Using Secure Software Engineering Metrics to support the automated calculation and visualization of Team Security Maturity in Agile Development Projects

Timo Zandonella, Master Thesis, 08.08. 2022

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
wwwmatthes.in.tum.de

Motivation: Monitoring the security level in the development process with team security maturity model

With **Team Security Maturity**, it is possible to capture the capability of an agile development team to develop secure and security-compliant software [1-3]

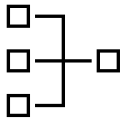
Within the Team Security Maturity Model (TSMM) [1], the maturity is measured in four maturity levels:

	No	Partly	Largely	Fully
Documentation				X
Build & Deployment		X		
Culture			X	
…				

The maturity level of the team gives insight for different stakeholders at different organizational levels:

Security Prediction before Deployment [4, 5]

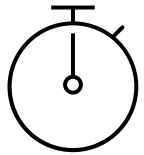
Award Team Autonomy [6]



Organizational Security Overview [4, 7]

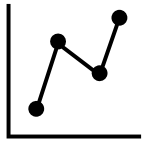
Goal: Support the maturity calculation with the automated collection of security metrics

Self-Assessments are used for the information collection, which can have some limitations:



They are time-consuming to execute [8]

Potential misjudgment [9]



They only give a periodic and not a continuous analysis [6]

🎯 Goal of the thesis:

We establish the continuous, automated tracking of Secure Software Engineering Metrics (Security Metrics):

e.g. Unit Test Coverage | Mean Time to resolve an issue | # of omitted security requirements

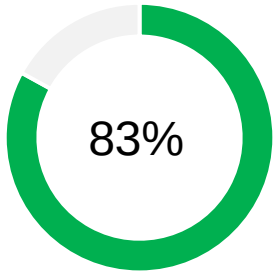
Together with the self-assessments we can improve the security maturity calculation of a team:

Self-Assessment

+

Team Security Metrics

- Product 1 Security Metrics
- Product 2 Security Metrics
- …



Research Questions

RQ1 Which Secure Software Engineering Metrics exist?

RQ2 How can Secure Software Engineering Metrics be used to assess the Security Maturity of an Agile Development Team?

RQ3 How can a Team‘s Security Maturity be calculated, represented and visualized in a Self-Assessment Tool?

RQ1: Towards a structured catalogue of security metrics

Security Metrics are quantifiable measurements used for assessing introduced security related **product imperfections** and the **teams' security efforts** during the software development process [4, 10, 11].

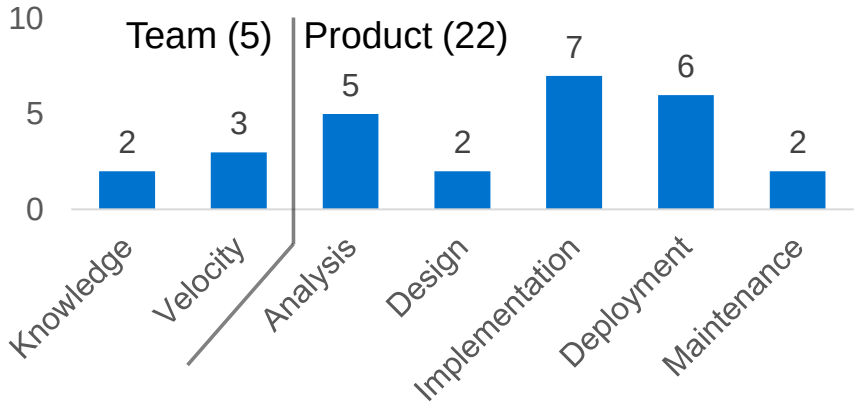
They can have different levels of quality and relevancy. The minimum baseline to be included in the catalogue is the fulfillment of the security metrics criteria according to Jaquith [4].

