Tampereen yliopisto

Continuous monitoring of teams performance and technical debt”

Davide Taibi
Associate Professor

@davidetaibi

Tampereen yliopisto

The Clowee Research Group

Service/Cloud Based Architecture optimization Anomaly detection

Francesco Lomio
Ph.D. student

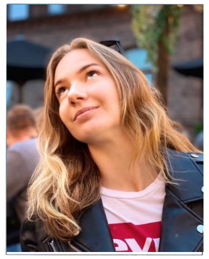
Ather Fayyaz
Ph.D. student (industrial)

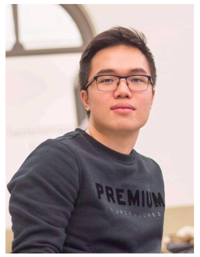
Fouzia Kahn
Ph.D. Student

Technical Debt, Software Quality

Nyyti Saarimäki
Ph.D. student

Open Ph.D. position


Savanna Lujan
Res. Assistant


Hung Nguyen
Res. Assistant

Open Source Quality and Assessment

Sergio Moreschini
Ph.D. student

Xiaozhou Li
Ph.D. student

Zheying Zhang
Lecturer

Collaborations / Projects

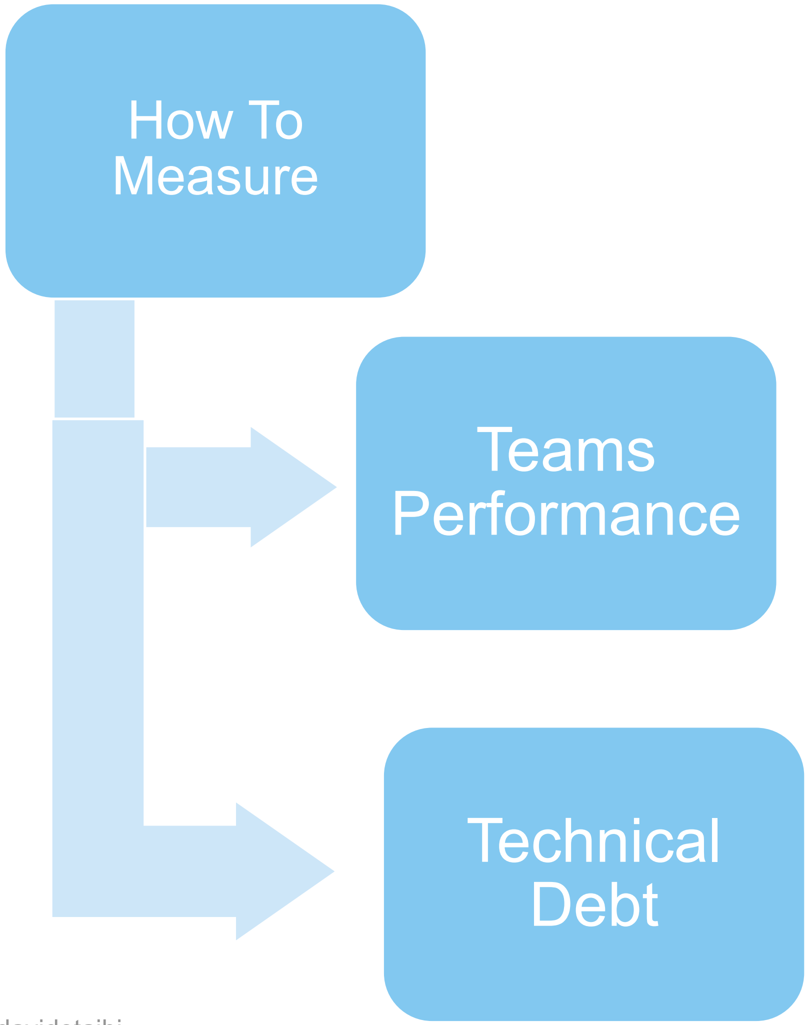
NOKIA

VENDASTA

HUAWEI

ABB

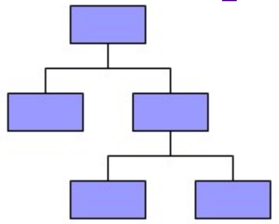
The Problem

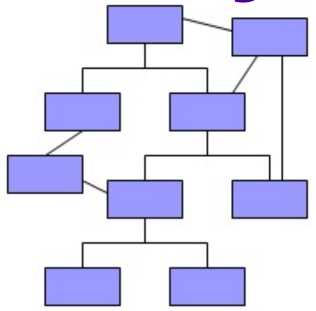


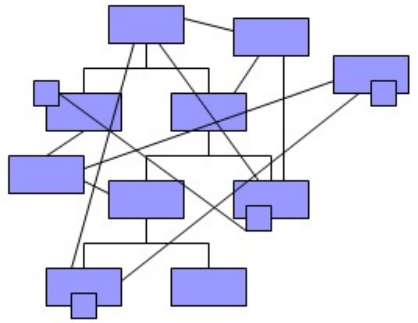
- KPI
- Metrics

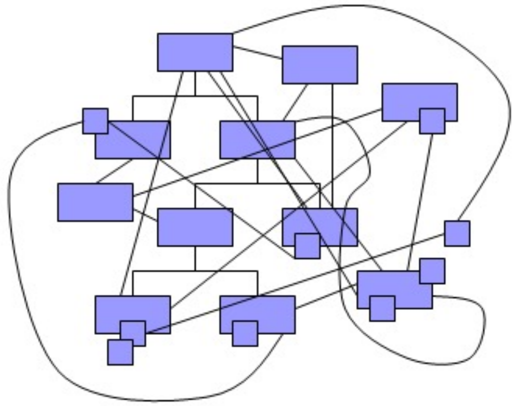
- What is technical debt
- When it accumulates
- How to Prioritize TD vs new Features?

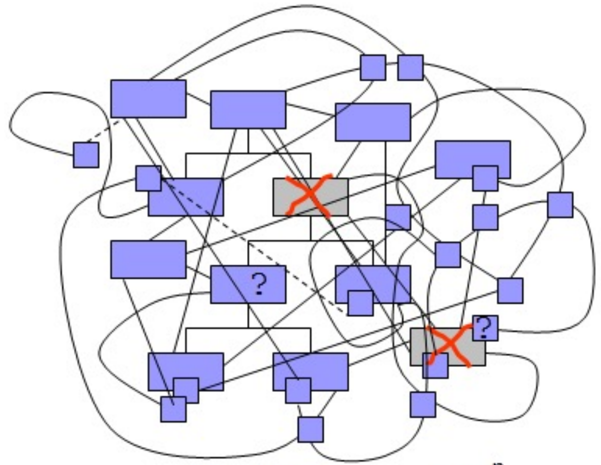
What happen to your source code


Cost of change: C


Cost of change: C + *n*


Cost of change: C x *n*


Cost of change: C^n



Cost of change: C^{n^n}

Technical Debt Definition

Debt = sub-optimal solution

- Save time by non-applying the optimal solution
    - You gain a benefit now (borrow money)
    - But, you pay the consequences later (you will pay the interest)

Technical Debt

Every minute spent on not-quite-right activities results in interest on that debt.

How to improve team performance while keeping technical debt under control?

Team Performance Measurement

- Team Productivity
  - Lines of code
  - User stories per month
  - #Issues per month
  - …

Organizational performance ↔ Software delivery performance

Measuring Team Performance

THE SCIENCE OF LEAN SOFTWARE AND DEVOPS
ACCELERATE
Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD
Jez Humble, and Gene Kim
with forewords by Martin Fowler and Courtney Kissler
and a case study contributed by Steve Bell and Karen Whitley Bell
Accelerate
Building and Scaling High Performing Technology Organizations

