DE GRUYTER

PROJECT RISK MANAGEMENT

MANAGING SOFTWARE DEVELOPMENT RISK

Edited by Kurt J. Engemann and Rory V. O'Connor

BUSINESS & ECONOMICS

Project Risk Management

Developments in Managing and Exploiting Risk

Volume I: Safety Risk Management
Volume II: Project Risk Management
Volume III: Organizational Risk Management
Volume IV: Socio-Political Risk Management

Editor-in-Chief
Kurt J. Engemann

Volume 2

Project Risk Management

Managing Software Development Risk

Edited by
Kurt J. Engemann and Rory V. O’Connor

DE GRUYTER

ISBN 978-3-11-064823-2
e-ISBN (PDF) 978-3-11-065232-1
e-ISBN (EPUB) 978-3-11-064829-4

Library of Congress Control Number: 2020949254

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at http://dnb.dnb.de.

Coverimage: German-skydiver/iStock/Getty Images Plus
Typesetting: Integra Software Services Pvt. Ltd.
Printing and binding: CPI books GmbH, Leck

www.degruyter.com

In Memory of
Rory V. O’Connor
Dear Friend and Esteemed Colleague

Kurt J. Engemann

Advances in project risk management

Introduction

Managing risk is important for any organization and there often appears to be an increasing emphasis on focusing on the downside aspect of risk. However, continuously concentrating on the negative characteristics of risk, without bearing in mind the positive attributes, may be unwise. Significant opportunities may be wasted by always choosing the secure road.

The objective of *Project Risk Management: Managing Software Development Risk*, is to provide a distinct approach to a broad range of risks and rewards associated with the design, development, implementation and deployment of software systems. The traditional perspective of software development risk is to view risk as a negative characteristic associated with the impact of potential threats, the development of risk mitigation plans and the avoidance of potential adverse consequences for software project objectives. The perspective of this book is to explore a more balanced view of software development risks, including the possibility of positive aspects to risk associated with potential beneficial opportunities, and present a view that risk does not always reflect only negative consequences. On the contrary some positive risks may actually represent previously unexplored benefits to a software development project and provide opportunities. Therefore, a balanced approach is required, where software project managers approach negative risks with a view to reduce the likelihood and impact on a software project, and approach positive risks with a view to increase the likelihood of exploiting the positive opportunities.

This volume explores software development risk both from a technological and business perspective. Issues regarding strategies for software development are discussed and topics including risks related to technical performance, outsourcing, cybersecurity, scheduling, quality, costs, opportunities and competition are presented. Bringing together concepts across the broad spectrum of software engineering with a project management perspective, this volume represents both a professional and scholarly perspective on the topic.

In this overview, we preview the book which consists of two parts: chapters covering fundamental concepts and approaches; and, chapters illustrating applications of these fundamental principles.

https://doi.org/10.1515/9783110652321-202

Fundamentals

Cyber-physical systems are systems that simultaneously act in the physical and digital space, comprising both physical and computational processes and involving people (Lee 2008). Examples include drones, robots, autonomous vehicles and smart grids. It is often unclear as a new cyber-physical system emerges, what the risks and opportunities are. With the advances in digitalization, the balance between software-related risks and opportunities is becoming a key decision, but without a thorough insight into the possibilities and liabilities of software, this is a difficult step to take. Hence, companies more commonly follow an approach of product evolution, and avoid large-scale changes in the system. The software architecture of a cyber-physical system is one of the main factors that determine its sustainability from the point of view of development, maintenance, and evolution. However, a software architecture is not inherently good or bad, it is just more or less fit for purpose, and software architecture assessment is an effective way to establish its fitness. In their chapter, Tuovinen, Christophe, Kettunen, Mikkonen and Männistö share their experiences of using a series of software architecture assessment workshops as a mechanism to identify risks and opportunities of an existing cyber-physical system software product line and to help in planning the renewal of the software system accordingly, taking into account the evolutionary line of new features as well as potential future disruptive technologies. The assessments take place at a company that provides industrial automation solutions and that takes the usual risk-oriented view to software engineering. The factors under study include feature creep, sensitivity for control points, and scaling the current product line to meet changing customer demand. They conclude that architecture assessment is an effective way of uncovering risks that bear on architectures' capability to support business.

Traditional approaches in software development assume that it is possible to anticipate a complete set of the requirements in an early phase of the project lifecycle, however, these approaches do not deal well with changes. Agile methods emerged as a response to the bureaucracy of traditional complex methods, the increasing changes in the business environment requiring faster changing, and the growing demand for efficient software development (Pavlič and Heričko 2018). Agile approaches are embraced widely as an answer to the failure of the traditional plan-driven waterfall-based approach (Gupta, George, and Xia 2019). An agile coach is an experienced user of agile methodologies, who can guide others through emphasizing best software engineering practices. In their chapter, Sánchez-Gordón and Colomo-Palacios discuss the role of the agile coach, and carry out a multivocal literature review devoted to identify the risks of introducing such a role by investigating both research and professional literature, including not only the negative consequences, but also the positive aspects that could lead to potential beneficial opportunities. Their chapter aims to benefit both researchers and practitioners by providing a comprehensive and balanced view of the topic.

Agile practices assert the application of a software project risk management activity. Nevertheless, software project delays, costs overruns, and failed projects are still reported in the literature, and Tavares et al. (2019) report that agile development practices lack risk management activities. Agile software project risk management projects have largely ignored decision support analytics to assist with its implicit management, planning, monitoring and evaluation of risks. In contrast, analytics are widely used in many business domains, as illustrated by a decision analytic methodology to address unintended consequences of new technologies (Miller and Engemann 2019). The goal of the chapter by Mora, Wang, Phillips-Wren and Gómez is to create awareness of the usefulness and value of decision-making support systems analytics in software project development. There are opportunities for fostering wider use of these analytics tools in both plan-driven and agile software project approaches which would assist in risk management activities.

Risk management is known to produce a number of benefits, including: identification of favourable alternative courses of action, reduced surprises, and more precise estimates (Bannerman 2008). However, recent research has also shown that these practices are not widely used in software development projects (Odzaly and Des Greer 2014). In their chapter, Dingsøyr and Petit present an exploratory case study of uncertainty management in a large software/hardware development project, with a focus on project/subproject and work package levels. They describe explicit and implicit practices for uncertainty management, and recommend practices to mitigate uncertainty. These exploratory findings offer opportunity for planning larger development projects.

Within the last few years, agile software development has become the mainstream software development paradigm. Among the existing agile methods, Scrum, which is a project management framework, is the most popular. However, Scrum does not explicitly recommend an approach to manage risk. Furthermore, Scrum is mainly based on tacit knowledge, which limits the reuse of information for risk management. In their chapter, Perkusich, Neto, Nunes, Gorgônio, Almeida and Perkusich propose to fill this gap by introducing a knowledge-based risk management approach for Scrum-based software development projects, focusing on risks (both positive and negative) related to the product delivery process. Ward and Chapman (2003) discuss that viewing risk management as uncertainty management enhances the focus on opportunity management, therefore, bringing balance on focusing on both types of risks (i.e., positive and negative). The proposed approach is based on a knowledge-based risk management framework, supported by a Bayesian network that models the main aspects of the Scrum product delivery process, which has been evaluated on industry projects in terms of its practical utility. With the use of the proposed approach, risk management of Scrum projects changes from being based on informal and tacit knowledge to being based on empirical evidence, as registered in the knowledge base. Therefore, instead of depending on the intuition of the project team, risk management decisions are informed and based on data.

Increasing use of algorithms and decision models in all facets of everyday life means that algorithmic bias may lead to some groups being treated less fairly than others. Disparate treatment may be thought of as affecting inputs and the algorithm, and disparate impact as affecting the outcome (Kleinberg et al. 2018). Algorithmic bias affects the effectiveness of algorithms and decisions, and may result in higher social, political, and economic costs and, as technologies evolve, may become a major issue of contention. The chapter by Miller and Engemann discusses bias in the context of algorithms and decision models, and develops various categories to classify biases of various types. Using these categories, they also examine several general bias-related risks, and discuss strategies for managing them.

Applications

Risks emerging out of the interactional nature of the discipline are yet to be examined in sufficient detail. Structures of developer communication vary according to information exchange needs at individual and collective levels. This leads to the emergence of various patterns of developer interaction. In their chapter, Datta, Bhattacharjee and Majumder posit that a clear understanding of the characteristics of developer communication over the software development life cycle is an essential step towards a deeper examination of related risks.. In a case study using development data from a large real-world system they construct an interaction network of developers. From this network, each developer's interaction profile in terms of her involvement in different motifs of communication is identified. After controlling for the effects of interest, engagement, experience, information dissemination and reception, they find statistically significant evidence that more uniform interaction profiles relate to higher workload for developers, while reducing the time taken to complete their tasks. This counter-intuitive nature of the results has a number of implications at individual and organizational levels. The results highlight the need for individual developers to engage in various interactive mores with equal attention. The results can also inform how tools and processes can be tuned at the organizational level, to engender a project culture that is both risk-aware and risk-resilient.

Innovative technical systems have been made possible through complex electrical subsystems that are embedded and interconnected. Complexity has increased on both the system level as well as the component level. Uncontrolled complexity and resulting weaknesses are systematically identified and exploited at the harm and expense of users and manufacturers. Only risk-oriented systematic engineering and management can ensure that security needs are met. The chapter by Ebert provides experience and guidance how information security can be successfully achieved in embedded systems, and outlines risk-oriented security engineering.

It illustrates applicability in the automotive domain, which currently has highest innovation speed across all information technology domains and thus best illustrates both risks and opportunities.

Access control policies prevent unauthorized access to an organization's resources. In a mobile environment, physical location plays an important role in determining whether to grant or to deny access. Incorporating location into access control policies reduces the risk of those resources being accessible to unauthorized personnel. Examples of such policies would be those restricting access to resources to managers only when they are physically in the office. In their chapter, Kozakevitch,, Collins, Lee and Shin discuss an uncertainty-embedded authorization model, enforcement algorithms, and how to handle user requests.

Organizations are increasingly adopting cloud computing to improve their operational capabilities and service effectiveness. Meanwhile, the shared infrastructure in a cloud computing environment exposes severe information security risks, making many organizations hesitate to migrate to a cloud environment. Cloud security control can be categorized into four major types: preventive, deterrent, detective, and corrective control mechanisms (Paul and Aithal 2019). In their chapter, Yoo and Li develop a framework of information security management in the cloud computing environment. The framework identifies different types of security concerns by linking the unique characteristics of cloud computing to the sources of security risks. The framework also discusses factors at different levels (e.g., organization, cloud computing technology, and individual employee) that could explain the occurrence of cloud security risks. By doing so, this chapter develops a set of managerial recommendations that help organizations manage cloud security and optimize investments in safeguarding cloud computing. Based on the framework of cloud security management, organizations can ride the wave of cloud computing by enhancing effectiveness while mitigating potential information security risks.

The market is demanding certifications for software companies to demonstrate how they can satisfy their customers, including an efficient and effective risk management approach. International standards have a key role in establishing consistent terminology, and a common understanding of the concepts is important from an analysis perspective, as it strongly influences the way both reliability and risk are assessed, managed and communicated (Selvik et al. 2020). The International Organization for Standardization (ISO) proposes management system standards, including within the domains of information systems, risk management, information security, and project management. In order to improve and integrate risk management in IT settings with ISO standards as the basis representing an international consensus of practices, Barafort, Mas and Mesquida explore how to improve risk management processes in IT settings from an integrated and management system perspective in multiple ISO standards. Their chapter explores risk management in IT settings from the perspective of ISO standards. A set of artefacts is proposed according to a design science methodology – a process reference model and a process assessment model for integrated risk management processes in IT settings based on ISO standards, with iterations and interactions for improving the proposed solution related to the problem to be solved.

Software development projects always involve a certain element of risk. Avoiding and mitigating those risks should be part of every project plan when embarking on such a project. While this holds true for most development projects, avoiding all risk when developing software is not always a good thing. The chapter by Malara discusses why sometimes business situations make it critical that organizations both embrace and encourage risk taking, especially if their business life is on the line. The appetite for risk taking usually depends on the magnitude of the reward. Calculated risk can be acceptable with the right mitigation practices identified and in place. Mitigation principles and practices unique to each situation make this a complicated topic with many implications. The focus is not always on avoiding and reducing risks, but when and how to embrace risk to increase the likelihood for project, and ultimately, business success.

Conclusion

Managing risk is an enormous challenge that all organizations encounter. Understanding the common characteristics of evolving risks can provide insight into identifying further threats and opportunities to organizations. With these common characteristics understood, risk analytics and professional management can aid in the recognition of additional obscured risks, and be beneficial in providing security and growth for an organization (Engemann 2019). Evaluating risk is demanding, in part because it involves uncertainty. Organizations are often acutely averse to risk to the detriment of exploiting possible opportunities. Although risk is often perceived as undesirable, a proportionate weight should also be given to the potential rewards, when appraising new undertakings.

References

Abrahamsen, E, J. Selvik, and K. Engemann, “On the New ISO Standard on Reliability and Maintenance Data Collection: Terms and Definitions,” *Safety and Reliability*, 2020.

Bannerman, P., “Risk and Risk Management in Software Projects: A Reassessment,” *Journal of Systems and Software*, Vol. 81, pp. 2118–2133, 2008.

Engemann, K., “Emerging Developments in Organizational Risk,” *Continuity and Resilience Review*, Vol.1 No. 1, pp. 26–35, 2019.

Gupta, M., J. George, and W. Xia., “Relationships between IT Department Culture and Agile Software Development Practices: An Empirical Investigation” *International Journal of Information Management*, Vol. 44, pp. 13–24, 2019.

Kleinberg, J, J. Ludwig, S. Mullainathan, and C. Sunstein, “Discrimination in the Age of Algorithms,” *Journal of Legal Analysis*, Vol. 10, pp. 113–174, 2018.

Lee, E., “Cyber Physical Systems: Design Challenges,” *11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, 2008.

Miller, H. and K. Engemann, “The Precautionary Principle and Unintended Consequences,” *Kybernetes*, Vol. 48 Issue: 2, pp.265–286, 2019.

Odzaly, E. and D. Des Greer, “Lightweight Risk Management in Agile Projects,” *26th Software Engineering Knowledge Engineering Conference*, Vancouver, Canada, 2014.

Paul, P., and P. Aithal, “Cloud Security: An Overview and Current Trend,” *International Journal of Applied Engineering and Management Letters*, Vol. 3 No. 2, pp. 53–58, 2019.

Pavlič, L., and M. Heričko, “Agile Coaching: The Knowledge Management Perspective,” *Knowledge Management in Organizations, Communications in Computer and Information Science*, (ed. L. Uden, B. Hadzima, and I. Ting), Springer International Publishing, pp. 60–70, 2018.

Selvik, J., E. Abrahamsen and K. Engemann, “Definition of Reliability and Maintenance Concepts in Oil and Gas – Validity Aspects,” *Safety and Reliability*, Vol. 39 No. 2, pp. 134–164, 2020.

Tavares, B., C. da Silva, and S. de Souza, “Practices to Improve Risk Management in Agile Projects,” *International Journal of Software Engineering and Knowledge Engineering*, Vol. 29 No. 03, pp. 381–399, 2019.

Ward, S. and C. Chapman, “Transforming Project Risk Management into Project Uncertainty Management,” *International Journal of Project Management*, Vol. 21 No. 2, pp. 97–105, 2003.

Contents

Part I: **Fundamentals**

Part II: **Applications**

Part I: **Fundamentals**

Antti-Pekka Tuovinen, François Christophe, Petri Kettunen, Tommi Mikkonen and Tomi Männistö

1 Managing risks and opportunities in cyber-physical systems with software architecture assessments

1.1 Introduction

When a new generation of cyber-physical system (CPS) emerges, it is often unclear which are risks and which are opportunities within the scope of the new generation. With the advances in digitalization, the balance between software risks and opportunities is becoming a key decision, but without a thorough insight into the possibilities and liabilities of software in the system, this is a difficult step to take. Hence, companies more commonly follow an approach where they have a linear model for product evolution, and try to avoid large-scale changes in the system as a whole. Such issues have been encountered in various contexts, including in particular mobile devices, but few practical approaches have been proposed. One of those that has been used in industry is planned staged investments (Savolainen et al., 2013), which divides the life cycle of the product into steps of investment and harvesting. During the former, an investment is made in the system under development by introducing new features and capabilities, and by improving quality. During harvesting, software is maintained at minimum cost, and no large investments in new features or improved quality are made.

The systems architecture of a CPS sets a framework for its key qualities and structures. The software architecture is one of the main factors that determine the sustainability of the system from the point of view of development, maintenance, and evolution. However, a software architecture is not inherently good or bad; it is just more or less fit for purpose. In order to assess the fitness of a software architecture for its particular context and requirements, the architecture can be assessed using established, mature methods. A software architecture assessment (a.k.a. software architecture evaluation) can also have specific goals – identifying risks when planning changes, or, considering the feasibility of further investment in a system vs. its replacement are common reasons for conducting a software architecture assessment, for example.

In this chapter, we share our experiences on using a series of software architecture assessment workshops as a mechanism to identify risks and opportunities of an existing CPS software product line and to help in planning the renewal of the software system accordingly, taking into account the evolutionary line of new features as well as potential future disruptive technologies. In terms of planned staged investments, the

https://doi.org/10.1515/9783110652321-001

goal is to identify opportunities to be gained during the next planned investment period, as well as to manage risks during the ongoing maintenance period.

The rest of the chapter is structured as follows. In Section 1.2, we introduce the case company's CPS domain, software product lines, and software architecture assessments. In Section 1.3, we discuss the role of architecture assessments as a risk management tool in the context of software product lines. In Section 1.4, we present our case study, executed together with a company operating in the domain of cyber-physical systems in industrial automation. In Section 1.5, we provide an extended discussion on our findings. Finally, in Section 1.6, we draw our conclusions.

1.2 Background

The background of this work consists of three different dimensions, Cyber-Physical Systems, Software Product Lines, and architecture assessments. In the following, we introduce briefly each of them in separate subsections.

1.2.1 Cyber-physical systems

Cyber-Physical Systems (CPS) are systems that simultaneously act in the physical and digital space, comprising both physical and computational processes and involving people (Lee, 2008). Typical examples of CPSs include drones, various robots, and autonomous vehicles and larger, complex systems such as Smart Grids. Since a major part in their development includes the design of physical, mechanical and electrical elements, the development has been executed under their terms and engineering disciplines and software has traditionally played only a minor role inside each device and system component independently. The situation is now changing rapidly, and software is becoming a major factor in innovation in CPSs (Lee et al., 2014) (Mikusz, 2014). Modern CPSs are increasingly interconnected and utilize multiple sources of data (Müller, 2017). Such capabilities are inherently software-based.

In the advent of the fourth industrial revolution, the Industrial Internet, software is becoming more and more entangled in physical machines, each of them playing a role in achieving a system level goal (Gilchrist, 2016). Such a goal is accomplished by machines forming a cyber-physical system (Jeschke et al., 2016): a network of machines executing software in a distributed and asynchronous way (Monostori et al., 2016). The impact of cyber-physical systems on industrial services in manufacturing is considerable (Herterich et al., 2015), turning companies that have been designing machinery to software companies.

Proficient design of modern, complex CPSs requires advanced competencies due to their heterogeneous nature, physical world concurrent processes, and timeliness requirements (Khaitan & McCalley, 2015) (Müller, 2017). Notably, there are considerable research problems concerning for example multidisciplinary integrated system architecture modeling. With the increasingly central role of software in most CPSs, developing such new system architecture designs requires extensive software architecture competencies.

As with any software, the architecture plays a key role in ensuring the continuous operation of any CPS. In particular, industrial systems need to be operated continuously regardless of out-of-order issues of components of this system. Due to high reliability requirements, software architecture plays a decisive role in all phases of the life cycle of a CPS with the software development phase having the most impact on the entire CPS sustainability (Törngren & Sellgren, 2018). To meet these goals at system level, software architecture of industrial CPSs needs to answer requirements of system orchestration, machine availability, predictive maintenance, and failure assessment. As for practical guidelines for meeting quality, interoperability and compatibility needs, the Industrial Internet Consortium has developed a reference architecture for designing software components for CPSs (Industrial Internet Consortium, 2015), highlighting the growing importance of software and software architecture in the CPS domain.

New CPS technologies offer significant opportunities, but they also pose considerable risks. The key source of development opportunities is the possibility to build new "smartness" and intelligence into the integrated and interconnected systems in totally new ways. For example, modern electricity network Smart Grids are large-scale CPSs with advanced control functions and automated metering services (Yu & Xue, 2016). However, they have also introduced new software-related risks such as cyber-security issues. Both recognizing such new opportunities and managing the risks call for advanced software architectural capabilities.

1.2.2 Software product lines

A software product line (SPL) (Van der Linden et al., 2007) is a collection of methods, techniques, tools, software components, and other assets that are used to create a collection of related products, sometimes referred to as a product family. The technical components that form the fundamental part of the product line are commonly referred to as core assets. These core assets are then reused in different products, and if necessary, they can be complemented with product-specific software components. While building on flexibility characteristics of software, SPLs can be applied in the design of CPSs (Niemelä & Ihme, 2001). Examples include cars, TVs, mobile phones, and many other mass-manufactured systems in which software plays a key role (Sangiovanni-Vincentelli & Martin, 2001) (Liggesmeyer & Trapp, 2009).

A key element of any SPL is product-line architecture (PLA) that defines how the core assets and product specific components are organized to create products.

In addition to the usual things included in software architectures, PLA also includes information about creating different variants. Building on PLA, a common way to partition an SPL is to organize core assets as a platform that can be extended, specialized, or tailored for product-specific use, at various levels (Myllymäki et al., 2002). Hence, two roles are needed: platform engineering and application engineering, both with different responsibilities. Platform engineering creates reusable components – the platform – that eventually make up the platform, which require assumptions on how future products will be built using them. In contrast, application engineering creates actual products, which requires a stable platform.

As platform and application engineering are run in parallel, but in the end share the same business goals, they need a common management function to steer the development. Examples of management decisions include resourcing of different flavors of engineering, schedules, and customer care. However, as the short-term technical goals of platform and application engineering are different, it is often difficult to balance between the different needs. Moreover, overlooking either type of engineering can lead to severe problems in the long run – focusing only on platform engineering leads to failing to deliver products in a timely fashion, and focusing only on products leads to increasing technical debt in core assets.

Planned Staged Investments (Savolainen et al., 2013) is a technique for managing and rebuilding SPLs in a sustainable way, based on technical and market needs. The overall aim is to manage more effectively SPLs when conflicting requirements simultaneously emerge from needs to redesign and reuse the software.

The key idea of Planned Staged Investments is to differentiate between two different operational modes – investment and harvesting – to coordinate the competing, parallel needs of redesign and reuse. These alternating modes can be characterized as follows:

- Investment: During investment, engineering effort is put into improving reusable asset creation. Development focuses on designing and improving product line’s core assets. In fact, they might even partly integrate product development. As an example, so-called lead products, commonly used in SPLs, are typically representatives of the first generation products built on a new generation of core assets forming the platform.
- Harvesting: During harvesting, benefits are gained from the investment in the form of simplified and faster product creation. The focus is placed on product development, and investments to core assets are minimized to only those that are critical for stability and robustness, thus reducing the need for product line engineering.

To summarize, the investment mode is a step change that requires careful planning, requirements, and technical surveys on technically feasible solutions. In contrast, harvesting mode supports iterative, rapid, and agile product creation. There are also pitfalls associated with the approach (Savolainen et al., 2013). The most obvious one is that a prolonged harvesting stage will always lead to decreased productivity and lower quality while product-specific needs become increasingly difficult to meet owing to accumulating technical debt. Then, the management may consider that an investment needed is actually an indication of poor engineering rather than a logical consequence of the overly extended harvesting period. Therefore, the harvesting period must be long enough to be profitable, and the investment phase must be extensive enough to renew the system. From the technical point of view, it is often difficult to developers to accept that the software made during harvesting contains numerous issues and problems that could be eliminated with some attention from them.

1.2.3 Software architecture assessment

Assessing software architectures is a practical necessity for ensuring that the designed architecture meets its functional and quality requirements (Bosch & Molin, 1999). Over the past twenty years or so, several methods for evaluating and assessing software architectures have been developed (see e.g., (Kazman et al., 2000) (Bengtsson et al., 2004) (Kettu et al., 2008) (Woods, 2012) (van Heesch et al., 2014) (Raatikainen et al., 2014) (Knodel & Naab, 2016)). Providing a comprehensive overview of the various methods falls beyond the scope of this chapter. However, in the following we introduce the salient properties of the prominent approaches that we have used, together with some first-hand experiences.

Two fundamentally different approaches to software architecture assessment exists: those based on experts asking questions and reviewing architectural artefacts (e.g., ATAM (Kazman et al., 1998) (Kazman et al., 2000)) and those based on measurements.

When performing reviews, the assessment team first collects information regarding the expectations of the stakeholders of the system. In scenario-based review approaches, the concerns and questions are posed as concrete scenarios involving a particular situation and stimuli that the system must respond to in a satisfactory manner. The scenarios exhibit important quality concerns of stakeholders, and they are evaluated together with the team responsible for the architecture. Evaluating a scenario means determining, with technical experts, whether or not the system will be able to produce a satisfactory response and identifying those aspects of the design that either support or inhibit reaching a favorable outcome. Scenarios can be predefined and reused in many different assessments virtually unmodified because they often address common situations related to, for example, security and maintainability.

As an example of another kind of review, the DCAR method (van Heesch et al., 2014) focuses on identifying architectural design decisions (meaning both a technical solution for a design issue and the actual resolution to use it), their rationale, and the relationships between the decisions. The decisions are then ordered by importance. In the evaluation part, the participants (typically the architect, the product owner, domain experts, and evaluation facilitators) discuss the forces affecting the most important decisions and their consequences (i.e., pros and cons) and vote whether each decision is good or needs to be reconsidered.

We have found it valuable to combine both the DCAR and ATAM approaches into a workshop style of architecture assessment (Tuovinen et al., 2017). The DCAR part of the workshop focuses on recovering the key aspects of the design and its history, while the scenario part can explore also future aspirations, opportunities, and risks and their impact on the architecture. This is the approach we have followed also in the case reported here.

Using measurements for assessing software architectures contrast expert reviews (like ATAM and DCAR above) in the way that the goal is not to raise questions about the system but to produce answers to questions as hard numbers. However, whereas not yet implemented designs can often be reviewed, measurements need a concrete object to measure: a simulation, prototype, or an at least partially implemented system in a test environment. For example, Sobhy et al. (2020) present an approach to evaluate architectural options by using reinforcement learning to find an optimal balance of incurred costs and benefits of alternative architectural choices run in a simulated system. It is important to recognize that such measurements are specific to a particular system and its quantified requirements. Indeed, there are no generally applicable, universal measures for the "goodness" of an architecture: an architecture is only more or less fit for its purpose as defined by the quality requirements of the system. Furthermore, relying too much on numbers (e.g., static metrics computed from code) can have a detrimental effect on quality – you will certainly get what you measure but that may not be what you actually need (Knodel & Naab, 2014 a). However, reviews and measurements can be used together as the RATE architecture assessment approach demonstrates (Knodel & Naab, 2014 b) (Knodel & Naab, 2014 a).

Scenario-based methods typically require effort and input from several stakeholders. A thorough assessment typically requires two or three full day meetings over a few weeks of calendar time and the participation of several key persons, adding up to tens (even hundreds) of person hours (Knodel & Naab, 2014 b). There is also a learning curve (Reijonen et al., 2010) (Woods, 2012). Unsurprisingly, scenario-based methods are often perceived as heavy by the practitioners (Banijamali et al., 2019) (Cruz et al., 2019). On the other hand, assessment results are valuable and usually well received (Knodel & Naab, 2014 b) (Reijonen et al., 2010) (Ferber et al., 2002) (Cruz et al., 2019), although they can be hard to quantify for management for decision-making (Knodel & Naab, 2014 b). As an example of usefulness of the results, in (Knodel & Naab, 2014 a) the authors state that 75% of the over 50 assessments they performed led to concrete actions. Scenarios are a powerful tool not only for assessing the adequacy of the system under evaluation but also for making the technical people aware of the needs of the business and for making the business people aware of the opportunities provided and the challenges and risks posed by technology (Knodel & Naab, 2014 b) (Reijonen et al., 2010).

There is evidence that scenario-based assessment and its derivatives are the most well-known methods in industry (Banijamali et al., 2019). For recent reports on industrial experiences on architecture assessment and assessment methods see e.g., (Cruz et al., 2019) (Tuovinen et al., 2017) (Bellomo et al., 2015) (van Heesch et al., 2014) (Knodel & Naab, 2014 b) (Woods, 2012) (Reijonen et al., 2010).

1.3 Software architecture assessment as a risk management instrument

In the world today, there is a vast number of software-intensive CPSs that have different missions, size, technological basis, and dependencies. There are systems that are in their inception and there are very mature systems that have been around for decades. Architecture assessment can be performed not only in the early phase of the life-cycle of a CPS but also in some significant turning point in its life. Therefore, assessments can have very different goals and each system has its unique characteristics (Knodel & Naab, 2016, p. 125). Still, there are common problems that many systems have to cope with. Typical assessment goals and questions include (paraphrased from (Knodel & Naab, 2014 b)):

- How suitable is the architecture as a basis for future products?
- Which framework or technology fits the needs best?
- How can performance, maintainability, or other important qualities be improved?
- How can the system be modularized to meet new productization and other business goals?
- What is the overall quality of the system and should it still be maintained or scrapped and redeveloped?
- How well does the designed architecture meet the key requirements?
- How can the system be modernized to meet new requirements and use modern technologies?

Identifying risks and is an integral part of ATAM (Kazman et al., 2000) (Clements et al., 2002) and a major motivation for software architecture assessment in general. However, an explicit link to risk management processes is usually missing from the descriptions of the assessment methods. We try to bridge this gap here by projecting the risk related assessment activities onto the risk management process defined in the international standard ISO/IEC 16085–2006 (International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) [ISO/IEC], 2006) that is compatible with the system and software life-cycle process standards ISO/IEC 15288 and ISO/IEC 12207.

When evaluating scenarios in ATAM, one outcome is to identify architectural risks. In this context, labeling an architectural design decision as a 'risk' means that the decision affects negatively an important quality attribute embodied in some scenario and hence poses a risk that the resulting system will not meet stakeholders' requirements. The formulation of a scenario should state the required response in as concrete terms as possible, which makes the *risk criteria* (ISO/IEC, 2006) explicit. So, identified risks are mainly about failing to reach the desired level of some capability. ATAM does not mandate how to document the risks in terms of *risk exposure* (ISO/IEC, 2006), for instance. However, because the evaluated scenarios reflect the most important stakeholder requirements, the risk of not satisfying them should be taken seriously.

ATAM recommends collecting risks that have a common (or closely related) root cause in 'Risk Themes' for easier linking to business goals and for reporting to decision makers. Themes correspond to *risk categories* (ISO/IEC, 2006), although themes are often fine grained focusing on technical aspects. The purpose of collecting risks and risk themes is to facilitate planning of mitigating actions thereof. However, proposing *risk treatments* (ISO/IEC, 2006) is out of scope of ATAM and software architecture assessment in general. So, in terms of the risk management process defined in (ISO/IEC, 2006), the role of architecture assessment is mainly as a task in the *performing risk analysis* activity – focusing on architectural design decisions and their consequences.

In (Bass et al., 2007), a retrospective analysis of 18 ATAM assessments was done to find patterns in the risk themes identified. A characterization of 99 themes into 15 categories was developed. The categories range from architecture (run-time & development-time qualities) to processes and organization, which demonstrates the wide range of issues that can come up in assessments where business goals act as a starting point for deriving assessment criteria (i.e., the scenarios). The main findings of the study were that twice as many risk themes stem from "omission" rather than "commission". That is, they concern design decisions not done, missing or misunderstood requirements, or other overlooked issues rather than the consequences of the architectural decisions already made. Interestingly, the study did not find any correlation between the risk themes identified and the requirements or the domain of the assessed system. That is, the type of a system does not seem to predict what kind of risks will come up. As a practical recommendation, the authors suggest that assessors should be acutely aware of risks stemming from the organizational context and the process of architecting rather than the kind of system under development while being on a constant lookout for important things missed.

Managing the risk related to changes in software is a major reason for doing architecture assessments according to Knodel and Naab (2014 a). They see two distinct ways in which architecture assessment can provide input for mitigating the risks related to software change requests: (1) by evaluating how the system and its architecture can accommodate a set of anticipated changes (that are more or less likely to happen), and (2) by determining the potential impact of concrete change requests currently at hand. The engineering branch of a development organization typically initiates these kinds of assessments. As examples of external initiators of assessment they give a potential customer who wants to gauge risks prior to investment, or an existing customer who wants to sort out known problems (Knodel & Naab, 2014 a) (Knodel & Naab, 2014 b). In a case study focusing on the adaptability of a CPS based on its software architecture (Mayrhofer et al., 2019), the authors evaluate alternative architecture designs using four criteria concerning well-known aspects of design and implementation that affect how well the system can adapt to changing needs and execution context.

Because of the wide range of modern new CPSs, their potential risks stem from a variety of sources. Not only the cyber parts but also the electronic, hardware and mechanical parts in conjunction to the humans involved must be taken into account. In addition, interconnected systems add to the complexity. System risk factors like safety and security are crosscutting. Consequently, engineering high-confidence CPSs requires advanced multidisciplinary competencies and co-development (Müller, 2017).

From the discussion above, it is clear that risk analysis in architecture assessment is typically focused on identifying things that could go wrong in the architecture and its development leading to a systemic failure. The findings are distilled and reported in terms that are understandable for business owners and managers so that the findings can be fed into the risk management process (e.g., into *the project risk profile* (ISO/IEC, 2006)) so that the managers can decide about the *risk action requests* (ISO/IEC, 2006) for treatments to mitigate or remove risks. Naturally, immediate corrective actions can be agreed on during assessment if managerial decisions are not required.

However, the literature is lacking examples of viewing risks as opportunities – of being proactive and recognizing options instead of just reacting to changes forced on by external developments. Some assessment methods do explicitly mention recognizing opportunities for architectural improvements as a motivation for assessments.

On the other hand, in addition to technical findings, other positive effects of assessments have been recognized. Because an architectural assessment usually means a deep discussion about product goals and technical possibilities, it not only helps to create a common frame of reference for the business and technology sides but it also provides a rare chance to share experiences, knowledge, and the rationale behind architectural choices (Reijonen et al., 2010) (Knodel & Naab, 2016, p. 6) in the organization. Also, it gives the opportunity to educate business owners about the potential of technology and the existing software assets. These ‘soft effects’ may in practice be even more valuable than the hard technical results (van Heesch et al., 2014). Therefore, we wanted to explore in our case study how to bring in the other side of risk analysis, recognizing opportunities, as an additional perspective to architectural assessment.

1.4 Case study

In this section, we present an improvement case study we conducted at a company that provides industrial automation solutions. The goal of the case study was to help the case company revisit its CPS software product line in preparation for the foreseen increasing role of the software in the future. A series of software architecture assessment workshops were used as the concrete mechanism to facilitate discussions and to identify risks and opportunities related to the software.

Giving a detailed account of the product line and the technical findings is not the purpose of this work and, consequently, we describe the product and the findings in general terms. Our focus is on describing the assessment process, its conduct, and the value of the outcomes.

1.4.1 Case company and case product

The company provides industrial automation solutions for controlling various devices and follows the usual risk-oriented view to software they are engineering. The factors they would like to study relate to potential risks like feature creep, sensitivity for control points, and scaling the current product line to meet changing customer demand. At the same time, various opportunities have been identified, including new business openings, widening the scope of the product line, reducing overheads and shortening lead-times in developing customer specific variants of the software, and reducing the need for bespoke device interfaces by promoting and embracing new standards in the field.

We call the case product under study Operations Control System (OCS). Figure 1.1 visualizes the functional scope of OCS. The system controls various industrial Devices using the Operations and Device Control Data provided by human operators. Individual Devices are typically combined into conglomerates that together perform an industrial process with the help of additional hardware (System HW). The OCS exchanges also information with Enterprise Resource Planning systems and other Value Adding Processes. OCS has gone through significant architectural and technological changes over its life cycle. The installed base of the system (base version and variants) is in the thousands.

From the company's perspective, the motivations for conducting a review of the software architecture of OCS comprised of the following questions:

1. Are the architecture and the technological choices on as sound a basis as we think?
2. Do outside experts see any risks or weaknesses?
3. How long can we keep on adding features to a single platform to serve growing customer needs and what would be the options?
4. How far does the performance of the system scale up in terms of the amount of operational data and the number of devices controlled?

