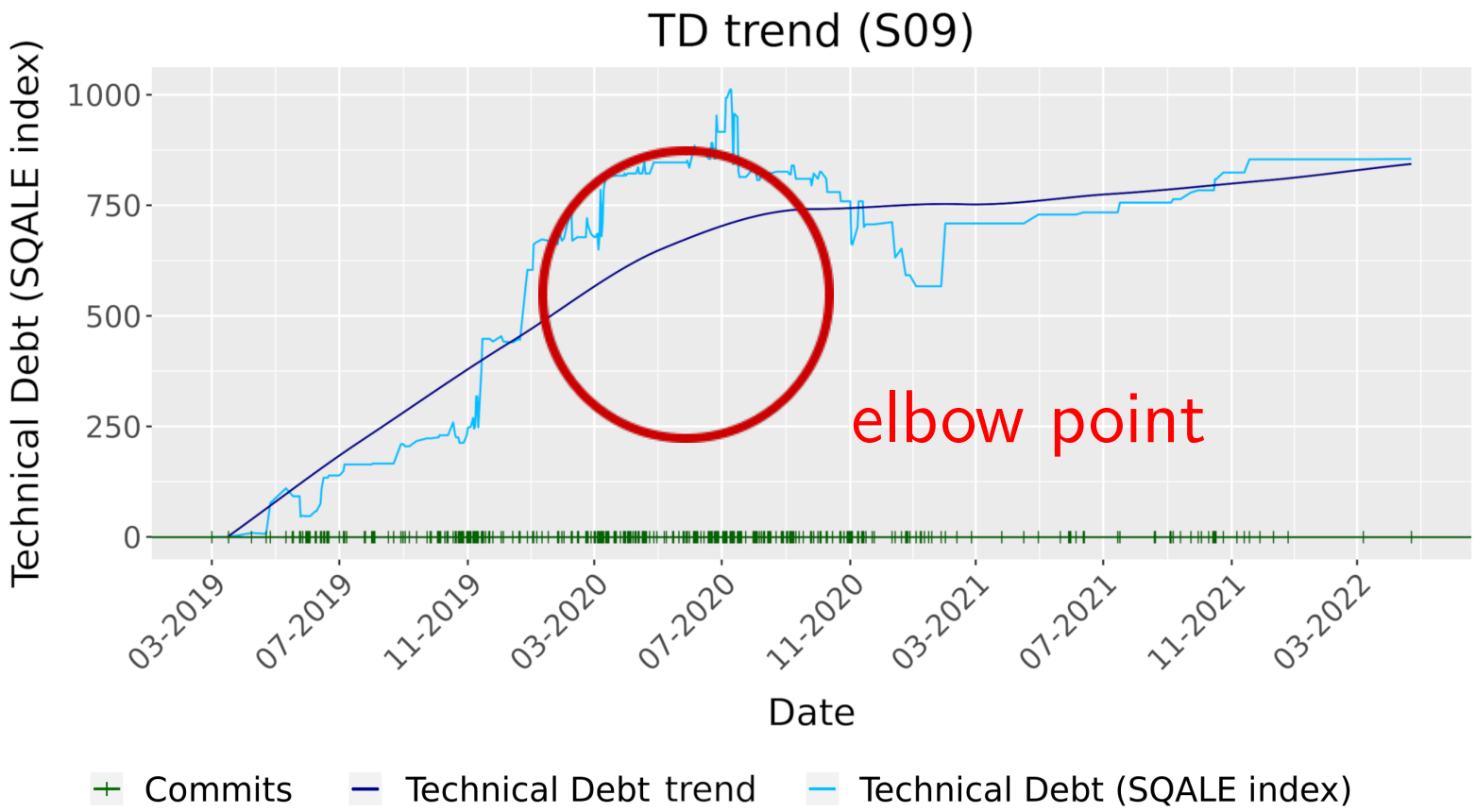
$\mathbf{H_0^{1.1}}$: *Technical Debt evolution does not change in time*