Additionally, many security metrics require supporting, general software metrics:

e.g. Product size (e.g. LOC) | Development Time spent | Number of Requirements

To collect the security metrics, a **structured literature review** is performed within five literature databases such as IEEE or ACM:

As of now 52 pieces of literature (incl. gray literature) were analyzed. The analysis resulted in 27 qualified metrics:

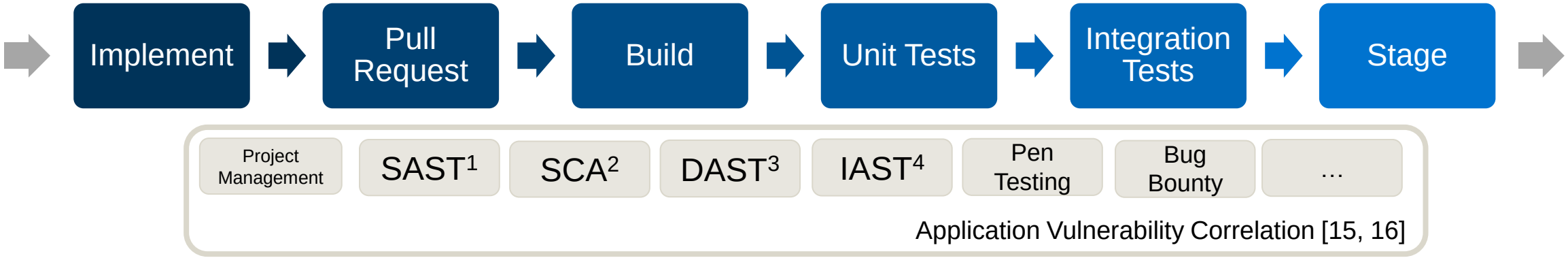


RQ1: Example Description of a Product Security Metric

Name	**Percentage of proposed Architecture Components subject to architectural risk analysis**
Description	This metric measures the ratio of architectural components that have undergone architectural risk analysis to the total number of components in the system. (Architectural risk assessment is a risk management process that identifies flaws in a software architecture and determines risks to business information assets that result from those flaws.)
Life Cycle Phase	Analysis → **Design** → Implementation → Deployment → Maintenance
Evaluation	Theoretical ~~Survey-based~~ ~~Tool-supported~~
Automatization	~~○ Manual~~ ◐ Semi automated ~~● Automated~~
Tool Category	Project Management
Data Type	Percentage
Metric Score [14]	85.4%: ~~Infant~~ ~~Evolving~~ Mature
Sources	(1) N. R. Mead and C. C. Woody, Cyber security engineering: a practical approach for systems and software assurance. Addison-Wesley, 2017. (2) N. Bartol and B. A. Hamilton, “Practical Measurement Framework for Software Assurance and Information Security,” Practical Software and Systems Measurement, 2008. …

RQ1: Measuring security metrics in a secure build pipeline

Automated and semi-automated product security metrics can be measured using a variety of tools in a **DevSecOps** Pipeline [12, 13].



[1] SAST: *Static Application Security Testing* scans the source code for vulnerabilities
[2] SCA: *Software Composition Analysis* identifies the open-source components in a code-base
[3] DAST: *Dynamic Application Security Testing* examines products for vulnerabilities in a deployed environment
[4] IAST: *Interactive Application Security Testing* tests the product with static and dynamic test cases

To validate the selection of security metrics, we will measure the security metrics with tools of a build pipeline on example open-source projects, e.g. Mozilla Firefox or OWASP WebGoat.

RQ1: Which Secure Software Engineering Metrics exist?
Methodology: Literature Review

Research Questions

RQ1 Which Secure Software Engineering Metrics exist?

RQ2 How can Secure Software Engineering Metrics be used to assess the Security Maturity of an Agile Development Team?

RQ3 How can a Team‘s Security Maturity be calculated, represented and visualized in a Self-Assessment Tool?