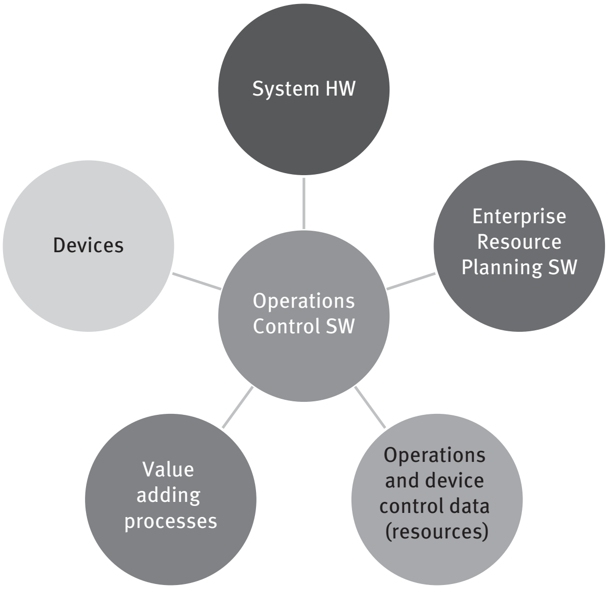

System HW
Devices
Enterprise Resource Planning SW
Operations Control SW
Value adding processes
Operations and device control data (resources)

Figure 1.1: Functional scope of Operations Control Software.

1.4.2 Data collection

Table 1.1 lists the participants of the series of workshops. Eleven people (the Informants I1–I11) participated from the case company and five from University of Helsinki (the Researchers R1–R5). As the table shows, the informants were very experienced and had core competences bearing on the product. The researchers had significant academic and industrial experience.

Table 1.2 lists in chronological order the face-to-face meetings held at the company premises during the study that were the primary way of collecting the data for the study. The actual conduct of the assessment process will be explained in required detail below. The table gives the duration of each event, lists the participants using the IDs given in Table 1.1, and explains the main outcomes or purpose of each event.

As we can see from Table 1.2, there was strong presence from the company in each event, which shows a high level of commitment. It is in fact remarkable that the key persons found time in their busy schedules for this work; it is a common experience that 'daily workload wins over architecture evaluation' (Knodel & Naab, 2016, p. 119).

Although the original planned timetable was not met, all the parties showed flexibility and resilience in seeing the work through. The meetings had clear goals and although the discussions did sometimes take a meandering course, they resulted in a wealth of high quality data. This is also reflected by the actual results

Table 1.1: Study participants.

Id	Role	Experience and expertise
Case Company Employees		
I1	Dept. manager	20 y, responsible for the study at the company
I2	Chief architect	15 y, responsible for architecture of the target system Dev. Dev.
I3	manager	20 y, responsible for the development of the target system
I4	Product Owner	10 y, product manager for digital services
I5	CDO	20 y, senior executive, chief digital officer of the company Sales
I6	agent	10 y, customer interface, sales support
I7	Engineer	15 y, developer, user experience
I8	Engineer	10 y, developer, robotics
I9	Engineer	10 y, developer
I10	Engineer	15 y, architect, user interface Programmer
I11		1 y, doing Master's thesis on the topic
Researchers (University of Helsinki)		
R1	Professor	10 y industrial experience, 15 y academic research in industrial collaboration
R2	Professor	3 y industrial experience, 25 y academic research in industrial collaboration
R3	Senior researcher	15 y industrial experience, 10 y academic research in industrial collaboration
R4	Researcher & lecturer	12 y industrial experience, 10 y academic research
R5	Post Doc. Researcher	3 y industrial experience, 10 y academic research

obtained in the end and in the expressed interest of the company to continue the co-operation with the researchers in this area.

1.4.3 Chronological description of activities and events

The activities of the study were centered on the main events recorded in Table 1.2 over approximately seven months of calendar time. The first meeting, Workshop 1, introduced the product and the company's current and projected future business needs. In

Table 1.2: Data collection events (at the company premises).

Event	Duration [h]	Participants	Focus of Outcomes
Workshop 1, Nov 2018	4	I1, I2, I3, I4, I5, I6, I7 R2, R3, R4, R5	Kick off and introductions, overview of the CPSs of the company and the short-term and longer-term business needs
Workshop 2, Feb 2019	3,5	I1, I2, I3, I4, I11 R1, R2, R3, R4, R5	Architecture assessment tutorial, setting goals for the assessment
Workshop 3, Feb 2019	5	I1, I2, I4, I7, I8, I11 R2, R3, R4, R5	Architecture presentation of the OCS core system, formulating the initial list of design decisions and scenarios
Workshop 4, May 2019	5	I1, I2, I3, I4, I7, I8, I9, I10, I11 R1, R2, R3, R4, R5	Review of the documented design decisions
Workshop 5, June 2019	5	I1, I2, I4, I7, I8, I11 R2, R3, R4, R5	Prioritization of scenarios and evaluation of the most important ones

this meeting, the general objectives for the study and the forms of co-operation were agreed. After the meeting, the research team formulated the first plan with an overall timetable.

Workshop 2 consisted of a tutorial about software architecture and software architecture assessment given by the research team. Video and other materials were provided for self-study at the company. In addition, the goals and the scope (focusing on the core functional parts of OCS device control and data management) were set in the meeting. After the meeting, a more detailed plan for the next workshop was produced. The idea was to follow the DASE approach of lean assessment explained in (Tuovinen et al., 2017). Following the approach, the research team prepared a preliminary list of design decisions and scenario sketches. The company representatives were asked to come up with their own suggestions for scenarios based on the researchers' list, which they did.

Workshop 3 began with a presentation by the OCS architect (Informant I2) about the design of the system under study and about the most recent changes it had gone through, as well as the reasoning behind. During the presentation, the researchers asked questions and collated lists of important aspects of the design in order to reconstruct a list of design decisions (decisions had not been systematically recorded before). In addition, possible scenarios were sketched during the first part of the workshop. The original plan was to select the most important decisions and to document them for analysis and voting ("OK" – "OK with some issues" – "Not OK") during the first part of the workshop and then, during the second part of the day, form a list of scenarios and evaluate them. However, this turned out to be an unrealistic plan. The architecture presentation, the questions, and the discussions on the aspects of the design and their rationale took almost all of the time. There was no time left for documenting decisions, but some time was used to go through a few scenarios prepared by the researchers in advance. However, a good picture of the architecture and the design issues was acquired. At the end of the day, it was clear that two full day meetings would be needed in order to analyze the design decisions and evaluate scenarios properly.

After Workshop 3, the researchers formulated a top list of architectural design decisions and sent them over to the company for commenting and documenting using the appropriate DCAR template.[1] The company was again asked to prepare scenarios. Over the next few months, the company representatives went through the initial list of decisions selecting the most important ones from the list and adding some decisions they thought were relevant. This resulted in eleven carefully documented decisions. They also worked on scenarios but they found that rather difficult. There was also a lack of time for the work.

The goal of Workshop 4 was to evaluate the design decisions documented by the company representatives. Thanks to the thorough preparations of the company people, the evaluation went smoothly and all documented decisions were analyzed and voted on. Only the informants with the relevant technical knowledge from the responsible development team were allowed to vote. Several issues were noted down. In addition, a brief look at the few scenarios prepared so far was taken. It was clear that effort and help from the researchers' side was needed to move this task forward. The document including the decisions and the voting results (marked using a 'traffic-light' coloring scheme for OK with some issues–not OK) as well as some comments was sent at the end of the workshop to the company representative.

In the final phase of the assessment, the researchers prepared fifteen scenarios divided into four themes. The themes addressed (1) the current strategic goals of the company, (2) potential technological and business developments that could present opportunities or pose risks, (3) threats, and (4) software development topics. The scenarios were partially documented using an ATAM-style scenario template, and a separate spreadsheet was prepared that listed the names and other characteristic attributes of the scenarios. The characteristics include the usual risk-related factors of probability and potential impact to business, the estimated time frame for the realization of the scenario, the difficulty or effort of realizing the scenario (where applicable), and whether or not the scenario includes opportunities or risks (or both). The company representatives ran their own scenario gathering sessions and added scenarios and filled in some of the known attributes of the scenarios in the sheet. This

[1] http://www.dcar-evaluation.com/?page_id=4

resulted in the final list of 22 scenarios in the four themes with a relatively even distribution.

Workshop 5 was started by first reviewing the list of scenarios and then selecting those that the participants considered the most important. This resulted in six scenarios with at least one from each theme. Next, the scenarios were evaluated by discussing how the architecture would either support or not achieving a favorable outcome. However, not all scenarios were actually stated in a way that would have allowed determining a definitive response. Some of the scenarios represented such a visionary state that they were very much outside the scope of the actual system under study. These could not be handled in a meaningful way and they were left for future when there would be an actual design to reflect on. Overall, six scenarios were evaluated thoroughly. Based on the session, three actions points were recorded for the company for immediate execution. The actions concerned the current version of the system.

In addition to the face-to-face information sharing in the workshops (Table 1.2), the case company provided during the study period supplementary documentation and the presented materials to the researchers. These were especially valuable given that the researchers were not experts of the industry domain of the case company.

1.4.4 Results

The major findings resulting from the reviews of the design decision and the scenarios of the assessed core part of the software system are listed in Table 1.3. The findings are categorized by the expected time frame for required actions from the company's side, ranging from Immediate (do now) to Long (in a few years), and by a uniting topic, or, risk theme, as they are called in ATAM. Each entry also shortly describes what kind of risk or opportunity is involved. Because the details of the findings are not important for this exposition, we describe the issues in general terms. We have included an indicator (*D* for Decision-based review and *S* for Scenario-based review) for the phase of review where the issue was discovered and recorded; some issues came up in both reviews.

Some of the scenarios turned out to be difficult to prioritize in the assessment. For example, although the participants from the platform team (responsible for developing the OCS core software) acknowledged the customer need for a cloud-based system solution, they also saw this approach risky for the time-critical functions of OCS. Consequently, there is a potential trade-off between important system qualities, and the company wanted to discuss the impact of different options confidentially with their customers.

At a general level, the cooperation between the company and researchers was mutually beneficial. In particular, the iterative nature of the approach that we followed was essential because it provided time for both parties to understand the details, practicalities and limitations of the other party. In other words, the researchers

Table 1.3: Identified risks and opportunities.

Time frame	Topic	Risk or Opportunity
Immediate (do)	Technical debt	A testing application suffers from feature bloat and is difficult to maintain, which is a minor but non-trivial risk for project work (D†)
Short (start)	Performance	Hard numbers about certain areas of performance are missing which is a potential risk for some projects (S‡)
	Security	OCS uses standard network security mechanisms that depend on the facilities provided by the customer. Because industrial processes are moving towards on-line computing, a thorough analysis of ensuing security risks should be conducted in order to device appropriate countermeasures. (D, S)
Medium (year)	Sharing of factory resources	Sharing of common operational resources between devices controlled by separate OCS instances is a potentially important feature that requires some system and software architecture work. This is both an opportunity and a risk because the demand for such solutions is increasing. (S)
Long (few years)	Technology dependency	A dependency on a technology was identified as a potential future risk that needs to be addressed if the situation changes. (D, S)
	Cloud-based system	There is an inevitable technology trend towards cloud-based services and data sharing in the industry domain, which was categorized potentially as an opportunity to be addressed in the future, but also as a risk if left unaddressed for too long. (S)

Note: † D = Decision-based review ‡ S = Scenario-based review

learned a lot about products and product development at the case company, and the company representatives had several lessons about software architecting and architecture assessments. For example, in the beginning, scenarios as a concept were not so well first understood by the company representatives and thinking of them spurred vivid discussion, but capturing them in text was easily left for later. However, during the course of the workshops, the company representatives were quick to pick up the idea of using a concrete example to demonstrate a technical detail in their design, to the extent that scenarios might become a permanent means to justify technical decisions in the case company. Overall, working with design decisions was easier for them than working with the scenarios.

When asked for feedback afterwards, the Department manager (Informant I1) stated that they found the results useful for the company. The findings will help in developing the current platform further and when doing the groundwork for a new architecture. They also valued the systematic way of evaluating architecture, and they appreciated the outside view that the researchers brought to the process. In this way, they found their time well spent.

1.5 Discussion

As already mentioned, our experiences are based on continuous, iterative cooperation with the case company. In the following, we provide an extended discussion to some of the key elements of our approach, and how they are reflected in our experiences with the case company.

1.5.1 Role of planned staged investments

Since architecture assessments utilize scenarios as a mechanism for identifying risks and potential problems, they at the same time are also an effective mechanism for identifying opportunities, or options that can be easily incorporated in the existing design. Furthermore, increasing risk awareness associated with the present design also enables considerations regarding actions to be taken to mitigate the risks as well as to improve the design in a rational, planned fashion rather than having to resort to hacks at the last possible moment on a per-customer basis.

To summarize, an evaluation of the present architecture with regard to risks it contains also enables thinking of potential directions for the future versions, thus unveiling potential opportunities. Furthermore, a timeline can be created to highlight the schedule for mitigating risks and grasping the opportunities. Based on the timeline, it is then possible to allocate different features to releases, using Planned Staged Investments as the strategy for the allocation.

Based on the experiences with the case company, it is clear that the most urgent issues will be directly included in the different products, with the present version of the platform as the baseline. Moreover, some of the features might even be patches to already existing systems, in particular when considering security-related risks in case of connected systems. In contrast, some road mapped value adding functionalities of the future may require a new platform so that they can be scheduled for release to the whole product line.

1.5.2 Lessons learned: Walking the line between risk and opportunity

Balancing between risks and opportunities turned out to be surprisingly difficult. The tendency was to always consider risks first, and opportunities only later. To some extent, this can be explained by the fact that the platform team is used to getting requirements from the product teams, and there is limited experience in being able to put in ideas for future features to product teams proactively. An exception to this observation is actions to renew the software technically from within, meaning that their newer counterparts could replace older, partly deprecated subsystems. We believe that this is a somewhat natural situation when considering platform teams operating in the CPS domain. The responsibility for implementing and maintaining the software platform weighs more on the teams than visioning new products.

To improve the situation, assessments would require even deeper participation by people from the customer interface who would be closer to the needs and everyday life of the customers. Moreover, they would be at a better position to consider the opportunities and their importance from the business perspective.

Overall, we experienced and discovered several notable learnings and findings in our industrial case study presented in Section 1.4:

- Planning and performing software architecture assessments in systematic ways require significant resources – particularly time – both from the assessors and from the software development organization.
- In case of large systems such as the OCS, the scope and focus of the assessments should be planned and prioritized according to realistic budgets.
- Because CPS software is by nature deeply coupled and intertwined with the other elements of the system and its operating environment, it is imperative to have sufficient high-level comprehension of the entire CPS in order to be able understand the role and dependencies of the software (e.g., hardware connections) in the whole system (c.f., Figure 1.1).
- Even when conducting just software architecture assessment, the key business drivers and particular company targets should be known at a general level. That helps rationalizing the design choices in the context. Consequently, the software assessors should have access to such information in advance and preferably also the business and product stakeholders participating in the actual assessment process as we did in our company case.
- In practical industrial settings, the architectural knowledge may be partially tacit and the documentation incomplete. This is understandable in particular in cases of large systems with very long life times (even tens of years).

The assessors should be ready to work on such knowledge constraints. It is then also important to be able to discuss directly with the senior software designers who can recollect the key information at the time of the architectural decision-makings possibly done many years ago.

1.5.3 Threats to validity

The validity of as study is basically about the knowledge claims that can be made based on the results (Shadish et al., 2002). As our intent was to gain experiences on the usage of particular architecture evaluation methods, one particular issue in terms of validity is that of the role of the evaluation approach itself in the results achieved. The separation of the approach used from the experience of the facilitators in the actions taken is fundamentally hard. In this study, the researchers, who acted as the facilitators, have a rather high level of experience in industrial software engineering and in software architecture research and practice in particular. This is something one may need to take into account if aiming to apply (generalize) the results in other cases. On the other hand, the approach to use was defined in advance and clearly documented while doing, so the guiding decisions made by the facilitators were not based on intuition or experience alone.

In terms of construct validity, even the central concepts of the study area are not uniformly defined and much of the domain terminology was not initially familiar to the researchers, and therefore, a risk for misunderstandings is real. However, as the collaboration with the company representatives and researchers was very tight, a form of member checking (Creswell, 2009) was continuously used, as the understanding of the researchers was reflected back to the company participants and special attention was paid on trying to ensure we were talking about the same thing. Furthermore, the lack of domain understanding potentially leading to misunderstanding by the researchers was, at least partially, alleviated by the emphasis on the need of the case company participants to understand the overall process and take the responsibility of the domain issues.

1.6 Conclusion

We have reported here the practical experiences we gained in using architecture assessments as a basis for identifying risks and opportunities in the domain of CPSs. The findings of our architecture assessments are two-fold. On the one hand, the case company found it easy to discuss scenarios that are close to its event horizon and build on business requirements from existing customers. These are hardly the key opportunities for future business, but rather contain potential risks. On the other hand, getting to a level where business benefits of extended digitalization and more elaborate software features will start to emerge requires in-depth connection with the case company and long-term commitment to elaborate the opportunities thoroughly. Additional discussions including the company top-management setting the business strategy and positioning of the particular product offering would be grounding.

We can conclude that architecture assessment is an effective way of uncovering risks that bear on architectures' capability to support business. This is especially true when examining an established system; the assessment will help to determine and affirm the limitations and the scope of the current design. However, addressing opportunities is not so straightforward. Although these can be recognized and discussed, they may not fit the current scope of the system and thus be difficult to analyze further – unless there already is a clear requirement for such features from business owners. A possible way forward would be to develop alternatives for a future architecture and assess them against the opportunistic scenarios to pave the way for creating a transition path from the current system to the new one.

Acknowledgments: The authors wish to express their sincere thanks to the case company. This research was funded by DIMECC[2] and the Finnish public funding agency Business Finland.

References

Banijamali, A., Heisig, P., Kristan, J., Kuvaja, P., & Oivo, M. (2019). Software Architecture Design of Cloud Platforms in Automotive Domain: An Online Survey. *2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA)* (pp. 168–175). Kaohsiung, Taiwan: IEEE. doi:0.1109/SOCA.2019.00032

Bass, L., Nord, R., Wood, W., & Zubrow, D. (2007). Risk Themes Discovered through Architecture Evaluations. *2007 Working IEEE/IFIP Conference on Software Architecture (WICSA'07)*. doi:10.1109/WICSA.2007.37

Bellomo, S., Gorton, I., & Kazman, R. (2015). Toward Agile Architecture: Insights from 15 Years of ATAM Data. *IEEE Software*, *32*, 38–45.

Bengtsson, P., Lassing, N., Bosch, J., & van Vliet, H. (2004). Architecture-level modifiability analysis (ALMA). *Journal of Systems and Software*, *69*, 129–147. doi:https://doi.org/10.1016/S0164-1212(03)00080-3

Bosch, J., & Molin, P. (1999). Software architecture design: evaluation and transformation. *Proceedings ECBS'99. IEEE Conference and Workshop on Engineering of Computer-Based Systems*, (pp. 4–10).

Clements, P., Kazman, R., & Klein, M. (2002). *Evaluating Software Architectures: Methods and Case Studies*. Addison-Wesley.

Creswell, J. W. (2009). *Research Design: Qualitative, Quantitative and Mixed Methods Approaches*. Sage Publications.

Cruz, P., Astudillo, H., Hilliard, R., & Collado, M. (2019). Assessing Migration of a 20-Year-Old System to a Micro-Service Platform Using ATAM. *IEEE International Conference on Software Architecture Companion (ICSA-C)*, (pp. 174–181). Hamburg.

Ferber, S., Heidl, P., & Lutz, P. (2002). Reviewing Product Line Architectures: Experience Report of ATAM in an Automotive Context. In F. van der Linden (Ed.), *Software Product-Family Engineering* (Vol. 2290, pp. 364–382). Springer. doi:10.1007/3-540-47833-7_33

[2] https://www.dimecc.com/

Gilchrist, A. (2016). *Industry 4.0: The Industrial Internet of Things*. doi:10.1007/978-1-4842-2047-4

Herterich, M. M., Uebernickel, F., & Brenner, W. (2015). The impact of cyber-physical systems on industrial services in manufacturing. *Procedia Cirp*, *30*, 323–328.

Industrial Internet Consortium. (2015). Industrial Internet Reference Architecture. *Technical Report*. doi:10.1371/journal.pone.0158503

International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC). (2006). International Standard ISO/IEC 16085:2006 Systems and software engineering – Life cycle processes – Risk management. ISO/IEC.

Jeschke, S., Brecher, C., Meisen, T., Özdemir, D., & Eschert, T. (2016). Industrial Internet of Things and Cyber Manufacturing Systems. doi:10.1007/978-3-319-42559-7_1

Kazman, R., Klein, M., & Clements, P. (2000). *ATAM: Method for Architecture Evaluation*. Tech. rep., Carnegie Mellon Software Engineering Institute.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., & Carriere, J. (1998). The Architecture Tradeoff Analysis Method. *Proceedings of the Fourth IEEE International Conference on Engineering of Complex Computer Systems, 1998. ICECCS '98*, (pp. 68–78). doi:10.1109/ICECCS.1998.706657

Kettu, T., Kruse, E., Larsson, M., & Mustapic, G. (2008). Using Architecture Analysis to Evolve Complex Industrial Systems. In R. de Lemos, F. Di Giandomenico, C. Gacek, H. Muccini, & M. Vieira (Eds.), *Architecting Dependable Systems V* (Vol. 5135, pp. 326–341). Springer. doi:10.1007/978-3-540-85571-2_14

Khaitan, S. K., & McCalley, J. D. (2015). Design Techniques and Applications of Cyberphysical Systems: A Survey. *IEEE Systems Journal*, *9*, 350–365. doi:10.1109/JSYST.2014.2322503

Knodel, J., & Naab, M. (2014 a). Mitigating the Risk of software change in practice: Retrospective on more than 50 architecture evaluations in industry (Keynote paper). *2014 Software Evolution Week – IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, (pp. 2–17). doi:10.1109/CSMR-WCRE.2014.6747171

Knodel, J., & Naab, M. (2014 b). Software Architecture Evaluation in Practice: Retrospective on More Than 50 Architecture Evaluations in Industry. *Software Architecture (WICSA), 2014 IEEE/IFIP Conference on*, (pp. 115–124). doi:10.1109/WICSA.2014.37

Knodel, J., & Naab, M. (2016). *Pragmatic Evaluation of Software Architectures*. Springer.

Lee, E. A. (2008). Cyber physical systems: Design challenges. *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, (pp. 363–369).

Lee, J., Kao, H.-A., & Yang, S. (2014). Service innovation and smart analytics for industry 4.0 and big data environment. *Procedia Cirp*, *16*, 3–8.

Liggesmeyer, P., & Trapp, M. (2009). Trends in embedded software engineering. *IEEE software*, *26*, 19–25.

Mayrhofer, M., Mayr-Dorn, C., Zoitl, A., Guiza, O., Weichhart, G., & Egyed, A. (2019). Assessing adaptability of software architectures for cyber physical production systems. In T. Bures, D. L., & I. P (Eds.), *ECSA 2019, Lecture Notes in Computer Science* (Vol. 11681, pp. 143–158). Springer Nature Switzerland AG. doi:10.1007/978-3-030-29983-5_10

Mikusz, M. (2014). Towards an understanding of cyber-physical systems as industrial software-product-service systems. *Procedia CIRP*, *16*, 385–389.

Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Ueda, K. (2016). Cyber-physical systems in manufacturing. *Cirp Annals*, *65*, 621–641.

Müller, H. A. (2017). The Rise of Intelligent Cyber-Physical Systems. *Computer*, *50*, 7–9. doi:10.1109/MC.2017.4451221

Myllymäki, T., Koskimies, K., & Mikkonen, T. (2002). Structuring product-lines: A layered architectural style. *International Conference on Object-Oriented Information Systems*, (pp. 482–487).

Niemelä, E., & Ihme, T. (2001). Product line software engineering of embedded systems. *ACM SIGSOFT Software Engineering Notes*, *26*, 118–125.

Raatikainen, M., Savolainen, J., & Männistö, T. (2014). Architecture Management and Evaluation in Mature Products: Experiences from a Lightweight Approach. *Proceedings of the 10th International ACM Sigsoft Conference on Quality of Software Architectures* (pp. 73–82). New York, NY, USA: ACM. doi:10.1145/2602576.2602583

Reijonen, V., Koskinen, J., & Haikala, I. (2010). Experiences from Scenario-Based Architecture Evaluations with ATAM. In M. Ali Babar, & I. Gorton (Eds.), *Software Architecture, 4th European Conference on (ECSA 2010)* (Vol. 6285, pp. 214–229). Springer. doi:10.1007/978-3-642-15114-9_17

Sangiovanni-Vincentelli, A., & Martin, G. (2001). Platform-based design and software design methodology for embedded systems. *IEEE Design & Test of Computers*, *18*, 23–33.

Savolainen, J., Niu, N., Mikkonen, T., & Fogdal, T. (2013). Long-term product line sustainability with planned staged investments. *IEEE software*, *30*, 63–69.

Shadish, W. R., Thomas, C. D., & Thomas, C. D. (2002). *Experimental and quasi-experimental designs for generalized causal inference*. Houghton Mifflin Company.

Sobhy, D., Minku, L., Bahsoon, R., Chen, T., & Kazman, R. (2020). Run-time evaluation of architectures: A case study of diversification in IoT. *Journal of Systems and Software*, *159*.

Tuovinen, A.-P., Mäkinen, S., Leppänen, M., Sievi-Korte, O., Lahtinen, S., & Männistö, T. (2017). Unwasted DASE: Lean Architecture Evaluation. In M. Felderer, D. Méndez Fernández, B. Turhan, M. Kalinowski, F. Sarro, & D. Winkler (Ed.), *18th International Conference on Product-Focused Software Process Improvement (PROFES 2017)* (pp. 128–136). Cham: Springer International Publishing. doi:https://doi.org/10.1007/978-3-319-69926-4_10

Törngren, M., & Sellgren, U. (2018). Complexity Challenges in Development of Cyber-Physical Systems. In M. Lohstroh, P. Derler, & M. Sirjani (Eds.), *Principles of Modeling: Essays Dedicated to Edward A. Lee on the Occasion of His 60th Birthday* (pp. 478–503). Cham: Springer International Publishing. doi:10.1007/978-3-319-95246-8_27

Van der Linden, F. J., Schmid, K., & Rommes, E. (2007). *Software product lines in action: the best industrial practice in product line engineering*. Springer Science & Business Media.

van Heesch, U., Eloranta, V.-P., Avgeriou, P., Koskimies, K., & Harrison, N. (2014). Decision-Centric Architecture Reviews. *Software, IEEE*, *31*, 69–76. doi:10.1109/MS.2013.22

Woods, E. (2012). Industrial Architectural Assessment Using TARA. *Journal of Systems and Software*, *85*, 2034–2047. doi:http://dx.doi.org/10.1016/j.jss.2012.04.055

Yu, X., & Xue, Y. (2016). Smart Grids: A Cyber–Physical Systems Perspective. *Proceedings of the IEEE*, *104*, 1058–1070. doi:10.1109/JPROC.2015.2503119

Mary Sánchez-Gordón and Ricardo Colomo-Palacios

2 Managing software development risk: Risks of introducing the role of agile coach – a multivocal literature review

2.1 Introduction

Agile software development is considered as one of the main avenues of research in current software engineering studies (Calefato & Ebert, 2019). In fact, agile methods have gradually gained popularity among both researchers and software practitioners until reaching complete dominance in the past 25 years of software engineering (Hoda et al., 2018). Agile methods emerged as a response to the "bureaucracy" of the traditional complex methods and the increasing change in the business environment revealed by the need of faster changing requirements and growing demand for efficient software development (Pavlič & Heričko, 2018). Traditional approaches could not deal with that change due to the fact that they assume that it is possible to anticipate a complete set of the requirements in an early phase of the project lifecycle (Abbas et al., 2008). Agile approaches are nowadays embraced widely as an answer to the failure of traditional plan-driven waterfall-based approach as well (Gupta et al., 2019).

In contrast, agile methods offered lightweight processes with a central focus on people and interactions, while they retain the rigor of engineering processes and best practices throughout the software development lifecycle process (Hoda et al., 2018). As a consequence of its popularity and effectiveness, agile methods are widely accepted in deployment of methods such as SCRUM, extreme programming (XP) and lean software development (Alahyari et al., 2019). Moreover, the success of agile methods for small, co-located teams has inspired companies to increasingly apply agile practices to large-scale efforts (Uludag et al., 2018). However, adopting agile practices, related to knowledge and experience is complex and requires lots of effort from the companies and teams along with cultural adaptation: it deals with egos and resistance to change and demands upper management sponsorship (Campanelli & Parreiras, 2015; Pavlič & Heričko, 2018). In sum, agile approaches are, like any other software method, intensive in human capital and need to be tackled taking into account the interests of all stakeholders (Colomo-Palacios et al., 2012). What is more important, given that software engineering and agile approaches are becoming more and more social (Mens et al., 2019), it is crucial to focus on social aspects of such teams.

In order to help companies to adopt agile methods smoothly, a new role, Agile Coach is gaining popularity among software practitioners (O'Connor & Duchonova, 2014). In fact, the vast majority (83%) of 1319 respondents of the survey (VersionOne,

https://doi.org/10.1515/9783110652321-002

2019) said their organization were below a high level of competency with agile practices —agile maturity—, further revealing opportunities for improvement through supporting training and coaching. Although the professionalization of that role seems well-known and consolidated by the conferences and certification programs to standardize the qualification process, the companies decide whether to use an Agile Coach for agile adoption or not and if so, what type of Agile Coach to use. Furthermore, in practice, different coaches have different styles and different focuses depending on the team needs and their own preferences (Bäcklander, 2019).

The authors are aware of the importance of the role of Agile Coach and aim to carry out a multivocal literature review (MLR) devoted to identify the risks of introducing such a role by investigating both research and professional literature including not only the negative consequences but also the positive aspects that could lead to potential beneficial opportunities. Therefore, this chapter aims to benefit the readers (both researchers and practitioners) by providing the most comprehensive and balanced view of the topic.

2.2 Understanding the relationship between risk and agile coach role

Risk is an uncertain event or condition that, if it occurs, has a positive or a negative effect on plans and goals of any software project (Project Management Institute, 2013). However, regardless of the outcome, risk management is a process that involves identifying risk, assessing and prioritizing risk, as well as monitoring and controlling risk. Therefore, risk is a necessary evil in the software processes, even those that are claimed to inherently reduce risk, such as in agile approaches (Cockburn & Highsmith, 2001). In fact, recent failures of projects that adopted agile software development and the reported challenges associated with it have drawn attention to its possible risks and the importance of identifying, assessing, and mitigating them (Elbanna & Sarker, 2016; Gold & Vassell, 2015). Moreover, due to the fact that agile approaches depend a lot on people involved in the projects and their motivation in applying agile practices, most issues encountered are related to the people and the practices involved (Parizi et al., 2014). This echoes one of the values in (*Manifesto for Agile Software Development*, 2001), i.e. "individuals and interactions over processes and tools". It implies that not having the right people doing the right process will be a source of risk. According to (Gold & Vassell, 2015), previous works stated that risk management is important but it is frequently overlooked in many projects, in particular, risks inherent in Scrum projects are categorized as people, organization and process.

One way on how organizations can reduce the risk when adopting agile methods is to use an Agile Coach (O'Connor & Duchonova, 2014). However, Agile Coach is an overloaded term. According to Lyssa Adkins (2010), an agile coach is an experienced user and teacher of agile methodologies, who can take on many roles, such as teacher, facilitator, coach-mentor, conflict navigator, collaboration conductor, problem solver, and so on, to help teams adopt and improve their use of agile methodologies. In this sense, Agile Coaches are meant to guide people on their path towards better expertise through emphasizing best software engineering practices (Rodríguez et al., 2016). Agile Coaches perform as agents of change and rely upon teamwork related skills as well as other social skills (Vikberg et al., 2013). It has been applied to advanced scrum masters, trainers, and leaders who are not sure where they fit in an agile organization (Gene Gendel & Erin Perry, 2015). However, Agile Coach is not a role mentioned in Scrum, Kanban, XP or any other agile framework or practice. The role and its importance have grown organically as organizations have realized the benefits of agility and appetite for long-lasting change has increased (VersionOne, 2018, 2019).

Agile coaching can be a role and career that requires a lot of skills, and may not be a natural role for everyone (White, 2018; Wick, 2018). With learning, practice, and awareness of oneself and others, many can learn and grow careers in coaching. This role has evolved naturally to provide coaching and mentoring to agile teams, but it is relatively new and little researched (O'Connor & Duchonova, 2014). However, such a role introduces risks which may not be fully understood and hence will not be properly mitigated. It raises the question: *Which are the risks related to the introduction of the role of Agile Coach?*

2.3 Research method

An MLR was conducted in order to identify all accessible literature on the Agile Coach role. An MLR is a systematic study of academic literature and grey literature which are constantly produced by SE practitioners outside of academic forums (Garousi & Mäntylä, 2016). The last one includes but is not limited to: blogs, post, white papers and articles. As far as the authors know, this is the first MLR on this combined topic although it is not the first secondary study for other roles, e.g. a systematic literature review about the Scrum Master's Role (Noll et al., 2017).

Figure 2.1 shows an overview of the search process that is based on a study protocol. In this MLR, the first author carried out the study selection process and the second author reviewed the process, verified the outcomes and supported the resolution of doubts. The search string was purposely broadened to identify factors related to the role of Agile Coach that could be a source of potential risks. This MLR was performed by June-July 2019. First, the search was performed on six database search engines using as search strings, "Agile coach" OR "Agile coaches". For academic literature, four full-text databases were estimated as enough, namely Elsevier ScienceDirect, SpringerLink, ACM digital library and IEEE Xplore. For the grey literature, InfoQ and

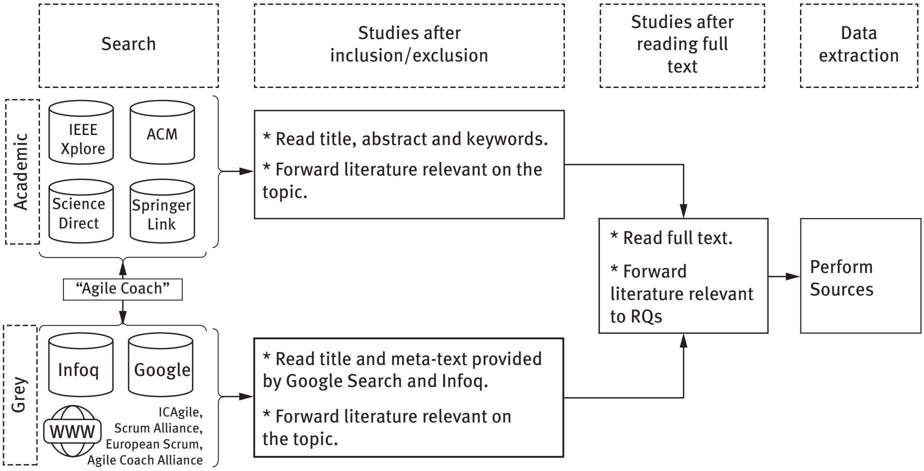

Search
Studies after inclusion/exclusion
Studies after reading full text
Data extraction
Academic
IEEE Xplore
ACM
Science Direct
Springer Link
"Agile Coach"
Grey
Infoq
Google
WWW
ICAgile, Scrum Alliance, European Scrum, Agile Coach Alliance
* Read title, abstract and keywords.
* Forward literature relevant on the topic.
* Read title and meta-text provided by Google Search and Infoq.
* Forward literature relevant on the topic.
* Read full text.
* Forward literature relevant to RQs
Perform Sources

Figure 2.1: An overview of the search process.

Google were selected due to this topic already stems from software industry. InfoQ provides software engineers with the opportunity to share experiences gained using innovator and early adopter stage techniques and technologies with the wider industry. However, InfoQ carefully peer review everything they publish. Moreover, four specialized websites related to Professional Certifications are included: ICAgile, Scrum Alliance, European Scrum, and Agile Coach Alliance. When we were using the Google's regular search engine, we use the stopping criteria called "effort bounded", i.e., only include the top N search engine hits based on the search engine page rank algorithm as recommend the guidelines proposed by (Garousi & Mäntylä, 2016) to restrict the search space in MLRs. Table 2.1 shows the number of search results per database. As one can see, we found 547 publications in the initial search.

Table 2.1: Summary of search results for primary study.

Studies	IEEE	ACM	ScienceDirect	SpringerLink	Google	InfoQ	Total
Studies retrieved	15	10	44	161	200	100	547
Studies after criteria	5	4	5	17	11	20	62
Studies after reading full text	2	2	1	6	7	4	22

We excluded articles based on title and meta-text provided by Google Search, while, we reviewed the titles, abstracts and keywords in the remaining databases.

The application of inclusion and exclusion criteria was conducted by the first author. Below are the inclusion criteria applied:

- Studies are about the Agile Coach role.
- Studies are in the field of software engineering and factors that could be related to potential risks in agile software development.
- Studies were published online in the period 2010 to 2019 (this study was conducted during July-August 2019).

When a study was excluded, the following criteria were applied:

- Studies not presented in English.
- Studies not accessible in full-text.
- Studies that are duplicates of other studies.

When a publication was clearly out of the inclusion criteria, it was not included in the following phases of the selection process. When a publication accomplished with the inclusion criteria, the publication was included in the next phase of the process. When in doubt, we were inclusive of taking the publication to full-text reading. After that, publications were thoroughly analyzed by reading the full text. In this way, we attempted to ensure that the publication certainly contains relevant information for this study. By full-text reading, it became obvious that further publications should be removed because they did not accomplish the inclusion and exclusion criteria. In this case, the primary studies are the union of the scientific and grey primary studies. At the end of the process, the list of items was formed by 22 sources (see Table 2.1). In this chapter, the sources are referred in the form of [S01], . . ., [S22] and these labels are the same as in Table A.1 in the Appendix.