$\rightarrow$ **REJECTED**

RQ_1: trend

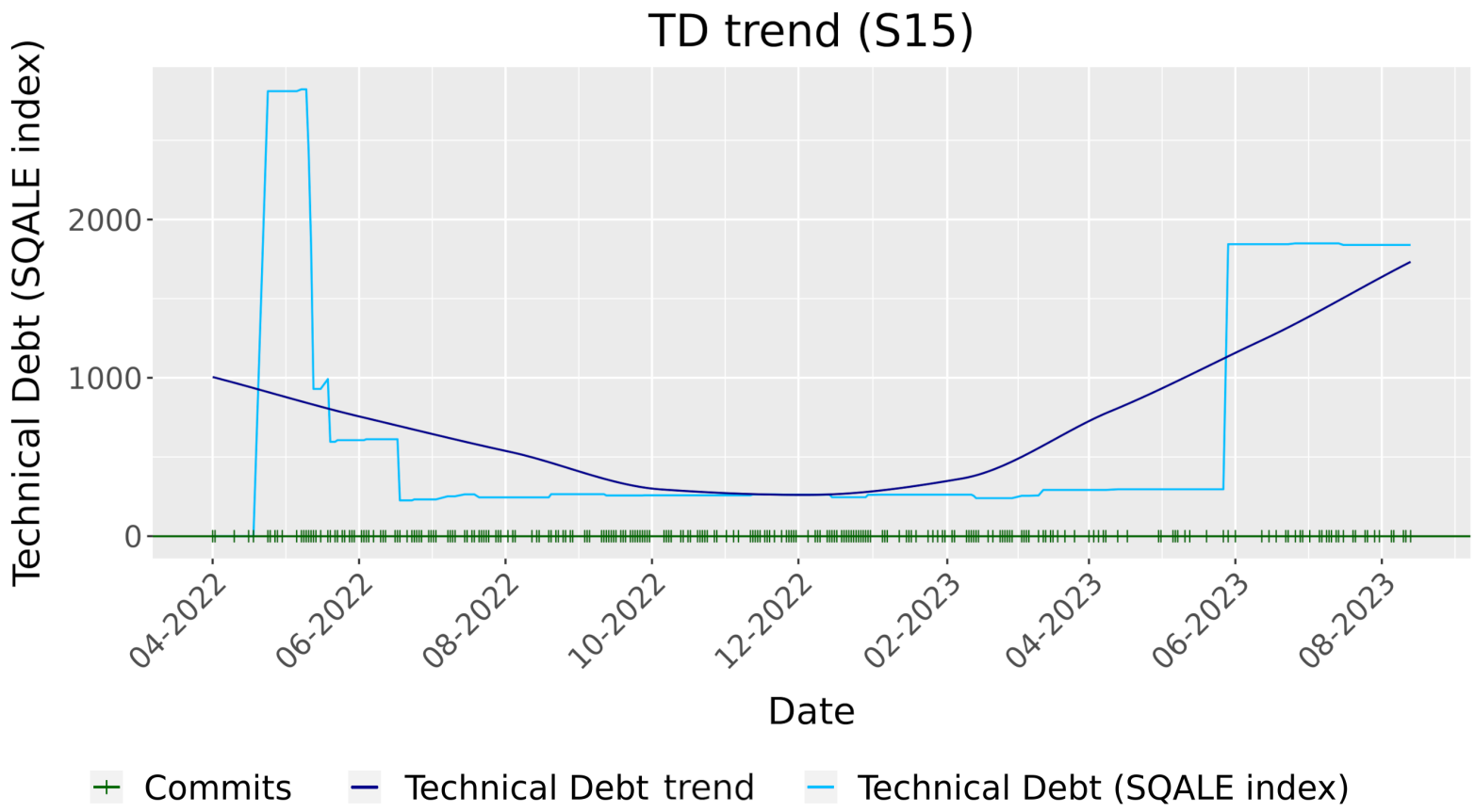

TD trend (S09)
Technical Debt (SQALE index)
1000
750
500
250
0
03-2019
07-2019
11-2019
03-2020
07-2020
11-2020
03-2021
07-2021
11-2021
03-2022
Date
Commits
Technical Debt trend
Technical Debt (SQALE index)