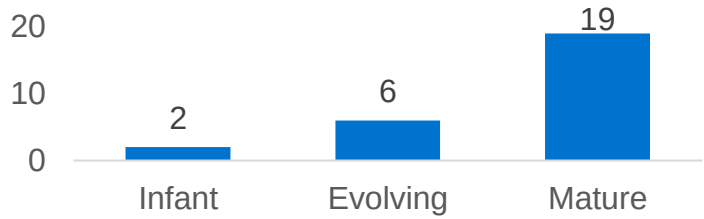
RQ2: Security Metrics have various levels of suitability for the calculation of team security maturity

The process of answering the research question is split into two parts:

1. Selecting a subset of security metrics, which are suitable for the calculation of team security maturity
2. *Introducing the suitable security metrics to the **team security maturity model***

a) Security metrics need to have a Security Metrics Maturity Score of 85% or higher (level “mature”) [14]:

Of the 27 security metrics analyzed so far, 19 reach this threshold.



b) Security metrics that relate a measured cardinal number to product size, number of requirements, etc. are preferred:

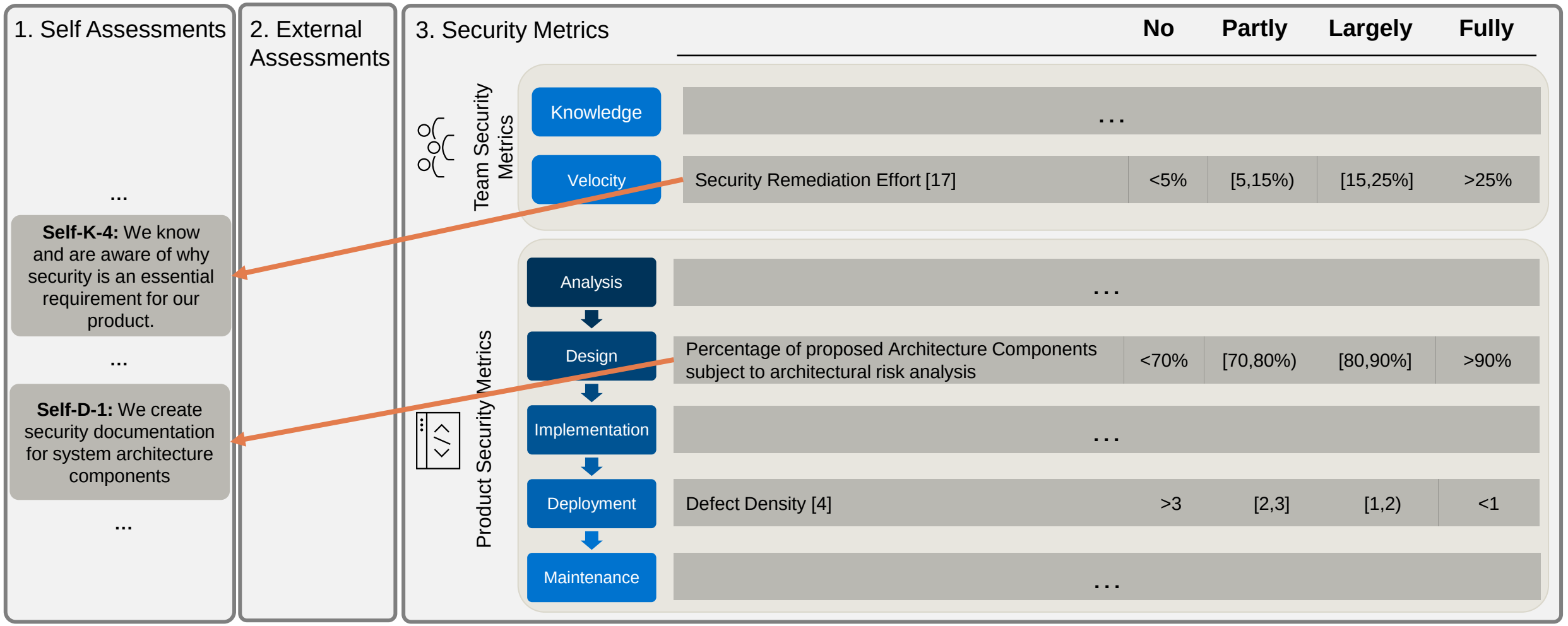
Vulnerabilities → $\frac{\text{Vulnerabilities}}{\text{Product Size}}$

The interpretation of relative metrics needs less context and may be applied product-independently [4, 7, 14].