In what follows, the results are presented based on the major categories related with the Agile Coach role: i) Coaching competencies, ii) Professional certifications, iii) Experience, iv) Style of Agile Coach, and iv) Focus and alignment, v) Internal/ External Coach, vi) Objectives of the coaches, vii) Target groups, viii) Value.

2.4 Results

Given that risks related to Agile Coach role are not explicitly mentioned in the literature, this section presents factors related to the role of Agile Coach and emerging human-related risks identified from the MLR. Then, in section 2.5, authors discuss the human-related risks of introducing the role of Agile Coach based on the identified factors. Table 2.2 shows two main categories, the first one is related to Agile Coach role itself and the second is related to business. In what follows, the main findings of each factor are presented.

Table 2.2: Summary of human-related risks.

Factor	Sources	Type	Human-related risks
Role			
Coaching competencies	[S01], [S02], [S06], [S07], [S09], [S11], [S14], [S22]	Negative	Lack of competencies Technical mistakes
		Positive	Solid skills set Wider-range ability to influence
Professional certifications	[S16–S21]	Negative	Not prove competence Wrong expectations
		Positive	Credibility Pertinent level of skills and Leaderfulness
Experience	[S06], [S07], [S08], [S10], [S11], [S22]	Negative	Lack of experience Technical mistakes Communication risks
		Positive	High value Lifelong learning
Style	[S11], [S13]	Negative	Short-lived impact
		Positive	Work engagement Healthy coaching (long-lasting change)
Business			
Focus and alignment	[S13], [S15]	Negative	Inappropriate focus and/or alignment Organizational dysfunctions Short-lived impact
		Positive	Share knowledge and experience Transformational success
Internal / External Coach	[S03], [S07], [S09], [S11], [S13]	Negative	Cost Wrong expectations
		Positive	Right balance Longer-term commitment to Agile
Objectives of the coaches	[S02], [S03], [S05], [S07], [S11], [S12]	Negative	Human factors Wrong expectations
		Positive	Work engagement Longer-term Commitment to Agile

Table 2.2 (continued)

Factor	Sources	Type	Human-related risks
Target Groups	[S02], [S03], [S04], [S06], [S07], [S08], [S11], [S13]	Negative	Human conflicts Wrong expectations Lack of recognition
		Positive	Work engagement Longer-term Commitment to Agile
Value	[S11], [S22]	Negative	Cost
		Positive	Reduction of organizational impediments Sustainable agile capability

2.4.1 Coaching competencies

In 2011, the agile industry needed a common definition and/or a learning path to grow in Agile Coaching [S14]. According to [S06], the best coach is one who is a "talker" and a "doer". Although coaching is a critical skill for Agile Coaches, ICAgile organized a panel in 2018 to discuss some key skills and attributes of Agile Coaches [S14]: (i) in control of themselves, (ii) devoted to the outcome, and hold the team and organization to that outcome, (iii) able to intervene: hard facilitation, give advice, raise awareness, be in service of a bigger outcome, empathy to meet a team where they are at, and patience. While coaching is a critical skill for Agile Coaches, they must also improve their skills in teaching and mentoring, as well as extend their competence in facilitating to include large, multi-team situations [S14], [S22].

Coaching Competencies are proficiencies that Agile Coaches are expected to demonstrate in their interactions with individuals and their organizations. According to [S14]:

- To be the change agent and work as a catalyst for the Coachee (client) organization. To be able to reach engagement with the whole organizational system and all the leaders that guide it. To have the ability to stimulate organizational reflection, learning and growth as well as connecting interdependencies.
- To be able to serve as an organizational mirror by accessing and surfacing the underlying system problems. To expose challenging symptoms and perform root cause analysis and be able to look below the surface.
- To be able to facilitate implementation, alignment and client agile adoption. During controversial moments and alignment-building activities to enable engagement to stakeholders. To keep non-biased views and facilitate a collaborative decision making.

- To keep a balance between the Coach's agile expertise with the Coachee's (client's) goals and intentions. To understand and keep the nature of the client-consulting relationship whether as consultant or employee. To guide the process of client self-discovery, to have the ability to lead by example and ask powerful questions.
- Educate and guide the Coachee's (client's) agile learning through the process of application and discovery. To be able to focus on stabilizing principles and varying practices that are aligned to the level of maturity of the Coachee's (client's) with an effective application of agility.

In other words, agile coaching is a subfield of coaching whose focus is to "*help teams or individuals adopt and improve agile methods and practice*" and "*rethink and change the way they go about development*" [S11]. Thus, an Agile Coach has to extract implicit and explicit knowledge in order to propose and introduce a novel (or appropriately adapted) leaner and agile development method [S09]. The attributes of *"an Agile Coach include experience in deploying Agile, in organizational change, in playing agile roles on a team, and in working with the business benefits of Agile"* [S07] . During the adoption of a new method, there are also many knowledge management-related issues linked with educating employees, measuring their confidence in new methods and based on this to fit new methods to the target organization [S09]. Furthermore, an Agile coach has the ability to challenge teams' perceptions of their capabilities and allow them to find their self-organizing behavior [S2]. Other roles facilitating self-organizing agile teams that could be played by Agile Coach are Champion, promoter and terminator [S01] (see Table 2.3).

Table 2.3: Roles Facilitating Self-Organizing Agile Teams [S01].

Role	Definition
Mentor	Provides initial guidance, understanding, confidence of Agile methods, and encourages continued adherence to Agile practices.
Champion	Gains the support of senior management to establish pilot teams and to propagate more self-organizing teams across the organization.
Promoter	Secures customer collaboration and involvement to support efficient functioning of Agile teams.
Terminator	Removes team members that hamper team productivity due to their inability to fit into the Agile way of working.

There are also coaching specialties which are based on a core skillset, expertise and knowledge that coaches possess [S13]. For instance [S22]: Technical/Product Research, Technical/Quality Practices, Development Operations, Development/Process Tools, Organizational Structures/Culture, Organizational Leadership, Scaling Agile/Enterprise Agility, Distributed Agile, Multi-Team Dynamics, Lean Principles and Lean Startup. However, sustainable organizational change implies Technical Mastery, Business Mastery and Transformation Mastery.

Based on the above mentioned, **potential risks** are *"Lack of competencies"* and *"Technical mistakes" (**negative**)* as well as *"Solid skill set"* and *"wider-range ability to influence" (**positive**)*

2.4.2 Professional certifications

The Agile coaching profession is relatively well-known and consolidated among professionals. There are specialized consulting companies and bodies of knowledge, specialized in Agile Coaching. Furthermore, conferences are being held on Agile Coaching where experienced practitioners share their ideas, and some of them even started to offer Agile Coaching courses in order to teach others how to become a qualified Agile Coach. According to Adkins [23], the Agile Coach role is designed to take care of performance and quality in an organization while they are part in the systemic reduction of organizational impediments and organizations build a sustainable agile capability. Table A.2 in the Appendix shows well known certifications on this field. Since May 2018, ICAgile [S16] has accredited more than 70 courses for the agile coaching track, and more than 11,700 certifications have been awarded to approximately 9,110 individuals by these courses. Scrum Alliance [S19] is another well-known organization that offers two professional certifications to become a Certified Agile Coach, Certified Team Coach (CTC) and Certified Enterprise Coach (CEC). Apart from them, European Scrum [S20] and Agile Coach Alliance [S21] provide two more professional certifications: Expert Agile Coach and Agile Coach Certification, respectively.

According to [S18], the learning path for agile coaching has established a common vocabulary, created an introduction to deeper learning paths, normalized the importance of professional coaching and professional facilitation. However, although the professional certifications offer continuing education certifications, some of them do not prove competence, and competence is what is needed [S18], [S22]. To address this gap, ICAgile creates practice and competence building programs that take the learner to the ICAgile Expert level but only 60 individuals have achieved that level up to 2018. Moreover, the ICE in Enterprise Agile Coaching (ICE-EC) [S17] will launch in June 2020.

Based on the above mentioned, **potential risks** are *"Not prove competence"* and *"Wrong expectations" (**negative**)* as well as *"Credibility"* and *"Pertinent level of skills and Leaderfulness" (**positive**)*

2.4.3 Experience

Apart from knowledge and a solid set of skills, Agile coach's experiences that support the mindset shift into the desired state of being are needed [S22]. During a typical coaching session, the Agile Coach explores the team dynamics non-intrusively and shares their Agile experiences and ideas to the team members with an intent to encourage him or her to learn and adapt based on the demands of the situation [S08]. Moreover, an experienced Agile Coach should have the experience of providing teachable moments without unnecessarily interrupting the flow of an event [S07].

Although, the professionalization of Agile Coach has emerged and the agile community keeps growing, there is evidence from numerous sources indicating a lack of qualified and well-experienced coaches to support the demand [S11]. In this sense, coaches could need mentors to observe them in action and provide targeted teaching, mentoring and professional coaching in-the-moment in order to get the opportunity of resetting core beliefs or boosting the Coach's learning [S22]. It means the improvement of coaching practice. On the other hand, the importance of experience for the Agile Coach role is empathized in a study carried out by [S10]. This study presents results acquired from student coaches (N=46) in a realistic setting at an early stage of their studies.

In the same line, [S06] states that many Agile Coaches consider following the "*Shu-Ha-Ri*" concept of learning. *Shu* can essentially be translated as following, *Ha* means to adopt the techniques and *Ri* translates to leave/transcend. In other words, first practice by textbooks, then, you are in a position to adapt and transcend. Moreover, it is recommended for executives to bring in Agile Coaches to help teams not only move to Agile but also help their staff shift to the behaviors that exemplify an Agile mindset [S07]. Finally, the research results in [S11] reveal that Certified Agile Coaches are more credible although they do not necessarily provide higher value than non-certified coaches since experience matters.

Based on the above mentioned, **potential risks** are *"Lack of experience"*, *"Technical mistakes"* and *"Communication risks" (**negative**)* as well as *"Lifelong learning"* and *"High value" (**positive**)*

2.4.4 Coaching style

A critical skill for healthy coaching is identifying the right situation, and properly transitioning between a directive style coaching to a supportive and reflective coaching [S13]. Table 2.4 shows some of the typical conditions under which a coach selects one style over another.

It can be appealing, especially for naturally directive leaders, to fall too often into the directive route [S13]. Half of the respondents in [S11] believe that supportive and reflective coaches provide higher value than directive coaches since they teach

Table 2.4: Coaching Style [S13].

Directive	Supportive and reflective
The coachee has low experience and knowledge for contextual learning. The coach has wide expertise in the subject matter.	The coachee levels of aptitudes, skillsets and expertise are really high regardless the level of expertise and skill set of the Coach.
The motivation and morale of the coachee are low.	The motivation and morale of the coachee are high.
The coachee is expected to follow the example of the coach. That is the way of leading of the coach.	The coach makes a reflection according to the coachee thoughts and makes the coachee to come to his/her own conclusions.

coaches how to be self-coaching. However, directive coach seems to be not only the easier form of coaching but also the less likely to leave a lasting impact on the coachee so that this means a purely directive route could ensure compliance, not engagement [S13]. Moreover, a study [S03] involving 46 agile practitioners reported that supportive coach help them to change themselves with less effort and time.

Based on the above mentioned, **potential risks** are *"Short-lived impact" (**negative**)* as well as *"Work engagement"* and *"Healthy coaching (long-lasting change)" (**positive**)*

2.4.5 Focus and alignment of coaching

In complex organizational settings, there could be two different coaching aspects, namely, focus and alignment. Focus areas include *enterprise (organizational) level* and *local (team) level*. According to [S13], team coaches are mainly focused on tools, frameworks and dynamics of multiple teams, with less emphasis on organizational transformation. On the contrary, enterprise coaches are more focused on organizational dynamics and more abstract elements of transformation with emphasis on senior leadership, upper management, organizational policies, and multiple organizational domains. Table 2.5 shows the level of coaching.

Regarding alignment, coaching could be placed: centrally or de-centrally. According to [S15], with agile coaching, being a centralized organizational function that owns transformation, one of its main deliverables becomes setting of standards and measures of success, by which the rest of an organization is measured. Although this could lead to organizational silos, it could make sense in small organizations. On the contrary, decentralized coaching is deep and narrow but takes time to cause significant and sustainable organizational changes. The coaches are locally aligned with teams, their customers and products, and immediately involved senior leadership.

Table 2.5: Focus of Coaching.

Organizational level	Local level
Educating senior leadership on inter-connection of various organizational elements within one Organizational Ecosystem.	
To try to influence and educate senior leadership and executives to become more agile across an entire organization.	To keep a balance between team growth and local optimization.
To asses team(s) and organization (s) through agile principles and practices to increase effectiveness.	To assist on the establishment of day-to-day interactions, ceremonies and agile roles.
To advise and give consultancy with organizations and leadership on different agile practices such as Lean, Scrum, Kanban and XP.	To advise teams with the adoption of basic agile frameworks (e.g. Kanban, Scrum, and XP).
To facilitate team (s) and groups to be able to achieve a higher quality on different aspects such as collaboration and to get a culture of continual learning and knowledge dissemination.	To enhance the improvement of the dynamics and maturity supporting single or multiple teams.
To develop a team, leadership and organizational agility by self-discovery and growth.	To give coaching to individual team members, scrum masters, and product owners.
To advise teams about the careful adoption of scaled agile frameworks as mechanism for organizational descaling (e.g. LeSS, SAFe, RAD)	To focus on test quality, coding standards and engineering practices.
To analyze systematic patterns, including norms, standards and behaviors.	To advise on different aspects such as metrics, living documentation agile requirements and communication
To enable an agile (Kaizen) culture and challenge the organizational and leadership status quo.	To defy the inappropriate behavioral problems that have been locally manifested (in isolation).

Source: Gendel and Perry 2015.

Based on the above mentioned, **potential risks** are *“Inappropriate focus and/or alignment”*, *“Organizational dysfunctions”* and *“Short-lived impact” (**negative**)* as well as *“Share knowledge and experience”* and *“Transformational success” (**positive**)*

2.4.6 Internal/External coaching

Another factor is based on whether an organization can employ its own Agile Coaches in order to achieve agile transformation or simply hire external coworkers, i.e. in-house/internal coaches and coach-consultants/external. According to [S11] value in Agile Coaching can be determined as the difference between the costs of hiring/using an Agile Coach and the benefits brought by the Agile Coach to the company in question. The value (benefits minus costs) provided by the coach can be also categorized as financial and non-financial.

On one hand, internal Agile Coaches could contribute with deeper knowledge of their own organizational structure and culture as well as organization's business and processes [S13], [S11]. On the other hand, a non-biased view of the organization and diverse experience can be provided by external Agile Coaches while they bring to the table experience of other organizations and industries, holistic and uninhibited views [S13], [S11]. However, an Agile coach should be hired before starting the Agile transition to manage the preparation phase [S03].

Depending on whether the Agile Coach stays with the team full-time and thus is coaching only one team at a time, or whether the coach stays with the team part-time and thus can coach multiple teams at once, [S11] also classify the coach either as a fulltime Agile Coach or a part-time Agile Coach. In this sense, participants in [S03] also pointed out the importance of having an on-site full-time coach during Agile transition. The study [S03] also recommended hiring a full-time on-site coach rather than an external coach as they can help teams in the right time when they are faced by various challenges.

In any case, an Agile Coach is confronted with the need to analyze the current state of processes, current level of employees' knowledge on processes, current satisfaction and obstacles, as well as the advantages offered by current development practices [S09]. The Agile Coach also should identify risks [S03] and understand both, the short-term and long-term pitfalls that can occur when a hierarchical organization is moving to Agile [S07]. Therefore, they can help mitigate the challenges ahead of time. They also should consider the ground conditions and make the winning strategy [S03].

Finally, every (internal/external) coach needs to define and discuss with coachee (individual or organization client) rules of engaging and disengaging [S13]. In other words, it must have a strategy in place for discontinuation of a coaching relationship. In case of internal coaches, they may fall back into their previous roles.

Based on the above mentioned, **potential risks** are *"Cost"* and *"Wrong expectations" (**negative**)* as well as *"Right balance"* and *"Longer-term commitment to Agile" (**positive**)*

2.4.7 Objectives of the coaches

Depending on the coach's mission, i.e. whether his/her objective is to manage the agile adoption of a team that is transferring to agile or to improve the performance of a team that has already started using agile and is struggling with it, one can identify adoption coaches and after-adoption coaches [S11]. Sometimes fully agile adoption is not possible. In this case, software companies need to do some activities in non-Agile ways [S03]. When teams are already applying agile practices, they seek coaching, they want to boost their performance in agile software development [S11], [S02]. Agile Coaches can address issues and challenges raised by teams that focus on adoption, culture, effect to customer value, work flow, and quality of the product being built [S07]. In addition, the focus on after-adoption requires to assess the relative effects of sustained agile use (more recent use) in comparison with their initial use (less recent use), this potentially biasing effect is minimized [S02].

Organizations also need to consider Agile coach's points of view in different stages of Agile transition such as hiring competent members, team set up, preparing an action plan, creating progress criteria, defining business goals, and so on [S03]. The agile adoption process is more difficult within large organizations as they usually have many established processes that conform numerous standards and involve globally distributed teams [S11].

The results study carried out by [S05], which involve 49 agile experts, identified different aspects of human-related challenges throughout Agile transition process. Table 2.6 shows the impediments to agile transition and the people's perceptions about agile transition reported in this study. In this line, a mapping of the market for Agile Coaches highlighted that client perceptions of their problem differ widely from the coaching perception [S12].

Table 2.6: Human-related challenges and issues [S12].

High level	Low level
Impediments to Agile transition	**Lack of knowledge** about Agile, its principles, and its values. This leads to other challenges such as low collaboration, wrong mindset, and unrealistic expectations.
	Cultural issues make the transition harder than expected. This challenge sometimes arises from organizational culture rather than people's culture.
	Resistance to change is related to the involved people's concerns about their jobs and afraid of losing their roles in development process.
	Wrong mindset mainly arises from perceptions and beliefs about the development process, required roles and responsibilities, and their fear of change.
	Lack of effective collaboration results in difficulty in setting up a cross-functional team.

Table 2.6 (continued)

High level	Low level
Perceptions about the change process	**Worried about the transition** involves about our future development approach.
	Enthusiastic but misguided. Lack of knowledge about Agile can make enthusiastic people misguided.
	Lack of belief in the change or lack of need for employing Agile methods represents a real risk for long-term success in the transition.
	Indifferent to the change means lack of enough motivation to start the change process.
	Unrealistic expectations or wrong expectations may lead to other challenges. Effective training, and full-time onsite coaching were reported as the most effective solutions that are useful to overcome this challenge.

Based on the above mentioned, **potential risks** are *"Human factors"* and *"Wrong expectations" (**negative**)* as well as *"Work engagement"* and *"Longer-term Commitment to Agile" (**positive**)*

2.4.8 Target groups

In agile settings, *group coaching* is typically focused on entire feature teams or Product Owner teams, where people are expected to have shared beliefs, norms and goals [S13]. Group coaching addresses team dynamics, roles, day-to-day interactions, metrics, reporting and so on.

Although, the Agile Coach can set up a dedicated session for group coaching or leverage existing group ceremonies (e.g. retrospective), group Coaching is often more structured and requires expert authority to be successful [S13]. Furthermore, an Agile Coach can provide consistency when multiple teams are adopting Agile at the same time while helping them both mechanically to do Agile and behaviorally be Agile [S07]. In other words, Agile Coach reinforces and ensures that the team continues both the expected practices and behaviors [S07]. Moreover, encouraging people to the changes, especially when facing problems, is also another duty of the Agile coach that facilitates the Agile transition [S03]. Supporting this, a study [S02] that involved 114 agile practitioners concluded that the role of an agile coach is a key factor in creating and sustaining well-balanced high performance software development teams by influencing agile usage. Moreover, an experiment [S04] that involved 10 teams of students revealed that coached teams outperformed non-coached teams since the Agile Coach emphasized the concept of "done criteria" and there was around 22% more coverage of software engineering practices.

On the contrary, *individual coaching* is one-on-one. Such coaching sessions are typically conducted in privacy [S13]. In this case, the Coach works with a single person on a personal level. Agile Coaches reach out to each individual member of the team, understand their expectations, beliefs and aspirations and help them to embrace the principles and practices of Agile [S08]. Therefore, individual sessions may address personal adaptation, happiness, job satisfaction, problems with management or subordinates, embracing roles and seeing career growth opportunities, dealing with personal challenges, reservations or fears [S13]. Despite that the coach motivates and influences the team, the coach wants the team to feel the ownership of the change to Agile [S07].

Individual coaching is often used to engage and support a Scrum Master or Product Owner as an individual [S13]. Here, the training is important but it is limited. Thus, training classes to get people oriented with new terminology and new concepts is a good approach, but how to be a good product owner goes beyond training classes since they exclude day-to-day competence [S06]. In consequence, the new product owner must be paired with a knowledgeable expert or Agile coach in order to do the work together [S06]. It means hands-on coaching in the related context of real-life projects is needed.

Both individual and group sessions can be pre-scheduled or situational/opportunistic, i.e. at moments, when Agile Coach finds ad-hoc appropriate moments to administer coaching [S13]. By providing the guidance of an expert the teams or individuals receive valuable information that speeds up the learning process and reduce the error rate [S11].

Based on the above mentioned, **potential risks** are *"Human conflicts"*, *"Wrong expectations"* and *"Lack of recognition" (**negative**)* as well as *"Work engagement"* and *"Longer-term Commitment to Agile" (**positive**)*

2.4.9 Value of an agile coach

The research results collected by [S11] from 8 Agile Coaches and 10 companies – 5 companies that used an Agile Coach and 5 companies that adopted agile without the help of an Agile Coach – can be summarized as shown in Table 2.7.

Finally, although, all companies claim the adoption was a success, a drawback of companies that adopted agile without the help of an Agile Coach was a larger learning curve [S11]. In support of that, an empirical investigation revealed that agile usage measured as intensity and extent of use of agile methods significantly impact agile effectiveness. The value of Agile Coaches is that they take part in the systemic reduction of organizational impediments so that organizations can build organizational agile capability based on their agile coaching capability [S22].

Table 2.7: Value of an Agile coach [S11].

Value	Considerations
The respondents believe that the benefits obtained through the Agile Coach exceeded the financial costs.	Half of the respondents think that Agile Coaches are perceived as expensive consultants.
All the respondents that during their experience of agile adoption used an Agile Coach would recommend it to other companies.	A certified Agile Coach could be more credible but not necessarily provides more value than another one that has no certification as experience matters.
An Agile Coach can provide financial and non-financial benefits through the adoption of agile methods.	There is difference in the value provided by different types of Agile Coaches.
The significant reduction of the risk of failure of agile adoption and the speed up of the adoption process is the value that Agile Coaches provide.	
Agile Coaches can assist with practicalities, such as how to do incremental design among many others.	
Benefits of using an Agile Coach are tailoring agile practices to company's needs, highlighting dysfunctions and waste in processes, sorting out industry related agile adoption challenges, and so on.	Different factors such as the company size, complexity of its processes, nature of the industry and company culture determine if a company should implement an Agile Coach.

Based on the above mentioned, **potential risks** are *"Cost" (**negative**)* as well as *"Reduction of organizational impediments"* and *"Sustainable agile capability"* *(**positive**)*

2.5 Discussion

Behavior in social – team, project and organizational – contexts is both an important enabler of risk management and a source category of potential project risks (Bannerman, 2015). As being software development is intensive in human capital, one of the main sources of risks in software development projects is the one connected to stakeholders and software people.

One way for reducing the risk of failure when adopting agile approaches is to use an Agile Coach (O'Connor & Duchonova, 2014). In fact, according to a survey carried out by (VersionOne, 2019), organizational culture issues remain the leading critical factor in the success of adopting and scaling agile. Thus, the three most significant challenges for agile adoption and scaling are reported as Organizational culture at odds with agile values (52%), General resistance to change (48%), and Inadequate management support and sponsorship (44%). Although other types of risks exist, our study is focused on these human-related challenges that are also source of human-related risks when introducing the role of Agile Coach.

From the nature of the Agile Coach, authors identify four factors related to the risks: Competencies, Professional certifications, Experience, and Style of the Agile Coach. **Professional certifications** and **Experience** are ways to build **Coaching competencies** to mitigate *Communication risks* and *Technical mistakes*. In the positive side of risks, *Solid skills set* and *Wider range ability to influence* based on *Credibility* and *Pertinent level of skills and Leaderfulness* along with *High value* and *Lifelong learning* should be part of the essence of an Agile coach. On the negative side of risk, *Lack of experience*, *Lack of competencies* along with *Wrong expectations* should be managed to avoid undermining the role of Agile Coach. Finally, the more supportive and reflective is the **Style of the Agile Coach** the more *Healthy coaching (long-lasting change)* and *Work engagement*. As opposite to directive style coaching results in *Short-lived impact*, i.e. negative risks.

From a Business perspective, authors identify five factors related to the risks: Focus and alignment, Internal/External Coach, Objectives of the coaches, Target groups, and Value of an Agile Coach. When **Focus and alignment** is pertinent, it could result in *Share knowledge and experience* and *Transformational success*. On the negative side of risk, *Organizational dysfunctions* and *Short-lived impact* could occur. **Internal/External Coach** implies *Right balance* and *Longer-term commitment to Agile in* the positive side however *Cost* and *Wrong expectations* should be mitigated. It is worthy to note that internal Agile Coaches is number one, being the most valuable in helping respondents scale agile practices in the surveys (VersionOne, 2018, 2019). On the other side, external Agile coaches were reported fourth one, behind "consistent practices and process across teams", and "implementation of a common tool across teams" in (VersionOne, 2018), and they did not appear in (VersionOne, 2019).

The risks related to **Objectives of the coaches** and **Target groups** could result in *Work engagement* and *Longer-term Commitment to Agile*. On the negative risks, both factors are related to *Human conflicts* and *Wrong expectations* while *Lack of recognition* may occur among Target groups. Finally, **Value of an Agile Coach** is hindering mainly by the *"Cost"* but it seems that positive risks – i.e, *Reduction of organizational impediments* and *Sustainable agile capability* – outweigh negative ones.

In consequence, Agile Coach role is also a source of risk factors that could impact software development since that role is closely connected to people and social interactions. Moreover, it is worth noting that coaching in Agile approaches is slightly different from coaching in traditional approaches (Parizi et al., 2014). Indeed, other Agile roles such as Scrum Masters could do "Agile Coaching" and they could become agents of change for their organizations using some of the skills that would be associated with Agile Coaching. However, an Agile coach brings also a wide spectrum of aspects from conflict management, facilitation, teaching, mentoring and professional coaching. This perspective focuses on identifying and developing personal skills and organizational capabilities that are important in successfully managing agile projects, i.e. Risk Management as a Capability as mentioned (Bannerman, 2015). According to (Bannerman, 2015), it recognizes that managing risk is about the ability to “do it”, not just “plan it” – particularly in dynamic, uncertain and complex environments – which fits in Agile Software Development.

2.6 Conclusion

The goal of any coaching initiative should be to bring Coachees to a healthy state where learning and self-improvement are happening organically (Gene Gendel & Erin Perry, 2015). It means that high performing organizations, high performing teams, and high performing people do not often happen organically but they are a return on investment. Coaching could keep them on their agile journey and help apply the mindset, process, and skills properly. In this context, an Agile Coach role offers an appealing option although there are companies that prefer to do their agile journey without an Agile Coach.

Our study identifies risks including not only the negative aspects but also the positive aspects that could lead to potential beneficial opportunities. In consequence, this chapter highlights issues for software practitioners and organizations to think about, as they decide whether to and how to, include an Agile Coach role. The main limitation is that our findings imply complex intangibles – such as individual and organizational culture – that are difficult to explore in research and measure in practice. Therefore, more research is needed.

From a software development perspective, authors, after the research conducted, underline that Agile Coaches could lead to a reduction of potential threats in software production. Being software development intensive in human capital, one of the main sources of risks in software development projects is the one connected to stakeholders and software people. Although pure agile approaches are not necessarily connected with the notion of project management (Leybourn & Hastie, 2019), agile is adopted both in project-oriented structures and in new #noprojects approaches. In both scenarios, Agile Coaches can reduce the likelihood and impact of several risks.

Opportunities exist to extend the identification of risks and broaden how risk management is viewed and studied in both practice and research. Future work will be twofold. Firstly, it is aimed to investigate the impact of Agile Coaches in the previously defined scenarios measuring differences among them in terms of efficiency and efficacy and develop a risk management plan. Secondly, it is aimed to shed some light into the role of Agile Coaches as cultural coaches in global software development arenas.

Appendix

Table A.1: List of primary studies included in the MLR.

ID Source	Authors	Year	Title	BdD
S01	Hoda, Rashina; Noble, James; Marshall, Stuart	2010	Organizing Self-organizing Teams	ACM
S02	Senapathi, Mali; Srinivasan, Ananth	2014	An Empirical Investigation of the Factors Affecting Agile Usage	ACM
S03	Parizi, R. M.; Gandomani, T. J.; Nafchi, M. Z.	2014	Hidden facilitators of agile transition: Agile coaches and agile champions	IEEE Xplore
S04	Rodríguez, G.; Soria, Á; Campo, M.	2016	Measuring the Impact of Agile Coaching on Students' Performance	IEEE Xplore
S05	Javdani Gandomani, Taghi; Ziaei Nafchi, Mina	2016	Agile transition and adoption human-related challenges and issues: A Grounded Theory approach	Science Direct
S06	Kulak, Daryl; Li, Hong	2017	Getting Coaching That Really Helps	Springer Link
S07	Moreira, Mario E.	2013	Being Agile: Your Roadmap to Successful Adoption of Agile	Springer Link
S08	Boral, Sumanta	2016	Domain VII: Continuous Improvement (Product, Process, People)	Springer Link
S09	Pavlič, Luka; Heričko, Marjan	2018	Agile Coaching: The Knowledge Management Perspective	Springer Link
S10	Vikberg, Thomas; Vihavainen, Arto; Luukkainen, Matti; Kurhila, Jaakko	2013	Early Start in Software Coaching	Springer Link
S11	O'Connor, Rory V.; Duchonova, Natalia	2014	Assessing the Value of an Agile Coach in Agile Method Adoption	Springer Link
S12	Bulloch, Elaine; Frumkin, Alexander; de la Maza, Michael	2018	Mapping the Market for Agile Coaches	Infoq
S13	Gene Gendel; Erin Perry	2015	Agile Coaching – Lessons from the Trenches	Infoq
S14	Wick, Angela	2018	Defining the Competencies of Agile Coaching	Infoq

Table A.1 (continued)

ID Source	Authors	Year	Title	BdD
S15	Gendel, Gene	2018	Centralized vs. Decentralized Coaching	Infoq
S16	ICAgile		ICAgile > Home	Google
S17	ICAgile		ICAgile > Learning Roadmap > Enterprise Agile Coaching > Agility in the Enterprise	Google
S18	ICAgile		ICAgile > Learning Roadmap > Agile Coaching > Agile Team Facilitation	Google
S19	Scrum Alliance		Scrum Alliance Certified Enterprise Coach (CEC) Certification	Google
S20	EuropeanScrum		Agile Coach Certification	Google
S21	Agile Coach Alliance		Home of Agile Coach	Google
S22	Adkins, Lisa		Developing an Internal Agile Coaching Capability	Google

Table A.2: Summary of Certifications.

Organization	Certification	Acronym
International Consortium for Agile [S16] https://icagile.com/	ICAgile Certified Professional – Agile Team Facilitation ICAgile Certified Professional – Agile Coaching ICAgile Certified Expert – Agile Coaching ICAgile Certified Professional – Agility in the Enterprise ICAgile Certified Professional – Coaching Agile Transitions ICAgile Certified Expert – Enterprise Coaching	ICP-ATF ICP-ACC ICE-AC ICP-ENT ICP-CAT ICE-EC
Scrum Alliance [S19] https://www.scrumalliance.org/	Certified Enterprise Coach Certified Team Coach	CEC CTC
European Scrum [S20] http://www.europeanscrum.org/	Expert Agile Coach	EAC
Agile Coach Alliance [S21] https://www.agilecoachalliance.org/	Agile Coach Certification Agile Coach Organization	

References

Abbas, N., Gravell, A. M., & Wills, G. B. (2008). Historical Roots of Agile Methods: Where Did "Agile Thinking" Come From? In P. Abrahamsson, R. Baskerville, K. Conboy, B. Fitzgerald, L. Morgan, & X. Wang (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (Vol. 9, pp. 94–103). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-68255-4_10

Alahyari, H., Gorschek, T., & Berntsson Svensson, R. (2019). An exploratory study of waste in software development organizations using agile or lean approaches: A multiple case study at 14 organizations. *Information and Software Technology*, *105*, 78–94. https://doi.org/10.1016/j.infsof.2018.08.006

Bäcklander, G. (2019). Doing complexity leadership theory: How agile coaches at Spotify practise enabling leadership. *Creativity and Innovation Management*, *28*(1), 42–60. https://doi.org/10.1111/caim.12303

Bannerman, P. L. (2015). A Reassessment of Risk Management in Software Projects. In C. Schwindt & J. Zimmermann (Eds.), *Handbook on Project Management and Scheduling Vol. 2* (pp. 1119–1134). Springer International Publishing. https://doi.org/10.1007/978-3-319-05915-0_20

Calefato, F., & Ebert, C. (2019). Agile Collaboration for Distributed Teams [Software Technology]. *IEEE Software*, *36*(1), 72–78. https://doi.org/10.1109/MS.2018.2874668

Campanelli, A. S., & Parreiras, F. S. (2015). Agile methods tailoring – A systematic literature review. *Journal of Systems and Software*, *110*, 85–100. https://doi.org/10.1016/j.jss.2015.08.035

Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *Computer*, *34*(11), 131–133. https://doi.org/10.1109/2.963450

Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., Misra, S., & García-Peñalvo, F. J. (2012). Analyzing human resource management practices within the GSD context. *Journal of Global Information Technology Management*, *15*(3), 30–54.

Elbanna, A., & Sarker, S. (2016). The Risks of Agile Software Development: Learning from Adopters. *IEEE Software*, *33*(5), 72–79. https://doi.org/10.1109/MS.2015.150

Garousi, V., & Mäntylä, M. V. (2016). A systematic literature review of literature reviews in software testing. *Information and Software Technology*, *80*, 195–216. https://doi.org/10.1016/j.infsof.2016.09.002

Gene Gendel, & Erin Perry. (2015). *Agile Coaching – Lessons from the Trenches*. InfoQ. https://www.infoq.com/articles/agile-coaching-lessons/

Gold, B., & Vassell, C. (2015). Using risk management to balance agile methods: A study of the Scrum process. *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, 49–54. https://doi.org/10.1109/KBEI.2015.7436020

Gupta, M., George, J. F., & Xia, W. (2019). Relationships between IT department culture and agile software development practices: An empirical investigation. *International Journal of Information Management*, *44*, 13–24. https://doi.org/10.1016/j.ijinfomgt.2018.09.006

Hoda, R., Salleh, N., & Grundy, J. (2018). The Rise and Evolution of Agile Software Development. *IEEE Software*, *35*(5), 58–63. https://doi.org/10.1109/MS.2018.290111318

Leybourn, E., & Hastie, S. (2019). *# noprojects: A Culture of Continuous Value*. Lulu.com.

Lyssa Adkins. (2010). *Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition*. Addison-Wesley Professional, Boston, MA.

Manifesto for Agile Software Development. (2001). http://agilemanifesto.org/

Mens, T., Cataldo, M., & Damian, D. (2019). The Social Developer: The Future of Software Development [Guest Editors' Introduction]. *IEEE Software*, *36*(1), 11–14. https://doi.org/10.1109/MS.2018.2874316

Noll, J., Razzak, M. A., Bass, J. M., & Beecham, S. (2017). A Study of the Scrum Master's Role. In M. Felderer, D. Méndez Fernández, B. Turhan, M. Kalinowski, F. Sarro, & D. Winkler (Eds.), *Product-Focused Software Process Improvement* (pp. 307–323). Springer International Publishing, Cham.

O'Connor, R. V., & Duchonova, N. (2014). Assessing the Value of an Agile Coach in Agile Method Adoption. In B. Barafort, R. V. O'Connor, A. Poth, & R. Messnarz (Eds.), *Systems, Software and Services Process Improvement* (Vol. 425, pp. 135–146). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-43896-1_12

Parizi, R. M., Gandomani, T. J., & Nafchi, M. Z. (2014). Hidden facilitators of agile transition: Agile coaches and agile champions. *2014 8th. Malaysian Software Engineering Conference (MySEC)*, 246–250. https://doi.org/10.1109/MySec.2014.6986022

Pavlič, L., & Heričko, M. (2018). Agile Coaching: The Knowledge Management Perspective. In L. Uden, B. Hadzima, & I.-H. Ting (Eds.), *Knowledge Management in Organizations* (pp. 60–70). Springer International Publishing, Cham.

Project Management Institute. (2013). *A Guide to the Project Management Body of Knowledge (PMBOK Guide)* (5th ed.). Project Management Institute, Newtown Square, PA, USA.