RQ_1: trend



Conjecture: passage from development to maintenance phase
Evidences: reduced commit frequency, notable plateaus

RQ_1: trend


TD trend (S15)
Technical Debt (SQALE index)
2000
1000
0
04-2022
06-2022
08-2022
10-2022
12-2022
02-2023
04-2023
06-2023
08-2023
Date
Commits
Technical Debt trend
Technical Debt (SQALE index)

RQ_1: hotspots investigation

Activities that can introduce TD:

- **add components** (microservices, infrastructural elements, UI, ...)
  `add implementation for policy-service`
  `dashboard service initial release`
- **evolve business logic**
  `make update sequence atomically updated [...]`
  `update product rest api`
- add/upgrade dependencies
  `refactor remote catalog/config events to not require dependency [...]`
  `disable jaeger`
- refactoring
  `refactor comx`
  `code refactoring`

"Size" of commits: apparently no strong correlation with TD variations

RQ_1: seasonality

No seasonality in any systems

> $H_0^{1.2}$: *Technical Debt evolution does not present periodic trend*
>
> → **ACCEPTED**

RQ_1: seasonality

No seasonality in any systems

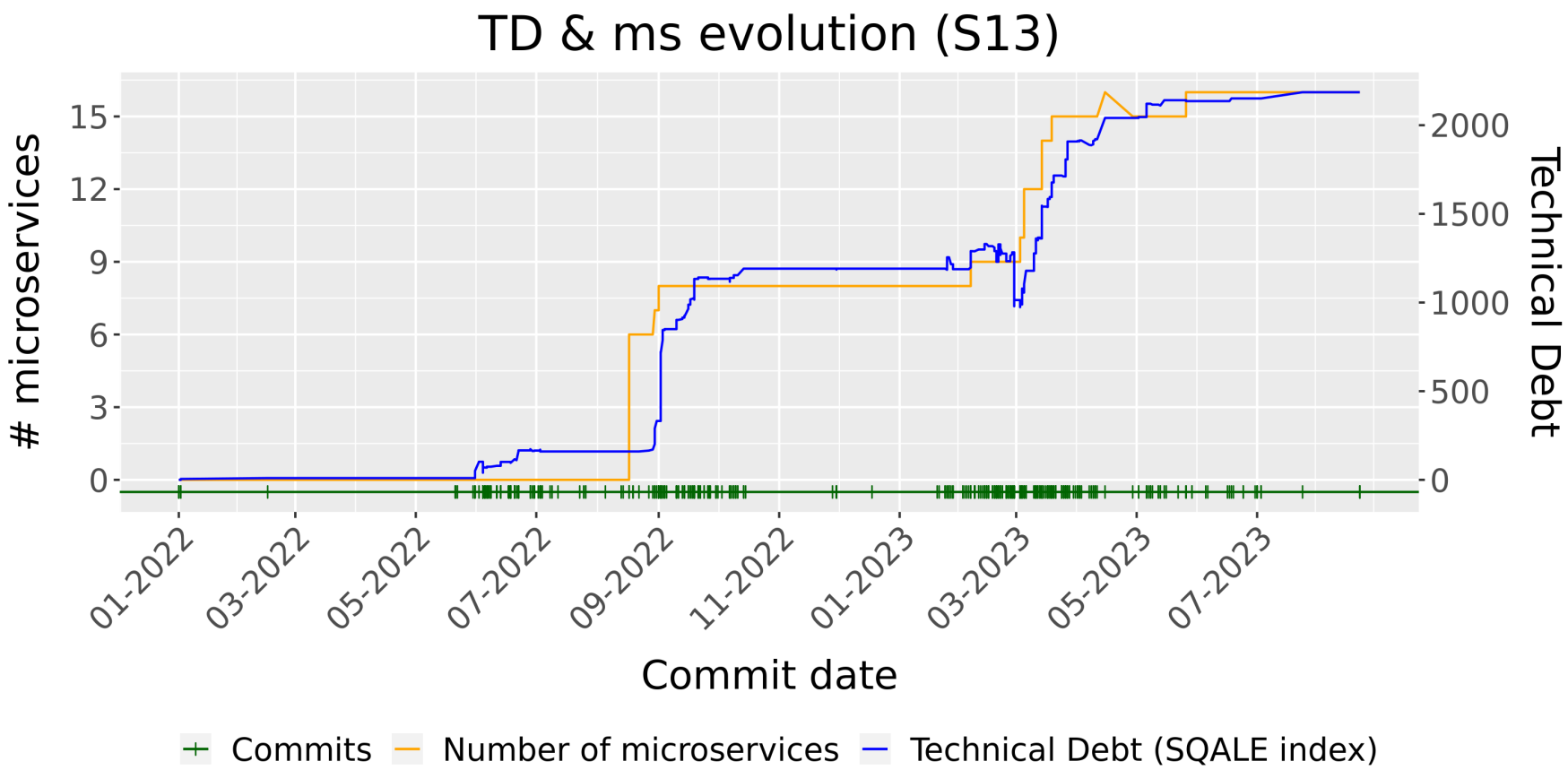
> $\mathbf{H_0^{1.2}}$: *Technical Debt evolution does not present periodic trend*
>
> $\rightarrow$ **ACCEPTED**

