Risk-based Software Quality and Security Engineering in Data-intensive Environments (Invited Keynote)

Michael Felderer[1,2](✉)

[1] University of Innsbruck, Innsbruck, Austria
michael.felderer@uibk.ac.at
[2] Blekinge Institute of Technology, Karlskrona, Sweden

Abstract. The concept of risk as a measure for the potential of gaining or losing something of value has successfully been applied in software quality engineering for years, e.g., for risk-based test case prioritization, and in security engineering, e.g., for security requirements elicitation. In practice, both, in software quality engineering and in security engineering, risks are typically assessed manually, which tends to be subjective, non-deterministic, error-prone and time-consuming. This often leads to the situation that risks are not explicitly assessed at all and further prevents that the high potential of assessed risks to support decisions is exploited. However, in modern data-intensive environments, e.g., open online environments, continuous software development or IoT, the online, system or development environments continuously deliver data, which provides the possibility to now automatically assess and utilize software and security risks. In this paper we first discuss the concept of risk in software quality and security engineering. Then, we provide two current examples from software quality engineering and security engineering, where data-driven risk assessment is a key success factor, i.e., risk-based continuous software quality engineering in continuous software development and risk-based security data extraction and processing in the open online web.

Keywords: Risk assessment · Software quality engineering
Security engineering · Data engineering

1 Introduction

The concept of risk as a measure for the potential of gaining or losing something of value has successfully been applied in software quality and security engineering to support critical decisions.

In software quality engineering, the concept of risk has for instance been applied in risk-based testing, which consider risks of the software product as the guiding factor to steer all phases of a test process, i.e., test planning, design, implementation, execution, and evaluation [1–3]. Risk-based testing is a pragmatic, in companies of all sizes widely used approach [4,5] which uses the straightforward idea to focus test activities on those scenarios that trigger the most critical situations of a software system [6]. In general, a risk is an event that may possibly occur and, if it occurs, it has (typically negative) consequences. Therefore, risks are determined by the two factors probability and impact. For testing purposes, the factor probability describes the likelihood that the negative event, e.g., a software failure, occurs and impact characterizes the cost if the failure it occurs in operation. Assessing the risk exposure of a software feature or component requires estimating both factors. Impact can in that context usually be derived from the business value associated to the feature defined in the software requirements specification. Probability is influenced by the implementation characteristics of the feature or component as well as the usage context in which the software system is applied.

In security engineering, the concept of risk in particular and risk management in general receives even more attention than in software quality engineering. Risks are often used as a guiding factor to define security measures throughout the software development lifecycle. For instance, Potter and McGraw [7] consider the process steps creating security misuse cases, listing normative security requirements, performing architectural risk analysis, building risk-based security test plans, wielding static analysis tools, performing security tests, performing penetration testing in the final environment, and cleaning up after security breaches. In security engineering, risk is determined by the probability that a threat will exploit a vulnerability and the impact of the resulting adverse consequence, or loss [8]. A threat is a cyber-based act, occurrence, or event that exploits one or more vulnerabilities and leads to an adverse consequence or loss. A vulnerability is a weakness in an information system, system security procedures, internal controls, or implementation that a threat could exploit to produce an adverse consequence or loss.

The overall risk management comprises the core activities risk identification, risk analysis, risk treatment, and risk monitoring [9]. In the risk identification phase, risk items are identified. In the risk analysis phase, the likelihood and impact of risk items and, hence, the risk exposure is estimated. Based on the risk exposure values, the risk items may be prioritized and assigned to risk levels defining a risk classification. In the risk treatment phase the actions for obtaining a satisfactory situation are determined and implemented. In the risk monitoring phase the risks are tracked over time and their status is reported. In addition, the effect of the implemented actions is determined. The activities risk identification and risk analysis are often collectively referred to as risk assessment while the activities risk treatment and risk monitoring are referred to as risk control.

Several methods to assess software or security risks are available (e.g., RisCal [10] for software risks and the Security Engineering Risk Analysis (SERA) Framework [8] for security risks). In practice, both, in software quality engineering and in security engineering, risks are typically assessed manually, which tends to be subjective, non-deterministic, error-prone and time-consuming.

However, in modern data-intensive environment like open online environments, continuous software development or IoT, the online, system or development environments continuously deliver data, which provides the possibility to automatically assess and utilize software and security risks. In the following two sections, we sketch two examples from software quality engineering and security engineering, where data-driven risk assessment plays a key role, i.e., risk-based continuous software quality engineering and risk-based security data extraction and processing.