Rodríguez, G., Soria, Á., & Campo, M. (2016). Measuring the Impact of Agile Coaching on Students' Performance. *IEEE Transactions on Education*, *59*(3), 202–209. https://doi.org/10.1109/TE.2015.2506624

Uludag, Ö., Kleehaus, M., Caprano, C., & Matthes, F. (2018). Identifying and Structuring Challenges in Large-Scale Agile Development Based on a Structured Literature Review. *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*, 191–197. https://doi.org/10.1109/EDOC.2018.00032

VersionOne. (2018). *12th Annual State of Agile Report*. https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report

VersionOne. (2019). *13th Annual State of Agile Report*. https://explore.versionone.com/state-of-agile/13th-annual-state-of-agile-report

Vikberg, T., Vihavainen, A., Luukkainen, M., & Kurhila, J. (2013). Early Start in Software Coaching. In H. Baumeister & B. Weber (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (pp. 16–30). Springer Berlin Heidelberg.

White, S. K. (2018, August 8). *What is an agile coach? A valuable role for organizational change*. CIO. https://www.cio.com/article/3294700/agile-coach-role-defined.html

Wick, A. (2018). *Defining the Competencies of Agile Coaching*. InfoQ. https://www.infoq.com/news/2018/08/competencies-agile-coaching/

Manuel Mora, Fen Wang, Gloria Phillips-Wren and Jorge Marx Gómez

3 The role of DMSS analytics tools in software project risk management

3.1 Introduction

A Software Project Management process and a Risk Management activity are mandatory for all plan-driven software development methodologies, as well as for heavyweight and lightweight software process standards and models. Plan-driven methodologies like SOA-RUP (Péraire et al., 2007), that is a modern variant of RUP (Kruchten, 2004), heavyweight software process standards and models like ISO/IEC 12207 (ISO/IEC, 2008) and CMMI-DEV (CMMI Product Team, 2010), and lightweight standards like ISO/IEC 29110 (ISO/IEC, 2011) include Software Project Management process and a Risk Management activity. For the case of software development agile practices such as Scrum (Sutherland & Schwaber, 2017; Eclipse Foundation, 2019), Project Management is exercised at the most of them (Abrahamsson et al., 2010), but Risk Management activity is usually diluted as a set of practices for mitigating software project risks, rather than as an explicit and documented step-by-step activity.

According to SWEBOK (Bourque & Farley, 2014), Software Project Management is one of the three levels of Software Engineering Management processes that include Organizational and Infrastructure Management, as well as Measurement Program Management. Software Engineering Management refers to the application of management processes and activities to achieve the "Iron Triangle" (Agarwal & Rathod, 2006) (i.e. to achieve cost, schedule, and quality Software Project metrics). SWEBOK (Bourque & Farley, 2014) proposes the following sub-processes for the Software Project Management level: 1) Initiation and Scope Definition, 2) Software Project Planning, 3) Software Project Enactment, 4) Review and Evaluation, and 5) Closure. The Risk Management activity is included in the sub-process of Software Project Planning. In the Project Management Institute (PMI, 2013), there are five process categories: 1) Initiating Process Group, 2) Planning Process Group, 3) Executing Process Group, 3) Monitoring and Controlling Process Group, and 4) Closing Process Group. These five process groups are related to ten Knowledge Areas, where one of them is Risk Management, which is mainly linked to the 2) Planning Process Group.

Project Management can be defined as: "*The application of knowledge, skills, tools, and techniques to a program to meet the program requirements and to obtain benefits and control not available by managing projects individually*" (PMI, 2013; p. 553). A Project is "*a temporary endeavor undertaken to create a unique product, service, or result*" (PMI, 2013; p. 3) that has an agreed start and end. Project Software, thus, is any planned effort with an agreed start and end for achieving a software product.

https://doi.org/10.1515/9783110652321-003

A Risk Management activity concerns "*identifying, analyzing and monitoring risks as well as designing and implementing risk treatments for those risks*" (NIST, 2012). Software Project Risk Management, therefore, refers to managing potential risks occurring in Software Projects. A comprehensive notion of *Risk* is about an undesirable uncertain (estimated with a qualitative *Likelihood*) or probabilistic (estimated with a quantitative *Likelihood*) event, generated by *Threat Agents*, that would cause a negative *Impact* on a valuable *Asset*, that has a degree of *Vulnerability*, in the case of its occurrence without any applied *Countermeasure* (Aven, 2016). Consequently, the plan-driven software development methodologies, the heavyweight software process standards and models (i.e. ISO/IEC 12207, CMMI-DEV), and the lightweight software process standards (i.e. ISO/IEC 29110) have included explicitly a Software Project Management process and Risk Management activity.

In the case of plan-driven software development methodologies and heavyweight process standards and models, the Software Project Management process and Risk Management activity are well-structured and documented. In the case of agile practices or lightweight standards (i.e. Scrum and the ISO/IEC 29110), the Software Project Management process is essentially documented, but the Risk Management activity is implicitly considered in varied optional tasks.

Hence, the application of a Software Project Management process and a Risk Management activity should prevent or minimize failed Software Projects. However, software project delays, costs overruns, and low-quality software projects are still reported (El Emam & Koru, 2008; Cerpa & Verner, 2009; Marques et al., 2017). This negative evidence suggests difficulty correctly applying an appropriate Software Project Management process and a Risk Management activity (Boehm & DeMarco, 1997; Boehm & Turner, 2003).

In this chapter, we consider computer-based systems, referred to as Decision-Making Support Systems (DMSS) (Forgionne et al., 2009; Delen & Demirkan, 2013), as useful tools to improve the Software Project Management process and the Risk Management activity. DMSS tools have been extensively used in many business domains (Eom & Kim, 2006; Sharda et al., 2015; Eom, 2016), but they are rarely used for the Software Project Management process and the Risk Management activity. Thus, we illustrate 14 exemplary cases of DMSS tools utilization for the Software Project Management process and the Risk Management activity.

Our goal is to create awareness of the usefulness and value of DMSS tools to Software Project managers to support their Software Project Management process and Risk Management activity. For this aim, we provide a brief review of Software Project Risk Frameworks and Risk Management processes proposed in the primary literature in Section 3.2. In Section 3.3 we present a review of DMSS Analytics tools. Section 3.4 provides linkages between DMSS Analytics tools and Software Project Risk Management, as well as illustrative cases of DMSS Analytics tools applied to Software Project Risk Management. In Section 3.5 we discuss implications for Risk

Project Managers, and finally, we conclude with contributions, trends, and challenges of the utilization of DMSS Analytics tools for the Software Risk Project Management process.

3.2 Review of software project risk management process

Relevant literature on Software Risk Project Management has been published in the last 30 years (Schultz, 1984; Boehm, 1991; Higuera & Haimes, 1996; Wallace & Keil, 2004; Bannerman, 2008; Tavares et al., 2019a; Menezes et al., 2019; Masso et al., 2020).

Risk Management in Software Projects can be traced to a software military workshop on Software Cost Estimation (Schultz, 1984), where Boehm led an analysis group suggesting the inclusion of Risk Management "*to identify and eliminate sources of cost, schedule, or performance risk in the development and support of software*" Schultz, 1984; p. 16). From several industrial software projects, Boehm (1988) reported his Spiral Model of Software Development, where Risk Analysis is one of the four phases. This Spiral Model is reported as a risk-driven development approach in contrast to document-driven or code-driven. Basically, in the Spiral Model, Risk Analysis consists of identifying risks, proposing risk resolutions (i.e. *countermeasures*), and applying or evaluating the potential results of the proposed risk resolutions. Boehm et al. (1998), extended this model in the Win-Win Spiral Model, where they included negotiation tasks that pursue a win-win for every Software Project stakeholder. More recently, the relevance of a Risk Management activity was recognized for both primary software process frameworks (CMMI-DEV, ISO/IEC 12207, and ISO/IEC 29110) and primary software development methodologies (RUP-SOA and Scrum).

3.2.1 Software project risk checklists and taxonomies-frameworks

To guide the execution of Risk Management, several Software Project Risk Checklists (Boehm, 1991; Schmidt et al., 2001; Tiwana & Keil, 2004) and Taxonomies-Frameworks (Higuera & Haimes, 1996; Ropponen & Lyytinen, 2000; Wallace & Keil, 2004; Han & Huang, 2007) have been proposed in the literature (Bannerman, 2008). To our best knowledge, for the agile approach Software Project Risks Taxonomies-Frameworks have not been reported, but just agile risk recommendations (Tavarez et al., 2019a).

Software Project Risk Checklists contain a ranked list of software project risks usually derived from empirical experience from authors (Boehm, 1991) or derived from empirical surveys on Software Projects (Schmidt et al., 2001; Tiwana & Keil, 2004). Software Project Risk Taxonomies-Frameworks contain organized classifications/subclassifications of software project risks categories and they have also been derived from the empirical experience of authors (Higuera and Haimes, 1996) or empirical surveys on Software Projects (Ropponen & Lyytinen, 2000; Wallace & Keil, 2004; Han & Huang, 2007). Both conceptual tools (Software Project Risks Checklists and Taxonomies-Frameworks) aim to assist Software Project Managers in an adequate application of a Risk Management activity. Most of these conceptual tools were proposed by a plan-driven Software Engineering approach (Boehm, 1991; Han & Huang, 2007), and one for an agile Software Engineering approach is also available (Tavares et al., 2019a).

Tables 3.1 and 3.2 summarize, respectively, the main Software Project Risk Checklists and Taxonomies-Frameworks reported in the literature.

Table 3.1: Summary of Software Project Risk Checklists.

Risks	Boehm' List	Schmidt et al.'s List	Tiwana et al.'s List	Tavares et al.'s List
Lack of qualified software development team.	√	√	–	–
Optimistic shortened software project required schedule.	√	–	–	√
Optimistic shortened software project required budget.	√	–	–	√
Development of wrong software functionalities and properties.	√	√	–	√
Addition of unnecessary functionalities.	√	–	–	√
Volatile requirements.	√	√	√	√
Lack of qualified external human teams required for the software project.	√	√	–	–
Lack of high-quality external artifacts required for the software project.	√	–	–	–
Lack of adequate computational tools and machines.	√	–	–	–
Lack of top management support	–	√	–	–
Lack of user involvement	–	√	√	√
Introduction of new technology or a new unknown project or inherent project complexity.	–	√	√	√
Human conflicts.	–	√	–	√
Use of inappropriate development methodology.	–	–	√	–
Lack of formal Project Management practices.	–	–	√	–

Table 3.2: Integrative Summary of Software Project Risk Taxonomies-Frameworks.

Software Project Risk Frameworks	**Risk Categories**			
	Project Risks			**User Risks**
	Engineering Design and Build Risks	**Functional Project Management Risks**	**Economic Project Management Risks**	**Customer and Users Risks**
Higuera and Haimes's Framework	– Changing/non-stable system requirements risk – Non-feasible system requirements risk – Non-valid systems requirements risk – Wrong system performance risk	– No team familiarity/training with development process/tools risk – No adequate Project Management process risk – No experienced Project Manager risk	– Wrong schedule estimation risk – Wrong budget estimation risk – Excessive number of external contracts risk	–
Ropponen and Lyytinen's Framework	– Changing/non-stable system requirements risk – Non-feasible system requirements risk – Non-valid systems requirements risk – Wrong system performance risk	– No team familiarity/training with development process/tools risk – No adequate Project Management process risk – No experienced Project Manager risk	– Wrong schedule estimation risk – Wrong scheduling estimation risk – Excessive number of external contracts risk	–

(continued)

Table 3.2 (continued)

Software Project Risk Frameworks	Risk Categories			
	Project Risks			User Risks
	Engineering Design and Build Risks	Functional Project Management Risks	Economic Project Management Risks	Customer and Users Risks
Wallace and Keil's Framework	– Changing/non-stable system requirements risk – Non-feasible system requirements risk – Non-valid systems requirements risk – Conflicting system requirements risk	– No team familiarity/training with development process/tools risk – No adequate Project Management process risk – No experienced Project Manager risk – Use of a new/immature technology risk	– Wrong schedule estimation risk – Wrong budget estimation risk – Excessive number of external providers risk – Organizational changes risk	– Lack of top management support risk – Lack of user involvement risk – Lack of user commitment risk – Users conflicts risk
Han and Huang's Framework	– Changing/non-stable system requirements risk – Non-valid systems requirements risk	– No team familiarity/training with development process/tools risk – No adequate Project Management process risk – No experienced Project Manager risk – Use of a new/immature technology risk	– Wrong schedule estimation risk – Wrong budget estimation risk – Organizational changes risk	– Lack of user involvement risk – Lack of user commitment risk – Users conflicts risk

Interesting findings can be derived from Table 3.1 as follows: 1) the unique common risk refers to highly dynamic requirements and it leads to the emergence of agile practices; 2) the oldest Checklist (Boehm, 1991) did not consider human issues (i.e. top management support, and user involvement) except by technical skills required for the development team; 3) the newest Checklist (Tiwana & Keil, 2004) uniquely identifies the need of an adequate development methodology as well as the use of formal Project Management practices; and 4) the agile practices per se address usual budget, schedule, requirements, and user risks.

Similarly, interesting findings can be derived from Table 3.2 as follows: 1) the oldest Software Project Risks Taxonomies/Frameworks (Higuera & Haimes, 1996; Ropponen & Lyytinen, 2000) did not consider customer and user risks but the newest ones did (Wallace & Keil, 2004; Han & Huang, 2007); 2) common risks were reported in all Software Project Risks Taxonomies/Frameworks such as changing/non-stable system requirements risk, non-valid systems requirements risk, no team familiarity/training with development process/tools risk, no adequate Project Management process risk, no experienced Project Manager risk, wrong schedule estimation risk, and wrong budget estimation risk; 3) three of the four all Software Project Risks Taxonomies/Frameworks accounted for the excessive number of external contracts risk; and 4) the acknowledgment of a wrong system performance risk and the use of a new/immature technology risk was reported by, respectively, the two oldest and newest ones.

3.2.2 Plan-driven risk management – CMMI-DEV, ISO/IEC 12207, and RUP-SOA

The current primary software process frameworks and software development methodologies formally recognize Risk Management as an essential process that accompanies the entire software project life cycle throughout its Planning, Execution, Control, and Completion phases.

As outlined in ISO/IEC 12207 standard (ISO/IEC, 2008), the purpose of Risk Management is to identify, analyze, treat, and monitor risks continuously. Early and aggressive detection of risk (Boehm, 1991; Kwak & Stoddard, 2004; Brun et al., 2013) is important because it is typically easier, less costly, and less disruptive to make changes and correct work efforts during the earlier, rather than the later, phases of the project. Effective risk management includes early and aggressive risk identification through collaboration and the involvement of relevant stakeholders. Accordingly, the Risk Management Process is a continuous process for systematically addressing risk throughout the life cycle of a system or software product or service. It can be applied to risks related to the entire lifecycle of the acquisition, development, maintenance, or operation of a system. It is generally agreed that Risk Management takes a continuing, forward-looking approach for managing risks that include activities and phases such as identification of risk parameters, risk assessments, and risk mitigation.

Specifically, according to ISO/IEC 12207 standard (ISO/IEC, 2008), the project shall implement the following activities and tasks concerning the Risk Management Process: 1) Risk management planning, which defines the Risk management policies describing the guidelines under which risk management is to be performed; 2) Risk profile management, which defines the context of the Risk Management Process including a description of stakeholders' perspectives, risk categories, and the technical and managerial objectives, assumptions and constraints; 3) Risk analysis, during which the probability of occurrence and consequences of each risk identified shall be estimated while each risk shall be evaluated against its risk thresholds; 4) Risk treatment, which recommends alternatives for risk treatment in risk action requests; 5) Risk monitoring, which indicates that all risks and the risk management context shall be continuously monitored for changes; and 6) Risk management process evaluation, during which information shall be collected throughout the project's life cycle for purposes of improving the Risk Management Process and generating lessons learned.

ISO/IEC 12207 standard (ISO/IEC, 2008) and CMMI-DEV (CMMI Product Team, 2010) list Risk Management as one of the 22 core process areas in the CMMI development framework. It is associated with the Project Management category and has a maturity level of 3 (using a simple numbering scheme of 1–5, with 5 being the highest maturity level). According to CMMI-DEV (CMMI Product Team, 2010), Risk Management can be divided into the following parts and processes: 1) Prepare for Risk Management, defining a risk management strategy, including the following tasks: 1.1) Determine Risk Sources and Categories, 1.2) Define Risk Parameters, and 1.3) Establish a Risk Management Strategy; 2) Identify and Analyze Risks, including 2.1) Identify Risks and 2.2) Evaluate, Categorize, and Prioritize Risks; and 3) Mitigate Risks, handling identified risks including 3.1) Develop Risk Mitigation Plans and 3.2) Implement Risk Mitigation Plans.

Comparatively, in the main software development methodologies such as the Rational Unified Process for Service-Oriented Architecture (RUP-SOA), Risk Management is introduced as early as in the first Inception Phase of the RUP-SOA lifecycle (Péraire et. al., 2007). At the end of the Inception Phase, all relevant risks have been identified and a mitigation strategy exists for each risk, which corresponds to the Risk Identification and Assessment processes as specified in the main software process frameworks (CMMI-DEV and ISO/IEC 12207). A Risk List (about 25% complete) will be among the major outputs from the Inception Phase activities. In the second Elaboration Phase, the identified major risk elements have been addressed and have been credibly resolved, which corresponds to the Risk Mitigation and Resolution processes as specified in the main software process frameworks (CMMI-DEV and ISO/IEC 12207). An updated Risk List (about 50% complete) is included as an essential work product at the end of the Elaboration Phase. In the remaining two phases, the Construction Phase and Transition Phase, an assessment of the current iteration and a reevaluation of the risks will be conducted. The Risk List will reach 75% complete at the end of the Construction Phase and fully complete upon completion of the Transition Phase. It is also noted that an iterative software development process has the benefit of early risk deduction as well as higher predictability throughout the software project life cycle.

Table 3.3 summarizes an integrated view of the phases, risks, and risk controls from a plan-driven perspective considering the current main software process frameworks (CMMI-DEV, ISO/IEC 12207) and main software plan-driven development methodology (RUP-SOA).

Table 3.3: Plan-Driven Risk Management – ISO/IEC 12207, CMMI-DEV and RUP-SOA.

ISO/IEC 12207 / CMMI-DEV / RUP-SOA Phases and Activities/Tasks	**Typical Outputs**	**Risk Tools/Resources**
Risk Management Planning – Determine Risk Sources and Categories – Define Risk Parameters – Establish a Risk Management Strategy/ Plan	A Risk List (about 25% complete): – Risk sources, categories, and parameters – Risk evaluation and prioritization criteria – Risk Management Plan	– Risk management databases – Risk taxonomies – Brainstorming – Checklists – Structured interviews
Risk Identification & Analysis – Identify Risks – Evaluate, Categorize, and Prioritize Risks	An updated Risk List (about 50% complete) – List of identified risks, including the context, conditions, and consequences of risk occurrence – Risk priorities	– Risk assessments – Process, project, and product performance models – Cost models – Network analysis – Quality factor analysis
Risk Mitigation & Resolution – Develop Risk Mitigation Plans – Implement Risk Mitigation Plans	The Risk List will reach 75% complete – Documented handling options for each risk – Risk mitigation plans – Contingency plans	– Risk mitigation tools – Prototyping tools – Modeling and simulation tools

Table 3.3 (continued)

ISO/IEC 12207 / CMMI-DEV / RUP-SOA Phases and Activities/Tasks	Typical Outputs	Risk Tools/Resources
Risk Monitoring & Evaluation – Monitor and reevaluate the solved and open risks – Conduct an assessment and reevaluation of the Risk Management Process	Fully completed Risk List – Risk status reports (number of risks identified, managed, tracked, and controlled) – Reevaluation of risk management process (e.g., Comparison of estimated versus actual risk mitigation effort and impact)	– Periodic review and re-assessment – Risk assessments – Process, project, and product performance models

3.2.3 Agile and lightweight risk management – Scrum and ISO/IEC 29110

Agile and Lightweight development practices claim the utilization of Risk Management controls (Schwaber, 1997; Rising & Janoff, 2000; Sutherland & Schwaber, 2017; ISO/IEC, 2011; O'Connor & Laporte, 2012). Other literature on agile development practices report the lack of traditional Risk Management activities (Gold & Vassell, 2015; Tavares et al., 2019a; 2019b) but proposes enhancements for including these required practices.

The Scrum process (Schwaber, 1997; Sutherland & Schwaber, 2011) can be described for the following phases and activities: Pre-Game (with Backlog Planning, Architecture Design, and Sprint Planning activities), Game (with Daily Meeting, Sprint, Sprint Review, and Sprint Retrospective activities), and Post-Game (with System Integration, User Documentation, Project Closure). For some Scrum literature (Schwaber, 1997; Rising & Janoff, 2000; Sutherland & Schwaber, 2017), this software development process (also self-considered as a framework of agile loose practices) uses control managerial practices for inherent risks in any software development project. Overall, Scrum assumes that the software development process is inherently unpredictable and thus exercises specific agile practices as risk countermeasures. Scrum is self-defined as an "*iterative, incremental approach to optimize predictability and control risk*" (Sutherland and Schwaber, 2017; p. 4). Risk Controls are exercised through Backlog Planning with an "*assessment of risk and appropriate risk controls*" (Schwaber, 1997; p. 71), and Daily Meeting and Sprint Review where risks are reported, reviewed and adequate countermeasures are defined to be applied during the current and next Sprint (Sutherland & Schwaber, 2017).

Complementary Scrum literature (Gold & Vassell, 2015; Tavares et al., 2019a; 2019b) suggests several additions to Scrum for enhancing the implicit assumed Scrum Risk Management. The main ones additions are 1) a risk taxonomy; 2) qualitative risk analysis; 3) quantitative risk analysis; and 4) risk response plans; and the total inclusion of a Management of Risk activity exercised during the Pre-Game and Game Scrum phases (i.e. Agile Risk Identification, Agile Risk Assessment, Agile Risk Response, and Agile Risk Review and Monitoring). Regarding the Software Project Risks frameworks from section 3.2.1, Scrum does not report a Risk Taxonomy. Thus, the Scrum Master and Scrum Team are responsible for agreeing on risk types based on their expertise and additional sources. Scrum complementary literature (Tavares et al., 2019a) has proposed a Taxonomy of Scrum Risks classified by Artifacts, Events, Features, Roles, and Techniques and Methods.

Regarding the Lightweight software process frameworks (ISO/IEC, 2011; O'Connor & Laporte, 2012), Risk Management is exercised through the Project Management subprocess. The ISO/IEC 29110 standard (Basic Profile) contains two sub-process: Project Management (with 4 activities and 7 objectives) and Software Implementation (with 6 activities and 7 objectives). Project Management sub-process activities are PM.1 Project Planning, PM.2 Project Plan Execution, PM.3 Project Assessment and Control, and PM.4 Project Closure. Risk Management is exercised specifically through the objective PM.O5 "*Risks are identified as they develop and during the conduct of the project*" (ISO/IEC, 2011; p. 6), the tasks PM.1.9 "*Identify and document the risks which may affect the project*" (ISO/IEC, 2011; p. 11), PM.2.3 "*Conduct revision meetings with the Work Team, identify problems, review risk status, record agreements and track them to closure*" (ISO/IEC, 2011; p. 13), and the activity PM.3 Project Assessment and Control where a "*Review of project risks and identification of new risks*" is conducted (ISO/IEC, 2011; p. 14). Regarding the Software Project Risks frameworks from section 2.2.1, the ISO/IEC 29110 standard (Basic Profile) does not report a Risk Taxonomy. Thus, the Project Manager is responsible for guiding this process based on expertise and additional sources.

Table 3.4 summarizes an integrated view of the Scrum and ISO/IEC 29110 phases, risks, and risk controls from an agile/lightweight perspective.

3.3 Review of decision-making support systems – a modern analytics classification

Decision-Making Support Systems (DMSS) have been defined as *computer-based systems designed to support some, several or all phases of a decision-making process* (Forgionne et al., 2009). Origins of DMSS tools can be traced to the early 1980s, but they have evolved from the classical DMSS tools to modern versions during these last 40 years (Shim et al., 2002; Dersen & Demirkan, 2013). Classical DMSS tools

Table 3.4: Agile Risk Management – Scrum and ISO/IEC 29110.

Scrum / ISO/IEC 29110 Phases and Activities/Tasks	Typical Risks	Risk Controls
Pre-Game / Project Planning – Backlog Planning – Architecture Design – Sprint Planning	– Under or over effort estimation – Missed requirements / Technical Risks – Undervalue delivery	– Team Poker Planning – Spikes / List of Risks-Impacts-Countermeasures – User Story Prioritization
Game / Project Execution – Daily Meeting – Sprint	– Unproductive day – Sprint delays	– 3-question technique – Burndown Charts
Game / Project Assessment and Control – Sprint Review – Sprint Retrospective	– Technical mistakes – Human conflicts	– Next Sprint Re-planning – Support teams
Post-Game / Project Closure – System Integration – User Documentation – Closure	– Release an under quality software and/or over cost and/or over the scheduled project	– Daily Meetings / Sprint Review / Sprint Retrospective / TDD / Product Owner total involvement

(Forgionne et al., 2009) are Decision Support Systems (DSS), Executive Information Systems (EIS), Expert Systems/Knowledge-based Systems (ES/KBS), Knowledge Management Systems (KMS), Group Decision Support Systems (GDSS), and intelligent DMSS (i-DMSS). Modern DMSS tools (Sharda et al., 2015) are Data Warehouse-based Business Intelligence tools (DW/BI), and Data Mining and Business Statistical Analytics (DM/BSA) tools. Another related study (Dersen & Demirkan, 2013) has proposed a DMSS framework that integrates both classic and modern views under the concept of Analytics. Delen and Demirkan (2013) refer to Analytics as *a set of classic and modern decision-making tools for transforming data to information and knowledge usable for decision makers*. Delen and Demirkan (2013) classify Analytics tools by their main support aim as Descriptive Analytics tools, Predictive Analytics tools, and Prescriptive Analytics tools.

Descriptive Analytics tools correspond to classic Executive Information Systems (EIS) and modern Data Warehouse-based Business Intelligence tools (DW/BI). These tools focus on standard/ad-hoc/on-demand/interactive-dynamic reporting, querying, and visualization support for knowing *what happened or what is happening* toward the timely *detection of business problems and/or opportunities*. Predictive Analytics tools correspond to Data Mining and Business Statistical Analytics DMSS (DM/BSA) tools. These tools focus on statistical, mathematical, and artificial intelligence techniques to discover data patterns and trends useful for estimating *what could happen and why could happen in this way* toward the *accurate estimation of future business states*. Prescriptive Analytics correspond to classic Decision Support Systems (DSS), Expert Systems/ Knowledge-based Systems (ES/KBS), Knowledge Management Systems (KMS), Group Decision Support Systems (GDSS), and Intelligent (optimization) DMSS (i-DMSS). These tools focus on a variety of modeling-simulation, knowledge representation and processing mechanisms, knowledge storages and communication mechanisms, collaborative group decisional mechanisms, and intelligent optimization techniques for knowing *what should be done and why it should be done* towards determining the best plausible courses of action on complex decision-making problems.

A complete decision-making process (Forgionne & Kohli, 2000; Mora et al., 2014) is a managerial process composed of the phases of Intelligence, Design, Choice, Implementation, and Learning. Each type of DMSS tool is designed usually to support a partial decision-making process (i.e. a particular decision-making process phase). The Intelligence phase refers to collecting all relevant elements associated with a business problem or/and opportunity. The Design phase refers to organizing methodically such elements into a decision model. The Choice phase refers to using and interacting with the decision model (i.e. enacted in a DMSS tool) to determine the best courses of action regarding the business problem or opportunity. The Implementation phase refers to putting into practice the selected course of action and monitoring its results. Finally, the Learning phase refers to a post-mortem analysis of the implemented decision, its expected, and real outcomes. Classical and modern DMSS tools have been widely used in organizations because when they are successfully implemented, they provide benefits such as improved organizational performance, better decision quality, improved communication, enhanced mental models, amplified analytical skills of decision-makers, and reduced decision times among others (Turban & Aronson, 1998; Forgionne & Kohli, 2000; Phillips-Wren et al., 2004; Eom & Kim, 2006; March & Hevner, 2007; Davenport, 2006; Phillips-Wren et al., 2009; Delen & Demirkan, 2013; Watson, 2014; Sharda et al., 2015).

Table 3.5, from the previous analyzed literature, summarizes the main decision support characteristics provided by various types of classical and modern DMSS Analytics tools.

Table 3.5: Classical and Modern DMSS Analytics Tools.

Descriptive DMSS Analytics	Predictive DMSS Analytics	Prescriptive DMSS Analytics
Executive Information Systems (classic tools) ***EIS*** – Standard/ad-hoc/on-demand/interactive-dynamic reporting, querying, and visualization support – Executive dashboards on data warehouses **Data Warehouse-based Business Intelligence (modern tools)** **DW-BI** – Standard/ad-hoc/on-demand/interactive-dynamic reporting, querying, and visualization support – Multiple dashboards on data warehouses	**Data Mining (modern tools)** **DM** – Patterns detection – Trends detection – Associations detection – Affinities detection **Business Statistical Analytics (modern tools)** **BSA** – Forecasting models – Regression models – Clustering models – Classification models	**Decision Support Systems (classic tools**) **DSS** – Modeling-simulation – What-if, goal-seeking, sensitivity analysis – Multicriteria analysis – Numerical decision analysis **Expert Systems/Knowledge-based Systems (classic tools)** **ES/KBS** – Knowledge processing – Knowledge modeling **Knowledge Management Systems (classic tools)** **KMS** – Knowledge repositories – Knowledge communication **Group Decision Support Systems (classic tools)** **GDSS** – Group decision modeling – Group decision analysis – Group decision communication **Intelligent (optimization) DMSS (classic tools) i-DMSS** – Analytics optimization – Heuristic optimization

3.4 Integrated DMSS analytics tools – software project risk management framework and illustrative cases

Based on the results reported previously in Sections 3.2 and 3.3, we elaborate an Integrated DMSS Analytics Tools – Software Project Risk Management framework (see Table 3.6) that accounts for both plan-driven and agile Software Project Risk

Table 3.6: Cases of DMSS Analytics Tools applied to Plan-Driven & Agile Software Project Risk Management.

Types of DMSS Analytics Tools		Cases of DMSS Analytics Tools	Plan-Driven Software Risk Management Process				Agile Software Risk Management Process			
			Risk Management Planning	Risk Identification & Analysis	Risk Mitigation & Resolution	Risk Monitoring & Evaluation	Pre-Game / Project Planning	Game / Project Execution	Game / Project Assessment and Control	Post-Game / Project Closure
Descriptive	EIS / DW-BI Tools	None.	–	–	–	–	–	–	–	–
Predictive	DM / BSA Tools	C.02 – BSA with Bayesian Belief Networks (BBNs)	■	■	■	–	–	–	–	–
		C.03 – BSA with Bayesian Belief Networks (BBNs)	■	■	–	–	–	–	–	–
		C.05 – BSA with Logistic Regression Models	–	■	–	–	–	■	■	–
		C.06 – BSA with Multiple Linear Regression Models	–	■	■	–	–	–	–	–
		C.07 – BSA with Logistic Regression Models	–	■	–	–	–	–	–	–
		C.08 – DM with Random Forests for prediction) and i-DMSS with Linear PBO optimization	–	■	■	–	–	–	–	–

Table 3.6 (continued)

Types of DMSS Analytics Tools		Cases of DMSS Analytics Tools	Plan-Driven Software Risk Management Process				Agile Software Risk Management Process			
			Risk Management Planning	Risk Identification & Analysis	Risk Mitigation & Resolution	Risk Monitoring & Evaluation	Pre-Game / Project Planning	Game / Project Execution	Game / Project Assessment and Control	Post-Game / Project Closure
		C.09 – DM with Decision Trees enhanced with an Ensemble method	–	■	–	■	–	–	–	–
		C.10 – DM with Bayesian networks with causality constraints	–	■	■	■	–	–	–	–
		C.12 – BSA with a Linear Mixed Model	–	■	–	–	–	–	–	–
Prescriptive	DSS / ES-KBS / KMS / GDSS / i-DMSS Tools	C.01 – DSS with System Dynamics simulation and a KMS	–	■	■	■	–	–	–	–
		C.04 – DSS with Discrete Event and Monte Carlo Simulation	■	■	■	–	–	–	–	–
		C.11 – ES/KBS with Fuzzy Analytic Hierarchy process and Fuzzy Inference Systems	–	■	–	–	–	–	–	–
		C.13 – DSS with Analytical Hierarchy Process	■	■	■	■	–	–	–	–
		C.14 – DSS with DEMATEL and Network Theory	–	■	–	–	–	–	–	–

Management processes versus the nine types of DMSS Analytics tools grouped in their respective three DMSS Analytics tool categories (i.e. Descriptive, Predictive and Prescriptive). This simple framework is useful for identifying what is provided and what is missed from decisional support for any type of DMSS Analytical tool compared to a particular activity performed in either a plan-driven or agile Software Project Risk Management process.

To identify these issues in the scientific literature associated with the topics of DMSS Analytics tools and Software Project Risk Management, we conducted a selective literature review (Glass et al., 2004). A selective literature review qualifies as a descriptive research approach and literature analysis research method (Glass et al., 2004). A selective literature review differs from a systematic literature review (Brereton et al., 2007) in the magnitude of the sample of studies selected once fixed the selection criteria because it does not analyze all studies from an exhaustive search but from a reduced and selective sample of studies. A selective literature review differs also from a mapping study (Petersen et al., 2008) in the purpose of extracting core findings related to specific research questions rather than elaborating a visual multidimensional classification of topics regarding the dimensions of interest for the researchers.

We applied the following steps: 1) to define general and specific knowledge inquiries; 2) to define the criteria for selecting journals; 3) to define the generic document search statement as well as the document inclusion and exclusion criteria; 4) to execute knowledge search procedures; 5) to analyze the abstract of the located documents and decide its inclusion or exclusion for detailed analysis; 6) to elaborate a detailed analysis of included documents; and 7) to populate the framework and report the type of DMSS Analytics tool, and the type and activities supported by a Software Project Risk Management process, the scope of application (i.e. real setting or laboratory test), and benefits. Following these steps:

1) To define the general and specific knowledge inquiries. We stated the general knowledge inquiry as *What kind of DMSS Analytics tools have been used to support a plan-driven and/or agile Software Project Risk Management process in the 1990–2019 period?* Specific knowledge inquiry was stated as: *What are the main characteristics (i.e. type of DMSS Analytics tool, category of Analytics support, type and activities supported by a Software Project Risk Management process, the scope of application (i.e. real setting or laboratory test), and benefits) found in each supported case?*
2) To define the criteria for selecting journals. C.1) journal already reported as a leading journal in previous literature or journal listed in the JCR index with at least 1.0 of impact factor if not suggested as a leading journal in previous literature; and C.2) journal devoted to Software Engineering, and/or Decision-Making Support Systems Analytics, and/or on Project Management / Risk Management disciplines. We limited the number of selected journals to a total number of 15 by scope-time research limitations. Combining criteria C.1 and C.2 for the Software Engineering discipline

(Wong et al., 2011; Garousi & Fernandes, 2016) we selected the journals JSS (Journal of Systems and Software), IST (Information and Software Technology), EMSE (Empirical Software Engineering), TOSEM (ACM T. on Software Engineering and Methodology), and SPE (Software Practice and Experience). Similarly, for the Decision-Making Support Systems Analytics discipline (Forgionne & Kohli, 2001; Serenko & Dohan, 2011) we selected the journals DSS (Decision Support Systems), DS (Decision Sciences), ESWA (Expert Systems with Applications), GDSN (Group Decision and Negotiation), and OMG (Omega). For Project Management / Risk Management disciplines we did not find a previous study on leading journals, so we selected three journals listed in the JCR index with the highest impact factors. These journals were IJPM (Int. Journal of Project Management), PMJ (Project Management Journal), RMIJ (Risk Management), RA (Risk Analysis), and JRR (Journal of Risk Research).

3) To define the generic document search statement as well as the document inclusion and exclusion criteria. The generic search statement was defined as "journal (journal target ISSN) AND period (1990–2019) AND (abstract (("risk management" OR "software risk" OR "software project risk") AND ("decision" OR "knowledge" OR "analytics"))". The unique document inclusion criterion was stated as "*authors agree that the abstract refers a case where a DMSS Analytics tool (any type) is reported as decisional support for a Software Project Risk Management process*", and the exclusion criteria as the logical negation of the inclusion criteria.
4) To execute the knowledge search procedures. We applied the generic search statement and located 396 research articles in the 1990–2019 period. Table A.1 in the Appendix reports the number total of articles located in the 15 leading selected journals.
5) To analyze the abstract of the located documents and decide its inclusion or exclusion for detailed analysis. We applied the inclusion-exclusion criteria to the 396 located articles resulting in a final list of 14 valid articles, all of which were selected to be analyzed in further detail. Table A.2 in the Appendix reports the list of these 14 articles.
6) To elaborate on a detailed analysis of the included documents. The first two authors analyzed every full article separately, elaborated a descriptive profile case (i.e. type of DMSS Analytics tool, type and activities supported of a Software Project Risk Management process, the scope of application (i.e. real setting or laboratory test), and benefits), and shared their profile cases for agreeing to a unique profile case.
7) To populate the framework and report the profile cases. Finally, the framework was populated (see Table 3.6). The acronyms used in Table 3.6 are defined in Table 3.5.