RQ_1 answer

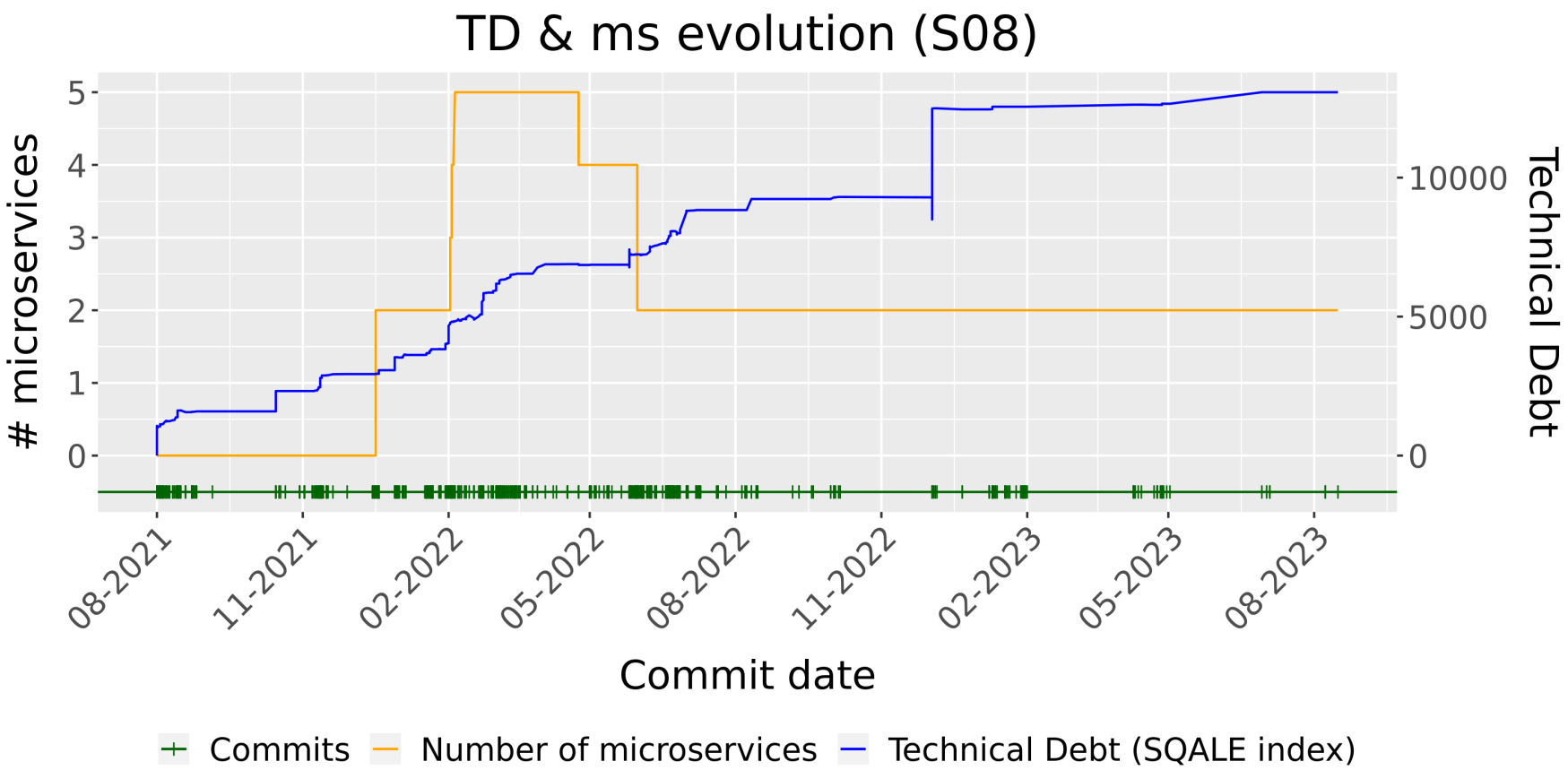
RQ_1 answer (*Technical Debt evolution trend in MSA*)

- Technical Debt **overall increasing trend in time**, above all in the initial development phase
- Technical Debt **variations caused by a variety of activities**, first of all adding components and evolving business logic
- Technical Debt presents **no seasonality**

RQ_2: correlation?



RQ_2: correlation?



RQ_2: correlation and causality

General correlation (with phase shift) between TD and microservices

# systems	Correlation
5	very strong
4	strong
4	absent or very weak

Not general causality between TD and microservices

# systems	Granger causality
4	Yes
5	No

H_0^2: *Technical Debt evolution does not depend on number of microservices*

$\rightarrow$ **REJECTED**

RQ_2: correlation and causality

General correlation (with phase shift) between TD and microservices

# systems	Correlation
5	very strong
4	strong
4	absent or very weak

Not general causality between TD and microservices

# systems	Granger causality
4	Yes
5	No

H_0^2: *Technical Debt evolution does not depend on number of microservices*

→ **REJECTED**

RQ_2: correlation and causality

General correlation (with phase shift) between TD and microservices

# systems	Correlation
5	very strong
4	strong
4	absent or very weak

Not general causality between TD and microservices

# systems	Granger causality
4	Yes
5	No

$\mathbf{H_0^2}$: *Technical Debt evolution does not depend on number of microservices*

$\rightarrow$ **REJECTED**

RQ_2: correlation (growth rate)