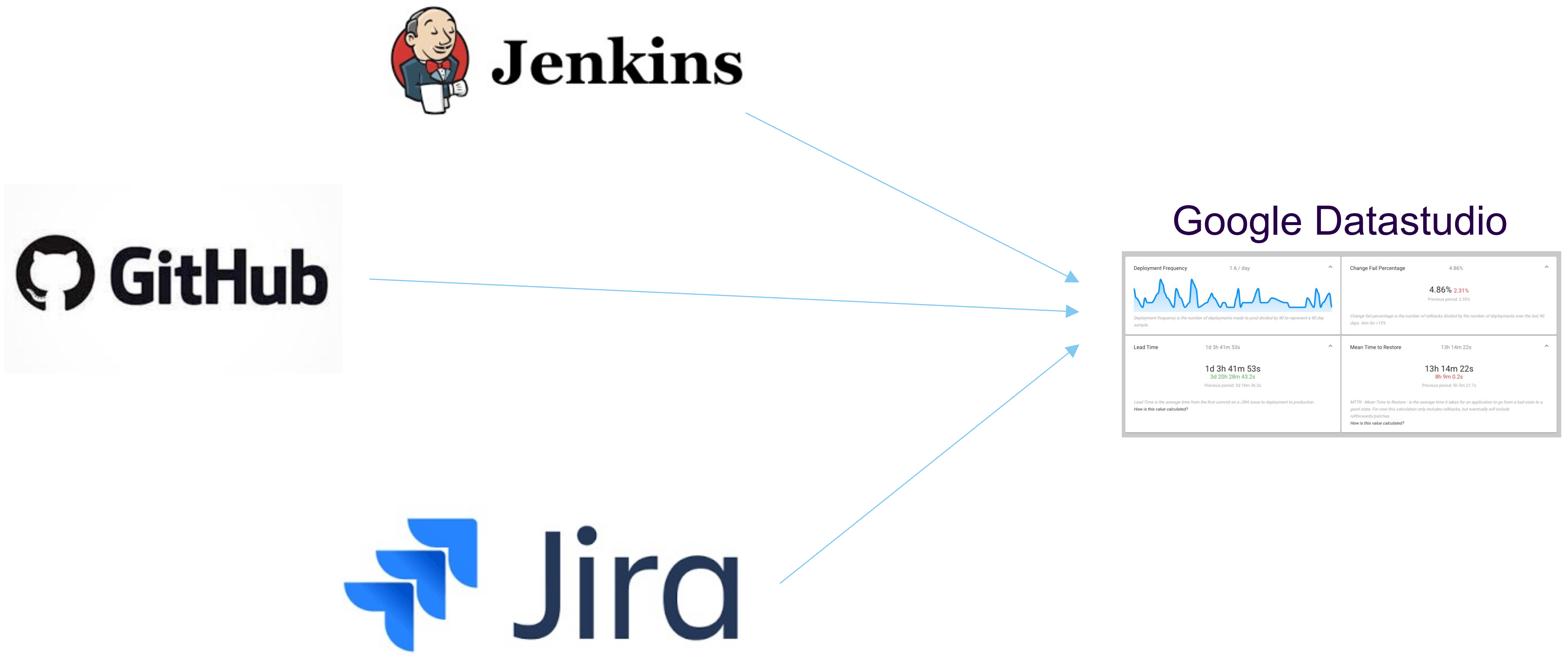
Accelerate metrics

- **Lead Time** — the average amount of time it takes from the time code is checked in to the version control system to the point in time where it is deployed to production.
- **Deployment Frequency** — the number of times deploys to production occur in a time period.
- **Mean Time to Restore** — how long it takes to resolve or rollback an error in production.
- **Change Fail Percentage** — what percentage of changes to production (software releases and configuration changes) fail.

Accelerate Metrics Use Case

- Applied by >40 teams
- 120 microservices
- Each team is responsible of 2-5 microservices

The Accelerate Metrics Dashboard



Deployment Frequency 1.6 / day

Deployment frequency is the number of deployments made to prod divided by 90 to represent a 90 day sample.