3.5 Discussion

Applying DMSS Analytics tools to plan-driven and agile Software Project Risk Management is still in its infancy. Our literature review for the 1990–2019 period on 15 top specialized journals on Software Engineering, Decision-Making Support Systems, and Project Management, while produced 396 potential papers, the final inclusion filter applied as " to report a DMSS tool applied to some or several stages of the Risk Management processes in software projects" was limited finally to just 14 papers.

From Table 3.6, relevant findings can be derived. First, the application of DMSS Analytics tools has been linked mainly to the plan-driven Software Engineering approach, which is a logical consequence given that Risk Management is a mandatory process for this approach. In contrast, despite the almost 20 year-period of the utilization of the Agile Software Engineering approach, we located only one application case in the top literature. The Agile approach claims to perform Risk Management implicitly (or embedded) by applying their usual activities and tasks, but explicitly a well-documented Risk Management process is not performed. We identified that BSA with Logistic Regression Models applied to Game / Project Execution and Game / Project Assessment and Control was the unique case found for the Agile approach. Thus, DMSS Analytics tools applications to support the agility issues in Agile Software Projects are open research questions at present. Second, into the plan-driven Software Engineering approach, as Table 3.6 shows, Risk Identification, and Analysis activity is the primary application for both predictive and prescriptive analytics methods. An emerging area is in Risk Mitigation and Resolution, followed by some applications to Risk Management Planning, and Risk Monitoring and Evaluation. It is relevant to remark that this Risk Identification and Analysis activity has been, thus, supported by diverse Analytics approaches such as Bayesian Belief Networks, Logistic Regression, Linear Regression, Random Forests, and Decision Trees with a predictive purpose, and with System Dynamics simulation, Discrete Event simulation, and Fuzzy AHP methods from a prescriptive purpose. However, it is interesting to remark that well-established Artificial Intelligence (AI) techniques included in Intelligent optimization DMSS (i.e. i-DMSS) and Expert Systems (i.e. ES) tools have not been widely adopted. This appears to offer opportunities for future research to apply AI to Software Project Risk Management. Third, the relevant Risk Monitoring & Evaluation activity has been practically not supported as well as the utilization of the descriptive DMSS Analytics tools. This identified situation opens important research opportunities given that the main purpose of Risk Monitoring & Evaluation is to track software project heath status via risk tracking to timely identify critical deviations to expected plans and apply the adequate corrective actions. This tracking and monitoring activity is naturally supported by dashboards and visualizations techniques naturally provided by descriptive DMSS Analytics tools.

In brief, we can claim that DMSS Analytics tools, in their three types (descriptive, predictive, and prescriptive ones) have been scarcely used for supporting Risk

Management activities in Software Projects. The plan-driven Software Engineering approach appears to be naturally supported by DMSS Analytics tools given the mandatory Risk Management process in this plan-driven approach. In contrast, the application of DMSS Analytics tools in the Agile Software Engineering approach opens relevant research questions for supporting their implicit and discretionary way of performing Risk Management activities. We suggest the following ones: 1) what are the managerial and technical DMSS acceptance factors for software developers in the agile approach? 2) what type and sub-types of DMSS Analytics tools are more appropriated for supporting the agile Risk Management practices? 3) what characteristics are shared and unshared in the risk frameworks and taxonomies for plan-driven and agile software development approaches? 4) how can DMSS analytics capabilities be embedded in current agile project management tools? And 5) how can an agile maturity level risk management framework be devised?

3.6 Contributions and conclusions

This chapter has explored a research area that is important to Software Project Risk Management and that has been scarcely examined in the past. Specifically, the contributions of this chapter are:

- Examination of the application of DMSS Analytics tools to plan-driven and agile Software Project Risk Management.
- Identification of opportunities to improve decision making and support specific tasks in Software Project Risk Management using existing tools.
- Illustration of the use of DMSS Analytics tools to support tasks in plan-driven and agile Software Project Risk Management in the literature.

Through a structured selective literature review, we found evidence that DMSS Analytics tools have been applied to some plan-driven Software Project Risk Management projects to identify and analyze risk, and secondarily to mitigate and resolve risks in some cases. The number of successful cases found of management, planning, monitoring, and evaluation of risk with DMSS Analytics tools in several projects using plan-driven Software Project Risk Management was reduced. However, this has provided empirical evidence, on the effectiveness of supporting Risk Management activities. In contrast, Agile Software Project Risk Management projects have largely ignored decision support tools as aids to assist with its implicit management, planning, monitoring, and evaluation of risks. Hence, there are still open opportunities and research inquiries to be addressed for fostering wider use of these DMSS Analytics tools in both plan-driven and agile Software Project approaches which would assist in the Risk Management activities, and lately to help alleviate software project delays, costs overruns, and even failed projects.

Appendix

Table A.1: Research Articles located in the 15 selected Journals.

Journal	ISSN Journal	Discipline	# articles located	valid articles	# articles selected
JSS	0164–1212	Software Engineering	7	4	4
IST	0950–5849	Software Engineering	1	0	0
EMSE	1382–3256	Software Engineering	3	3	3
TOSEM	1049–331X	Software Engineering	0	0	0
SPE	0038–0644	Software Engineering	0	0	0
		SUB-TOTAL	**11**	**7**	**7**
DSS	0167–9236	DMSS Analytics	13	3	3
DS	0011–7315	DMSS Analytics	9	0	0
ESWA	0957–4174	DMSS Analytics	30	2	2
GDSN	0926–2644	DMSS Analytics	3	0	0
OMG	0305–0483	DMSS Analytics	9	0	0
		SUB-TOTAL	**64**	**5**	**5**
IJPM	0263–7863	Project Management	36	1	1
PMJ	8756–9728	Project Management	2	1	1
RM	1460–3799	Risk Management	15	0	0
RA	0272–4332	Risk Management	186	0	0
JRR	1366–9877	Risk Management	82	0	0
		SUB-TOTAL	**321**	**2**	**2**
		TOTAL	**396**	**14**	**14**

Note: The Scopus search query was "ISSN (01641212) AND ABS ("risk management" OR "software risk" OR "software project risk") AND ABS ("decision" OR "knowledge" OR "analytics") AND PUBYEAR > 1989 AND PUBYEAR < 2020", where the ISSN was changed for each specific journal. This example corresponds to the JSS.

Table A.2: List of the 14 valid and selected Research Articles located in the 15 selected Journals.

Journal	Discipline	# articles selected
JSS	Software Engineering	C.01 Barros, M., Werner, C., & Travassos, G. (2004). Supporting risks in software project management. *Journal of Systems and Software, 70*(1–2), 21–35. C.02 Fan, C., & Yu, Y. (2004). BBN-based software project risk management. *Journal of Systems and Software, 73*(2), 193–203. C.03 Houmb, S., Franqueira, V., & Engum, E. (2010). Quantifying security risk level from CVSS estimates of frequency and impact. *Journal of Systems and Software, 83*(9), 1622–1634. C.04 Uzzafer, M. (2013). A simulation model for strategic management process of software projects. *Journal of Systems and Software, 86*(1), 21–37.
EMSE	Software Engineering	C.05 Costa, D., McIntosh, S., Treude, C., Kulesza, U., & Hassan, A. (2018). The impact of rapid release cycles on the integration delay of fixed issues. *Empirical Software Engineering, 23*(2), 835–904. C.06 Han, W. (2014). Validating differential relationships between risk categories and project performance as perceived by managers. *Empirical Software Engineering, 19*(6), 1956–1966. C.07 Takagi, Y., Mizuno, O., & Kikuno, T. (2005). An empirical approach to characterizing risky software projects based on logistic regression analysis. *Empirical Software Engineering, 10*(4), 495–515.
DSS	DMSS Analytics	C.08 Hu, Y., Du, J., Zhang, X., Hao, X., Ngai, E., Fan, M., & Liu, M. (2013). An integrative framework for intelligent software project risk planning. *Decision Support Systems, 55*(4), 927–937. C.09 Hu, Y., Feng, B., Mo, X., Zhang, X., Ngai, E., Fan, M., & Liu, M. (2015). Cost-sensitive and ensemble-based prediction model for outsourced software project risk prediction. *Decision Support Systems, 72*, 11–23. C.10 Hu, Y., Zhang, X., Ngai, E., Cai, R., & Liu, M. (2013). Software project risk analysis using Bayesian networks with causality constraints. *Decision Support Systems, 56*(1), 439–449.
ESWA	DMSS Analytics	C.11 Rodríguez, A., Ortega, F., & Concepción, R. (2016). A method for the evaluation of risk in IT projects. *Expert Systems with Applications, 45*, 273–285. C.12 Shi, X., Tsuji, H., & Zhang, S. (2011). Eliciting experts' perceived risk of software offshore outsourcing incorporating individual heterogeneity. *Expert Systems with Applications, 38*(3), 2283–2291.
IJPM	Project Management	C.13 Neves, S., da Silva, C., Salomon, V., da Silva, A., & Sotomonte, B. (2014). Risk management in software projects through knowledge management techniques: Cases in Brazilian incubated technology-based firms. *International Journal of Project Management, 32*(1), 125–138.

Table A.2 (continued)

Journal	Discipline	# articles selected
PMJ	Project Management	C.14 Hwang, W., Hsiao, B., Chen, H., & Chern, C. (2016). Multiphase assessment of project risk interdependencies: Evidence from a university ISD project in Taiwan. *Project Management Journal, 47*(1), 59–75.
	TOTAL	**14**

References

Abrahamsson, P., Oza, N., & Siponen, M. T. (2010). Agile software development methods: a comparative review. In T. Dingsøyr, T. Dybå, & N. B. Moe (Eds.), *Agile Software Development* (pp. 31–59). Springer, Heidelberg.

Aven, T. (2016). Risk assessment and risk management: review of recent advances on their foundation. *European Journal of Operational Research*, *253*(1), 1–13.

Agarwal, N., & Rathod, U. (2006). Defining 'success' for software projects: An exploratory revelation. *International Journal of Project Management*, *24*(4), 358–370.

Bannerman, P. (2008). Risk and risk management in software projects: a reassessment. *Journal of Systems and Software*, *81*(12), 2118–2133.

Boehm, B. (1988). A spiral model of software development and enhancement. *IEEE Computer*, *21*(5), 61–72.

Boehm, B. (1991). Software risk management: principles and practices. *IEEE Software*, *8*(1), 32–41.

Boehm, B., & DeMarco, T. (1997). Software risk management. *IEEE Software*, *14*(3), 17–19.

Boehm, B., Egyed, A., Kwan, J., Port, D., Shah, A., & Madachy, R. (1998). Using the WinWin spiral model: a case study. *IEEE Computer*, *31*(7), 33–44.

Boehm, B., & Turner, R. (2003). Using risk to balance agile and plan-driven methods. *IEEE Computer*, *36*(6), 57–66.

Bourque, P., & Fairley, R. E. (2014). *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0.* IEEE Computer Society Press, Los Alamos, CA.

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, *80*(4), 571–583.

Brun, Y., Holmes, R., Ernst, M. D., & Notkin, D. (2013). Early detection of collaboration conflicts and risks. *IEEE Transactions on Software Engineering*, *39*(10), 1358–1375.

Bryson, J. M., Berry, F. S., & Yang, K. (2010). The state of public strategic management research: A selective literature review and set of future directions. *The American Review of Public Administration*, *40*(5), 495–521.

Cerpa, N., & Verner, J. M. (2009). Why did your project fail? *Communications of the ACM*, *52*(12), 130–134.

CMMI Product Team (2010). *CMMI for Development (CMMI-DEV). Version 1.3*. Technical Report, CMU/SEI-2010-TR-033, Software Engineering Institute.

Davenport, T. H. (2006). Competing on analytics. *Harvard Business Review*, *84*(1), 1–8.

Delen, D., & Demirkan, H. (2013). Data, information and analytics as services. *Decision Support Systems*, *55*(1), 359–363.

Eclipse Foundation (2019). Document online at ---http://www.eclipse.org/downloads/download.php?file=/technology/epf/Scrum/published/scrum_published_1.5_20080820.zip

El Emam, K., & Koru, A. G. (2008). A replicated survey of IT software project failures. *IEEE Software*, *25*(5), 84–90.

Eom, S., & Kim, E. (2006). A survey of decision support system applications (1995–2001). *Journal of the Operational Research Society*, *57*(11), 1264–1278.

Eom, S. (2016). Longitudinal author cocitation mapping: The changing structure of decision support systems research (1969–2012). *Foundations and Trends in Information Systems*, *1*(4), 277–384.

Forgionne, G., & Kohli, R. (2000). Management Support System effectiveness: further empirical evidence. *Journal of the Association for Information Systems*, *1*(3), 1–39.

Forgionne, G., & Kohli, R. (2001). A multiple criteria assessment of decision technology system journal quality. *Information & Management*, *38*(7), 421–435.

Forgionne, G., Mora, M., Gupta, J. N., & Gelman, O. (2009). Decision-making support systems. In: *Encyclopedia of Information Science and Technology*, 2ND Ed. (pp. 978–984). IGI Global.

Garousi, V. and Fernandes, J. (2016). Highly-cited papers in software engineering: the top-100. *Information and Software Technology*, *71*, 108–128.

Glass, R. L., Ramesh, V., & Vessey, I. (2004). An analysis of research in computing disciplines. *Communications of the ACM*, *47*(6), 89–94.

Gold, B., & Vassell, C. (2015). Using risk management to balance agile methods: a study of the Scrum process. In: *2nd. International Conference on Knowledge-Based Engineering and Innovation (KBEI)* (pp. 49–54). IEEE.

Han, W., & Huang, S. (2007). An empirical analysis of risk components and performance on software projects. *Journal of Systems and Software*, *80*(1), 42–50.

Higuera, R., & Haimes, Y. (1996). *Software Risk Management*. SEI Report CMU/SEI-96-TR-012, Carnegie Mellon University, Pittsburgh PA.

ISO/IEC (2008). *ISO/IEC 12207 – Systems and software engineering-software – life cycle processes*. International Organization for Standardization, Geneva, Switzerland.

ISO/IEC (2011). ISO/IEC TR 29110-5-1-2 Software engineering – Lifecycle profiles for Very Small Entities (VSEs) Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile. International Organization for Standardization, Geneva, Switzerland.

Jones, C. (2004). Software project management practices: Failure versus success. *CrossTalk: The Journal of Defense Software Engineering*, *17*(10), 5–9.

Keil, M., Cule, P., Lyytinen, K., & Schmidt, R. (1998). A framework for identifying software project risks. Communications of the ACM, 41(11), 76–83.

Kwak, Y. H., & Stoddard, J. (2004). Project risk management: lessons learned from software development environment. *Technovation*, *24*(11), 915–920.

Kruchten, P. (2004). *The rational unified process: an introduction*. Addison-Wesley Professional.

March, S., & Hevner, A. (2007). Integrated decision support systems: A data warehousing perspective. *Decision Support Systems*, *43*(3), 1031–1043.

Masso, J., Pino, F. J., Pardo, C., García, F., & Piattini, M. (2020). Risk management in the software life cycle: A systematic literature review. *Computer Standards & Interfaces*, *71*, (in press).

Marques, R., Costa, G., Silva, M., & Gonçalves, P. (2017). A survey of failures in the software development process. In: *Proc. of the 25th European Conf. on Information Systems (ECIS)*, (pp. 2445–2459).

Menezes, J., Gusmão, C., & Moura, H. (2019). Risk factors in software development projects: a systematic literature review. *Software Quality Journal*, *27*(3), 1149–1174.

Mora, M., Phillips-Wren, G., Marx-Gomez, J., Wang, F., & Gelman, O. (2014). The role of decision-making support systems in IT service management processes. *Intelligent Decision Technologies, 8*(2), 147–163.

NIST (2012). *Guide for Conducting risk managements – NIST Special Publication 800–30 R1*. National Institute of Standards and Technology, Gaithersburg, MD, USA.

O'Connor, R. V., & Laporte, C. Y. (2012). Software project management in very small entities with ISO/IEC 29110. In: *European Conference on Software Process Improvement* (pp. 330–341). Springer, Heidelberg.

Péraire, C., Edwards, E., Fernandes, A., Mancin, E., & Carroll, K. (2007). *The IBM Rational Unified Process for System z*. IBM Corp: IBM® Redbooks® publication.

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). Systematic mapping studies in software engineering. In: *12th International Conference on Evaluation and Assessment in Software Engineering (EASE)* (pp. 1–10).

Phillips-Wren, G., Hahn, E., & Forgionne, G. (2004). A multiple criteria framework for the evaluation of decision support systems. *Omega*, *32*(4), 323–332.

Phillips-Wren, G., Mora, M., Forgionne, G., & Gupta, J. (2009). An integrative evaluation framework for intelligent decision support systems. *European Journal of Operations Research*, *195*(3), 642–652.

PMI (2013). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Fifth Edition*. Project Management Institute, PA.

Ropponen, J., & Lyytinen, K. (2000). Components of software development risk: How to address them? A project manager survey. *IEEE Transactions on Software Engineering*, *26*(2), 98–112.

Rising, L., & Janoff, N. (2000). The Scrum software development process for small teams. *IEEE Software*, *17*(4), 26–32.

Sharda, R., Delen, D., Turban, E., Aronson, J., Liang, T., & King, D. (2015). *Business Intelligence and Analytics: Systems for Decision Support*. Upper Saddle River, NJ, Pearson Education.

Shim, J., Warkentin, M., Courtney, J., Power, D., Sharda, R., & Carlsson, C. (2002). Past, present, and future of decision support technology. *Decision Support Systems*, *33*(2), 111–126.

Schmidt, R., Lyytinen, K., Keil, M., & Cule, P. (2001). Identifying software project risks: An international Delphi study. *Journal of Management Information Systems*, *17*(4), 5–36.

Schultz, H. (1984). *Software Cost Estimation Workshop Report (No. MTR-9165)*. MITRE Corporation, Bedford, MA.

Schwaber, K. (1997). Scrum development process. In: *Business object design and implementation*, (pp. 117–134). Springer, London.

Serenko, A., & Dohan, M. (2011). Comparing the expert survey and citation impact journal ranking methods: example from the field of Artificial Intelligence. *Journal of Informetrics*, *5*(4), 629–648.

Sutherland, J., & Schwaber, K. (2011). *The Scrum Papers. Nut, Bolts, and Origins of an Agile Framework*. Scrum Inc.

Sutherland, J., & Schwaber, K. (2017). *The scrum guide – The definitive guide to scrum: The rules of the game*. Document online at http://www.scrum.org.

Tavares, B., da Silva, C. E. S., & de Souza, A. (2019a). Practices to Improve Risk Management in Agile Projects. *International Journal of Software Engineering and Knowledge Engineering*, *29*(03), 381–399.

Tavares, B., da Silva, C., & de Souza, A. (2019b). Risk management analysis in scrum software projects. *International Transactions in Operational Research*, *26*(5), 1884–1905.

Tiwana, A., & Keil, M. (2004). The one-minute risk assessment tool. *Communications of the ACM*, *47*(11), 73–77.

Turban, E., & Aronson, E. (1998). *Decision Support System and Intelligent System*. Upper Saddle River, NJ, Prentice-Hall.

Wallace, L., & Keil, M. (2004). Software project risks and their effect on outcomes. *Communications of the ACM*, *47*(4), 68–73.

Wallace, L., Keil, M., & Rai, A. (2004). How software project risk affects project performance: an investigation of the dimensions of risk and an exploratory model. *Decision Sciences*, *35*(2), 289–322.

Watson, H. J. (2014). Tutorial: Big data analytics: Concepts, technologies, and applications. *Communications of the Association for Information Systems*, *34*(1), 1247–1268.

Webster, J., & Watson, R. (2002). Analyzing the past to prepare for the future: writing a literature review. *MIS Quarterly*, *26*(2), (pp. xiii–xxiii).

Wong, W. E., Tse, T. H., Glass, R. L., Basili, V. R., & Chen, T. Y. (2011). An assessment of systems and software engineering scholars and institutions (2003–2007 and 2004–2008). *Journal of Systems and Software, 84*(1), 162–168.

Torgeir Dingsøyr and Yvan Petit

4 Managing layers of risk: Uncertainty in large development programs combining agile software development and traditional project management

4.1 Introduction

Software development projects have a track record of schedule and cost overruns, and have often faced challenges with delivering expected quality (Flyvbjerg & Budzier, 2011). In the nineties many expressed concern about software projects and used the word “crisis” (Kraut & Streeter, 1995). Consequently, project management professional associations and some authors have proposed that software development projects should adopt and implement the traditional risk management approaches as a contributor to success (Dey et al., 2007). This is quite natural since risk management has been considered one of the core knowledge areas in project management for many decades. Literature abounds in this field (Chapman & Ward, 2003; Jaafari, 2001; Johansen et al., 2019; Kendrick, 2015; Persson et al., 2009; Project Management Institute, 2019; Raz et al., 2002) and most general project management books include at least a section on risk management (Andersen, 2008; Dinsmore & Cabanis-Brewin, 2014; Gray & Larson, 2014; Kerzner, 2017; Nicholas, 2017). Risk management has also been shown to bring a number of benefits such as identification of favourable alternative courses of action, reduced surprises, and provided more precise estimates (Bannerman, 2008). However, recent research has also shown that these practices are not widely used in software development projects (Bannerman, 2008; Odzaly & Des Greer, 2014).

Risk Management is covered in the PMBOK Guide® (Project Management Institute, 2017) which defines a project risk as: “an uncertain event or condition that, if it occurs, has a positive or negative effect on a project objective” (p. 720) with processes for risk identification, risk categorization, risk qualitative and quantitative analysis, plan risk responses, monitor risks and implement risk response. Both the Project Management Institute (PMI) and Association for Project Management (APM) define a risk as an uncertain “event” which might have positive effects (opportunities) or negative effects (threats) (Association for Project Management, 2008). In general, however, project managers and the literature tend to focus on threats rather than on opportunities (Johansen, 2015). One of the most used textbooks in software engineering, for example, defines risks as “something you´d prefer not to happen” (Sommerville, 2016). Several techniques have been developed to assess the probability of occurrence and the potential

https://doi.org/10.1515/9783110652321-004

impacts to projects. A typical classification of risks is based on the level of knowledge about the possibility for the risk to take place (known or unknown) and the level of knowledge about the impact (known or unknown).

In projects undertaken in rapidly changing environments where uncertainty may be unavoidable, such as software development projects, managers need to go beyond traditional risk management; adopting roles and techniques oriented less towards planning and more towards flexibility and learning (De Meyer et al., 2002; Loch et al., 2006; Pich et al., 2002; Platje & Seidel, 1993). Some authors have therefore advocated for the use of the broader concept of *uncertainty management* instead of *risk management* (Cleden, 2009; Ibrahim et al., 2009; Perminova et al., 2007, 2008; Ward & Chapman, 2003). "Uncertainty management is not just about managing perceived threats, opportunities and their implications. [. . .] It implies exploring and understanding the origins of project uncertainty before seeking to manage it, with no preconceptions about what is desirable or undesirable" (Ward & Chapman, 2003, p. 98–99). An uncertainty management perspective draws attention to the need to understand variability in future activities; the individual's inability to assign probabilities to events, and their inability to predict accurately what the outcomes of a decision might be (Duncan, 1972). This has been shown to be particularly important in safety-critical projects (Saunders, 2015a, 2015b; Saunders et al., 2016). Dönmez and Grote (2018) summarize the difference between risk and uncertainty as follows: "risk refers to an unknown event that leads to one outcome from a set of known outcomes, each of which can be assigned a probability (however estimated). Uncertainty, in contrast, relates to a lack of knowledge about which outcomes are possible, including both their nature and associated probabilities. An 'uncertainty' is thus an unknown event from an unknown set of possible outcomes" (p. 95).

Uncertainty might apply to multiple facets of a project and some authors have tried to define categories of uncertainties. This ranges from simple models with two categories such as endogenous vs. exogenous uncertainty (Ahsan et al., 2010), to more extensive models. A review of uncertainty in project management by Jalonen (2011) uses the following categories: technology, markets, regulation, social/ political, acceptance/legitimacy, managerial, timing, and consequence uncertainty. More suitable to the software development projects are the three categories requirement uncertainty, resource uncertainty and task uncertainty proposed by Dönmez and Grote (2018) or the model by Ropponen and Lyytinen (2000) which was used in this study as it is widely used in previous studies to categorize software project risks: scheduling and timing, system functionality, subcontracting, requirement management, resource usage and performance and personnel management. Such categorization helps practitioners and researchers to identify adequate responses and techniques according to the uncertainty category (Cagliano, 2015).

Through multiple short iterations in conjunction with more frequent and earlier feedback, agile approaches such as Scrum (Hossain et al., 2009; Schwaber, 2004; Schwaber & Sutherland, 2013) have attempted to implicitly address the first category of uncertainty i.e., requirement uncertainty (Tuunanen et al., 2015). However, a qualitative study from 2014 (Siddique & Hussein, 2014) reveals that many Agile practitioners handle risk as in a traditional waterfall approach. Some authors also claim that, while having addressed the concern with requirement uncertainty, additional and new risks have also been introduced by the adoption of agile approaches such as new development cycle risks, development environment risks and programmatic risks (Walczak & Kuchta, 2013). Although agile approaches have brought product owners (or clients) closer to development there remains a separation of development and operations. Agile has also exacerbated the impact of technical debt[1] (Kruchten et al., 2012).

Agile approaches have initially been developed for small development projects with one team but recently they have been scaled to include multiple teams in larger organizations (Leffingwell, 2015; Vaidya, 2014), for example in the oil industry (Grewal & Maurer, 2007), in large software development organizations (Bick et al., 2018; Gruver & Mouser, 2015; Lindvall et al., 2004), in regulated environments (Fitzgerald et al., 2013) and in the public sector (Dingsøyr et al., 2018a). Software development projects might vary from simple one-team development to very large-scale projects with more than ten teams (Dingsøyr et al., 2014). However, studies on project success suggest that agile methods outperform traditional methods in contexts with high dynamism (Butler et al., 2019) and both in small and large projects (Jørgensen, 2019). Scaling agility have introduced a range of new challenges which have only been recently studied (Conboy & Carroll, 2019; Hobbs & Petit, 2017a, 2017b), including approaches to uncertainty management.

In general, we have few recommendations on risk management for large software projects. Sommerville´s textbook on software engineering (Sommerville, 2016) recommends establishing a risk register with consequence analysis and to establish a risk management plan which should be revisited throughout the project. A practice which is recommended for risk analysis is to identify and monitor the “top 10 risks”. However, the author notes that in agile development, risk management is “less formal”. Agile methods are believed to reduce risks related to requirements, but, on the other hand, increase risks related to staff turnover as documentation is more limited and communication more informal.

“Agile software development teams rely on high levels of autonomy. This has implications for the way in which they approach uncertainty. When left without clear guiding structures, their choice regarding how to deal with uncertainty becomes an uncertainty itself” (Dönmez & Grote, 2018). As a consequence, in contexts of multi-team projects, some additional levels of monitoring and coordination must

[1] Technical debt has been introduced as a metaphor in software development, describing the cost of taking an easy solution now that is cheaper to implement than a solution which will be more robust over time.

therefore be introduced. This applies to all types of activities, including risk/uncertainty management. Inter-team coordination could be done by standards (using the rules by which something is done), by plans (achieved by specifying what is to be produced, by whom and when), by formal mutual adjustment, and by informal mutual adjustment (Dingsøyr et al., 2018b; McBride, 2008; Sabherwal, 2003).

Many of the strategies to manage risks in Agile development are implicit (Hijazi et al., 2012; Nyfjord & Kajko-Mattsson, 2007; Odzaly & Des Greer, 2014). Moran argues that "risk management in agile projects remains a passive and implicit activity that can be misdirected and often misunderstood [. . .] whilst most developers have little difficulty explaining which features they are working on (e.g., user stories) or to what level of quality they should be completed (e.g. definitions of done), few can comment on the capacity of their work to reduce (or exploit) project risk" (Moran, 2014, p. 33). There is some disagreement as to whether explicit methods of risk management should be used in projects which are executed according to Agile methodology or if the implicit risk management built into Agile methodologies is sufficient (Walczak & Kuchta, 2013), some authors argue that this might depend on the context (Howell et al., 2010; MacCormack & Verganti, 2003).

Most software development projects are today using agile methods, including large projects with multiple development teams. The special issue on large-scale agile development in *IEEE Software* (Dingsøyr et al., 2019) describes a transition from first generation large-scale agile methods such as advice from project management frameworks like Prince2 combined with Scrum, to second generation frameworks such as the Disciplined Agile Delivery, Scaled Agile Framework and Large-Scale Scrum. These new frameworks add practices, roles and new artefacts to manage software development. An example of a program using a first generation large-scale agile development method, the "Perform" programme for the Norwegian State Pension Fund was organised in four main projects focusing on establishing business requirements, the technical architecture, development and test (Dingsøyr et al., 2018a). The rich description does not describe explicit practices for risk management, but the programme was organised into 12 releases, all development teams used three-week iterations and there were a number of arenas to ensure customer engagement and internal programme coordination.

Prior studies on uncertainty management in agile software development have focused on several types of uncertainty as discussed above (Dönmez & Grote, 2015). But we also find studies which focus on specific areas such as knowledge sharing, uncertainty in assessing value and cost and management of technical risks: To manage knowledge sharing risks in agile development, Ghobadi and Mathiassen (2017) have developed a model with seven risk areas such as "team diversity" and "project technology" and five areas of resolution strategies such as "strengthen resources" and "improve processes". This model is based on studies of four single-team projects. A practically oriented method to assess value and cost in agile software development projects is described in a magazine article by Hannay et al.

(2019). The method is described as suitable for large projects but does not focus on organisational layers. We also find studies which suggest models on how to handle technical risks as expressed in the software architecture by Nord et al. (2014).

The second-generation frameworks for agile development introduce a number of new practices, roles and artefacts to manage software development. With several teams working on the same project, new risks are introduced and advice on how to handle risks in single-team development may no longer be sufficient. We have tried to search literature for research on risk management at multiple layers i.e., how risks are managed at project level, release level, team level. Despite the fact that risk management has been researched for many decades, this specific topic seems to have been neglected, especially in projects using agile approaches. Nkukwana and Terblanche observed that "implementation teams value the governance role that project managers fulfil on agile projects, particularly with regards to project delivery, risk management, reporting and budgeting" (Nkukwana & Terblanche, 2017, p.6) but they have not tried to assess how this is done in practice.

How risks are managed implicitly and explicitly at multiple levels of agile projects has not been extensively studied (Nelson, 2008) and there is a need to investigate how risk management can be used in large agile projects (Odzaly & Des Greer, 2014). This is the objective of this exploratory study which investigates the following research question: *How does a large software/hardware development project using agile practices manage uncertainty at project/subproject and work package levels?* We believe this study of a first-generation agile development project will be important for researchers addressing uncertainty management in second-generation frameworks.

4.2 Method

We conducted an exploratory case study (Runeson & Höst, 2009; Yin, 2009) of a "very large" development project (Dingsøyr et al., 2014) of a new product, the Joint Strike Missile project at Kongsberg Defence & Aerospace. The case was selected as a project which combined traditional project management with agile development methods. The company had a traditional internal risk and opportunity process, and should be considered a "typical" project with respect to managing uncertainty. The case involved both hardware and software development. We chose a single-case study design in order to get a thorough understanding, as we could not find rich case descriptions on this topic in the literature and we were uncertain about the volume and range of practices employed. Single-case studies allow for generalisation of findings (Flyvbjerg, 2006) and can provide significant contributions to scientific development.

A survey distributed by the company amongst participants in the project indicated that uncertainty management was an area with potential for improved practices, as the score on the question "the project has plans for uncertainty management" was particularly low compared to other questions on project management. The survey was completed by 60 project participants and was followed up with a half-day workshop on uncertainty management where researchers provided recommendations from software engineering and project management literature. Data collection was done when the project was near completion of the third of four phases. We were granted access to interview 11 participants (see Table 4.1), three subproject managers, the assistant project manager (with responsibility for uncertainty management) – we refer to these in the following as managers. Further, four work package leaders from three subprojects (referred to as work package leaders) and finally three employees working in work packages in three subprojects (work package team members). This choice of informants, though a small number, allowed us to get diverse opinions on practices at the two levels.

Table 4.1: Overview of informants for study.

Level	Assistant project manager	Subproject manager	Work package leader	Work package team member
Project	1			
Subproject1		1	1	1
Subproject2		1	2	1
Subproject3		1	1	1
Total	**1**	**3**	**4**	**3**

The semi-structured interviews were conducted by the first author. We used the interview guide in Appendix 1 to get informants to describe their background, approaches to uncertainty management, and practices used in agile development. We also focused specifically on how they had managed opportunities so far in the project, as prior research has indicated a focus on threats. The interviews lasted from 30 to 45 minutes, a total of 120 pages of text were transcribed and sent back to informants for validation. We also included a quantitative second part, where we asked interviewees to indicate current and future importance of risk factors in the Ropponen and Lyytinen (2000) model.

Interview material was imported into NVivo for qualitative analysis, where the material was coded into groups that described explicit and implicit practices for uncertainty management at different levels of the project (see Table 4.2 for resulting categories). The project was located at two development sites, most subprojects were conducted at the main site, but some subprojects have work packages conducted at

Table 4.2: Mitigating practices to manage uncertainty at project and work package levels; divided into explicit practices and implicit practices.

	Explicit		Implicit	
	Common	**Specific**	**Common**	**Specific**
Project level/ Subproject	ABCD-reports Burndown chart Issue board Risk matrix	Progress meeting with customer Risk register Subproject meetings Top Five Risks	Ad-hoc handling of risks Code Review Estimation Integrations Project plan Early testing Technical debt Tasks to other subprojects	Task prioritization
Work package level		None identified		Problem analysis using A3s No Gold plating Pair design Retrospective

the other site which meant that possibilities for informal coordination was limited. Subprojects 1 and 2 were at the main site, while the work package leader and team members in Subproject 3 were from the other site.

As the interviews were conducted in Norwegian, the analysis was performed by the first author of this article with help from one researcher (see acknowledgement). The second author contributed in writing the theory section and discussing findings and contrasting this with findings from prior literature. The discussion was developed in a series of phone meetings. Results from the analysis were presented to some of the informants for validation.

4.3 Results

4.3.1 Case description

The project developed a product which consists of hardware and software components and the development has taken more than ten years and for a long period involved about 200 engineers. Main subprojects were for hardware, for software, and for integration and testing. The project had more than 50 subcontractors. The product had strict non-functional requirements for performance, security and safety, and a set of functional requirements which were mainly known in early phases of development. Most requirements were defined early in the project and a contract with the client was written based on these requirements. The project was managed as a traditional project using a V-model but followed agile development methods primarily for the software components (a first-generation large-scale agile project). The project was divided into six subprojects, with up to six work packages. The work packages had a work package leader and a team. Some of the software teams used the Scrum development process with two-week iterations, starting with iteration planning and ending with an iteration review. There were numerous dependencies between the subprojects and these dependencies were both technical and organisational.

4.3.2 Project risks

When asked to assess risks after the framework suggested by Ropponen and Lyytinen (2000), the informants rated “scheduling and timing risks” and “system functionality risks” as the two most important. One respondent stated, “the last two years, we have been pressed hard on time and cost . . . it has been a large focus on just implementing what must be implemented” (work package team member). Another respondent was asked what was most important and stated that it was “functionality and performance . . . but it is always connected to schedule”.

Risks are mainly interpreted as threats, but sometimes also opportunities.

> *We have tried to talk about threats and opportunities and uncertainty, but we notice that we easily fall back on talking about threats. (Manager)*

However, some state that the reporting structure of the project focuses on reporting benefits from work package level and up (see more on ABCD-reporting later), and people report things like:

> *this is an opportunity – here we have a chance to work more efficiently. (Work package manager)*

4.3.3 Explicit and implicit practices at different levels of the project

We show the mitigation practices to manage risks at the project/subproject level and show the practices we characterise as explicit and implicit in Table 4.2. The table also shows which practices were common to both project/subproject and work package levels, and which were specific. In the following sections, we provide descriptions of each of these practices.

4.3.4 Explicit uncertainty mitigation

4.3.4.1 Common

ABCD-reports: Every second week the project managers wrote a report, called the “ABCD-report”, indicating what had been **A**chieved, resulting **B**enefits, **C**oncerns and what they planned to **D**o next. The subproject managers received the ABCD reports from their work package leaders, and risks were reported as concerns. It was possible to indicate at what level a risk should be handled. Another option was to present issues requiring concrete actions directly to the project manager in project meetings.

Burndown charts: These were used at subproject and work package levels. At work package level, there were multiple variants of burndown charts – as a board, in different software tools which changed during the course of the project.

Issue boards: Issue boards were used to register and follow up risks, bugs, problems and improvement suggestions at subproject and work package levels. In a work package, one informant stated:

> *this is something we use for a number of purposes [. . .] we meet once a week and discuss what has been registered. (Work package manager)*

Risk matrix: The risk matrix was established early at project and subproject levels. The matrix was updated before progress meetings described below, but also on a need-be basis if something came up. Although it was not mandated by the project managers, some work package leaders discussed risks at work package level and would bring them into the risk matrix at subproject level. An informant stated that:

> *work package leaders are clearly involved in developing the risk matrix at subproject level. (Work package team member)*

4.3.4.2 Specific for project/subproject level

Progress meeting: Three to four times a year, the project had a meeting on progress with the customer. They walked through the risk matrix and risk register as preparation for the meeting. This was done on project level, based on preparations in all subprojects where subproject managers assessed their risk registers.

