Towards Agile Security Risk Management in RE and Beyond

Virginia N. L. Franqueira*† , Zornitza Bakalova†, Thein Than Tun‡ and Maya Daneva†
† University of Twente
Enschede, The Netherlands
Email: {franqueirav, z.bakalova, m.daneva}@ewi.utwente.nl
‡ The Open University
Milton Keynes, UK
Email: t.t.tun@open.ac.uk

***Abstract*—Little attention has been given so far to the process of security risk management at the early stages of system development. Security has been addressed by isolated security assurance practices, some of which consider risks and mitigations but they do not provide an overview of the overall security state of the system being developed. This paper takes the position that (1) these isolated security assurance practices should be fully integrated and should be embedded in short iterations of risk assessment, treatment and acceptance, providing input for updating security requirements and for security risk management, and that (2) available empirical data from public catalogs and databases should be used as a source of expertise, to leverage past experiences, and therefore reduce, although not eliminate, subjectivity of human judgment. Borrowing from the agile software development and project management philosophy, we introduce the idea of a light weight, agile approach to security risk management integrated to the development life cycle.**

***Index Terms*—Information Security Risk Management; Agile Software Development; Secure Engineering; Security Assurance.**

I. Introduction

"Just about every software system deployed today must defend itself from malicious adversaries" [1]. Building defenses, however, requires a clear overview of security risks which change across the development life cycle for many reasons, e.g., new threats arise, new vulnerabilities are reported, risks are introduced at different phases of development. Therefore, assuming that security requirements should be neatly engineered up-front and frozen for the rest of the development life cycle (independent of the development paradigm adopted) is rather unrealistic. In practice, new security requirements are *discovered* along the development life cycle, some of them will have to be addressed by the system and some will be accepted as residual risks. Security risk management has the potential to help in making such decisions in an informed way, providing a clearer overview of the state of system security, when deployed.

Our position is that security risk management should be embedded into the development life cycle. However, the traditional security risk management process is intrinsically similar to the traditional waterfall life cycle model, and therefore inherits several of its drawbacks. Adopting it in a traditional way would cause too much overhead during development, and would not gain wide acceptance. We argue that, the same way as the agile software development and project management philosophy addresses drawbacks of the waterfall life cycle, it is also a promising direction for a light weight security risk management. In this paper, we introduce the idea of an agile security risk management into RE and beyond, i.e., along the whole development life cycle. It is noted that the idea represents a first step in this direction, and remains at a high level of abstraction. Although we focus on security across the whole life cycle, this has an immediate implications on requirements and the way they are managed.

This paper is structured as follows. Section II summarizes the problems intrinsic to security risks and explores a possible solution that uses empirical data. Section III reviews the traditional security risk management process and discusses what it can learn from the agile philosophy. Section IV presents the idea of embedding security risk management in an agile way into the development cycle. It also discusses research questions to guide empirical research to validate the feasibility of the idea, and to provide insights on limitations and improvements. Section V concludes the paper.

II. Dealing with Some Problems Intrinsic to Security Risks

A. State of Practice of Security Risk in RE and Beyond

A number of security assurance practices may be performed along a system development life cycle, either by internal or external parties. Figure 1 illustrates some of these practices [2], [3], and indicates that while project risk management happens across the whole development process, security risk management does not. Note that practices listed here are representative, rather than exhaustive.

Security assurances practices are applied to the RE phase mainly with the purpose of eliciting requirements, e.g., [4]–[6] (to name a few), without any follow-up on the consequences of security risks addressed and not addressed. Moreover, risks have mainly been approached in RE from two perspectives:

*First author is supported by the research program Sentinels (http://www.sentinels.nl), second and fourth authors by the Netherland's NWO under the QUADREAD project, and third author by the EU SecureChange project.