RQ2: Security Metrics as the third pillar of the team security maturity model

The process of answering the research question needs to be split into two parts:

1. *Selecting a subset of security metrics, which are suitable for the calculation of team security maturity*
2. Introducing the suitable security metrics to the **team security maturity model**



Research Questions

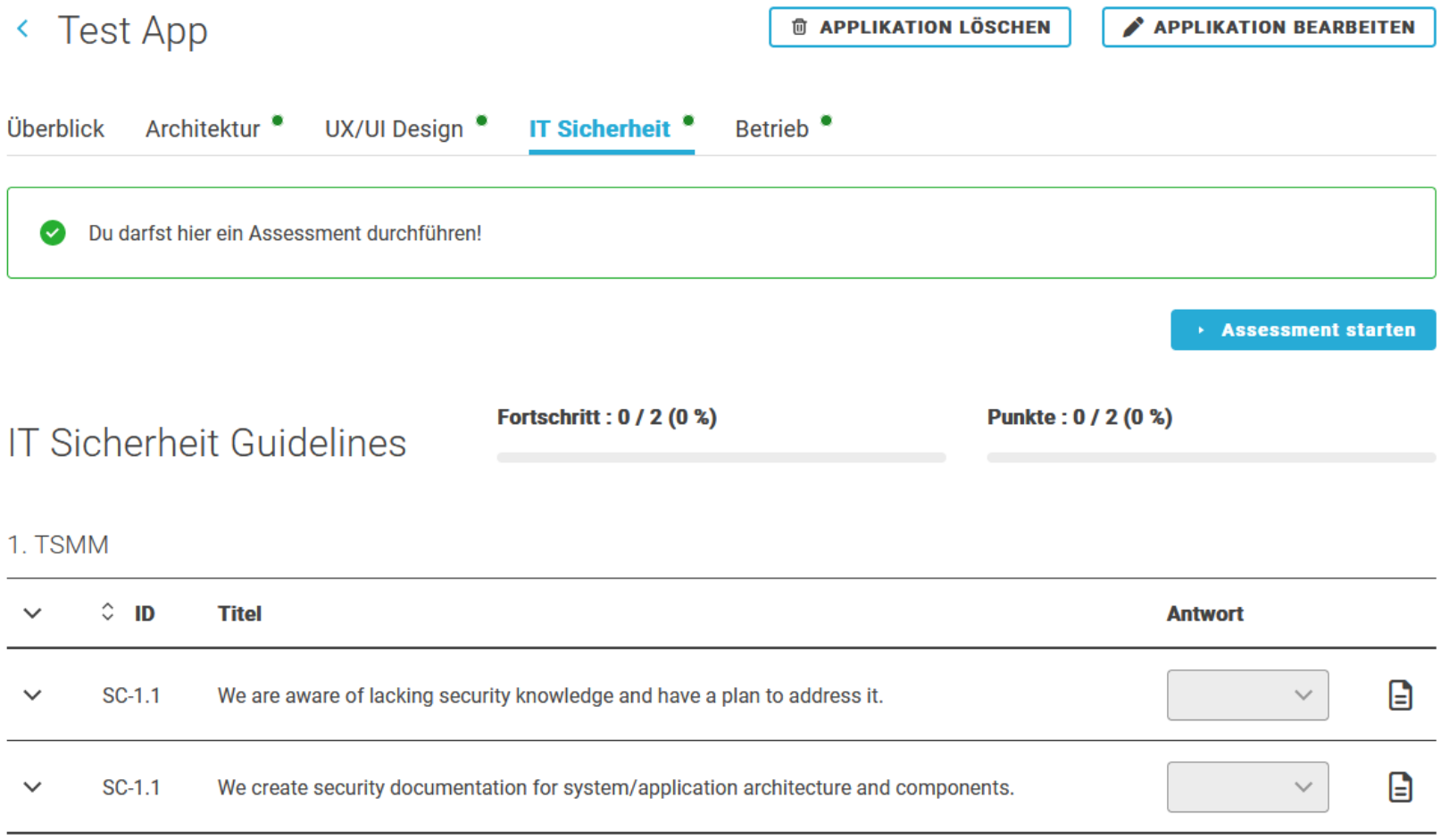
RQ1 Which Secure Software Engineering Metrics exist?