Risk register: The program established a risk register in the beginning of the program by brainstorm meetings at subproject level. The items in the register were given a probability and a consequence. A person was given responsibility of mitigating actions. Status of actions were discussed with the customer at the progress meetings.

Subproject meetings: Risks were discussed in subproject meetings, where the subproject managers met with work package managers. This was based on discussions between work package team leaders and team members, not necessarily in a separate meeting on this topic.

Top Five Risks: Every subproject manager brought a list of “top five” internal risks to a project meeting, and the management discussed the risks and developed a list of the project “top five”. This list was communicated internally through the project Intranet. In our interviews, this was only mentioned by people in management positions, the awareness of this list might be low in the project organisation.

4.3.4.3 Specific to work packages

From our interview material, we could not identify explicit practices that were specific to work packages. There were no obligations for explicit practices at this level.

4.3.5 Implicit uncertainty mitigation

4.3.5.1 Common

Ad-hoc handling of risks: Several informants said risks were handled ad-hoc as they were discovered.

> *What can we do when things stop? . . . we find solutions to ensure that we deliver as planned (Manager).*

Another informant stated that at work package level,

> *if something shows up, we call for a meeting right away, but there is not necessarily any process or description of it. (Work package manager)*

Code Review: Many said they did multiple code reviews both in the work packages and in subprojects. Sometimes code was reviewed by several peers if it was a large or complicated part. If it was a minor change then just one person might review it. One work package had a practice of having at least two people reviewing the code, as code quality was particularly critical in this part of the project.

Estimation: Because of the complexity and innovations of the tasks, accurate estimations were one of the major challenges throughout the project. The work tended to be underestimated, in particular the work regarding integration of hardware and software components. At work package level, tasks were individually estimated in man-hours. In one work package, they tried group estimates, but it was not a practice that they maintained.

Integrations: The project integrated deliverables from all subprojects at certain intervals, but because some deliverables include hardware, the increments could be very large, in the order of one to two years of work (while many of the software work packages used two-week iterations).

> *This means that groups that consist of system engineers, software architects and software developers work on the same functions, but not at the same time. . . . when we pick up a topic, the system engineers have forgotten what they talked and thought about. (Software developer)*

Although many expressed that integrations were too infrequent, they described integrations as a major factor to reduce risks:

> *because you go through the whole process over a short period and can find uncertainties in the whole V. (Work package team member)*

Some said that it had been difficult to integrate at the planned milestone dates, because of the difficulty to synchronize parts from so many people at an agreed milestone:

> *We are in a situation now, where we are depending on a delivery from another work package, they do not know exactly when they can deliver the first version, and we know there will be errors. So, we need to test, which will take time, and we are unsure about how long. Then someone else is waiting at the other end to make use of the delivery. (Work package manager)*

Project plan: Some risks were identified during planning, for example when the project team needed to commit to dates and milestones.

Early Testing: Taking up agile practices such as early integration led to much earlier testing. In previous projects testing was only done at the end of the project. The early testing led to a reduction of risks.

Technical debt: Previous studies on risk management in agile projects identified technical debt as one important concern. In our case, the informants expressed a varying degree of awareness on technical debt. One said that he was not aware of the term, but has experienced that

> *we cannot always perform tasks the way we want to, there is not enough time for that. (Work package team member)*

Other informants reported that they had frequent discussions on technical debt and registered all debt. The debt was then regularly assessed. The technical debt had so far not been shown in the risk matrix, they talk about technical debt, but had not identified it as risk.

Tasks to other subprojects: Tasks were regularly assigned from one work package to another and they tried to make dependencies visible. In particular this was important when limited resources could be assigned to certain tasks, especially if they were considered high priority tasks.

4.3.5.2 Specific: Project/subproject level

Task prioritization: Some stated that the project was in a better position to manage risks compared to previous projects due to the uptake of agile methods and frequent re-prioritization and focus on work tasks. According to the project manager: "The method makes you have more focus". A subproject manager explained that they had influence on prioritisation in other subprojects:

> *we have worked with getting them to understand what is important to us. (Manager)*

4.3.5.3 Specific: Work package level

Problem analysis using A3s: A technique borrowed from lean manufacturing used to document a problem on an A3 sheet, including: Background, what has been done regarding the problem and a conclusion. This practice was used by the work package teams in this phase of the project. These were stored to give background if a problem reoccurred.

No gold plating: Most informants expressed that over-investing in quality or functionality was not a problem in the project.

> *The last two years, we have been pressed hard on time and costs, so towards the end of the project there has been a strict focus on just implementing what needs to be implemented. (Work package team member)*

The work package leader would be informed of the team's activities and ensured that activities were within scope. According to a work package member:

> *we focus on being finished rather than the "nice to have"-stuff. (Work package team member)*

Pair design: Both pair design and pair programming have been used to some extent, but there was no project-wide policy on the use of these techniques. Pair programming allows code review to be done while writing the code, which is believed to increase the code quality. Also, discussions while doing development could lead to more optimal design decisions which could reduce risks related to code quality.

Retrospective: Some of the work packages conducted regular retrospectives, while others did not. Some that did retrospectives combined this with a planning meeting where most of the time was spent thinking ahead, and little on discussing lessons learned during the last iteration. Retrospectives could be used to discuss estimation precision, but our informants did not express such use of these meetings.

4.4 Discussion

We return to our research question, *how does a large software/hardware development project using agile practices manage uncertainty at project/subproject and work package levels?*

The case studied was a large project developing a product consisting of hardware and software components. The software engineering field has experienced many changes in recent years due to agile development methods which bring in new practices and new terminology such as «technical debt». The project has taken up agile methods through more frequent integration of components than in previous projects as well as in using agile development methods mainly at work package level. We described the case project as a first-generation large-scale agile development project.

The results show a total of 21 uncertainty mitigation practices, where we have classified eight as explicit and 13 as implicit. Further, five practices are specific to the project and subproject levels (four explicit and one implicit), while four are specific to the work package level (all implicit). This finding echoes findings on coordination in large-scale agile development, where studies show that there are far more practices in use in successful projects than are described in recommendations in existing frameworks (Dingsøyr et al., 2018b).

Although there were a large number of practices, their use varied much between work packages and between subprojects. For example, estimation techniques are used to varying degrees, the awareness of «technical debt» was strong in some work packages while others were unaware of the term, and the agile practice of retrospectives was used to varying degrees. Some of the explicit practices such as the risk matrix and risk register were used across the subprojects and also at some work packages. Our study is to our knowledge the first to identify practices at different organisational layers in projects. We also identified a larger variety of practices than can be seen in empirical studies of small agile development projects (Dönmez & Grote, 2018; Elbanna & Sarker, 2016; Siddique & Hussein, 2016) or from studies of uncertainty management in general.

An interesting observation was that at work package level there were no specific explicit practices. All explicit practices had been deployed at project level and most of these techniques required some flow up or down between the hierarchy of the project. This was different for the implicit practices where multiple specific techniques were used at the work package level. The specific implicit practices at work package level were primarily used to manage “system functionality” risks (Ropponen & Lyytinen, 2000) and consisted of practices from agile development such as retrospectives and pair programming and from lean production with the use of A3s to document problems and solutions. In contrast, the risk management techniques used at project level were focused mainly on “scheduling and timing risks” (Ropponen & Lyytinen, 2000).

The interviewees did not make distinctions between risk and uncertainty management. However, going back to the definitions presented in the literature review section, it became clear during the data analysis that the project was focusing mainly on risks (i.e., possible events which might impact the project) using traditional explicit risk management techniques deployed throughout the project. As in previous findings, this program focused their risk management primarily on threats and not on opportunities. In comparison, the work packages were focusing primarily on implicit techniques for uncertainty management (i.e., their inability to determine the scope and the task durations precisely for the whole project).

Previous studies of risk management in agile development have identified challenges with separation of development and operations and a growing technical debt (Elbanna & Sarker, 2016). The separation of development and operations was not directly addressed in the mitigation practices identified in our study, while some brought this topic up in interviews. Technical debt was a topic with varying practices across work packages. For large projects, a crucial topic identified in previous studies has been coordination between different teams. This was mainly done through a traditional project organisation in our case, and for example not through meetings such as Scrum of Scrums. However, the informants stated that "scheduling and timing" were the main risks in the project, and this was much related to coordination between teams. A practice to manage this type of uncertainty was to give "tasks to other subprojects".

4.5 Conclusion

This study shows how uncertainty is managed in a large project with several subprojects, using a combination of practices from project management and from agile development methods. In line with previous findings, we found that uncertainty management practices were mainly focusing on handling threats, and to a smaller degree on opportunities.

We are not aware of previous studies describing how uncertainty is managed at the different levels in large projects. The main contribution of this study is that our case shows the high number of practices in use, some are used on all levels, while others are used only the project/subproject level or at the work package level.

The use of many of the practices vary, for example some work packages were very conscious on registering technical debt, while others did not use this term. The specific practices at project/subproject level are mainly practices that explicitly handle uncertainty while the practices at work package level are practices that implicitly handle uncertainty. Further, some of the practices link between the layers in the project, such as the ABCD reports, the issue boards and the registration of technical debt.

With the increasing importance of software in many large projects, we believe more projects will have to manage layers of risks with a wide range of practices in the future. This exploratory study highlights characteristics of uncertainty mitigation in first generation large-scale agile projects that blend "traditional" project management and agile methods.

4.6 Limitations

The major limitation of this study is that it is an exploratory study with a limited data collection. The case was selected as a “typical” case of a first-generation large-scale agile development project. An “extreme” case with more effort needed on uncertainty management might have opened up other topics for further investigation.

The interviews were done during the third of four project phases, rather than longitudinally over the life of the project. Given the type of study, with semi-structured interviews of informants on how uncertainty was managed, we do not have detailed information on use of all practices identified.

4.7 Implications for theory and practice

We have presented findings from a case study of a first-generation large-scale agile development project and what practices were in use for uncertainty management at two levels. This study gives important directions when attention is given to second generation frameworks which are increasingly used in the software industry. In particular, we would like to highlight five directions:

First, a number of challenges have been identified in large-scale agile development (see for example (Bass, 2019)) which includes topics such as interteam coordination, technical architecture and assigning priority to user needs. How do the resolution strategies suggested in the various second-generation large-scale agile methods work in practice?

Second, an implicit practice to manage uncertainty in agile development is to assign work to teams. A team is more robust than individuals but studies of large-scale development suggests that teamwork in this context is different from teamwork in single-team settings (Lindsjørn et al., 2018). How can second-generation frameworks facilitate good teamwork while managing project uncertainty?

Third, the technical architecture is a major source of uncertainty. Prior studies of first generation large-scale agile development suggests that the architecture has been stable (Dingsøyr et al., 2018a) while advice is to work more iteratively also with architecture (Nord et al., 2014). How do projects address this type of uncertainty when using second generation methods?

Fourth, large-scale agile development projects are organised as multi-team projects, often with a structure as in our case with subprojects and work package teams. What alternative linkage strategies exist, and how are these strategies used in practice?

Finally, we think there is a need to revise existing published risk frameworks such as (Ropponen & Lyytinen, 2000) in order to fit needs in large projects both for first- and second-generation large-scale agile development methods.

Although this is an exploratory study primarily intended to generate directions for new research, we believe the description of uncertainty management practices in the case has several implications for practice:

First, the case study shows a large number of explicit and implicit practices in use to manage uncertainty. Prior studies on coordination in large-scale agile development projects have also found a large number of arenas for coordination. It seems reasonable that large projects will need a large number of practices to manage the diverse range of uncertainty during project execution.

Second, we believe the list of practices identified in Table 4.2 can be valuable when planning new large projects or seeking inspiration to manage certain types of uncertainty during project execution.

Acknowledgements: This article was written with partial support from the Agile 2.0 project, supported by the Research Council of Norway through grant 236759 and companies Equinor, DNV GL, Kantega, Kongsberg Defence & Aerospace, Sopra Steria and Sticos. We are very grateful to Andreas Bredesen and Elin Lintvedt at Kongsberg Defence & Aerospace for administering a project-wide survey and setting up individual interviews. Further, we would like to thank Finn Olav Bjørnson at the Department of Computer and Information Science at the Norwegian University of Science and Technology for helping with the analysis of the interview material. A prior version of this paper was presented at the IRNOP2018 conference, we are grateful to comments from conference and book reviewers and to conference participants for discussions. We would also like to thank Agnar Johansen at the Norwegian University of Science and Technology for discussions on study design and input on the interview guide. We are further grateful to Amany Elbanna at the University of London for comments on the interview guide and discussion of findings.

Appendix 1: Interview guide

Will use "uncertainty management" as managing *"uncertain event or conditions that, if they occur, has a positive or negative effect on one or more project objectives such as scope schedule, cost and quality" (PMI Body of Knowledge).*

Part 1

Background

1. Could you describe your role in the project?
2. Could you describe the work method in your part of the project?
3. Are your tasks affected by other work packages or subprojects? How?
4. Do you work in a distributed team? Hardware or software?

Uncertainty management

[Risks]

5. Do you identify risks to project progress in your part of the project? What hinders project progress in your part? Are there particular challenges you are facing?
6. How do you work to identify risks? (practices? tools?) what do you normally do about them (issues/ challenges/ problems)?
7. Are you aware of (implicit) practices that reduce risks to project progress? Is there something done on the work package/subproject/project level about them?
8. How are risks communicated in the project organization? Do you think your technical lead or PM is aware of this? Are they on denial or can/do something about it?
9. How do you work to identify opportunities? (practices? tools?)

[Questions about identified risks in agile projects]

10. Would you say that the product has technical debt? How much debt you think the team accumulates? Do you think there is a way to pay part of this debt back during the development? Do you think some could be paid back after the project? How?
11. Do you have established approached to manage technical debt? Do you keep a log or any other way to keep track of these debts?
12. Are there challenges in handing over products from development to operations/maintenance?
13. What PM tools are you using? Is this available for all teams in the organisation? How are project management tools used in the project? Alignment?
14. Is knowledge regarding work tasks preserved for later maintenance? Have you been involved in system upgrade or maintenance tasks?

[Opportunities]

15. Are you aware of (implicit) practices to identify opportunities in the project?
16. How is this handled?
17. How are opportunities communicated in the project organization?

18. Do you estimate the effort involved in work tasks? How? (practices? tools?)
19. How is progress on work tasks communicated to other parts of the project?
20. Do you see potential for improvement for managing
    * Risks?
    * Opportunities?

Part 2

Questionnaire: Project risks

Please indicate how important the following risks are in your current project. Use the following scale:
1 – Not important
2 – Slightly important
3 – Moderately important
4 – Important
5 – Very important

Please indicate how important you think the following risks will be in future similar projects. Use the following scale:
1 – Not important
2 – Slightly important
3 – Moderately important
4 – Important
5 – Very important

Risks to be assessed (from Ropponen and Lyytinen (2000)):

Scheduling and timing risks
System functionality risks
Subcontracting
Requirement management risks
Resource usage and performance risks
Personnel management risks

References

Ahsan, M., Haried, P., & Musteen, M. (2010). Understanding the Relationship between Uncertainty and International Information Technology Sourcing Strategy: A Conceptual Framework. *Academy of Information and Management Sciences Journal*, *13*(2), 1–24.

Andersen, E. S. (2008). *Rethinking Project Management: An Organisational Perspective*. Harlow (UK): Prentice Hall/Financial Times.

Association for Project Management. (2008). *Prioritising Project Risks*. Cambridge (UK): APM.

Bannerman, P. L. (2008). Risk and Risk Management in Software Projects: A Reassessment. *Journal of Systems and Software*, *81*, 2118–2133.

Bass, J. M. (2019). Future Trends in Agile at Scale: A Summary of the 7th International Workshop on Large-Scale Agile Development. Paper presented at the XP2019, Montreal, Canada.

Bick, S., Spohrer, K., Hoda, R., Scheerer, A., & Heinzl, A. (2018). Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings. *IEEE Transactions on Software Engineering*, *44*(10), 932–950. doi:10.1109/TSE.2017.2730870

Butler, C. W., Vijayasarathy, L. R., & Roberts, N. (2019). Managing Software Development Projects for Success: Aligning Plan-and Agility-Based Approaches to Project Complexity and Project Dynamism. *Project Management Journal*, 8756972819848251.

Cagliano, A. C. (2015). Choosing project risk management techniques. A theoretical framework. *Journal of risk research*, *18*(2), 1–17.

Chapman, C. B., & Ward, S. (2003). *Project Risk Management: Processes, Techniques, and Insights* (Second ed.). New York: Wiley.

Cleden, D. (2009). *Managing Project Uncertainty*. Farnham (UK): Gower.

Conboy, K., & Carroll, N. (2019). Implementing Large-Scale Agile Frameworks: Challenges and Recommendations. *IEEE Software*, *36*(2), 44–50. doi:10.1109/MS.2018.2884865

De Meyer, A., Loch, C. H., & Pich, M. T. (2002). Managing Project Uncertainty: From Variation to Chaos. *MIT Sloan Management Review*, *43*(2), 59–67.

Dey, P. K., Kinch, J., & Ogunlana, S. O. (2007). Managing risk in software development projects: a case study. *Industrial Management + Data Systems*, *107*(2).

Dingsøyr, T., Fægri, T., & Itkonen, J. (2014). What Is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development. In A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Männistö, J. Münch, & M. Raatikainen (Eds.), *Product-Focused Software Process Improvement* (Vol. 8892, pp. 273–276): Springer International Publishing.

Dingsøyr, T., Falessi, D., & Power, K. (2019). Agile Development at Scale: The Next Frontier. *IEEE Software*, *36*(2), 30–38. doi: 10.1109/MS.2018.2884884

Dingsøyr, T., Moe, N. B., Fægri, T. E., & Seim, E. A. (2018a). Exploring Software Development at the Very Large-Scale: A Revelatory Case Study and Research Agenda for Agile Method Adaptation. *Empirical Software Engineering*, *23*(1), 490–520. doi: https://doi.org/10.1007/s10664-017-9524-2

Dingsøyr, T., Moe, N. B., & Seim, E. A. (2018b). Coordinating Knowledge Work in Multi-Team Programs: Findings from a Large-Scale Agile Development Program. *Project Management Journal*, *49*(6), 64–77. doi:10.1177/8756972818798980

Dinsmore, P. C., & Cabanis-Brewin, J. (2014). *The AMA Handbook of Project Management* (Fourth ed.). New York: AMACOM.

Dönmez, D., & Grote, G. (2015). The Two Faces of Uncertainty: Threat vs Opportunity Management in Agile Software Development. In C. Lassenius, T. Dingsoyr, & M. Paasivaara (Eds.), *Agile Processes, in Software Engineering, and Extreme Programming, Xp 2015* (Vol. 212, pp. 193–198). Berlin: Springer-Verlag Berlin.

Dönmez, D., & Grote, G. (2018). Two sides of the same coin – how agile software development teams approach uncertainty as threats and opportunities. *Information and Software Technology*, *93*, 94–111. doi: 10.1016/j.infsof.2017.08.015

Duncan, R. B. (1972). Characteristics of Organizational Environments and Perceived Environmental Uncertainty. *Administrative Science Quarterly*, *17*(3), 313–327.

Elbanna, A., & Sarker, S. (2016). The Risks of Agile Software Development Learning from Adopters. *IEEE Software*, *33*(5), 72–79. doi:10.1109/ms.2015.150

Fitzgerald, B., Stol, K.-J., O'Sullivan, R., & O'Brien, D. (2013). *Scaling Agile Methods to Regulated Environments: An Industry Case Study*. Paper presented at the 2013 International Conference on Software Engineering.

Flyvbjerg, B. (2006). Five Misunderstandings about Case Study Research. *Qualitative Inquiry*, *12*(2), 219–245.

Flyvbjerg, B., & Budzier, A., 89(9), 23–25. (2011). Why Your IT Project May Be Riskier than You Think. *Harvard Business Review*, *89*(9), 23–25.

Ghobadi, S., & Mathiassen, L. (2017). Risks to Effective Knowledge Sharing in Agile Software Teams: A Model for Assessing and Mitigating Risks. *Information Systems Journal*, *27*(6), 699–731. doi: 10.1111/isj.12117

Gray, C. F., & Larson, E. W. (2014). *Project Management – The Managerial Process* (Sixth ed.). New York: McGraw-Hill.

Grewal, H., & Maurer, F. (2007). Scaling Agile Methodologies for Developing a Production Accounting System for the Oil & Gas Industry. Paper presented at the Agile Conference (AGILE), 2007.

Gruver, G., & Mouser, T. (2015). *Leading the Transformation: Applying Agile and DevOps Principles at Scale*. Portland (OR): IT revolution.

Hannay, J. E., Benestad, H. C., & Strand, K. (2019). Agile Uncertainty Assessment for Benefit Points and Story Points. *IEEE Software*, *36*(4), 50–62. doi: 10.1109/ms.2018.2875845

Hijazi, H., Khdour, T., & Alarabeyyat, A. (2012). A Review of Risk Management in Different Software Development Methodologies. *International Journal of Computer Applications*, *45*(7), 8–12.

Hobbs, B., & Petit, Y. (2017a). Agile Methods on Large Projects in Large Organizations. *Project Management Journal*, *48*(3), 3–19.

Hobbs, B., & Petit, Y. (2017b). *Agile Methods on Large Projects in Large Organizations*. Newton Square (PA): Project Management Institute.

Hossain, E., Babar, M. A., & Paik, H.-y. (2009, 13–16 July 2009). *Using Scrum in Global Software Development: A Systematic Literature Review*. Paper presented at the 2009 Fourth IEEE International Conference on Global Software Engineering.

Howell, D., Windahl, C., & Seidel, R. (2010). A Project Contingency Framework Based on Uncertainty and its Consequences. *International Journal of Project Management*, *28*(3), 256.

Ibrahim, H., Far, B. H., Eberlein, A., & Daradkeh, Y. (2009). *Uncertainty Management in Software Engineering: Past, Present, and Future*. Paper presented at the Electrical and Computer Engineering, 2009. CCECE'09. Canadian Conference on.

Jaafari, A. (2001). Management of Risks, Uncertainties and Opportunities on Projects: Time for a Fundamental Shift. *International Journal of Project Management*, *19*(2), 89–101.

Jalonen, H. (2011). The Uncertainty of Innovation: A Systematic Review of the Literature. *Journal of Management Research*, *4*(1).

Johansen, A. (2015). Project Uncertainty Management: A New Approach – The 'Lost Opportunities' Practical uncertainty management seen from a project joint perspective. (Ph.D.), NTNU, Trondheim.

Johansen, A., Olsson, N. O., Jergeas, G., & Rolstadås, A. (2019). *Project Risk and Opportunity Management: The Owner's Perspective*: Routledge.

Jørgensen, M. (2019). Relationships Between Project Size, Agile Practices, and Successful Software Development: Results and Analysis. *IEEE Software*, *36*(2), 39–43. doi: 10.1109/MS.2018.2884863

Kendrick, T. (2015). Identifying and Managing Project Risk – Essential Tools for Failure-Proofing Your Project (Third ed.). New York: AMACOM.

Kerzner, H. (2017). Project Management: A Systems Approach to Planning, Scheduling, and Controlling (Twelfth ed.). Hoboken (NJ): John Wiley & Sons.

Kraut, R. E., & Streeter, L. A. (1995). Coordination in software development. *Communications of the ACM*, *38*(3), 69–81.

Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical Debt: From Metaphor to Theory and Practice. *IEEE Software, November"December*, 18–21.

Leffingwell, D. (2015). SAFe – Scaled Agile Framework. Retrieved from http://www.scaledagileframework.com/

Lindsjørn, Y., Bergersen, G. R., Dingsøyr, T., & Sjøberg, D. I. K. (2018). *Teamwork Quality and Team Performance: Exploring Differences Between Small and Large Agile Projects.* Paper presented at the XP2018, Porto, Portugal.

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., May, J., Kahkonen, T. (2004). Agile Software Development in Large Organizations. *Computer*, *37*(12), 26–34.

Loch, C., De Meyer, A., & Pich, M. T. (2006). *Managing the Unknown: A New Approach to Managing High Uncertainty and Risk in Projects*. Hoboken (NJ): John Wiley & Sons.

MacCormack, A., & Verganti, R. (2003). Managing the Sources of Uncertainty: Matching Process and Context in Software Development. *Journal of Product Innovation Management*, *20*(3), 217–232.

McBride, T. (2008). The Mechanisms of Project Management of Software Development. *Journal of Systems & Software*, *81*, 2386–2395.

Moran, A. (2014). *Agile Risk Management*: Springer.

Nelson, C. R. (2008). Explicit Risk Management in Agile Processes. In P. Abrahmasson, R. Baskerville, K. Conboy, B. Fitzgerald, L. Morgan, & X. Wang (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (pp. 190–201). Berlin: Springer-Verlag.

Nicholas, J. M. (2017). Project Management for Business and Engineering: Principles and Practice (Fifth ed.). Amsterdam: Elsevier.

Nkukwana, S., & Terblanche, N. H. D. (2017). Between a Rock and a Hard Place: Management and Implementation Teams' Expectations of Project Managers in an Agile Information Systems Delivery Environment. *South African Journal of Information Management*, *19*(1).

Nord, R. L., Ozkaya, I., & Kruchten, P. (2014). Agile in Distress: Architecture to the Rescue. In T. Dingsøyr, N. B. Moe, R. Tonelli, S. Counsell, C. Gencel, & K. Petersen (Eds.), *Agile Methods: Large-Scale Development, Refactoring, Testing, and Estimation* (Vol. 199, pp. 43–57).

Nyfjord, J., & Kajko-Mattsson, M. (2007). *Commonalities in risk management and agile process models*. Paper presented at the Software Engineering Advances, 2007. ICSEA 2007. International Conference on.

Odzaly, E. E., & Des Greer, D. S. (2014). *Lightweight Risk Management in Agile Projects*. Paper presented at the 26th Software Engineering Knowledge Engineering Conference, Vancouver, Canada.

Perminova, O., Gustafsson, M., & Wikström, K. (2007). Managing Uncertainty in Projects: Merging Theory and Practice. *Project Perspectives*, *29*, 65–67.

Perminova, O., Gustafsson, M., & Wikström, K. (2008). Defining Uncertainty in Projects: A New Perspective. *International Journal of Project Management*, *26*(1), 73–79.

Persson, J. S., Mathiassen, L., Boeg, J., Madsen, T. S., & Steinson, F. (2009). Managing Risks in Distributed Software Projects: An Integrative Framework. *IEEE Transactions on Engineering Management*, *56*(3), 508–532.

Pich, M. T., Loch, C. H., & Meyer, A. D. (2002). On Uncertainty, Ambiguity, and Complexity in Project Management. *Management Science*, *48*(8), 1008–1023.

Platje, A., & Seidel, H. (1993). Breakthrough in Multiproject Management: How to Escape the Vicious Circle of Planning and Control. *International Journal of Project Management*, *11*(4), 209–213.

Project Management Institute. (2017). *A Guide to the Project Management Body of Knowledge (PMBOK Guide)* (Sixth ed.). Newtown Square (PA): Project Management Institute.

Project Management Institute. (2019). *The Standard for Risk Management in Portfolios, Programs, and Projects*. Newtown Square (PA).

Raz, T., Shenhar, A. J., & Dvir, D. (2002). Risk Management, Project Success, and Technological Uncertainty. *R&D Management*, *32*(2), 101–109.

Ropponen, J., & Lyytinen, K. (2000). Components of Software Development Risk: How to Address Them? A Project Manager Survey. *IEEE transactions on software engineering*, *26*(2), 98–112.

Runeson, P., & Höst, M. (2009). Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering*, *14*(2), 131.

Sabherwal, R. (2003). The evolution of coordination in outsourced software development projects: a comparison of client and vendor perspectives. *Information and organization*, *13*(3), 153–202.

Saunders, F. C. (2015a). Conceptualising uncertainty in safety-critical projects: A practitioner perspective. *International Journal of Project Management*, *33*(2), 467–478.

Saunders, F. C. (2015b). Toward High Reliability Project Organizing in Safety-Critical Projects. *Project Management Journal*, *46*(3), 25–35. doi:10.1002/pmj.21498

Saunders, F. C., Gale, A. W., & Sherry, A. H. (2016). Responding to project uncertainty: Evidence for high reliability practices in large-scale safety–critical projects. *International Journal of Project Management 34 (2016) 34*, 1252–1265.

Schwaber, K. (2004). *Agile Project Management with Scrum* Redmond (WA): Microsoft Press.

Schwaber, K., & Sutherland, J. (2013). The Scrum Guide™ – The Definitive Guide to Scrum: The Rules of the Game.

Siddique, L., & Hussein, B. A. (2014, 12–15 June 2014). *Practical insight about risk management process in agile software projects in Norway*. Paper presented at the 2014 IEEE International Technology Management Conference.

Siddique, L., & Hussein, B. A. (2016). Managing Risks in Norwegian Agile Software Projects: Project Managers´ Perspective. *International Journal of Engineering Trends and Technology*, *41*(2), 56–65.

Sommerville, I. (2016). *Software Engineering* (Tenth edition ed.). Harlow, England: Pearson Education Limited.

Tuunanen, T., Vartiainen, T., Ebrahim, M., & Liang, M. (2015). *Continuous Requirements Risk Profiling in Information Systems Development*. Paper presented at the 48th International Conference on System Sciences, Hawaii.

Vaidya, A. (2014). *Does DAD Know Best, Is it Better to do LeSS or Just be SAFe? Adapting Scaling Agile Practices into the Enterprise*. Paper presented at the Pacific Northwest Software Quality Conference (PNSQC), Portland (OR).

Walczak, W., & Kuchta, D. (2013). Risks Characteristic of Agile Project Management Methodologies and Responses to Them. *Operations Research & Decisions*, *23*(4), 75–95. doi:10.5277/ord130406

Ward, S., & Chapman, C. B. (2003). Transforming Project Risk Management into Project Uncertainty Management. *International Journal of Project Management*, *21*(2), 97–105.

Yin, R. K. (2009). *Case Study Research: Design and Methods* (4th ed.). Los Angeles (CA): Sage.

Mirko Perkusich, Ademar Sousa Neto, Joao Nunes,
Kyller Gorgônio, Hyggo Almeida and Angelo Perkusich

5 A knowledge-based risk management approach for Scrum software development projects

5.1 Introduction

Within the last few years, Agile Software Development (ASD) has become the mainstream software development paradigm (Hoda et al., 2018; Campanelli & Parreiras, 2015) It consists of a change-driven approach to developing software in the context of volatile requirements (Hoda et al., 2017). Contrasting with traditional plan-based project management methods, ASD focuses on people, is oriented to face-to-face communication, flexible, fast, light, responsive, and driven for continuous learning and improvement (Fontana et al., 2014). ASD has its values and principles stated in the Agile Manifesto (Fowler et al., 2001), which are manifested by agile methods. The literature presents several agile methods. Some focus on project management such as Scrum (Sutherland & Schwaber, 2017); others on software development such as Extreme Programming (XP) (Beck & Gamma, 2000); and others on both such as Crystal Clear (Cockburn, 2002) and Feature-Driven Development (FDD) (Palmer & Felsing, 2001). Scrum is the most popular agile method (VersionOne, 2019), and, given this, it is the focus of this work. Scrum is a framework to manage work on complex products (i.e., having high uncertainty regarding the requirements and solution) composed of events, artifacts, and roles (Schwaber, 2004).

Regarding risks, Scrum manages them empirically. It does not consider an explicit risk log to manage them, which is one of the key project management processes (Raz & Michael, 2001; Chapman & Ward, 1996). Despite this, the empirical nature of Scrum, having short feedback loops, assists in controlling risk. Further, Scrum relies mainly on tacit knowledge. This characteristic might limit the reuse of information for risk management throughout the organization. As a result, this might hinder organizational learning and maturity because the knowledge is highly coupled to the people, and not to processes. A solution is to use knowledge management practices to make knowledge regarding risk management explicit and creating a corporate risk memory (or database). Such memory is continuously updated throughout the execution of projects. Further, it enables learning-based risk management in which project managers can use the knowledge in memory to assist in their decision making.

Several researchers have explored using knowledge to support risk management. Dikmen et al. (2008) presented a structured process to support learning based risk

https://doi.org/10.1515/9783110652321-005

management for construction projects. Its assumption is that using a generic list of potential risk sources can help with risk identification, making knowledge generalized, and enabling cross-project learning. Alhawari et al. (2012) combined the fields of knowledge management and risk management within a conceptual framework to support Information Technology projects. Currie (2003) presented a knowledge-based risk assessment framework for web enabled application outsourcing projects. According to our knowledge, there is no learning-based risk management approach tailored for agile software development projects, in particular, focusing on Scrum.

Furthermore, with the recent revolution on data analytics and artificial intelligence (Perkusich et al., 2020), there is an opportunity to improve the state-of-the-art on managing risks on Scrum projects by combining learning-based risk management with such technologies. Such an endeavor can promote organizational maturity by enabling the reuse of risk-related knowledge and assisting in decision-making.

To explore this opportunity, we developed a knowledge-based risk management framework for Scrum software development projects. Ward et al. (2003) discussed that viewing risk management as uncertainty management enhances the focus on opportunity management, therefore, bringing balance on focusing on both types of risks (i.e., positive and negative). A popularly used technique for assisting on managing risk by treating uncertainty is Bayesian networks (Ancveire et al., 2015; Hu et al., 2013; Okutan & Yildiz, 2014; Chin et al., 2009; Lee et al. 2009; Fenton et al. 2008; Fenton & Neil, 2012). Positive aspects regarding Bayesian networks for risk management are its suitability for causal and diagnostic reasoning, suitability for decision making, and clarify of inference (Verbert et al., 2017). Thus, the proposed approach relies on modeling the main aspects of the Scrum product delivery process with a causal model, more specifically, a Bayesian network.

This chapter describes the proposed framework, focusing on its main components. Further, it describes an overview of the underlying Bayesian network, which is used as the basis for managing risks within the proposed framework. It is not within the scope of this paper to present details about the Bayesian networks. For more details, refer to Perkusich et al. (2017) and Perkusich et al. (2015). This chapter is organized as follows. Section 5.2 synthesizes background knowledge on Scrum (Section 5.2.1) and Bayesian networks (Section 5.2.2). Section 5.3 presents the proposed solution. Section 5.4 presents our final remarks and future works.

5.2 Background

Scrum is an iterative and incremental process to optimize the ability to foresee and control risks. Scrum's process is sustained by three pillars: transparency, inspection, and adaptation (Griffiths, 2012). At the end of each iteration, called *sprint*, the development team delivers a functional product increment to be verified and validated by the other stakeholders. Essential aspects of the process, such as acceptance criteria, must be visible to all stakeholders. At the end of each sprint, the stakeholders inspect the project's progress and adapt the product, if necessary. We present an overview of the Scrum process in Figure 5.1.

Scrum describes five events: the Sprint and four ceremonies, namely, the Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective (Sutherland & Schwaber, 2017). The sprints are two to four-week iterations in which a potentially releasable product increment is delivered. The Sprint Planning is a meeting that occurs at the beginning of each sprint and aims to define their goal and execution plan. The Daily Scrums are meetings to inspect the progress of the sprint and synchronize the team's work to mitigate risks. An example of questions to be answered by each member during the Daily Scrum is presented in what follows:

– What did I do yesterday that helped the Development Team meet the Sprint Goal?
– What will I do today to help the Development Team meet the Sprint Goal?
– Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?

The Sprint Review is a meeting that occurs at the end of each sprint to inspect the delivered product increment and, if necessary, to adapt the product backlog to changes requested by the business team. The Sprint Retrospective is a meeting that occurs after the Sprint Review meetings to assess the interactions between people, relationships, processes, and tools. During this meeting, the team identifies problems, defines action points (i.e., corrective and preventive actions), and defines a plan to apply the action points.

Scrum contains three primary artifacts: the product backlog, the sprint backlog, and the increment. The product backlog is an ordered list that represents the product's features, requirements, improvements, and bug fixes. It should be ordered given business value, risk, effort, and technical dependencies (Silva et al., 2017). Furthermore, it is emergent and should adapt to business changes (Pichler, 2010). The sprint backlog is composed of product backlog items that were allocated to a given sprint (Griffiths, 2012), further decomposed into technical tasks, and process improvement items identified during Sprint Retrospective meetings. The increment is the result of completing the product backlog items during a sprint summed to the value of the increments of all previous sprints.