2 Risk-Based Continuous Software Quality Engineering

In the data-intensive environment of modern continuous software development based on cloud technologies, system testing and release management merge and have to be performed continuously ranging from automated system testing (for critical system software potentially based on model-based testing), over manual acceptance testing to live online experimentation at runtime. There is unexploited potential to improve system testing, on the one hand by intelligent automation and on the other hand by complementing it with live experimentation. Live experimentation at runtime [11] allows to deploy faster and thus gaining the competitive advantage of giving customers earlier access to new functionality, to reach a larger population than possible with acceptance testing and to check functional as well as non-functional behavior. However, live experimentation can only be implemented for uncritical software components to avoid that critical defects or hazards occur during runtime. Therefore, a suitable software structure and software risk assessment based on automated data analytics (leading to risk analytics) is required to avoid the issue of live experimentation for critical software components prior to sufficient system testing. The three continuous software quality improvement aspects of risk analytics, intelligent test automation and live experimentation are shown together with their characteristics in Fig. 1.

The first aspect is automated software risk analytics. It processes structured, semi-structured and unstructured software product data (e.g., data from source code, test specifications, defects, design models, or requirements specifications), organizational data (e.g., data about the teams developing specific services), process data (e.g., data from the version control system, issue tracking data, or deployment and runtime data), and business data (e.g., data about the business value, market potential or cost of specific software services), which allows to automatically determine probability and impact for risk assessment. The risk information is then applied to perform intelligent test automation to support decisions on what to automate (test-case design, test data generation and test execution of specific components, scenarios or services) and when to automate (in which sequence and iteration) as second aspect. Finally, as a third aspect, if the risk level is moderate, even live experimentation can be performed to test functional and non-functional system properties.

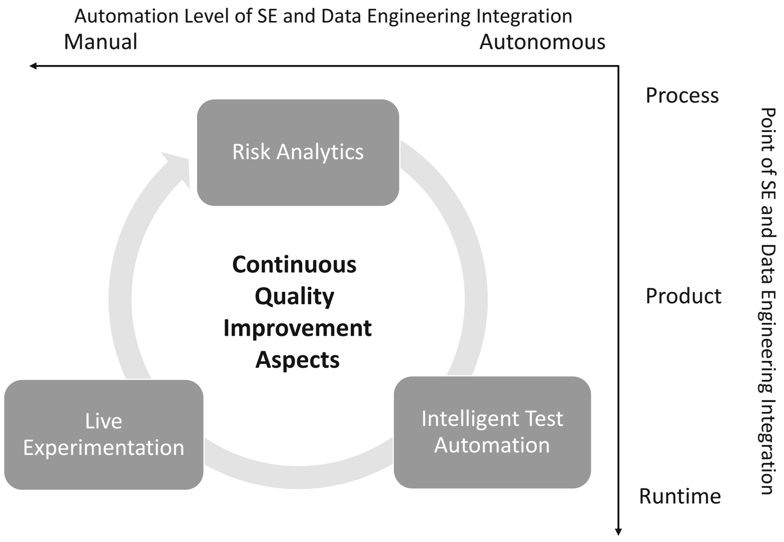
Automation Level of SE and Data Engineering Integration
Manual
Autonomous
Process
Product
Runtime
Point of SE and Data Engineering Integration
Risk Analytics
Continuous Quality Improvement Aspects
Live Experimentation
Intelligent Test Automation

Fig. 1. Risk-based continuous software quality engineering

3 Risk-Based Security Data Extraction and Processing

The proposed approach to risk-based security data extraction and processing in the data-intensive environment of the open online web consists of two major components, i.e., a Security Data Collection and Analysis Component as well as a Security Knowledge Generation Component. The approach was originally presented in [12] and we refer to it here. Figure 2 shows the approach.

The Security Data Collection and Analysis Component is responsible for the data extraction from various data sources, quality assessment of data and data merging in order to provide the data in a processable form. It considers extraction from several online sources including vulnerability knowledge bases like Common Vulnerabilities and Exposures (CVE) [13] or the Malware Information Sharing Platform (MISP) [14], social media like Twitter as well as security forums or websites. Once the data is extracted and available, it must be formatted, the quality assessed and then merged. Because of the type of information being handled and the fact that there are different data fields to deal with, this is a highly complex task. In order to overcome differences, a general format is proposed, which includes information such as name, type, year, target platform, description and reference. It is the basis to automatically assess security risks.