Not significant correlation between TD growth rate and microservices

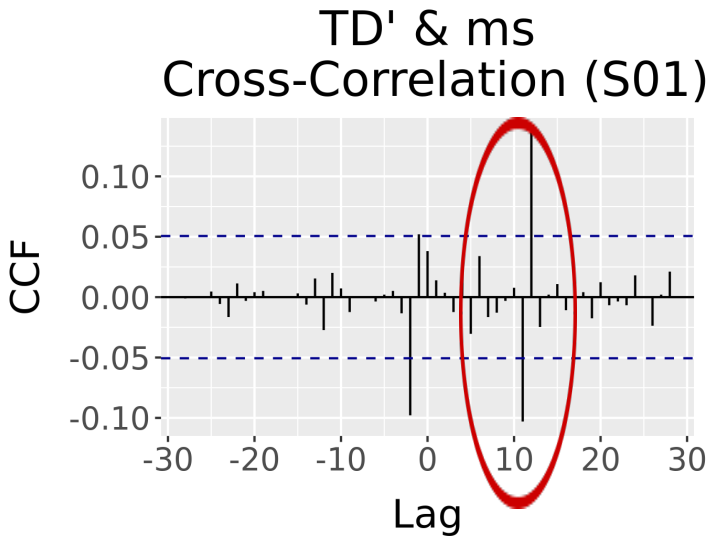
# systems	Cross-Correlation
3	very strong
3	strong
7	absent or very weak

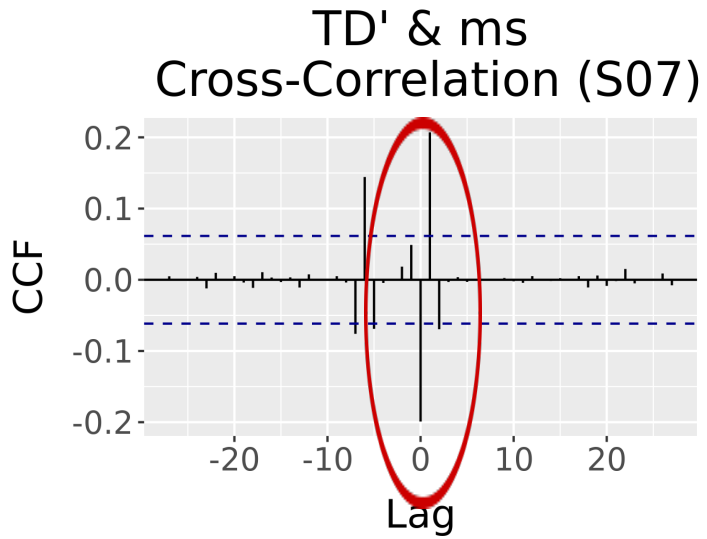
Conjecture: consequence of adherence to MSA principle of independence

RQ_2: correlation (growth rate)