There are three roles: Product Owner, Scrum Master, and developer (Sutherland & Schwaber, 2017). The Product Owner is responsible for maximizing the product's value. He should serve as an interface between the technical team and the business team and is responsible for managing the product backlog. The Scrum Master serves as a servant-leader and is responsible for ensuring that Scrum's theories, rules, and practices are correctly applied in the project. The developers are responsible for

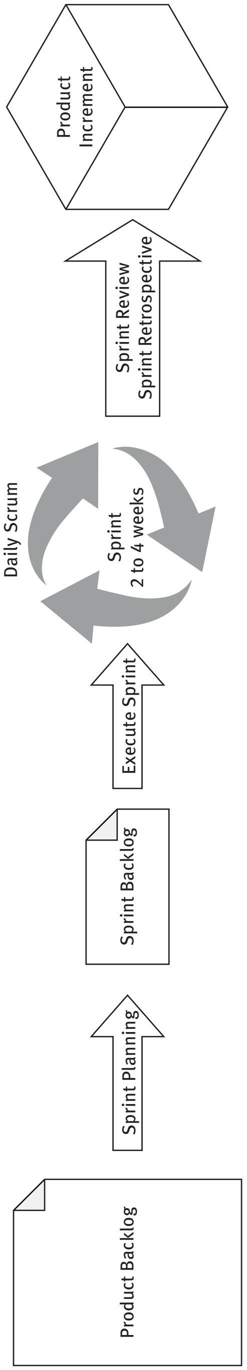


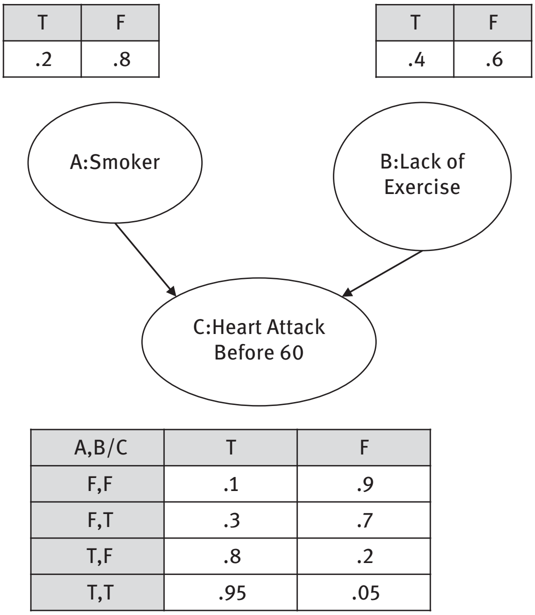
Figure 5.1: Overview of Scrum.

executing any activity related to delivering the product increment at the end of the sprints (e.g., design, implement, and test).

5.2.1 Bayesian networks overview

Bayesian networks are probabilistic graph models and used to represent knowledge about an uncertain domain (Ben-Gal, 2007). A Bayesian network is a directed acyclic graph that represents a joint probability distribution over a set of random variables (Friedman, 1997). It consists of two main parts: the structure and the uncertainty quantification (i.e., probability functions). The structure is a directed acyclic graph representing the causality between the random variables. Nodes represent the random variables. Arcs represent the causal relationship between the nodes. Each node is represented by a probability function that quantifies the relationships between the node and its parents if any.

We present an example of a Bayesian network in Figure 5.2. Ellipses represent the nodes and arrows represent the arcs. The probability functions are usually represented as node probability tables. Even though the arcs represent the causal connection's direction between the variables, information might propagate in any direction, following the rules of D-separation (Pearl & Russel, 1995).



T	F
.2	.8

T	F
.4	.6

A,B/C	T	F
F,F	.1	.9
F,T	.3	.7
T,F	.8	.2
T,T	.95	.05

Figure 5.2: A Bayesian network example.

One of the advantages of Bayesian networks is its explicit representation of causality, easing the construction of its structure by domain experts; and, as a consequence, having a high degree of interpretability. Further, in general, Bayesian networks have many advantages such as suitability for small and incomplete data sets, structural learning possibility, the combination of different sources of knowledge, an explicit treatment of uncertainty, support for decision analysis, and fast responses (Uusitalo, 2007). Given this, they have been extensively used to assist in managing risks in several domains, such as software development (Perkusich et al., 2015; Freire et al., 2018) and large engineering projects (Lee et al., 2009). As example of open source Bayesian network tools, there is the UnBBayes,[1] JavaBayes,[2] and the Github[3] project made available by Paul Govan.

5.3 Proposed solution

This section presents a knowledge-based framework to manage risks for Scrum software development projects. As requirements, the framework must conform to agile mindset, enable diagnosing root cause (i.e., causal analysis) and business impact prognosis, flexible to metrics programs, and promote cross project reuse of risk knowledge. Such a solution should help in identifying risks based on historical data and defining action plans to mitigate them. The proposed approach relies on a framework to reuse risks and modeling the main aspects of the Scrum product delivery process with a causal model, more specifically, a Bayesian network. In what follows, presents the causal model, Section 5.3.1 presents the causal model, which is used by the framework, and Section 5.3.2 presents the proposed framework.

5.3.1 Causal model

The causal model is a Bayesian network that models the main process factors of Scrum projects, in other words, its artifacts, roles, and events. Since the causal model represents the main components of a Scrum project enabling diagnosis (i.e., root cause analysis) and prognosis (i.e., impact analysis), we assume that modeling these factors help managers in managing risks, positive and negative. This assumption logically follows from observing that risk can be classified in terms of the product, process, project, or organization (Wallace & Keil, 2004), which, in turn, can be modeled by a Bayesian network (i.e., in our case, the developed Bayesian network

[1] https://sourceforge.net/projects/unbbayes/

[2] https://www.cs.cmu.edu/~javabayes/

[3] https://github.com/paulgovan/BayesianNetwork

does not handle organizational risks). Further, Bayesian networks can handle distinct types of variables (e.g., discrete, continuous, ordinal, nominal, and Boolean); therefore, being flexible to metric programs. This explains why Bayesian networks are heavily applied to managing risks (Fenton & Neil, 2012). Thus, we believe that having a Bayesian network for the Scrum process might help Scrum teams to deliver the correct product to the client, on time. We built the model from the project's Scrum Master perspective because it is the role responsible for the accountability over the Scrum process. Figure 5.3 presents an overview of the model.

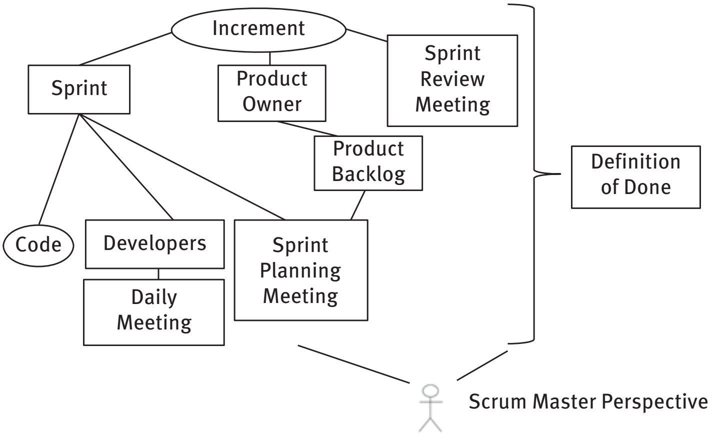


Figure 5.3: Overview of the causal model.

Figure 5.4 presents the complete model. In what follows, we explain the main model fragments. For details regarding the validity and applicability of the model, refer to Perkusich et al. (2017).

To develop such a model, we followed a top-down approach. We defined as the top-level node the (product) **Increment**, since the main goal of agile development is to satisfy customers with work code. In Scrum, the increment is developed during sprints. The Product Owner is responsible for maximizing the value of the product. Furthermore, the increment must be evaluated during Sprint Review meetings. Therefore, we added the nodes *Sprint*, *Product Owner* and *Review meeting* as parents of the node *Increment*.

During Sprint Review meetings, the product must be inspected and, if necessary, adapted by the customers. As previously explained, not using this practice or not having the customers participating might result in rework and implementation of the wrong features. Therefore, we added the nodes *Inspection* and *Adaptation* as parents of the node *Review meeting*. To input data into the model, the Scrum Master must observe the Sprint Review meetings and judge these two factors. As shown in

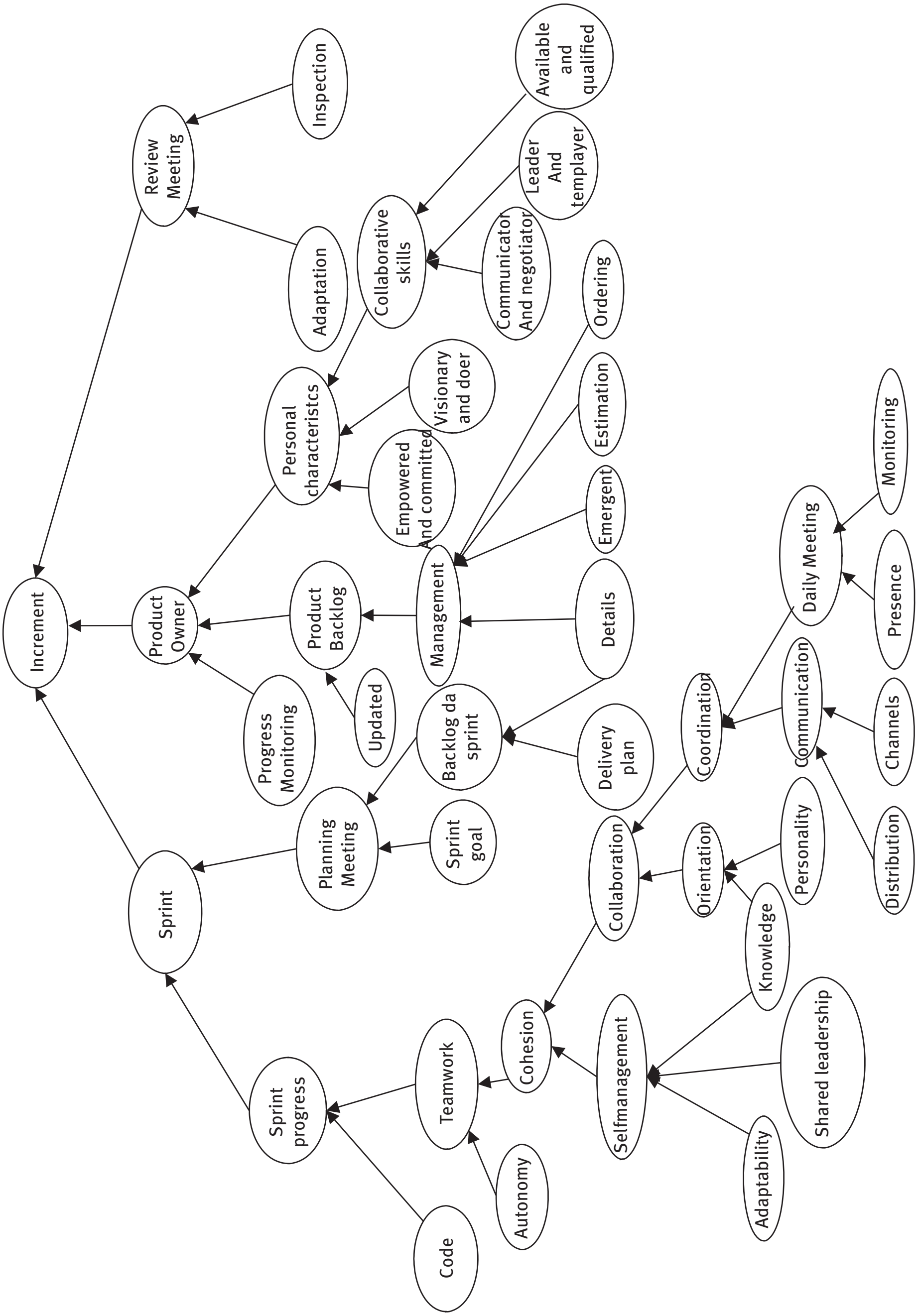


Figure 5.4: DAG of the base BN.

Perkusich et al. (2017), it is also possible to complement this fragment with metrics to indicate the evidence of these input nodes.

The causal model consists of process factors that act as predictors for the product increment quality. The team might map each process factor to a set of issues, risks, and indicators. The indicators might be collected automatically by integrating the Bayesian network tool to CASE tools, or manually through questionnaires or during Sprint Retrospective meetings.

5.3.2 Risk management framework

The goal of the framework presented herein is to integrate knowledge engineering and risk management for assisting in decision making in an agile software development environment. The framework consists of mechanisms for transforming tacit knowledge regarding risks into explicit knowledge through the use of an organizational risk memory. For this purpose, each project must register information regarding risks and issues into risk artifacts, namely, the project's risk log, issue log, and lessons report. Figure 5.5 shows how the proposed framework enables the reuse of risks by focusing on the flow of the identification of risks creation and reuse. In what follows, Section 5.3.2.1 presents how the risk artifacts are produced in conformance with Scrum practices, Section 5.3.2.2 presents details of the artifacts, Section 5.3.2.3 describes the main activities of the framework.

5.3.2.1 Risk identification in Scrum

This section discusses how risks are identified in Scrum projects. As shown in Figure 5.5, each project produces a set of risk artifacts containing the risks, issues, and lessons registered for the given project. In Scrum, each meeting is an opportunity for the team to manage risk, and, consequently, update these artifacts. The main risk management activities are risk identification, analysis, planning, implementation, and communication Bennett (2017).

During the first planning meeting, which might be a Release Planning meeting or the first Sprint Planning meeting, the team might identify an initial set of risks for the given project, and register them on the project risk log. The risk should be assigned, whenever possible, to one of the variables from the model presented in Section 5.3.1. Otherwise, it should be classified as Customer mandate, scope, and requirements, environment, or execution (Keil et al., 1998). The list of identified risks might be updated during future Scrum ceremonies.

As mentioned in Section 5.2.1, during the Daily Scrums, the team report impediments for the team reaching the goal of the current sprint. Such impediments might be risks or issues. If a reported impediment has not occurred yet, it is a risk; if it has

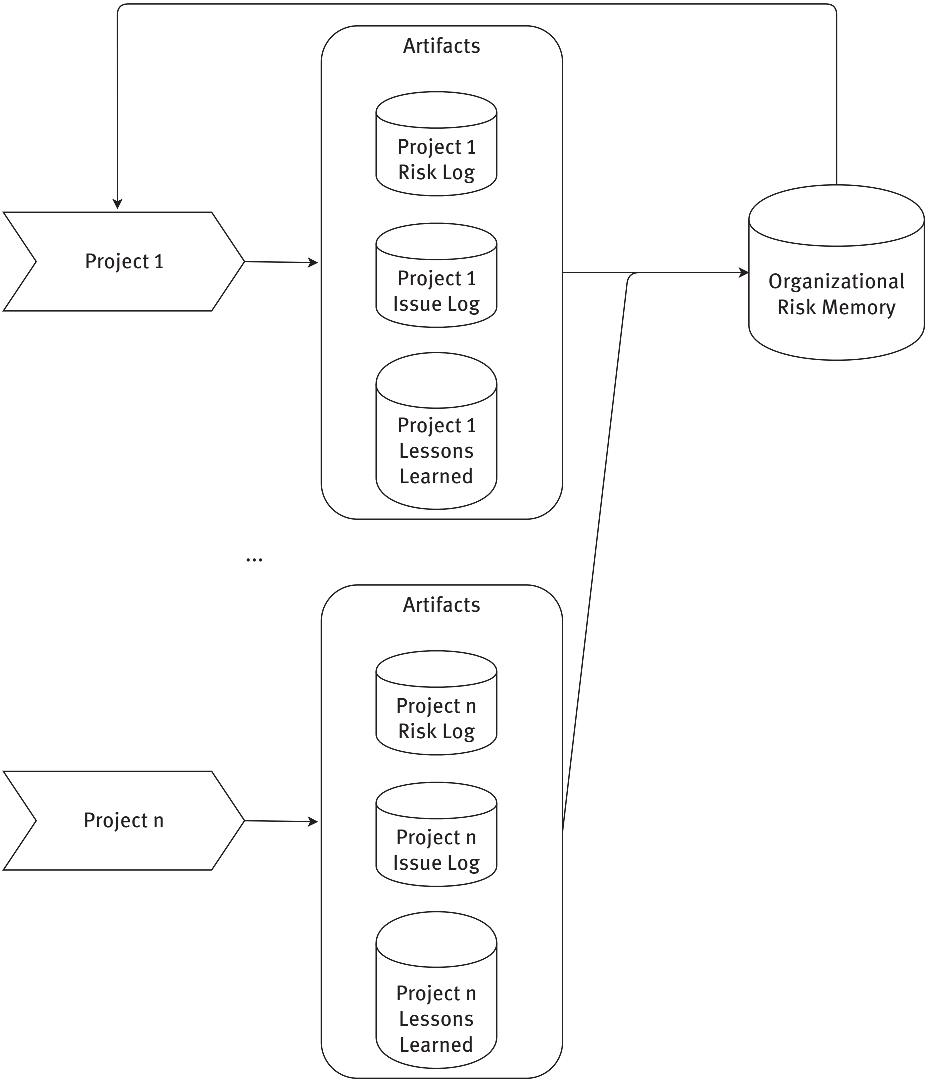


Figure 5.5: Flow of reusing risk with the proposed framework.

already occurred, it is an issue. Given this, they should be registered in the risks or issues log properly.

During Sprint Review meetings, the product is inspected and adapted. At this point, risks related to potential changes in the marketplace can be discussed, and given the team's progress, project variable restrictions such as time, scope, and cost might also be evaluated. Therefore, if necessary, the risk and issues logs should be updated after the Sprint Review meeting.

The Sprint Retrospective is an opportunity that the team has to focus on its continuous improvement by reflecting in what is working well, and what is not. Therefore, at each Sprint Retrospective meeting, the team increments knowledge regarding the project, identifying lessons, and defining improvement plans. Given this, the team can incrementally build the lessons report during these meetings. We recommend the team to document each raised point from Sprint Retrospective meetings into the team's Retrospective log, and only points that the team sees potential value for other projects should be exported to the lessons report.

Finally, continuously, the team incurs into technical debt (Cunningham, 1993; Alves et al., 2016). Technical debt refers to the debt that the team incurs when it chooses a quick approach to solve an issue, focusing on the short term, but with the potential of a negative long-term impact. It can refer to any aspect of the software, such as documentation, testing, code complexity, and known defects. We recommend the team to manage technical debt using, for instance, an issue tracking system and relate each debt to the related risks if any.

Notice that, to avoid losing focus of each Scrum meetings' goal, we recommend that, during them, the team only identifies the risks. The team should perform risk analysis and implementation in separate meetings, specific for risk management. These meetings should also be time-boxed and facilitated by the Scrum Master.

5.3.2.2 Artifacts description

This section describes the risk artifacts. The risk log contains information regarding the risks registered for the given project. Table 5.1 shows the minimum attributes for the risk log.

The issue log contains information regarding issues registered for the given project. An issue is, generally, a risk that has occurred. It might be the case that the risk was not identified previously by the project team. Table 5.2 shows the minimum attributes for the issue log.

The lessons report consists of the lessons registered by the team. It might contain information useful for risk management. Table 5.3 shows the minimum attributes for the lessons report.

5.3.2.3 Main activities

This section describes the three main activities of the proposed framework:

- Risk memory creation.
- Risk memory maintenance.
- Risk management.

Each activity of the framework is composed of tasks and relationships, shown in Figure 5.6. The framework is composed of the following roles:

- *Risk manager*, responsible for managing risks, which includes their identification, assessment, and control. In Scrum, since the team is crossfunctional, this role is shared by the Product Owner, Scrum Master, and the development team.
- *Risk memory manager*, responsible for promoting the risk reuse process and managing the risk memory. This role might be played by a program manager or the Project Management Office (PMO).

The **risk memory creation** activity has the goal of creating the risk memory by analyzing an existing organization's database of risks, issues, and lessons learned or risks documented in the literature. The risk memory manager is responsible for, in partnership with project managers, to select in the existing database or literature the risks that have the potential to be reused by other projects. The risk memory manager is also responsible for making adjustments in the language of the documented risks, for making them reusable. For instance, an example of adjustment would be removing the names of people and companies from the risks' description.

Table 5.1: The minimum attributes of the risk log.

Attribute	Description
Risk identifier	A unique identifier for the risk
Risk type	Threat or opportunity
Risk category	Each risk should be assigned, whenever possible, to one of the variables from the model presented in Section 5.3.1. Otherwise, it should be classified as Customer mandate, scope, and requirements, environment, or execution (Keil et al., 1998). If necessary, such a classification might be tailored to fit the organization's culture.
Risk description	A textual description of the risk. We recommend to, initially, use a catalog such as the one presented in Schmidt et al. (2001). The description should be written, making explicit the cause, event, and effect of the given risk.
Probability impact	Choose a value from an agreed scale to represent the probability impact of the risk.
Response category	There are six possible responses for threats and four for opportunities. For threats: avoid, reduce, fallback, transfer, accept, and share. For benefits: share, exploit, enhance, and reject.
Risk response	A textual description of the response strategy adopted. There might have multiple actions to handle risk.
Risk status	Current status of the risk: opened or closed.
Risk owner	The person responsible for handling the risk.

Table 5.2: The minimum attributes of the issue log.

Attribute	Description
Issue identifier	A unique identifier for the issue
Issue impact	Positive or negative
Issue description	A textual description of the issue
Associated risk	If applicable, relate the issue to an existing risk on the organizational risk memory.
Date raised	The date in which the issue was identified
Raised by	The person that identified the issue
Issue priority	Choose a value from an agreed scale to represent the priority of the issue (e.g., high, medium, low).
Issue severity	Choose a value from an agreed scale to represent the severity of the issue (e.g., high, medium, low).
Issue status	Current status of the issue: opened or closed.
Closure date	The date in which the issue was closed.

Table 5.3: The minimum attributes of the lessons report.

Attribute	Description
Lesson identifier	A unique identifier for the lesson.
Lesson impact	Positive or negative.
Lesson recommendation	A textual description of the recommendation.
Lesson category	If applicable, determines the relationship of the given lessons with risks on the organizational risk memory.
Lesson priority	Choose a value from an agreed scale (e.g., high, low, medium) to represent the importance of the lesson.

For companies with low-risk management maturity, we recommend to use an existing risk catalog to populate the risk memory such as the ones presented in Schmidt et al. (2001) and Eloranta et al. (2016). Furthermore, the risks should be assigned, whenever possible, to one of the variables from the model presented in Section 5.3.1. Otherwise, they should be classified as Customer mandate, scope, and requirements, environment, or execution (Keil et al., 1998).

The **risk memory maintenance** activity consists of a continuous effort to maintain the quality of the risk memory. Whenever new data (i.e., risks, issues, or

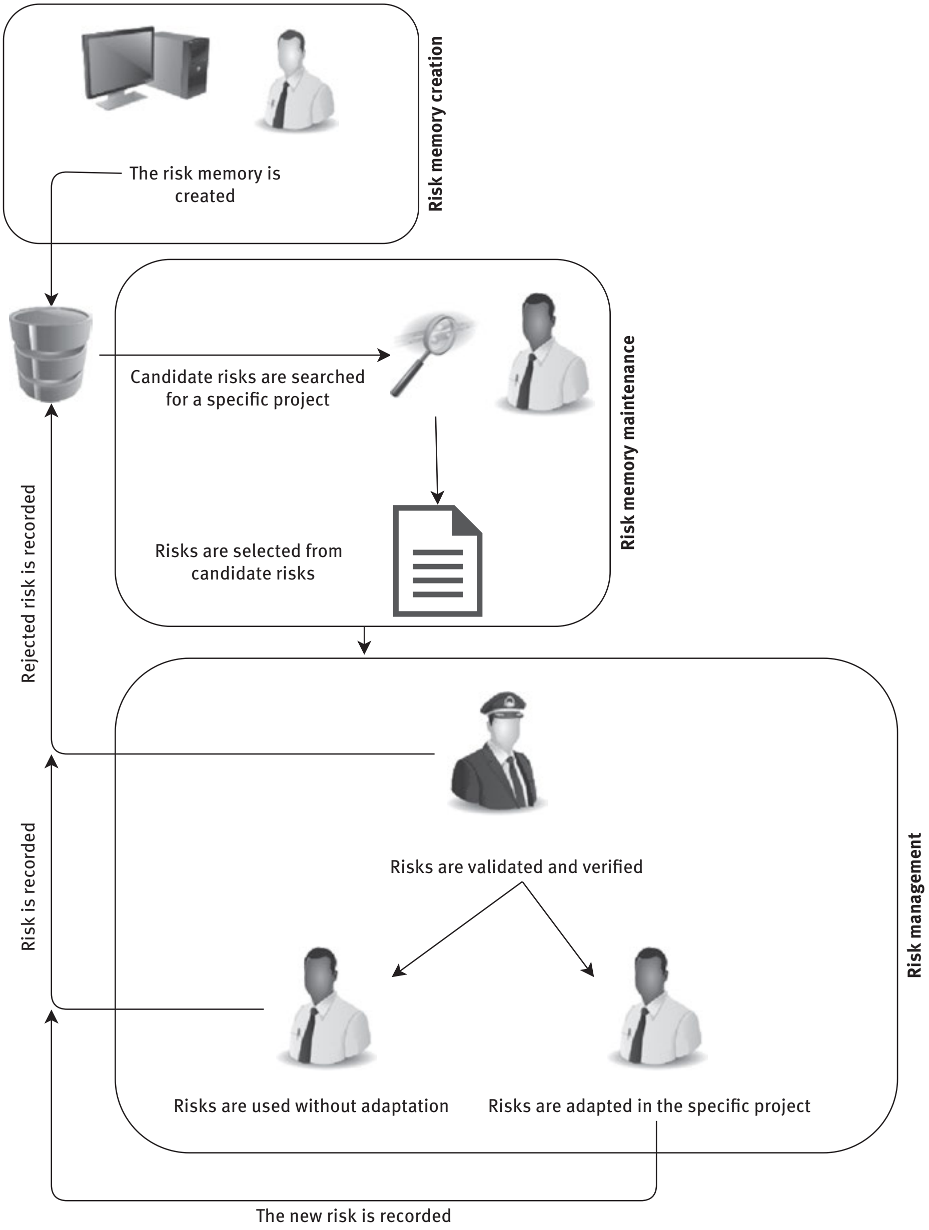

Risk memory creation
The risk memory is created
Candidate risks are searched for a specific project
Risks are selected from candidate risks
Risk memory maintenance
Rejected risk is recorded
Risk is recorded
Risks are validated and verified
Risks are used without adaptation
Risks are adapted in the specific project
Risk management
The new risk is recorded

Figure 5.6: Relationship between the framework's activities.

lessons learned) are available from projects to be added to the risk memory, the risk memory manager must validate them to ensure that the language is adequate and that the information is consistent. In the case of adjustments in existing risks, the risk memory manager must collaborate with the project managers that originated them to ensure correctness. Furthermore, through traceability relationships, any changes to a risk notify the original creator of the risk, allowing him to review if the changes are valid. Finally, for safety reasons, a version control process should be performed on the risk memory.

The **risk management** activity consists of the Scrum team managing risks, by identifying them and reusing the information stored in the risk memory. Notice that the team reuses not only the risk itself but also the response strategies. Since the risks are mapped to the Bayesian network shown in Section 5.3.1, the project team can use the Bayesian network for prognosis or diagnosis purposes. On the project level, by applying the prediction and diagnosis capabilities of the Bayesian network, the team can identify threats (i.e., negative risks) and opportunities (i.e., positive risks). Whenever a threat or opportunity is identified, the team registers them and associates them with one (or more) of the nodes (i.e., process factors) of the Bayesian network.

Notice that risk might be reused with or without adaptation. If a risk is reused with an adaptation, for instance, defining a new response (i.e., intervention), the new risk must be created and related to its parent. New risks must be validated by the risk memory manager for ensuring its validity and correctness. Rejected risks must also be recorded since this information might be useful for future risk managers.

For clarity, in what follows, we present examples of how to classify risks using the activities previously discussed. For example, a risk number 4 for the factor *Product Owner* (see Figure 5.4), classified as a threat and related to the anti-pattern *Customer Product Owner* presented by Eloranta et al. (2016), could be registered as:

- <Risk identifier≥ 4
- <Risk type≥ threat=t
- <Risk category≥ Product Backlog Ordering
- <Risk description≥ Unordered Product Backlog. The Product Backlog is not ordered, but Teams select items based on their own judgment. Thus, as Product Backlog is unordered, the Team might be lacking vision of the risky or valuable elements of the product. As a result, the Team might be building wrong features which do not have value to the customer or are just rarely used. Only features which are fun to implement get implemented as the Team starts to pick whatever they like from the backlog. Features which are hard to implement or test are left to the backlog and implemented last. This increases the risk of problems arising in the late stages of development. **Exceptions**: Projects where the requirements are fixed and given up-front by the customer, and these requirements are not going to change. Some initiatives by governments and military might have such requirements.
- <Probability impact≥ Very high=vh
- <Response category≥ Reduce

- <Risk response≥ Organize Product Backlog refinement meetings every other week where the top of the Product Backlog is ordered according to the value of items and taking care of the dependencies between work items.
- <Risk status≥ opened

Notice that, for the risk identified by the *Risk identifier = 4*, namely, *Customer Product Owner*, the risk is not explicitly represented as a variable on the model. Conversely, such a risk is categorized (i.e., mapped) to one of the Bayesian network's variables. In this case, it is mapped to the variable *Ordering*. If such a variable uses an ordinal scale (e.g., Bad, Moderate, Good), we might consider that, if such a risk becomes an issue (i.e., it becomes reality), it means that we have a *Bad* Ordering of the Product Backlog. By having a *Bad* Ordering of the Product Backlog, it is expected to deliver a product with low chances of having market fit. Such inference can be observed in Figure 5.7, in which even if we consider that the remaining Scrum's processes are good, we have less than 40% of delivering a product of *High* quality, considering a 5-point Likert scale.

As an example of an opportunity to be explored, there is the use of *Estimate with Planning Poker*, which can be classified as a risk associated with the variable *Estimation* (see Figure 5.4). It could be registered as:

- <Risk identifier ≥ 3
- <Risk type≥ opportunity=o
- <Risk category≥ Estimation
- <Risk description≥ Estimate with Planning Poker.
- <Probability impact≥ High=h
- <Response category≥ Exploit
- <Risk response≥ Try to arrange training for managers and developers to demonstrate the benefits that would be gained from having effort estimation with Planning Poker and Story Points.
- <Risk status≥ opened

By registering the risks, incrementally, a knowledge base is created, containing data regarding non-conformities (and solutions) related to the attributes of the leading entities of the product delivery process modeled by the Bayesian network (see Section 5.3.1). On the organization level, teams of future Scrum projects can use these data to manage risks related to the product delivery process. For instance, the teams will be able to predict process-related threats and evaluate the use of preventive and corrective actions for their projects. With the use of the proposed approach, risk management of Scrum projects changes from being based on informal and tacit knowledge to being based on empirical evidence, as registered in the knowledge base. Therefore, instead of depending on the intuition of the project team, risk management decisions are informed and based on data.

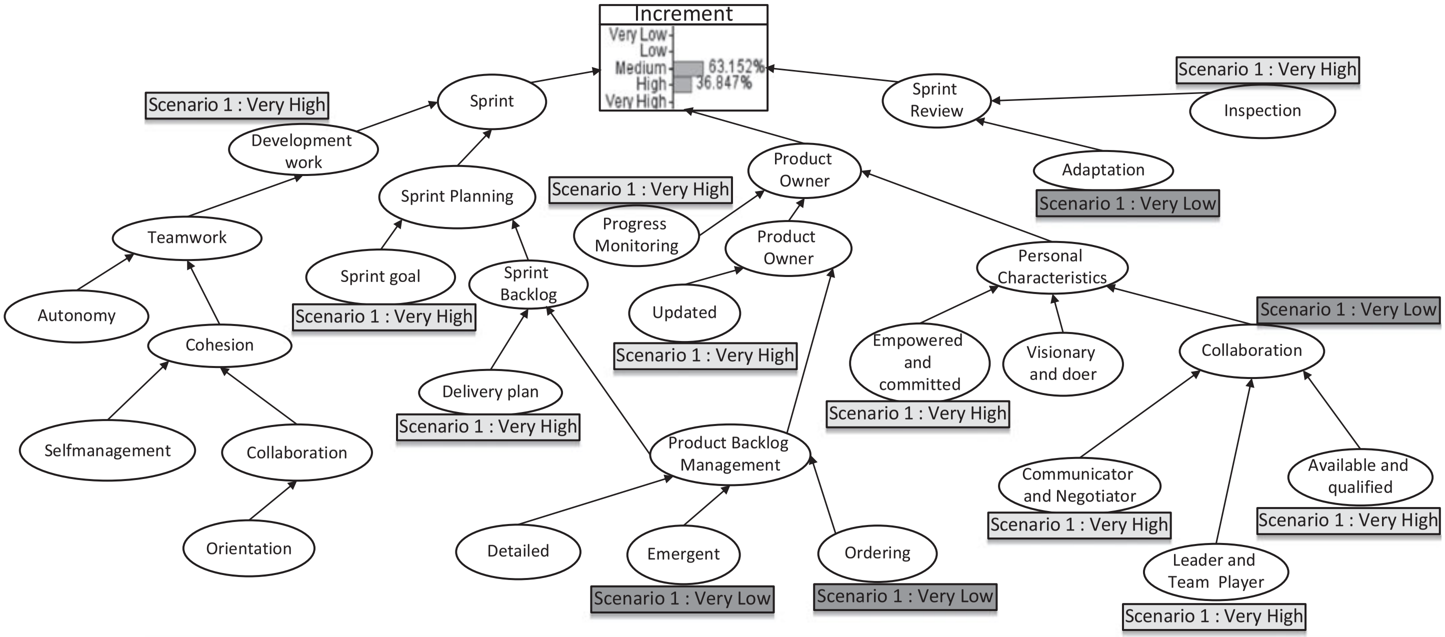


Figure 5.7: Example of risk analysis.

5.4 Conclusion

In this chapter, we presented a knowledge-based risk management framework for Scrum software development projects, focusing on positive and negative risks related to the product delivery process. For this purpose, we used the causal model presented in Perkusich et al. (2017) as the basis to relate risks to key process factors, enabling a causal analysis of risks. Furthermore, we presented a set of activities and roles to enable reuse-driven risk management in an agile software development context.

As future work, we foresee several research directions. First, we plan to refine and expand the model presented in Perkusich et al. (2017) to model variables with a low level of control such as risks related to customer mandate factors and to model interventions (i.e., actions) and their effect such as done in Constantinou et al. (2016). Furthermore, we intend to refine the framework by incorporating integration with a network approach and dialogue for knowledge transfer to include the potential knowledge which cannot be made explicit, following the perspective presented by Blackler (1995). We also plan to implement tools to assist the risk memory manager to validate risks, issues, and lessons using Natural Language Processing capabilities. Further, given that an organization uses the proposed framework, a history of risk reuse is generated. Such history enables the development of recommender systems, which can assist project managers in identifying risks and planning interventions. With regards to the causal model, we plan to expand it to include a variable related to the Sprint Retrospective and to transform it into a Dynamic Bayesian Network to handle the dynamic, iterative nature of Scrum projects. Finally, we plan to evaluate the proposed framework through case studies in real-world settings.

References

R. Hoda, N. Salleh, J. Grundy, The rise and evolution of agile software development, IEEE Software 35 (5) (2018) 58–63 (2018).

A. S. Campanelli, F. S. Parreiras, Agile methods tailoring–a systematic literature review, Journal of Systems and Software 110 (2015) 85–100 (2015).

R. Hoda, N. Salleh, J. Grundy, H. M. Tee, Systematic literature reviews in agile software development: A tertiary study, Information and Software Technology 85 (2017) 60–70 (2017).

R. M. Fontana, I. M. Fontana, P. A. da Rosa Garbuio, S. Reinehr, A. Malucelli, Processes versus people: How should agile software development maturity be defined?, Journal of Systems and Software 97 (2014) 140–155 (2014).

M. Fowler, J. Highsmith, et al., The agile manifesto, Software Development 9 (8) (2001) 28–35 (2001).

J. Sutherland, K. Schwaber, The Scrum Guide, https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum- Guide-US.pdf, accessed in: 08- 19-2019(2017).

K. Beck, E. Gamma, Extreme programming explained: embrace change, addison-wesley professional, 2000 (2000).

A. Cockburn, Agile software development, Vol. 177, Addison-Wesley, Boston, USA. 2001 (2001).

S. R. Palmer, M. Felsing, A practical guide to feature-driven development, Prentice Hall, New Jersey, USA., 2002 (2002).

VersionOne, 13th Annual State of agile development survey results, https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508, accessed in: 08- 23-2019(2019).

K. Schwaber, Agile project management with Scrum, Microsoft press, Redmond, Washington, United States 2002 (2002).

T. Raz, E. Michael, Use and benefits of tools for project risk management, International journal of project management 19 (1) (2001) 9–17 (2001).

C. Chapman, S. Ward, Project risk management: processes, techniques and insights, John Wiley, Hoboken, New Jersey, United States 1996 (1996).

I. Dikmen, M. Birgonul, C. Anac, J. Tah, G. Aouad, Learning from risks: A tool for post-project risk assessment, Automation in construction 18 (1) (2008) 42–50 (2008).

S. Alhawari, L. Karadsheh, A. N. Talet, E. Mansour, Knowledge-based risk management framework for information technology project, International Journal of Information Management 32 (1) (2012) 50–65 (2012).

W. L. Currie, A knowledge-based risk assessment framework for evaluating web-enabled application outsourcing projects, International Journal of Project Management 21 (3) (2003) 207–217 (2003).

M. Perkusich, L. C. e Silva, A. Costa, F. Ramos, R. Saraiva, A. Freire, E. Dilorenzo, E. Dantas, D. Santos, K. Gorgônio, et al., Intelligent software engineering in the context of agile software development: A systematic literature review, Information and Software Technology 119 (2020) 106241 (2020).

S. Ward, C. Chapman, Transforming project risk management into project uncertainty management, International journal of project management 21 (2) (2003) 97–105 (2003).

I. Ancveire, I. Gailite, M. Gailite, J. Grabis, Software delivery risk management: application of bayesian networks in agile software development, Information Technology and Management Science 18 (1) (2015) 62–69 (2015).