The Security Knowledge Generation Component processes the extracted security information in order to provide it for different roles and various purposes, for instance as knowledge to stakeholders in the agile development process or to generate attack models. For instance, a developer can be provided with a security dashboard showing security risks or concrete guidelines on how code can be secured or security properties can be tested. As for the product owner, they can

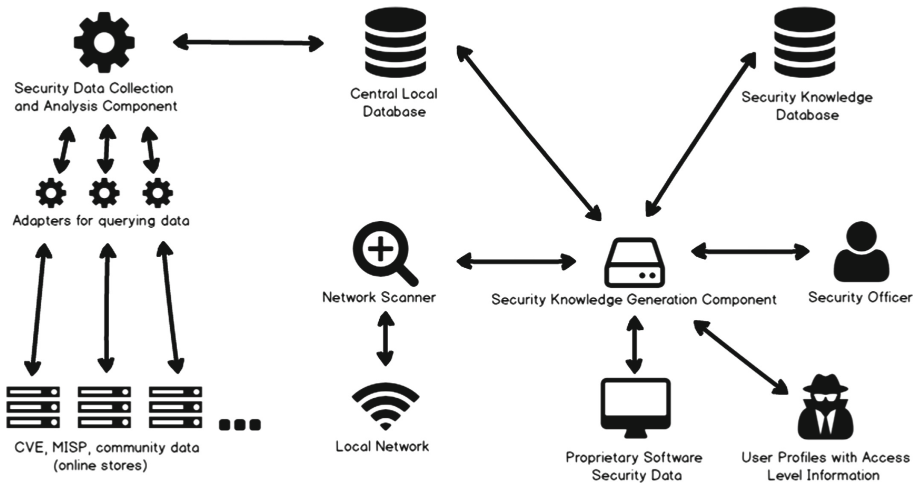
Security Data Collection and Analysis Component
Central Local Database
Security Knowledge Database
Adapters for querying data
Network Scanner
Security Knowledge Generation Component
Security Officer
CVE, MISP, community data (online stores)
Local Network
Proprietary Software Security Data
User Profiles with Access Level Information

Fig. 2. Risk-based security data extraction and processing approach [12]

receive guidelines on security requirements and risk management. Finally, when developing a safety critical system, a developer who is responsible for the system architecture can be provided with generated attack models annotated with risk information that can be integrated with available system models to perform a combined safety-security analysis [15] or model-based security testing [16].

4 Conclusion

This paper sketched to approaches to automated risk assessment in data-intensive environments, i.e., risk-based continuous software quality engineering in continuous software development as well as risk-based security data extraction and processing in the open online web.

References

1. Gerrard, P., Thompson, N.: Risk-Based E-business Testing. Artech House Publishers, Norwood (2002)
2. Felderer, M., Ramler, R.: Integrating risk-based testing in industrial test processes. Softw. Qual. J. **22**(3), 543–575 (2014)
3. Felderer, M., Schieferdecker, I.: A taxonomy of risk-based testing. Int. J. Softw. Tools Technol. Transf. **16**(5), 559–568 (2014)
4. Felderer, M., Ramler, R.: A multiple case study on risk-based testing in industry. Int. J. Softw. Tools Technol. Transf. **16**(5), 609–625 (2014)
5. Felderer, M., Ramler, R.: Risk orientation in software testing processes of small and medium enterprises: an exploratory and comparative study. Softw. Qual. J. **24**(3), 519–548 (2016)

6. Wendland, M.F., Kranz, M., Schieferdecker, I.: A systematic approach to risk-based testing using risk-annotated requirements models. In: ICSEA 2012, pp. 636–642 (2012)
7. Potter, B., McGraw, G.: Software security testing. IEEE Secur. Priv. **2**(5), 81–85 (2004)
8. Alberts, C., Woody, C., Dorofee, A.: Introduction to the security engineering risk analysis (SERA) framework. Technical report, Carnegie Mellon University Software Engineering Institute, Pittsburgh, Pennsylvania (2014)
9. ISO: ISO 31000 - Risk Management (2018). http://www.iso.org/iso/home/standards/iso31000.htm
10. Haisjackl, C., Felderer, M., Breu, R.: Riscal-a risk estimation tool for software engineering purposes. In: 2013 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pp. 292–299. IEEE (2013)
11. Auer, F., Felderer, M.: Current state of research on continuous experimentation: a systematic mapping study. In: EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2018). IEEE (2018)
12. Felderer, M., Pekaric, I.: Research challenges in empowering agile teams with security knowledge based on public and private information sources (2017)
13. MITRE: Common vulnerabilities and exposures. https://cve.mitre.org/
14. Andre, D.: Malware information sharing platform. http://www.misp-project.org/
15. Chockalingam, S., Hadžiosmanović, D., Pieters, W., Teixeira, A., van Gelder, P.: Integrated safety and security risk assessment methods: a survey of key characteristics and applications. In: Havarneanu, G., Setola, R., Nassopoulos, H., Wolthusen, S. (eds.) CRITIS 2016. LNCS, vol. 10242, pp. 50–62. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71368-7_5
16. Felderer, M., Zech, P., Breu, R., Büchler, M., Pretschner, A.: Model-based security testing: a taxonomy and systematic classification. Softw. Test. Verif. Reliab. **26**(2), 119–148 (2016)