Change Fail Percentage 4.86%

4.86% 2.31%

Previous period: 2.55%

Change fail percentage is the number of rollbacks divided by the number of deployments over the last 90 days. Aim for <15%

Lead Time 1d 3h 41m 53s

1d 3h 41m 53s

3d 20h 28m 43.2s

Previous period: 5d 10m 36.2s

Lead Time is the average time from the first commit on a JIRA issue to deployment to production.

How is this value calculated?

Mean Time to Restore 13h 14m 22s

13h 14m 22s

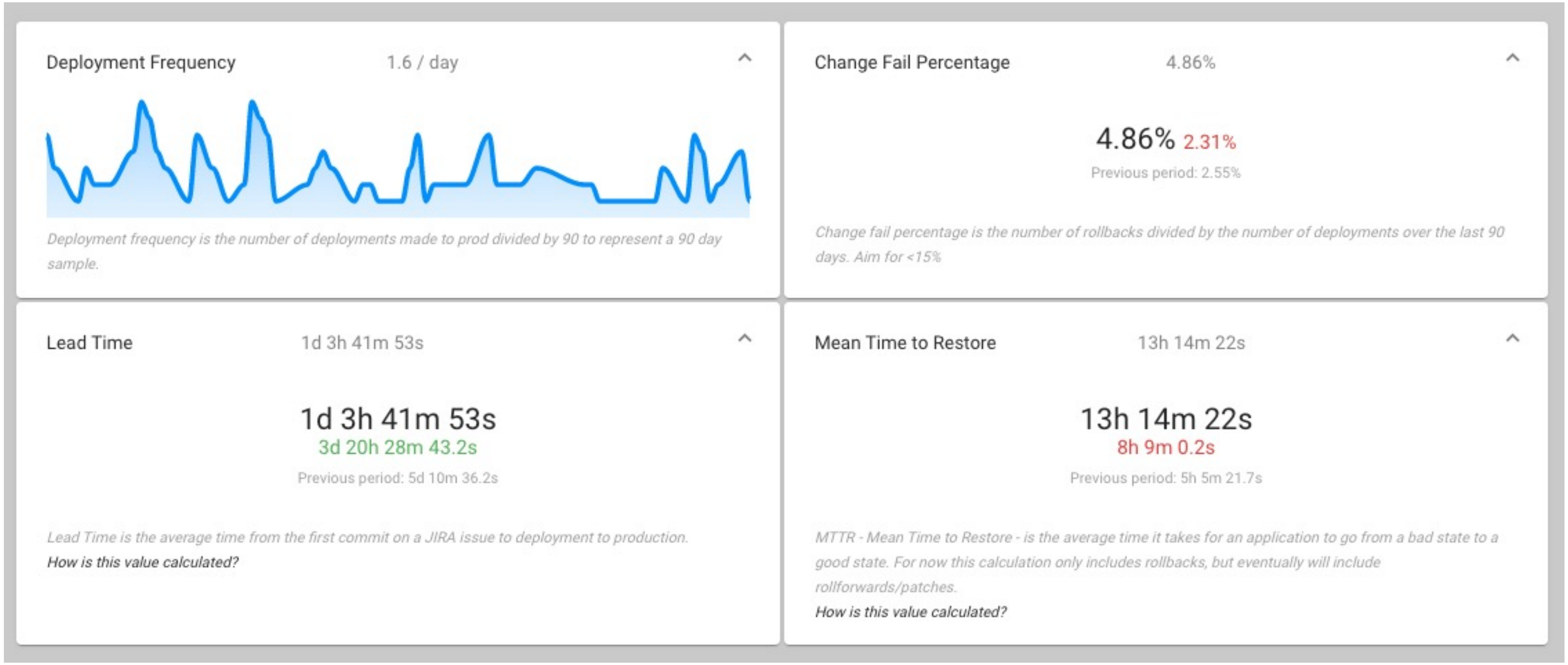
8h 9m 0.2s

Previous period: 5h 5m 21.7s

MTTR - Mean Time to Restore - is the average time it takes for an application to go from a bad state to a good state. For now this calculation only includes rollbacks, but eventually will include rollforwards/patches.