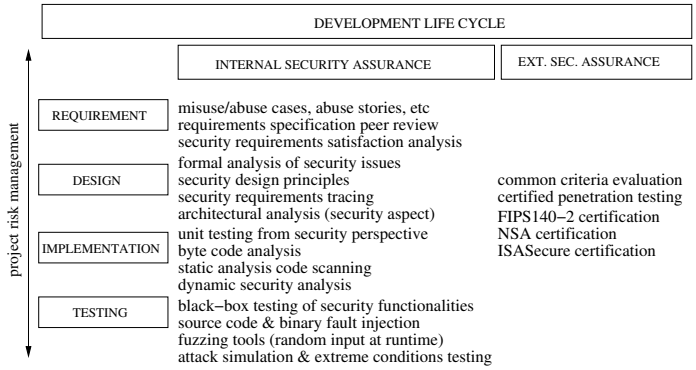

Fig. 1. Isolated security assurance practices at different phases of a system development life cycle, performed either internally or by external parties

(1) project risks, such as risks related to project resources and estimates, managed throughout the development life cycle, and (2) system risks, such as risks related to incomplete, inconsistent and imprecise requirements, managed throughout the requirements elicitation phase. Although project risks and *security risks* have common characteristics, such as uncertainty about future events, there are intrinsic differences. First, project risks mostly relate to a mature set of recurring and well documented risks. Second, project managers benefit from the availability of empirical data either organization-specific historical data on past projects or community-shared baseline data (e.g., the PERIL database [7]). This helps, not only to identify risks, but also to estimate and treat risks.

This state of the art practices highlight the following issues: assurance practices are (1) used in isolation without cross-cutting management, not only in RE but along the whole development cycle, (2) useful to identify threats, mitigations (e.g., via misuse cases, anti-goals), and vulnerabilities (e.g., via static analysis code scanning) in specific scenarios or pieces of code but fail to consider other aspects of security risks, such as "cost-effectiveness concerns" [8] or even the systematic prioritization of security risks, and (3) expensive in terms of expertise and past experiences required, and reliant on human judgment, a problem intrinsic to security risk assessment in general. We propose to address the first gap with an agile approach to security risk management (see Section III), and with the use of empirical data, publicly available (see Section II-B).

B. The Use of Empirical Data in Security Risk Management: Public Catalogs and Databases of Security Expertise

The National Cyber Security Division of the U.S. Department of Homeland Security has a strategic initiative to promote software assurance. Three projects under this initiative are particularly useful across the software development life cycle. (1) **CAPEC - Common Attack Pattern Enumeration and Classification** (http://cwe.mitre.org/): CAPEC is a catalog of known attacks; (2) **CWE - Common Weakness Enumeration** (http://capec.mitre.org/): CWE is a catalog of weaknesses, where each weakness can be either a direct source of vulnerabilities, or may indirectly contribute to an increase in the likelihood of an attack to happen and/or in the impact of the attack, if it succeeds; (3) **CVE - Common Vulnerabilities and Exposure** (http://nvd.nist.gov/) entries recorded in the National Vulnerability Database (http://nvd.nist.gov/): they represent concrete examples of vulnerabilities which have been detected in use in practice.

Such empirical data provide information, not only about threats and vulnerabilities, but also about generic solutions and mitigations at different phases of development, and about severity of risks involved. Such data benefit from input by a pool of security experts from several organizations (http://cwe.mitre.org/community/index.html), and are constantly being updated in new versions. This information is useful to prioritize risks and, in turn, to prioritize requirements or fixes in code, for instance.

In this section, we dealt with intrinsic problems of security risks and outlined a solution direction which leverages the use of empirical data, publicly available. In the next section, we deal with problems intrinsic to the security risk management process.

III. Dealing with Problems Intrinsic to the Traditional Security Risk Management Process

A. The Security Risk Management Process

Despite differences among Security Risk Management standards and methodologies (e.g. CRAMM [9], ISO 27005:2008 [10], AS/NZS4360:2004 [11], CORAS framework [12]), they follow a similar process. This typical process, adapted from ISO 27005:2008 [10], is illustrated in Figure 2.