RQ2 How can Secure Software Engineering Metrics be used to assess the Security Maturity of an Agile Development Team?

RQ3 How can a Team‘s Security Maturity be calculated, represented and visualized in a Self-Assessment Tool?

RQ3: Calculating the Security Maturity in practice

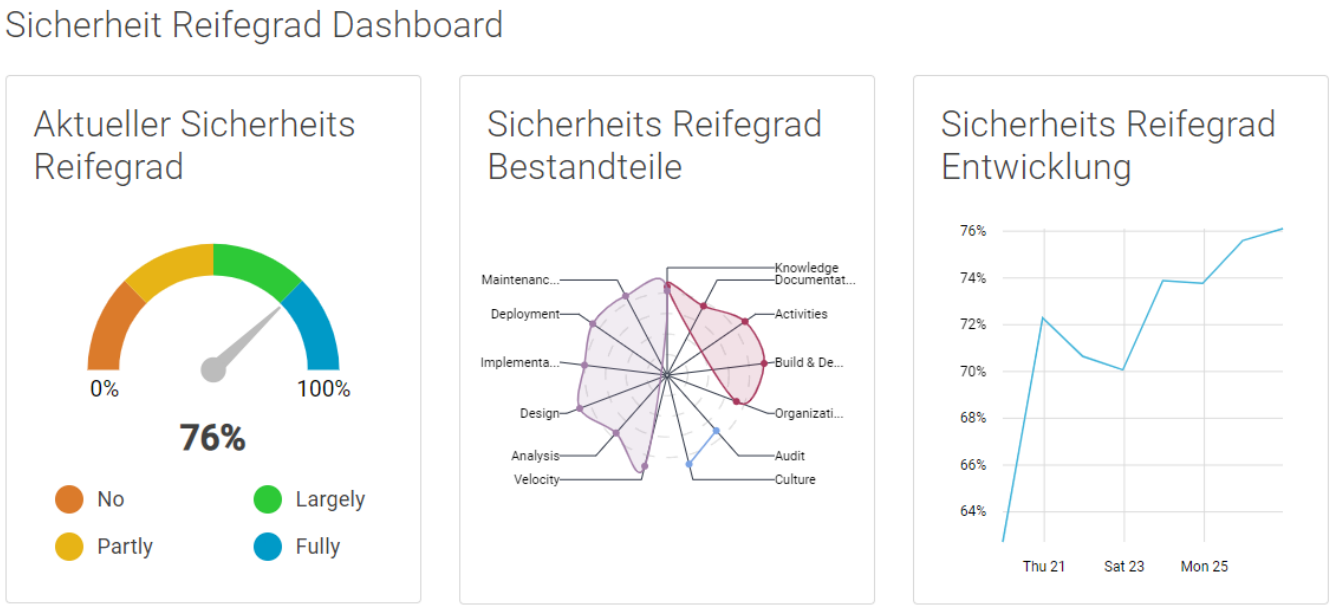
In the self-assessment tool Prince, two of the three pillars for calculating security maturity are already available: Self-Assessments and External-Assessments