Not significant correlation between TD growth rate and microservices

# systems	Cross-Correlation
3	very strong
3	strong
7	absent or very weak



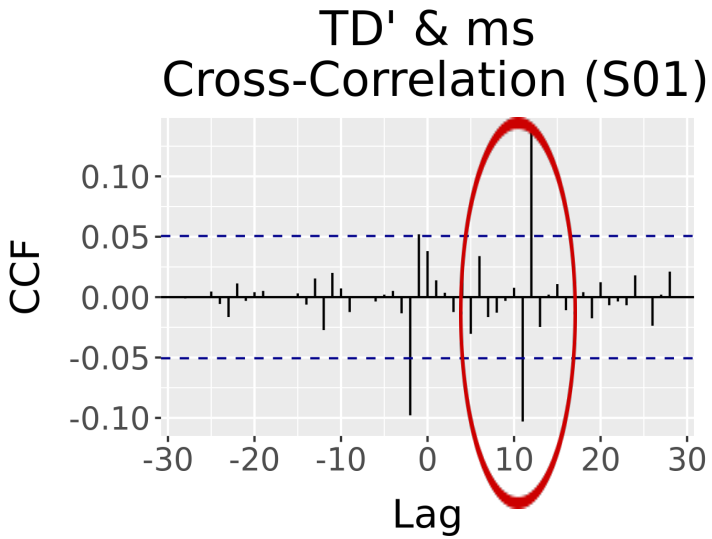


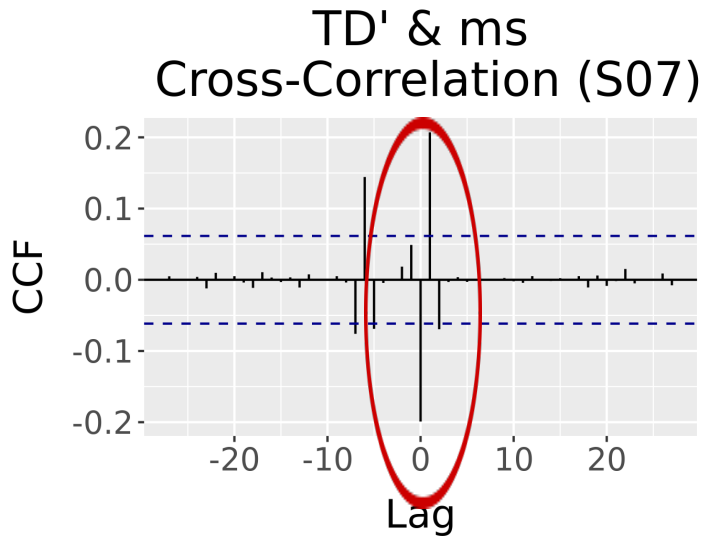
Conjecture: consequence of adherence to MSA principle of independence

RQ_2: correlation (growth rate)

Not significant correlation between TD growth rate and microservices

# systems	Cross-Correlation
3	very strong
3	strong
7	absent or very weak





Conjecture: consequence of adherence to MSA principle of independence

RQ_2 answer

RQ_2 answer (*Relation between Technical Debt and microservices*)

- Technical Debt and microservices number are **generally strongly correlated** (with a phase shift)
- In some cases also **a causality relation exists**
- **Addition or removal of a microservice does not impact** the growing rate of Technical Debt

Conclusion

Discussion:

- **maintaining a consistent level of TD is possible by monitoring it**, but its increase might be inevitable as the system grows
- developers should be **aware of the potential TD they incur with a variety of development activities**
- adherence to MSA principles can help to **keep TD compartmentalized within microservices**

Future Work:

- systematic evaluation and comparison of microservice detection method
- individual contribution of each microservice
- in-depth systematic analysis of TD hotspots

Conclusion

Discussion:

- **maintaining a consistent level of TD is possible by monitoring it**, but its increase might be inevitable as the system grows
- developers should be **aware of the potential TD they incur with a variety of development activities**
- adherence to MSA principles can help to **keep TD compartmentalized within microservices**

Future Work:

- systematic evaluation and comparison of microservice detection method
- individual contribution of each microservice
- in-depth systematic analysis of TD hotspots

Beyond this thesis

The "Cloud Native GeoServer" case study

Extension[4] with interview to leading developer just submitted:

- **results confirmed** also from its point of view
- **introduced TD monitoring** into its pipeline

"*The quantitative analysis was quite **enlightening to me**. I wanted to include a static code analysis for a long time. And maybe it would have never happen [...] if I didn't have this feedback from you*".

[4] R. Verdecchia, K. Maggi, L. Scommegna, and E. Vicario, "Technical Debt in Microservices: A Mixed-Method Case Study," *Under review*.

Grazie per l'attenzione

Candidato
Kevin Maggi

Relatori
Dott.Ric. Roberto Verdecchia
Prof. Enrico Vicario

Correlatore
Dott.Ric. Leonardo Scommegna

Conclusion

Discussion:

- **maintaining a consistent level of TD is possible by monitoring it**, but its increase might be inevitable as the system grows
- developers should be **aware of the potential TD they incur with a variety of development activities**
- adherence to MSA principles can help to **keep TD compartmentalized within microservices**

Future Work:

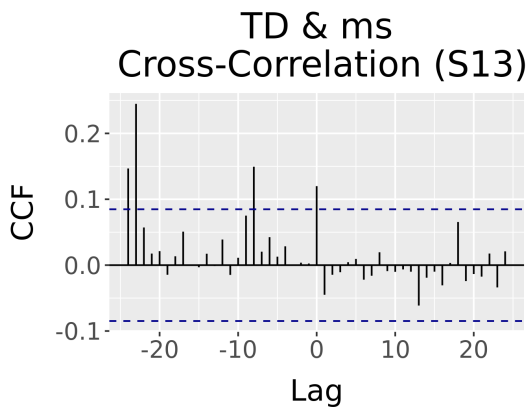
- systematic evaluation and comparison of microservice detection method
- individual contribution of each microservice
- in-depth systematic analysis of TD hotspots

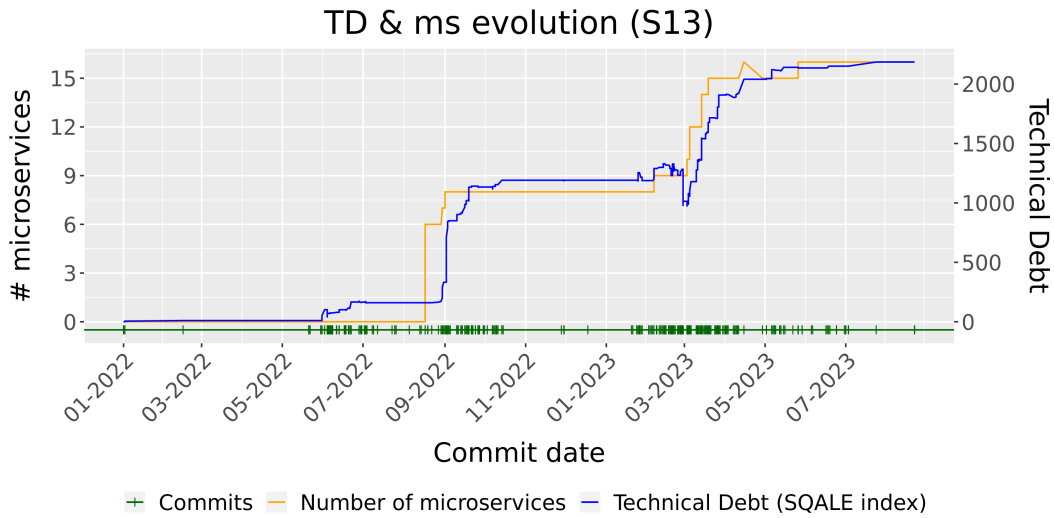
Mann-Kendall trend test

ID	Kendall's τ
$S01$, $S02$, $S03$, $S04$, $S07$, $S08$, $S10$, $S13$	$\tau \geq 0,79$
$S05$, $S09$, $S12$	$0,49 \geq \tau \geq 0,59$
$S14$	$\tau = -0,58$
$S15$	$\tau = 0,23$

Cross-Correlation TD/microservices

ID	Cross-Correlation (at some lag)
$S01$, $S02$, $S09$, $S10$, $S15$	very strong ($>>$ confidence level)
$S07$, $S12$, $S13$, $S14$	strong ($>$ confidence level)
$S03$, $S04$, $S05$, $S08$	absent or very weak ($<$ *or* $\approx$ confidence level)





Granger Causality test

ID	Granger causality
$S01$, $S07$, $S10$, $S15$	Yes
$S02$, $S03$, $S09$, $S12$, $S13$	No

Cross-Correlation TD growth rate/microservices

ID	Cross-Correlation (at some lag)
*S*07, *S*09, *S*10	very strong ($>>$ confidence level)
*S*01, *S*02, *S*12	strong ($>$ confidence level)
*S*03, *S*04, *S*05, *S*08, *S*13, *S*14, *S*15	absent or very weak ($<$ *or* $\approx$ confidence level)