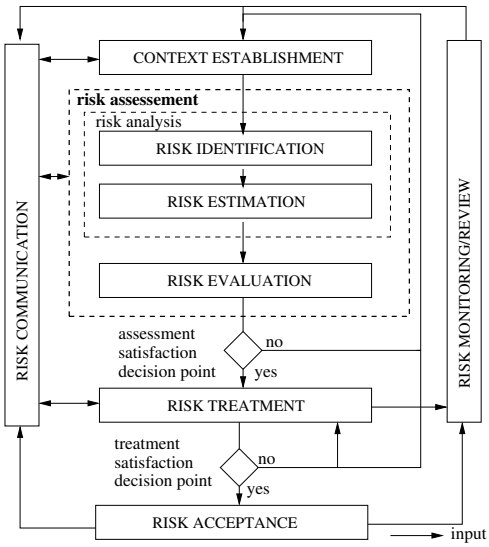

Fig. 2. Typical security RM process (adapted from ISO 27005:2008 [10])

The figure shows a sequence of activities from *context establishment* until the end of *risk assessment*, and two decision points defining iterations regarding risk *assessment satisfaction* and risk *treatment satisfaction*. The first decision point accommodates an increased "depth and detail of the risk assessment at each iteration" [10], while the second decision point accommodates the situation where "the risk treatment

will not immediately lead to an acceptable level of residual risk" [10] (i.e., risks expected to remain after risk treatment is enforced) and a new iteration of risk assessment and/or treatment is required.

This security risk management process falls under the traditional plan-do-check-act cycle of management [10]. Therefore, although security risk management may have decision points allowing iterations of risk assessment and/or treatment analysis, it is intrinsically similar to the traditional waterfall life cycle model: traditional security risk management works in fact on a top-down, plan-driven, documented-centric fashion [13]. Since agile approaches to software development and management, and in particular to RE, propose to address the main drawbacks of traditional development life cycle models, we review next the agile philosophy.

B. The Agile Philosophy

Agile approaches to software project delivery and to software development can be considered a paradigm, a project management philosophy, a culture, an attitude, and a state of mind. Compared to traditional software development, they rest on completely different understanding about the values and principles that represent the foundation of the development method. Namely, all agile methodologies rest on the *minimalist principle* of organizing work in the software development process, meaning a conscious choice in carrying out those tasks which directly create value for clients and leaving out anything that is deemed "waste" [14]. The latter refers to all work and deliverables not directly contributing to the development of the desired software, for example spending time on implementing features that are not specified by any user story or on producing an artifact not explicitly asked by the clients. The minimalist principle is fundamental to the ability of the agile approaches to cope with project uncertainties. In that sense, this principle can be seen as a reaction to the *plan-based* paradigm which assumes that problems are fully specifiable and that predictable solutions exist for every problem [14]. Another characteristic of agile approaches consists in their iterative and incremental nature. The work is organized in short time-boxed iterations, and the output of each iteration is a small increment of the final product. Within each iteration, all software development activities are present, i.e. planning, design, implementation and testing.

The agile community has addressed *security* in an agile software development process in many ways. Peeters [15] introduced the notion of abuser stories in the requirements domain. Boström et al. [13] proposed an extension of some agile practices, like planning game and coding guidelines, to aid formulating security-related user stories. Beznosov and Kruchten [2] classified security assurance methods and techniques with respect to their clash with agile development. Sonia and Singhal [16] proposed a technique for requirement elicitation within Agile Software Development which combines abuser stories and attack trees. We take a different approach towards embedding security risk management by adapting practices from the agile philosophy, independent on the software development paradigm adopted.

C. What Traditional Security Risk Management Can Learn from the Agile Philosophy

Table I summarizes the characteristics inherent to security risk management (Section III-A) against the philosophy behind agile development and management (Section III-B).