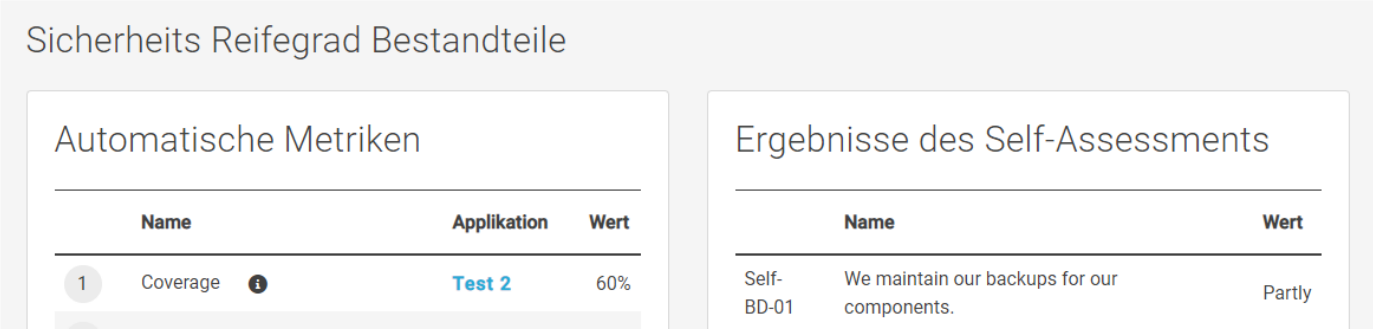


The goal is to additionally introduce the third pillar of the team security maturity model - security metrics.

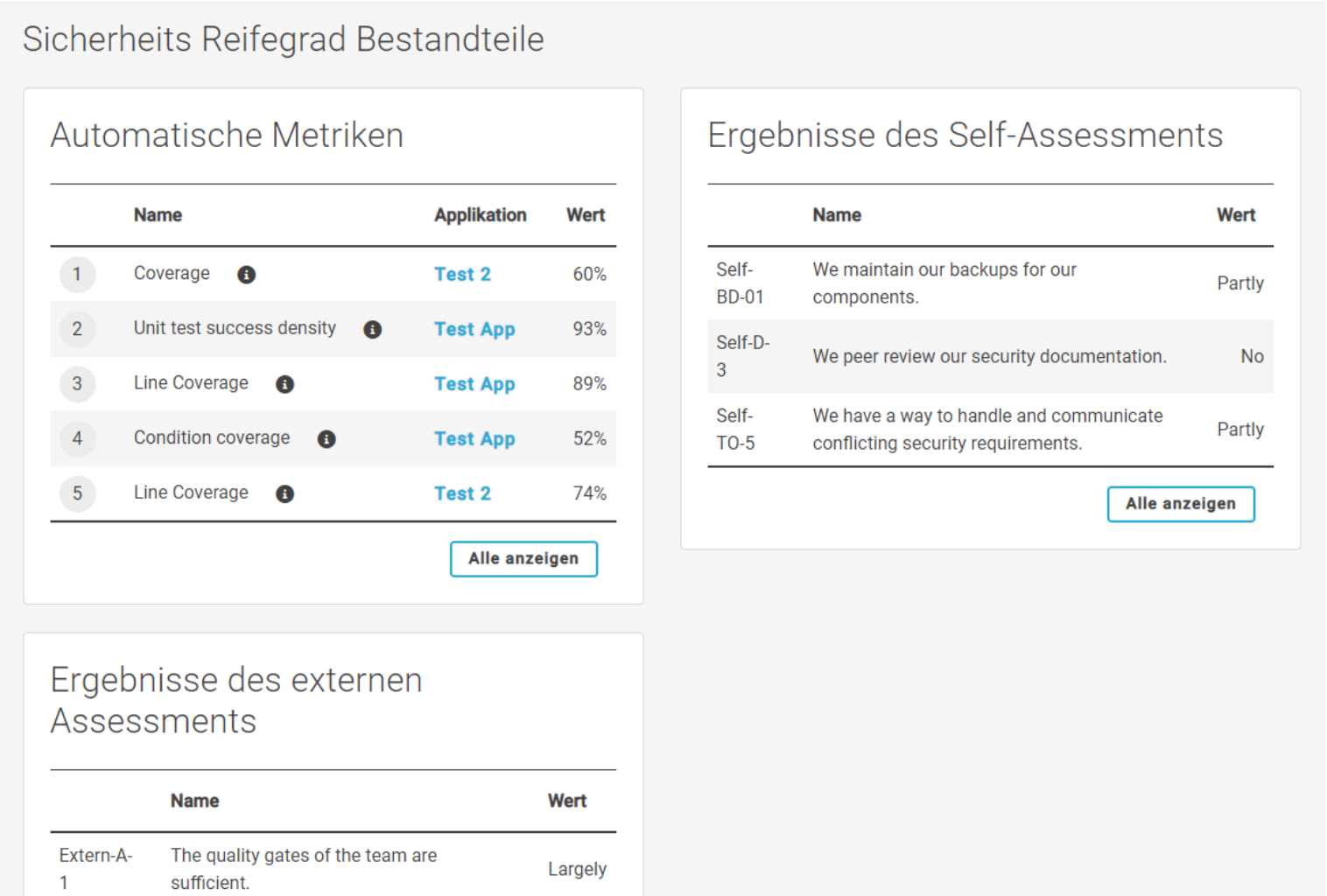
RQ3: Convert Prince from a self assessment tool to a team security maturity tool