How is this value calculated?

Accelerate Metrics



Deployment Frequency 1.6 / day

Deployment frequency is the number of deployments made to prod divided by 90 to represent a 90 day sample.

Change Fail Percentage 4.86%

4.86% 2.31%

Previous period: 2.55%

Change fail percentage is the number of rollbacks divided by the number of deployments over the last 90 days. Aim for <15%

Lead Time 1d 3h 41m 53s

1d 3h 41m 53s

3d 20h 28m 43.2s

Previous period: 5d 10m 36.2s

Lead Time is the average time from the first commit on a JIRA issue to deployment to production.

How is this value calculated?

Mean Time to Restore 13h 14m 22s

13h 14m 22s

8h 9m 0.2s

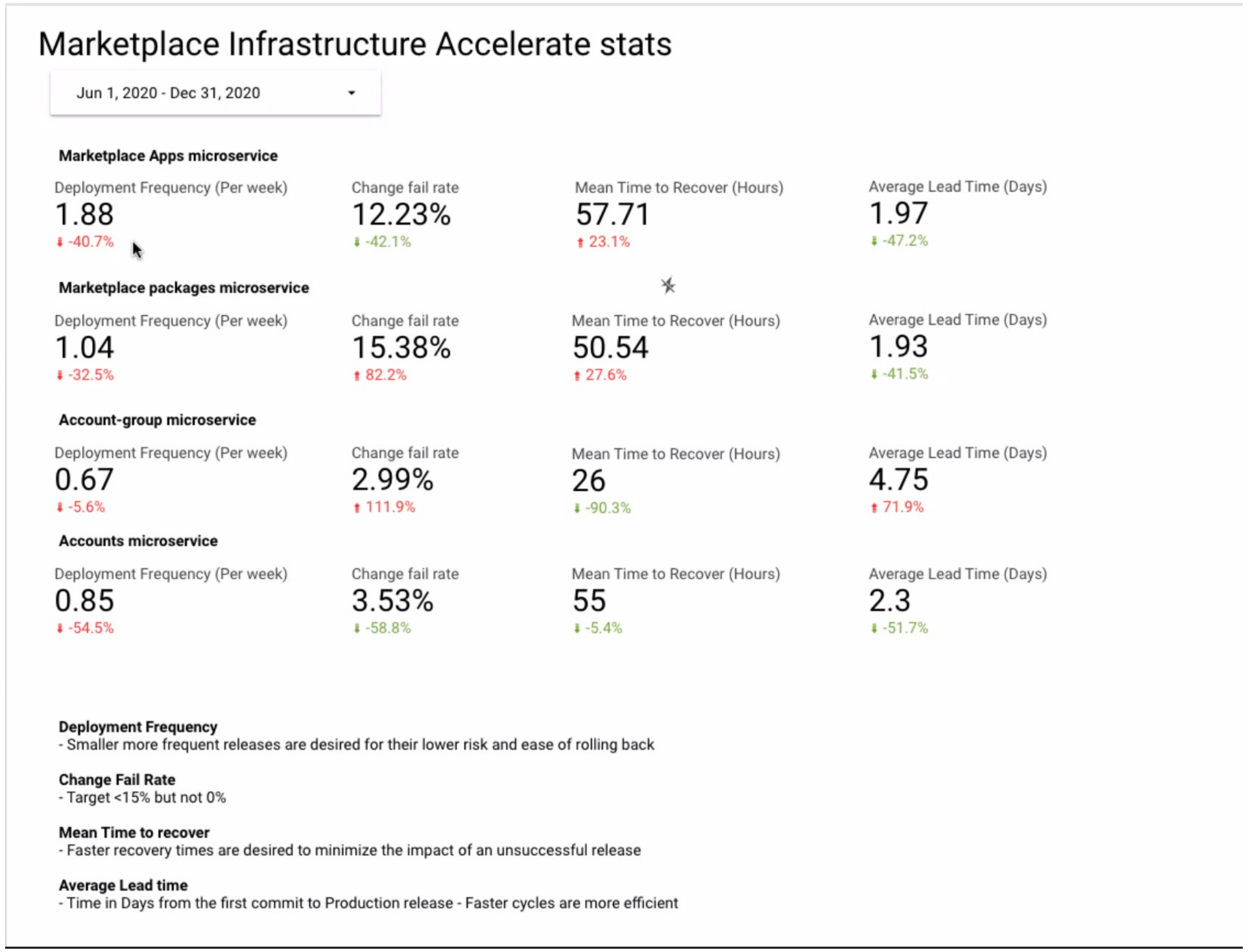
Previous period: 5h 5m 21.7s

MTTR - Mean Time to Restore - is the average time it takes for an application to go from a bad state to a good state. For now this calculation only includes rollbacks, but eventually will include rollforwards/patches.

How is this value calculated?

Tampereen yliopisto

Example of Detailed Dashboard



Marketplace Infrastructure Accelerate stats

Jun 1, 2020 - Dec 31, 2020

Marketplace Apps microservice

Deployment Frequency (Per week)	Change fail rate	Mean Time to Recover (Hours)	Average Lead Time (Days)