Security Risk Management	**The Agile Philosophy**
Top-down approach based on plan-do-check-act cycle	Incremental approach based on small speculate-collaborate-learn iterations and frequent feedback cycles
Upfront, fix planning which drives remaining risk management activities	Gradual planning of activities (planning game) driven by learning from feedback cycles
Documentation-centric approach which relies on documented knowledge	Light weight documentation driven by importance deemed by stakeholders; more tacit knowledge-oriented based on person-to-person communication
Upfront decision about level of security needed for the system reflected on a priori agreement on, e.g., risk evaluation criteria and risk acceptance criteria	Stakeholders establish an initial baseline for security level needed and adjust this along the way
Assumes complete and correct information and consensus about criteria used	Assumes that incomplete knowledge and uncertainty are part of the process, changes are inevitable and testing should start from the first iteration
Labor intensive and costly, causing time and budget overhead	Minimalist, lean approach which tends to be less demanding in terms of effort and time, therefore, less costly

TABLE I
COMPARATIVE BETWEEN TRADITIONAL SECURITY RISK MANAGEMENT AND THE AGILE PHILOSOPHY [2], [13], [17]

This table indicates possible improvements, borrowed from the agile philosophy, which could be made to the traditional security risk management process to make it more light weight, triggered by events, adaptable, and prone to a continuous build-up approach which fits to address the needs of a security risk management process in RE and beyond in the development, without the costs associated with a traditional heavy weight and costly security risk management process. Next, we propose an agile perspective to security risk management to fit the entire development life cycle.

IV. Towards an Agile Security Risk Management Applied to RE and Beyond

In this section we outline our idea; we borrow from the agile philosophy to address problems intrinsic to the traditional security risk management process, as summarized in Table I, and from the use of public empirical data to address problems intrinsic to security risks, as described in Section II.

Our proposal is illustrated in Figure 3 showing iterations of risk assessment, treatment and acceptance. These short iterations receive input (1) from different security assurance practices (see Figure 1), and (2) from empirical data stored in public catalogs and the NVD database. Their output

feed updates of security requirements and functionalities, and provide feedback to a security management process which incrementally collects residual risks. Such residual risks are not treated by mitigations introduced in the system being built up to the current iteration, and should be evaluated again in the next iteration, successively. At any time an informed decision can be made about delivering the system or a part of it based on the current list of prioritized residual risks.

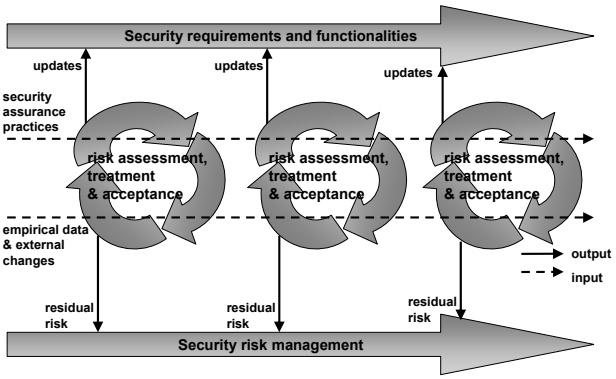

Fig. 3. Proposed agile security risk management process

The planning of security-related activities during RE and the whole life cycle is performed at the end of each iteration with participation of a security knowledgeable representative appointed by the client. This is a learning process which considers the security state of the system (i.e., residual risks) and external changes, such as system context changes, e.g., a relevant incident was reported, or new threats and vulnerabilities emerged (such as a new virus outbreak). It allows decision making about which mitigations to be incorporated in the next iteration, considering the level of security aimed versus the cost of implementing such mitigations.

The feasibility of the idea outlined in this section needs to be empirically evaluated. We plan to validate this feasibility by answering the following research questions: (RQ1) To what extent the use of security catalogs/databases contribute to reducing the amount of documentation currently being generated along the development cycle? (RQ2) How does the proposed idea change the way RE is done and what is the impact of change? (RQ3) What is the impact of considering residual risks related to different artifacts (i.e., requirements, design, code, system) on decision making? Do we need transformation steps to allow decision making? (RQ4) How to determine important/relevant security events (external changes) to be considered in the next iteration of risk assessment? What is the role of security catalogs/databases in this respect? (RQ5) How to deal with adjusted security level and risk acceptance criteria (learning process) without impact on documentation? and (RQ6) What is the impact of the idea for different settings, e.g., when different development paradigms are adopted and when different types of systems are developed? Answering these questions require a combination of in-depth interviews, case studies and action research studies, and will allow us to better understand the limitations of the idea, and ways to meet the needs of real-life development projects.