Enhancing Prince to include a calculation and visualization of the team's security maturity enhances the meaning of the results of the self-assessments





RQ3: Convert Prince from a self assessment tool to a team security maturity tool



Finally, we want to introduce the maturity visualization in Prince to software engineers at Allianz and DATEV trough expert interviews and usability studies.

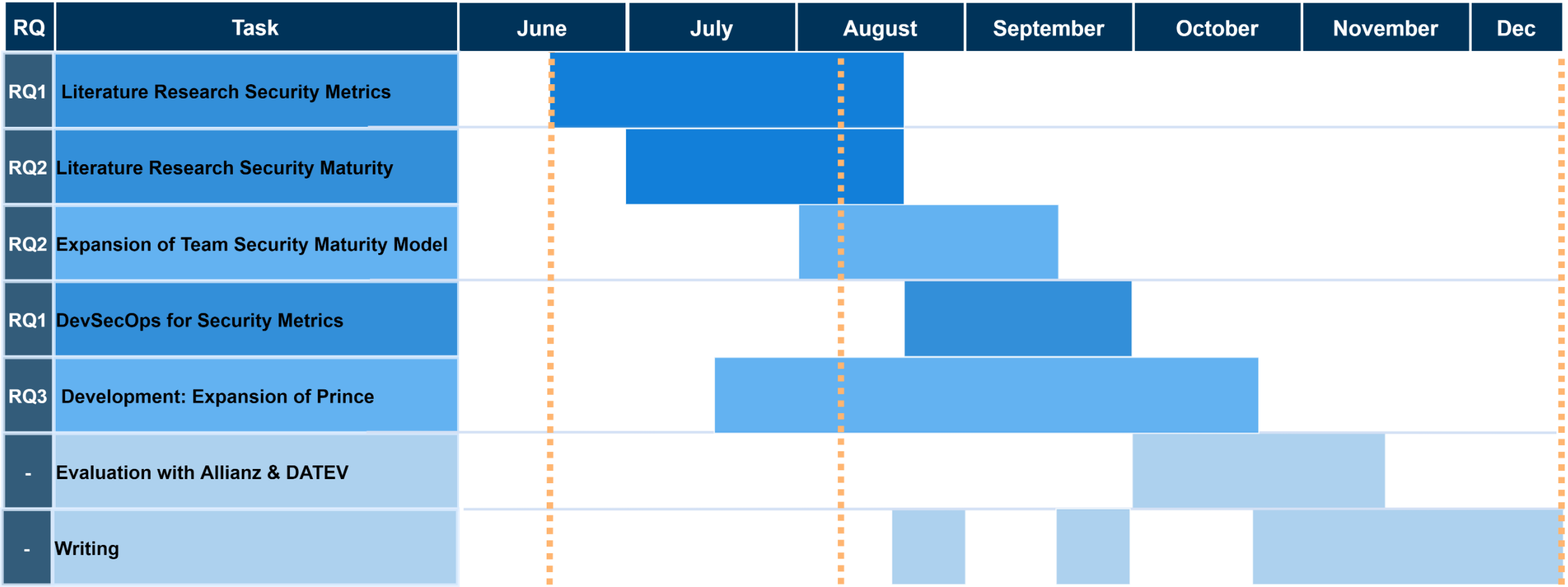
Research Questions

RQ1 Which Secure Software Engineering Metrics exist?