1.88	12.23%	57.71	1.97
↓ -40.7%	↓ -42.1%	↑ 23.1%	↓ -47.2%

Marketplace packages microservice

Deployment Frequency (Per week)	Change fail rate	Mean Time to Recover (Hours)	Average Lead Time (Days)
1.04	15.38%	50.54	1.93
↓ -32.5%	↑ 82.2%	↑ 27.6%	↓ -41.5%

Account-group microservice

Deployment Frequency (Per week)	Change fail rate	Mean Time to Recover (Hours)	Average Lead Time (Days)
0.67	2.99%	26	4.75
↓ -5.6%	↑ 111.9%	↓ -90.3%	↑ 71.9%

Accounts microservice

Deployment Frequency (Per week)	Change fail rate	Mean Time to Recover (Hours)	Average Lead Time (Days)
0.85	3.53%	55	2.3
↓ -54.5%	↓ -58.8%	↓ -5.4%	↓ -51.7%

Deployment Frequency
- Smaller more frequent releases are desired for their lower risk and ease of rolling back

Change Fail Rate
- Target <15% but not 0%

Mean Time to recover
- Faster recovery times are desired to minimize the impact of an unsuccessful release

Average Lead time
- Time in Days from the first commit to Production release - Faster cycles are more efficient

Accelerate Metrics and Technical Debt

- Developers are free to take technical decisions
  - Keep code quality under control
  - Keep accelerate metrics under a certain threshold