V. Conclusion

This position paper launched the idea of an agile security risk management process supported by the use of empirical data in the format of public catalogs and the NVD database to reduce the level of expertise and past experiences required, which is essential to managing security risks. While embedding security risk management in the development life cycle brings important potential benefits, the idea may not be applicable for some types of systems (e.g., safety-critical systems) and might be only partially applicable to systems requiring traditional security assurance imposed by external parties. Empirical research is required to validate these and other related research issues.

References

[1] P. T. Devanbu and S. Stubblebine, "Software Engineering for Security: a Roadmap," in *Proc. of the Conference on The Future of Software Engineering*, ser. ICSE '00. ACM Press, 2000, pp. 227–239.

[2] K. Beznosov and P. Kruchten, "Towards Agile Security Assurance," in *NSPW'04: Proc. of the 2004 Workshop on New Security Paradigms*. ACM Press, 2004, pp. 47–54.

[3] B. A. Hamilton, "Software Security Assessment Tools Review," National Institute of Standards and Technology, March 2009, available at http://samate.nist.gov/docs/NAVSEA-Tools-Paper-2009-03-02.pdf.

[4] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Engineering*, vol. 10, no. 1, pp. 34–44, 2005.

[5] A. van Lamsweerde, "Elaborating Security Requirements by Construction of Intentional Anti-Models," in *ICSE '04: Proc. of the 26th Int. Conf. on Software Engineering*, ser. ICSE '04. IEEE Press, 2004, pp. 148–157.

[6] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh, "Security Requirements Engineering: A Framework for Representation and Analysis," *IEEE Trans. Softw. Eng.*, vol. 34, no. 1, pp. 133–153, 2008.

[7] T. Kendrick, *Identifying and Managing Project Risk: Essential Tools for Failure-Proofing Your Project*, 2nd ed. New York: AMACOM - American Management Association, 2009.

[8] N. M. E. Dubois, P. Heymans and R. Matulevicius, *Intentional Perspectives on Information Systems Engineering*. Springer-Verlag, 2010, ch. A Systematic Approach to Define the Domain of Information System Security Risk Management.

[9] Walton-on-Thames: Insight Consulting, "CRAMM User Guide," July 2005, risk Analysis and Management Method, Version 5.1.

[10] ISO/IEC-27001/27005, "Information technology. Security techniques. (27001) Information security management systems; (27005) Information security risk management." 2008.

[11] AS/NZS-4360:2004, "Australian/New Zealand Standards, Risk Management," Sydney, NSW, 2004.

[12] F. den Braber, I. Hogganvik, M. S. Lund, K. Stølen, and F. Vraalsen, "Model-based security analysis in seven steps - a guided tour to the CORAS method," *BT Technology Journal*, vol. 25, no. 1, pp. 101–117, 2007.

[13] G. Boström, J. Wäyrynen, M. Bodén, K. Beznosov, and P. Kruchten, "Extending XP Practices to Support Security Requirements Engineering," in *SESS'06: Proc. of the 2006 Int. Workshop on Software Engineering for Secure Systems*. ACM Press, 2006, pp. 11–18.

[14] T. Dybå and T. Dingsøyr, "Empirical Studies of Agile Software Development: a Systematic Review," *Journal of Information and Software Technology*, vol. 50, pp. 833–859, 2008.

[15] J. Peeters, "Agile Security Requirements Engineering," 2005, presented at the Symposium on RE for Information Security.

[16] Sonia and A. Singhal, "Development of Agile Security Framework Using a Hybrid Technique for Requirements Elicitation," in *Advances in Computing, Communication and Control*. Springer, 2011, vol. 125, pp. 178–188.

[17] K. Beznosov, "Extreme Security Engineering: On Employing XP Practices to Achieve "Good Enough Security" without Defining It," in *BizSec: First ACM Workshop on Business Driven Security Engineering*, 2003.