RQ2 How can Secure Software Engineering Metrics be used to assess the Security Maturity of an Agile Development Team?

RQ3 How can a Team‘s Security Maturity be calculated, represented and visualized in a Self-Assessment Tool?

Timeline



Registration 15.06. 2022

Kickoff 08.08.2022

Deadline 15.12. 2022

Thank you for your attention

Q & A

Sources

[1] Watzelt, J.-P. (2022). Design and Implementation of a Team MaturityModel Assessing Security Compliance inLarge-Scale Agile Software Development. Technical University Munich.

[2] Yurum, O. R., and Demirors, O. (2017, August). Agile Maturity Self-Assessment Surveys: A Case Study. 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). https://doi.org/10.1109/seaa.2017.58

[3] Stray, V., Moe, N. B., and Hoda, R. (2018, May). Autonomous agile teams. Proceedings of the 19th International Conference on Agile Software Development: Companion. https://doi.org/10.1145/3234152.3234182

[4] Jaquith, A. (2007). Security Metrics. Addison-Wesley Professional. https://learning.oreilly.com/library/view/security-metrics-replacing/9780321349989/ch02.html

[5] Rao, M. (2017). Building your DevSecOps pipeline: 5 essential activities. Synopsys. https://www.synopsys.com/blogs/software-security/devsecops-pipeline-checklist/

[6] Schenk, N. (2022). An Adaptive Approach for Security Compliance in Large-Scale Agile Software Development. Technical University Munich.

[7] Hayden, L. (2010). IT security metrics. McGraw Hill. https://learning.oreilly.com/library/view/it-security-metrics/9780071713405/ch01.html

[8] Myers, A. M., Holliday, P. J., Harvey, K. A., and Hutchinson, K. S. (1993). Functional Performance Measures: Are They Superior to Self-Assessments? Journal of Gerontology, 48, M196–M206. https://doi.org/10.1093/geronj/48.5.m196

[9] Pemberton, R., Li, E. S. L., Or, W. W. F., and Pierson, H. D. (1996). Taking Control: Autonomy in Language Learning. Hong Kong University Press. https://books.google.de/books?id=pTHxAQAAQBAJ

[10] Cheng, Y., Deng, J., Li, J., DeLoach, S. A., Singhal, A., and Ou, X. (2014). Metrics of Security. In Advances in Information Security (pp. 263–295). Springer International Publishing. https://doi.org/10.1007/978-3-319-11391-3_13

[11] Jansen, W. (2009). Directions in Security Metrics Research. NISTIR 7564. https://csrc.nist.gov/publications/detail/nistir/7564/final

[12] DevSecOps Guide - Standard DevSecOps Platform Framework. (n.d.). https://tech.gsa.gov/guides/dev_sec_ops_guide/

[13] Crouch, A. (2017). DevSecOps: Incorporate Security into DevOps to Reduce Software Risk. Agile Connection. https://www.agileconnection.com/article/devsecops-incorporate-security-devops-reduce-software-risk

[14] Muthukrishnan, S. M., and Palaniappan, S. (2016, May). Security metrics maturity model for operational security. 2016 IEE Symposium on Computer Applications Industrial Electronics (ISCAIE). https://doi.org/10.1109/iscaie.2016.7575045

[15] Potter, B., and McGraw, G. (2004). Software security testing. IEEE Privacy Magazine, 2, 81–85. https://doi.org/10.1109/msp.2004.84

[16] Rangnau, T., v. Buijtenen, R., Fransen, F., and Turkmen, F. (2020, October). Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines. 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC). https://doi.org/10.1109/edoc49727.2020.00026

[17] SonarQube. (n.d.). Metric Definitions. https://docs.sonarqube.org/latest/user-guide/metric-definitions/