Tampereen yliopisto

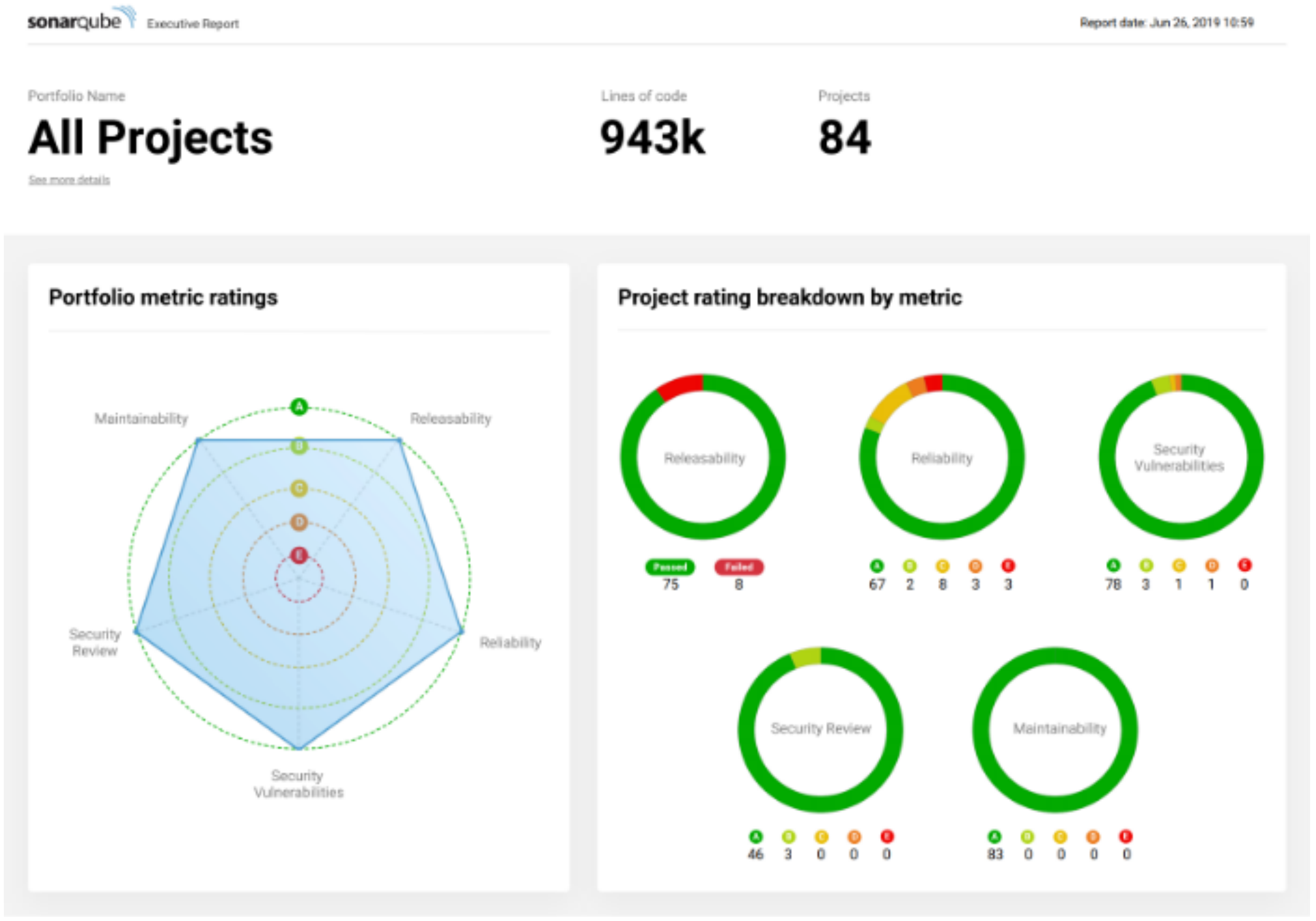
Automated Technical Debt Control

- Strict adoption of Static Analysis Tools

- Coding Rules and Design Patterns are fundamental
  - Increase code maintainability
    - Help other to understand the code
  - Reduce bugs significantly

Which rule, pattern and anti-pattern should be enforced, and which should be recommended?

Automated Technical Debt Control



sonarqube Executive Report

Report date: Jun 26, 2019 10:59

Portfolio Name

All Projects

See more details

Lines of code

943k

Projects

84

Portfolio metric ratings

Project rating breakdown by metric

Releasability	Passed 75	Failed 8			
Reliability	A 67	B 2	C 8	D 3	E 3
Security Vulnerabilities	A 78	B 3	C 1	D 1	E 0
Security Review	A 46	B 3	C 0	D 0	E 0
Maintainability	A 83	B 0	C 0	D 0	E 0

Conclusion

- Accelerate metrics seems to be effective to monitor team's performance
  - Ongoing validation with Vendasta, planning to start with ABB soon.

- Static analysis tools are of paramount importance
  - They need to be customized using historical data
  - Cannot be used out of the box

We are Hiring!

- Ph.D. position funded by ABB (4-years)
  - Continuous Impact Analysis of Architectural and Code Debt on software quality.
  - Development of dashboards for technical debt management.
- For more information: davide.taibi@tuni.fi