Y. Hu, X. Zhang, E. Ngai, R. Cai, M. Liu, Software project risk analysis using bayesian networks with causality constraints, Decision Support Systems 56 (2013) 439–449 (2013).

A. Okutan, O. T. Yıldız, Software defect prediction using bayesian networks, Empirical Software Engineering 19 (1) (2014) 154–181 (2014).

K.-S. Chin, D.-W. Tang, J.-B. Yang, S. Y. Wong, H. Wang, Assessing new product development project risk by bayesian network with a systematic probability generation methodology, Expert Systems with Applications 36 (6) (2009) 9879–9890 (2009).

E. Lee, Y. Park, J. G. Shin, Large engineering project risk management using a bayesian belief network, Expert Systems with Applications 36 (3) (2009) 5880–5887 (2009).

N. Fenton, M. Neil, W. Marsh, P. Hearty, Ł. Radliński, P. Krause, On the effectiveness of early life cycle defect prediction with bayesian nets, Empirical Software Engineering 13 (5) (2008) 499 (2008).

N. Fenton, M. Neil, Risk assessment and decision analysis with Bayesian networks, Crc Press, Boca Raton, Florida, United States 2012 (2012).

K. Verbert, R. Babuška, B. De Schutter, Bayesian and dempster–shafer reasoning for knowledge-based fault diagnosis–a comparative study, Engineering Applications of Artificial Intelligence 60 (2017) 136–150 (2017).

M. Perkusich, K. Gorgonio, H. Almeida, A. Perkusich, A framework to build bayesian networks to assess scrum-based software development methods, in: Proceedings of the 29th SEKE International Conference on Software Engineering & Knowledge Engineering, SEKE 2017, 2017 (2017).

M. Perkusich, G. Soares, H. Almeida, A. Perkusich, A procedure to detect problems of processes in software development projects using bayesian networks, Expert Systems with Applications 42 (1) (2015) 437–450 (2015).

M. Griffiths, PMI-ACP Exam Prep, RMC (Rita Mulcahy Companies) Publications, Minnetonka, Minnesota, USA 2012 (2012).

A. Silva, T. Araújo, R. Barbosa, F. B. A. Ramos, A. A. M. Costa,M. Perkusich, E. Dilorenzo, Ordering the product backlog in agile software development projects: A systematic literature review., in: Proceedings of the 29th SEKE International Conference on Software Engineering & Knowledge Engineering, SEKE 2017, 2017, pp. 74–80 (2017).

R. Pichler, Agile Product Management with Scrum: Creating Products that Customers Love, 1st Edition, Addison-Wesley Professional, Boston, USA 2010 (4 2010).

I. Ben-Gal, Bayesian Networks, John Wiley and Sons, Hoboken, New Jersey, United States 2007 (2007).

N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Machine Learning 29 (2–3) (1997) 131–163 (1997).

J. Pearl, S. Russell, Bayesian networks, Handbook of brain theory and neural networks (1995).

L. Uusitalo, Advantages and challenges of bayesian networks in environmental modelling, Ecological Modelling 203 (3–4) (2007) 312–318 (2007). .

A. Freire, M. Perkusich, R. Saraiva, H. Almeida, A. Perkusich, A bayesian networks-based approach to assess and improve the teamwork quality of agile teams, Information and Software Technology 100 (2018) 119–132 (2018)

L. Wallace, M. Keil, Software project risks and their effect on outcomes, Communications of the ACM 47 (4) (2004) 68–73 (2004).

N. Bennett., Axelos Prince2 Handbook, 1st Edition, Stationery Office Books, London, United Kingdom. 2017 (10 2017).

M. Keil, P. E. Cule, K. Lyytinen, R. C. Schmidt, A framework for identifying software project risks, Communications of the ACM 41 (11) (1998) 76–83 (1998).

W. Cunningham, The wycash portfolio management system, ACM SIGPLAN OOPS Messenger 4 (2) (1993) 29–30 (1993).

R. Schmidt, K. Lyytinen, M. Keil, P. Cule, Identifying software project risks: An international delphi study, Journal of management information systems 17 (4) (2001) 5–36 (2001).

V.-P. Eloranta, K. Koskimies, T. Mikkonen, Exploring scrumbut – an empirical study of scrum anti-patterns, Information and Software Technology 74 (2016) 194–203 (2016).

A. C. Constantinou, N. Fenton, W. Marsh, L. Radlinski, From complex questionnaire and interviewing data to intelligent bayesian network models for medical decision support, Artificial intelligence in medicine 67 (2016) 75–93 (2016).

F. Blackler, Knowledge, knowledge work and organizations: An overview and interpretation, Organization studies 16 (6) (1995) 1021–1046 (1995).

Holmes E. Miller and Kurt J. Engemann

6 Managing bias risk in algorithms and decision models

6.1 Introduction

Problems involving characterization, classification and choice are present in many facets of everyday life. Pick a topic and examples readily appear: Which students applying to college are offered admission and which are rejected; which applicants for home mortgage loans are approved to receive loans and which are denied; which job applicant is offered a job and which applicants are not; which convict is granted parole and which is not, which advertisement does a person see and which are ignored; who is offered a promotion and who are not. The list is endless.

Although these problems appear frequently in the press, nothing about them is new. A century ago, people applied to colleges and were admitted or were rejected; applied for loans that were or were not granted; were offered or were not offered jobs. The underlying problem of choice always was present, but what characterizes life in the 21st century is how these choices are made.

As with the underlying problems, methodologies involving logical methods of choice and decision analysis also are not new. For example, years ago bankers in the proverbial green eyeshades decided on who would receive loans and who would not, using paper-based data that was available. Executives used typewritten resumes, impressions from job interviews, and letters of recommendation to decide which candidates to hire. Paroles were and were not granted by processes that relied more on convention than on predictions based on what were then, hidden variables.

Now, computers and software facilitate this process in several ways. First, computer processing power and speed allow more decision makers to use many more variables to arrive at decisions that are not only made faster, but also involve heretofore unheard-of levels of complexity. Second, bringing to bear the benefits of mathematical and logical analysis means that the decisions often turn out to be better, when measured by appropriate metrics. Third, using software and computer processing power enables new technologies to automate many decisions that in the past could be made only by humans – for example, self-driving cars and machine intelligence embedded into many appliances. Finally, fourth, new machine learning methodologies at least superficially, promise to reduce or even eliminate the issue of human biases framing and shepherding the decision process. Thus, they create the promise for decisions that depend less on the decision maker, and more on the underlying facts of the decision terrain.

https://doi.org/10.1515/9783110652321-006

At the center of these changes is the concept of an *algorithm*, which the Merriam-Webster dictionary broadly defines as "a step-by-step procedure for solving a problem or accomplishing some end." The algorithmic process in a computer-based setting involves applying a "set of rules a machine follows to achieve a particular goal." Of course, algorithms need not always be computer driven. For example, people often develop and use personal algorithms to choose the best vegetables at the grocery, to solve puzzles, to find a lost set of keys, or to cook a meal. Indeed, the creation and application of a set of rules, both in computer mediated environments and non-computer mediated environments, creates the opportunity for the biases of the algorithm's developer, and the person who applies the algorithm, to enter into the choice process and to influence the final result.

Examples of bias occur when one group either is explicitly treated differently than others during the decision process, or when the impact on one group is markedly different from others with similar attributes. Kleinberg et al. (2018) discuss both disparate treatment and disparate impact from both legal and ethical perspectives. Disparate treatment may be thought of as affecting inputs and the algorithm, and disparate impact as affecting the outcome, regardless of whether or not disparate treatment occurred.

Groups that are adversely affected by bias in decision-making can be classified according to gender, ethnic origin, or class, where "class" may be defined by innumerable attributes, such as degree of physical ableness, age, or location. Examples include women, racial or ethnic minorities, people with physical or mental disabilities, or senior citizens. This isn't to say that the way algorithms treat all classes must always be equal, or that the outcomes need be equal. For example, when purchasing automobile insurance, people under 25 years often pay higher rates, all other things being equal, and a 70-year-old man would pay more for life insurance than a 30-year-old man. In each of these cases the algorithmic treatment generates different outcomes, because historical data validates the treating the various groups differently.

As Kleinberg et al. (2018) point out in discussing legal issues, "disparate impact means, in brief, that if some requirement or practice has a disproportionate adverse effect on members of protected groups (such as women and African-Americans), the defendant must show that the requirement or practice is adequately justified. Suppose, for example, that an employer requires members of its sales force to take some kind of written examination, or that the head of a police department institutes a rule requiring new employees to be able to run at a specified speed. If these practices have disproportionate adverse effects on members of protected groups, they will be invalidated unless the employers can show a strong connection to the actual requirements of the job."

Thus, a requirement that a prospective employee must be able to run 1.5 miles in 12 minutes or less and lift up to 200 pounds might be appropriate for firefighters, but they would be inappropriate for sports statisticians. In the case of the former, a woman might justifiably be denied employment, but in the latter case, as there is no relationship between the requirements and the job, imposing the requirement is an example of using a decision criterion, which may be embedded into an algorithm, to exclude a certain group of potential employees.

Disparate treatment involves treating different classes of applicants differently. Different treatment might occur based on the "taste" of the modelers or statistically arise from the underlying database used to develop the algorithm. Examples of "taste-based" bias would be excluding applicants deemed "unattractive" from becoming airline attendants or restaurant wait-staff, or members of an ethnic minority from applying for sales positions. Statistical bias can occur when the data used to develop a machine learning algorithm under-represents a minority group which biases the output. For example, the New York Times (Singer and Metz, 2019) reported that researchers for the National Institute of Standards and Technology found that a machine learning algorithm falsely identified African-American and Asian faces 10 to 100 times more than Caucasian faces. Assuming that the engineers who developed the algorithm were not biased, under-representation of minority groups in the database used by the training algorithm caused the decision algorithm itself to treat members of these two minority groups differently. Subsequent decisions could then adversely affect members of each group.

6.2 Decision algorithms

Two types of decision algorithms are 1) those where the model's developer consciously decides which variables will be used in the decision-making process, and 2) machine learning algorithms whose decisions are based on a learning algorithm that derives the underlying decision model from a database and a decision algorithm (e.g., from a neural network model) to use during the decision-making process. One attribute of case 1 is that an observer can identify the variables actually used in the model and identify the relationships among them. For example, multiple regression or mathematical programming models with known variable coefficients fall in this category. Machine learning algorithms, case 2, such as those developed by neural network modeling methods, after being trained may be effective in predicting outcomes but an observer may not know which variables exactly are being used in the prediction models, nor know how the variables are weighted, how they interact, or even know what is the relationship among them.

6.2.1 Classical algorithms

Here, we define *classical* algorithms as algorithms with known variables that are explicitly included in datasets used in model development and consciously incorporated into models by model developers, i.e., case 1 above, where the model developer decides what variables to include in a model and what variables to omit. The identity of the variables that are present and absent in the model creates an opportunity for bias, based on whether a variable is included or excluded. For example, in a mathematical programming model that might involve allocating funds for loans in various neighborhoods, might explicitly “redline” certain neighborhoods based on racial composition, either by adjusting weights in the model’s objective function, or by tailoring constraints to limit funds allocated. Here, the modeler’s bias enters, as the weights and constraints included in the model may be adjusted on modeler whim.

Modeler bias also may enter by excluding certain variables. For example, a model involving locating a plant might ignore certain variables, such as water usage, which could disadvantage a rural location where water is scarce. The result might be that the facility, owned by a company with deep pockets, may use most of the available water supply for itself, to the disadvantage of local residents. This may be a cost minimizing solution for the company, but one which omits the cost to local farmers, who now either may have no water, or who may incur higher expenses drilling deeper wells to obtain water.

Similar biases also hold for regression-based models, where modelers use datasets that might include or exclude data needed for an unbiased model. This would result in variables being present or absent in models, where the presence or absence reflects modeler bias. An example could arise from using a data set where the modeler manipulates the data so that it is aggregated (or disaggregated) in ways that serve the modeler’s biases, rather than in a neutral scientific manner. In a similar fashion, the data used could be an unrepresentative sample of the actual underlying population. In the model development process and in the model selection process, modelers could explicitly manipulate what the final model is, to reflect their biases. One way is to “throw out” data or models that they view as unfavorable, and by using the model development process, place their “thumbs on the scale” to ensure that the models that are used reflect their biases.

Engemann and Engemann (2017) explore safety climate, risk attitude, and risk decisions through their framework, the risk attitude chain, and discuss intersubjective processes as a foundation of behaviors and attitudes. A model builder’s behaviors and attitudes may bias a decision model through explicit or implicit assimilation of their own risk attitude. In certain situations, introducing a decision maker’s bias in a decision model is most appropriate. For example, evaluating risk strategies to enhance organizational resilience involves the determination of the likelihood of events. Engemann and Miller (2015) propose an algorithm for selecting strategies and determining sensitivity, based on the decision maker’s bias.

6.2.2 Machine learning algorithms

Machine learning algorithms may include a training algorithm applied to a dataset, which develops a final algorithm used for classification and the decision-making process. Kleinberg et al. (2018) discuss machine learning algorithms in the context of an applicant for a loan or a job. The final *screener* algorithm "produces an evaluative score (such as an estimate of future performance)." The other algorithm is the *trainer*, which "uses data to produce the screener that best optimizes some objective function." They go on to observe that "The Achilles' heel of all algorithms is the humans who build them and the choices they make about outcomes, candidate predictors for the algorithm to consider, and the training sample. A critical element of regulating algorithms is regulating humans."

In this sense, as with classical algorithms, machine learning algorithm, despite their appearance as being impartial and as being developed free of human intervention, at their core involve human interaction. Indeed, Kleinberg et al., regarding the presence of bias, in algorithms state that, "Our central claim, stated in simple form, is that safeguards against the biases of the people who build algorithms, rather than against algorithms per se, could play a key role in ensuring that algorithms are not being built in a way that discriminates (recognizing the complexity and contested character of that term)."

Thus, human issues involving algorithm builders and algorithm users exist when discussing bias in algorithms, regardless of their type. The next section discusses bias more completely, and presents a framework, which may be used to highlight how bias appears during the various phases of model development and model use.

6.3 Bias in decision algorithms

Three domains where bias can enter the decision-making process are the underlying data used to develop or train the algorithm, the decision algorithm itself, and the human biases of the algorithm's users who use the algorithm's output the make decisions (Smith, 2020). For example, consider a machine learning algorithm that recommends book purchases based on historical data involving purchases of people with similar demographic and general merchandize purchase histories. Suppose that a parent is looking for a book as a gift for a teenager. While the dataset used to develop the recommendation algorithm may have much information regarding similar members of the teen's demographic, there may be underrepresentation depending on specific attributes of the teen in question. For example, the teen may be disabled, may have interests in topics that most teens are uninterested in, or may belong to another underrepresented minority. The result is that the training algorithm used to develop the recommendation model may not reflect this particular teen's situation.

Algorithmic bias also can occur if the platform housing the recommendation process is biased to recommend books published by organizations who sponsor recommendations, e.g., by paying a fee to the platform owner in return for higher placement or greater likelihood that their products will be recommended. User bias can occur if the parent doing the purchase takes the list of recommended titles, and then eliminates those that they don't like, either based on their preferences or their perceptions regarding what their child should be reading.

The following discussion of these three domains is informed by several sources, including Silva et al. (2018), Mehrabi et al. (2019), and Miller (2015). While the specific model development process and structure used create differences in how some biases enter into the conversation, we will aggregate biases developed by what we call the *classical models* and *machine learning models*, noting differences when appropriate.

6.3.1 Biases in the data

Four categories where bias can occur concern how data might be included or excluded from the dataset before being used, how data is measured, how data is aggregated, and how data might be massaged by social pressure before being used.

Inclusion/Exclusion bias involves including too much data that advantages one group or excluding data about another group, so that the dataset used may not accurately represent the underlying population. This can be intentional (as the result of hidden agenda) of unintentional, as a result of a sampling method used to construct the data set. As noted above, one example of exclusion is that many datasets used to develop facial recognition algorithms underrepresent African American and other dark-skinned people, which makes prediction results less accurate. An example of overrepresentation is the famous prediction that Alf Landon, a Republican, would defeat Franklin Roosevelt, a Democrat in the 1936 election. Here, the data set was constructed using phone numbers and in 1936, during the depression, only the wealthier individuals who tended to favor the Republican candidate could afford phones.

Measurement bias occurs when a variable intended to measure one attribute, actually does not measure that attribute, or measures another attribute. For example, in developing a credit scoring algorithm, a bank might mistakenly use a high income as a measure for a person having high credit card balances, whereas this variable coupled with others, such as age, might reflect exactly the opposite, namely a propensity to keep low balances by repaying balances each month. Zip code also might be misused to measure propensity of default on loans, which biases those individuals living in that zip code who would never default.

Aggregation bias occurs when data is aggregated (or disaggregated) in ways that provide false messages, and bias the algorithm used in the decision model to recommend incorrect solutions. For example, aggregating all social classes in an undeveloped country might lead to a low average family income which might bias the algorithm to avoid lending to individuals from that country, or marketing to individuals in that country. In fact, the dispersion of wealth might be great so there may be various class subgroups that suggest that the opposite should be true.

Social bias occurs when social pressure of some sort distorts the data used to develop the algorithm. Mehrabi et al. (2019) report an example of this as being in a " . . . case where we want to rate or review an item with a low score, but when influenced by other high ratings, we change our scoring thinking that perhaps we are being too harsh." This would be true as rating a restaurant meal or movie, if we are not confident regarding our tastes relative to others.

6.3.2 Biases in the algorithm

Three categories where bias can occur in the algorithm itself concern how variables might be included or excluded from the model; how the algorithm evaluates, categorizes, and uses information; and how algorithm/user interaction might bias the decision algorithm.

Inclusion/Exclusion bias in what we call classical models involves what variables to use and what variables to omit. As discussed above, inclusion/omission may be valid and informed by the dataset and the underlying problem, such as in the case of the teenage driver getting auto insurance, but it also might reflect the biases of the modeler that are injected in the algorithm (e.g., pay women lower wages). For machine learning models, the algorithm that is used may be developed from the underlying data set, but doing this does not preclude inclusion of variables or misrepresentation of relationships. In both cases, biases not explicitly present in the underlying data may be inserted in the algorithm, consciously or procedurally.

Evaluation bias occurs in the process used to develop how the algorithm evaluates cases before presenting its output. The evaluation process may employ benchmarks which, if misapplied, may bias the results. For example, a facial recognition model may evaluate members looking a certain way or being a member of a racial minority as more likely to commit a crime, not based on the data, but on the misapplication of benchmarks that may confound innocent correlations with causation.

Interaction bias occurs when results of the algorithm are skewed based on interactions with users. Cases involve how data are presented to users and how users then can reframe presentations. Examples include how items reviewed favorably by users produce, via the algorithm structure, more favorable reviews. Algorithmic ranking of results causes a similar problem, and here, the ranking of results may be skewed unconsciously or by conscious intervention of the algorithm owner, e.g., after being paid a fee by an advertiser. Another case of interaction bias in social media applications involves user "likes" where regardless of the subject, videos presented often veer off, after several iterations, to extremes to better catch user attention (Nicas, 2018).

6.3.3 Biases caused by users

Three categories where bias can occur when users view the algorithm output and take further action, are when the output is misrepresented or deliberately misinterpreted, when the user puts a "thumb-on-the-scale" to bias results, and how different users respond behaviorally in different ways to the algorithm's output.

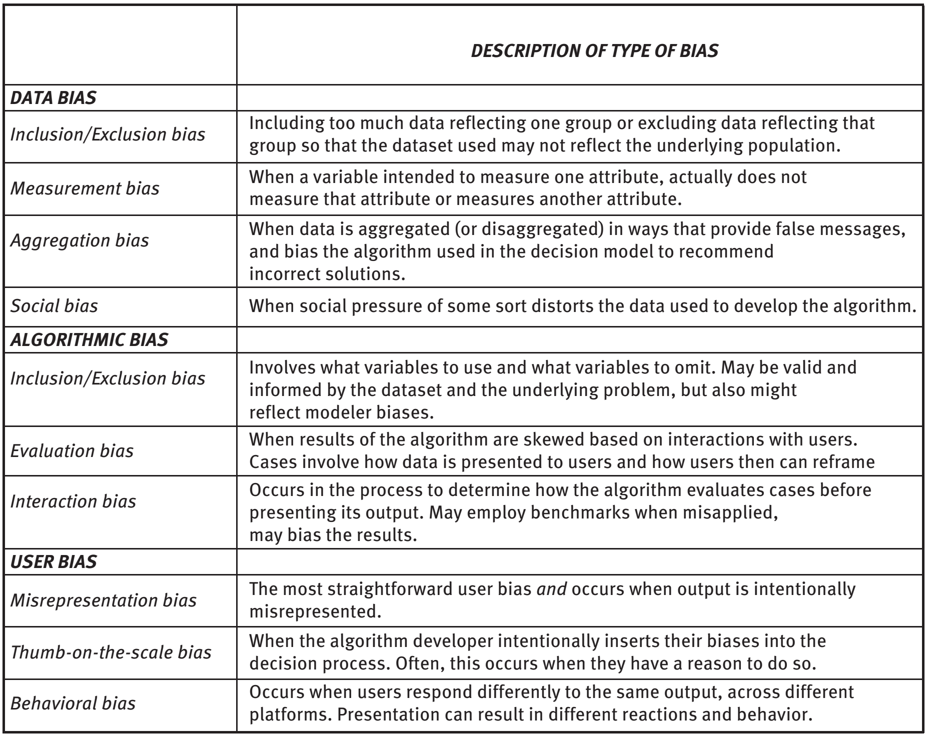
Misrepresentation bias is the most straightforward user bias and occurs when output is intentionally misrepresented. For example, when much output is generated, users can frame the results via graphical means or by other devices such as aggregating output. One frequently used example of doing this, is by using base years in reporting temperatures, where dramatic increases or even declines can result from judiciously picking the base year, or how results are aggregated by geographic region. Weighting output is another way to facilitate misrepresentation, for example by developing special purpose indices and then using these indices to prove the analyst's point.

Thumb-on-the-scale bias occurs when the algorithm developer intentionally inserts their biases into the decision process. Often, this occurs when they have a reason to do so. For example, results may be skewed to support a particular political point-of-view or to validate the desires of an agency or other group funding a study. Rather than misrepresenting results, selective reporting of results can be employed (e.g., by "cherry-picking" output), or results contrary to desired outcomes can be buried in appendices or clouded in convoluted language.

Behavioral bias occurs when users respond differently to the same output, across different platforms. Mehrabi et al. (2019) discuss this bias as occurring "where authors show how differences in emoji representations among platforms can result in different reactions and behavior from people and sometimes even leading to communication errors." An example of this concerns reporting predictions of deaths from flu epidemics, where user proclivity for risk, and users having (or not having) information to place results in context, can lead to different reactions and possibly different decisions made by decision-makers. Figure 6.1 summarizes the above biases and their effects.

6.3.4 How algorithmic bias leads to risk

A recent RAND study (Osoba and Welser, 2017) discusses "misbehaving algorithms" that highlight various generic algorithmic risks. Given the move toward machine learning and artificial agents, they focus on these risks, as increasingly decision-



	DESCRIPTION OF TYPE OF BIAS
DATA BIAS	
Inclusion/Exclusion bias	Including too much data reflecting one group or excluding data reflecting that group so that the dataset used may not reflect the underlying population.
Measurement bias	When a variable intended to measure one attribute, actually does not measure that attribute or measures another attribute.
Aggregation bias	When data is aggregated (or disaggregated) in ways that provide false messages, and bias the algorithm used in the decision model to recommend incorrect solutions.
Social bias	When social pressure of some sort distorts the data used to develop the algorithm.
ALGORITHMIC BIAS	
Inclusion/Exclusion bias	Involves what variables to use and what variables to omit. May be valid and informed by the dataset and the underlying problem, but also might reflect modeler biases.
Evaluation bias	When results of the algorithm are skewed based on interactions with users. Cases involve how data is presented to users and how users then can reframe
Interaction bias	Occurs in the process to determine how the algorithm evaluates cases before presenting its output. May employ benchmarks when misapplied, may bias the results.
USER BIAS	
Misrepresentation bias	The most straightforward user bias *and* occurs when output is intentionally misrepresented.
Thumb-on-the-scale bias	When the algorithm developer intentionally inserts their biases into the decision process. Often, this occurs when they have a reason to do so.
Behavioral bias	Occurs when users respond differently to the same output, across different platforms. Presentation can result in different reactions and behavior.

Figure 6.1: Possible Biases in the Decision process.

making will be occurring in algorithmic mediated environments, where a veil will cover the inner-working behind algorithmic recommendations. RAND points out that artificial agents are "not human" and that as a result, using them can lead to "incorrect, inequitable, or dangerous consequences."

Examples given include biased behaviors, such as those exhibited in the 1990s by the SABRE reservations system which via the model construction, advantaged flights from the system's owner, American Airlines. Similarly, medical resident match algorithms were shown to favor hospitals over medical residents (Friedman and Nissenbaum, 1996). These examples, both developed before the widespread introduction of today's machine learning algorithms, reflect bias in constructing models in the *classical* category mentioned above. Today, similar arguments are being made for firms such as Google and Amazon, who advantage search results to place sponsored content higher or even content, in the case of Amazon, for their private label branded products (Fussell, 2019; Duhigg, 2018).

Another generic risk that the RAND study points out is "algorithmic defamation", where "search engine autocompletion routines, fed a steady diet of historical user queries, learn to make incorrect defamatory or bigoted associations about people or groups of people." For example, a student researching a paper of terrorism or a scholar researching the pornography industry, could be labeled a terrorist or pedophile because of algorithmic inferences based on a history of web searches.

The RAND study also discusses a Citron and Pasquale (2014) study about what they call the "scored society and its pitfalls." Here, algorithms can be used to "produce authoritative scores of individual reputations that mediate access to opportunity. These scores include credit, criminal, and employability scores." Such scoring algorithms derive their bias not only from biases in the algorithm itself, but also in the biases of the datasets used to develop the algorithm. Increasingly these are embedded in processes opaque to the affected individuals.

Finally, the use of massive databases associated with big data creates problems concerning the misuse of consumer data and violation of privacy. This data may be used in decision-making as just discussed, or may be accessed to reveal information and possibly erroneous patterns of information that can negatively affect a person's reputation, employment status, or even legal status.

Initially, there was some thought that machine learning algorithms would always be less biased than individuals constructing algorithms, and perhaps even eliminate all bias, since for machine learning algorithms, people did not explicitly insert their biases in the algorithm's development. As discussed, problems with datasets still can introduce these biases, as do issues within the algorithm itself and user interpretation of results. But, in support of use of algorithms, Kleinberg et al. claim, "when algorithms are involved, proving discrimination will be easier – or at least it should be, and can be made to be. The law forbids discrimination by algorithm, and that prohibition can be implemented by regulating the process through which algorithms are designed."

Kleinberg et al. (2018) also discuss examples where omitting identifying variables, such as race, actually can be counterproductive regarding bias, as without explicit tags (such as race) the algorithm may make inferences from related variables that may be more disadvantageous to minority groups that if race itself were explicitly used as a variable in the algorithm's analysis. This is because explicitly including the variable could counteract implicit inferences and produce more unbiased outcomes.

6.4 Risk and algorithmic bias

Why is bias in algorithms problematic? Three reasons are that algorithmic bias may result in outcomes that are illegal, unethical, and detrimental. All three, lead to risks that must be managed.

Legal reasons are straightforward. As Kleinberg et al. (2018) point out, many cases exist where the law forbids discrimination by algorithm. Biased algorithms, whether the bias is explicit as in the coding of the algorithm itself, or implicit regarding biased outcomes disadvantaging specific groups, may violate the law and expose those who develop and use them to legal penalties.

Even when bias does not explicitly violate the law, biased results that negatively affect various groups may violate ethical and cultural norms. These violations not only harm the groups discriminated against, but also society as a whole. Moreover, when made public, such examples of bias may harm stakeholders including the individuals and organizations that develop and use the algorithms. This may lead to losses in reputation and brand image and equity.

Finally, biased algorithms are harmful as they fundamentally subvert the underlying objective of using the algorithms in the first place. A credit scoring algorithm that denies credit to otherwise credit worthy individuals because they belong to a minority, by definition is suboptimal and violates the economic objectives underlying employing the algorithm. An algorithm that implicitly discriminates against women applying for job on police forces or fire departments, rules out candidates who may elevate the performance of those departments. Ultimately, they weaken the decision-making process and impost opportunity costs. As Debrusk (2018) points out regarding machine learning algorithms, "Creators of the machine-learning models that will drive the future must consider how bias might negatively impact the effectiveness of the decisions the machines make. Otherwise, managers risk undercutting machine learning's potentially positive benefits by building models with a biased 'mind of their own.'"

Given these general reasons and the above discussion, five generic risks related to biased algorithms are as follow:

Legal risk involves the presence of algorithmic bias from selection processes and/or outcomes that violate the law. Legal risk can originate from bias present in the data set, the algorithm itself, or in how users interpret and act on the information produced by the algorithm. Harm can be harm of omission, such as when a person's resume is not selected for employment as a result of a biased process or outcome, or it can be harm of commission, as when bias in an algorithm results in an unlawful arrest or physical harm as a result of misidentification by a facial recognition system. Legal risk can result in legal penalties for individuals and for organizations.

Ethical risk involves the presence of bias in the algorithmic process that, while legal, result in processes and/or that violate ethical standards. Such violation can affect the culture of organizations, stakeholders, and the reputation of the organization with the broader community. In the long-term, regulations put in place to control ethical lapses can also harm the operation of the systems supported by the algorithm, by constraining future performance or even making the processes illegal.

Societal risk occurs when algorithmic models are employed in private or public settings in ways that fundamentally alter the way that society operates, often for the worse. Examples in the past have involved practices such as redlining regarding discriminating against African-Americans receiving mortgages, and perhaps in the future, using facial recognition systems to restrict individual movement and freedom, such as is reportedly taking place in China (Mozur and Krolik, 2019).

Short-term effectiveness risk involves the immediate effectiveness of systems supported by the algorithmic process. This could involve a student being denied admission to a college based on bias in a dataset or algorithm, or a good credit risk who is a member of a minority being denied credit based on bias based on their group membership. Short term effectiveness risk subverts the immediate objectives of the systems the algorithm has been developed to support.

Long-term effectiveness risk involves the long-term effectiveness of systems supported by the algorithmic process. This might involve many students over long periods of time being denied admission to a college based on bias admission processes reflected in datasets or embedded in algorithms, or many members of a group, such as women, receiving inequitable pay based on algorithmic models used to determine compensation. Long-term effectiveness risk subverts the long-term objectives of the systems the algorithm has been developed to support.

Figure 6.2 presents a grid that maps the forms of bias discussed against the five risks. An organization can use this grid, or one like it, to assess how various types of bias in a particular situation expose the organization to various types of risks. For example, the ranking could be on a three-point scale of Low-Medium-High, or by including specific scenarios, particular to that organization. When finished, a completed grid such as that in Figure 6.2 could provide information which decision-makers could use to not only assess where bias is present in the algorithmic process, but also estimate what is the organization’s risk exposure.

Finally, given that algorithms are used in the decision-making process, one may ask whether bias is greater with an algorithm where a modeler directly constructs the algorithm and knows the effects of the variables within the model, or with a machine learning algorithm where the relationships may be unknown and where the algorithm’s output is the result of a dataset and a training process.

In both cases, once the output appears, user bias can emerge. As noted, Kleinberg et al. (2018) argue that machine learning algorithms may enable proving discrimination easier than when non-algorithmic decision methods are used. Indeed, as noted above, they also discuss cases where explicitly tagging individuals in data sets with racial and other markers might actually reduce discrimination, since if untagged, the training and decision algorithm process itself might inject more bias into the overall process.

	TYPE OF RISK				
	Legal Risk	*Ethical Risk*	*Societal Risk*	*Short-term Effectiveness Risk*	*Long-term Effectiveness Risk*
DATA BIAS					
Inclusion/Exclusion bias					
Measurement bias					
Aggregation bias					
Social bias					
ALGORITHMIC BIAS					
Inclusion/Exclusion bias					
Evaluation bias					
Interaction bias					
USER BIAS					
Misrepresentation bias					
Thumb-on-the-scale bias					
Behavioral bias					

Figure 6.2: Interaction between Bias and Risk.

6.5 Strategies to reduce bias and manage risk

As algorithmic bias research is still in its infancy, few generally accepted methodologies with proven effectiveness exist for controlling algorithmic-based bias and the ensuing risks. Surveys and studies are ongoing (e.g., RAND by Osoba and Welser, 2017; McKinsey by Silberg and Manyika, 2019; Dwork et al. 2012) and the ideas below are informed by these results.

6.5.1 Database strategies

Database and Algorithmic Transparency shines a light on how data are collected, how databases are structured, how algorithms are structured and operate, and what results and recommendations are generated. A benefit from transparency is that the information underlying the decision process is made available to all, which enables heretofore bias in data and models to be uncovered. A downside of transparency is that since many databases and machine learning models are proprietary (e.g., Google searches, Amazon recommendations, facial recognition systems, bank credit scoring algorithms), private companies may be reluctant to share "their special sauce" with the general public.

One possibility to overcome this is for transparency at different levels for different groups, with the highest level being available to a select group of auditors. Dwork et al. (2012) proposed, as did Kleinberg et al. (2018), the counterintuitive idea involving making sensitive information available when doing data classification – e.g., tagging data with membership in a racial minority. This may actually be used to reduce bias in the ongoing analysis.

Statistical Approaches apply statistical analysis to potential bias in datasets and algorithms. The RAND study mentions one approach where, "Dwork et al. (2012) proposed using modified distance or similarity metrics when working with subject data. These similarity metrics are meant to enforce rigorous fairness constraints when comparing subjects in data sets. Sandvig et al. (2014) proposed a number of algorithm auditing procedures that compare algorithmic output with expected equitable behavior."

The RAND study points out that while statistical methods may correct for inequalities, using these methods may "often trade some predictive power for fairness." Beyond the potential power involved, embedding statistical approaches in the process my enable uncovering bias in ways similar to how statistical approaches in quality management are able to identify and control variation in manufacturing processes.

Causal Reasoning Approaches involve equipping machine learning algorithms with causal or counterfactual reasoning (Pearl, 2009; Loftus et al. 2018). Causal and counterfactual reasoning is employed in ascertaining algorithmic fairness. At its most elementary, causal reasoning is illustrated as in the case where, if A causes B, changing something in A will also change B, other elements held constant. This logic can be extended to evaluating algorithms and their output and inputs. The RAND study claims that, " . . . automated causal reasoning systems can present clear causal narratives for judging the quality of an algorithmic decision process. Accurate causal justifications for algorithmic decisions are the most reliable audit trails for algorithms." These approaches have been applied in examining bias in sentencing of inmates and may be applied to many of the other bias examples discussed above.

Auditing Approaches involve deciding on a measure of fairness regarding what bias is, and then examining inputs (e.g., the dataset), the algorithm, and the outputs in order to measure how much bias exists. From the results, risks can then be assessed and interventions made to correct biases. While auditing approaches can employ standard auditing methods, tailored for algorithmic environments, other approaches used in risk assessment and risk management also can be employed. One example is the take a preliminary risks assessment using an instrument such as that in Figure 6.2, and then to use this in the audit analysis. Another approach (Miller and Engemann, 1996) is to develop risk scenarios, and based on these, assess risk and develop mitigation and control solutions.

Personnel-based Approaches draw on human behavior to identify and eliminate bias. Transparency is one method, as shining a light on biased datasets and biased algorithms is a first step toward reducing bias and managing risk. Education in how bias enters the decision process creates awareness on the part of the engineers and scientists creating the models, and facilitates risk management. As many people developing algorithms perhaps have less formal training in ethics and perhaps even in algorithmic fairness, further education and training can raise their awareness and modify their behavior. A second method mentioned in the McKinsey study is increasing the diversity on the team of engineers and scientists developing models, which although not guaranteeing unbiased results, can raise awareness toward scenarios where bias can enter the process.

6.6 Conclusion

A truism in risk management is that risk cannot be eliminated, but can be managed. A similar statement can be made regarding bias. Because computer-based decision-making models have become so ubiquitous in everyday life, managing the risks associated with bias is increasingly important.

As we have discussed above, bias enter the process through multiple doorways: through data, through the algorithm and through the interpretation of results. Bias can be explicit or implicit. Explicit bias can be more easily identified and controlled, but machine learning algorithms, even when developed with no intention toward bias of any type, can evolve and generate results that themselves are biased. Moreover, these biases may be identified only ex-post, when the damages may be greater. Examples not only include biases toward groups as defined by gender, race, ethnicity, or disability, but also biases for or against certain brands, product types, and sources of information. The results can be as innocuous as poor placement on an internet search, to as serious as someone being wrongly imprisoned because of a facial recognition error.

In this chapter we have sought to highlight the issue and some of the challenges that lie ahead. Increasingly, bias may be seen not as only as something that exists, but as something that creates risks for the organization using a decision system, similar to other risks that the organization encounters. Just as with these risks, bias can be identified and then managed using strategies for mitigation and reduction. We have presented some methods of doing above, and they may be used to address the presence of bias. Attacking bias from multiple directions may lead to better management of bias risks, and lead to better solutions. In so doing, we can be more confident that the results of the algorithms that govern our 21st century lives reflect a level playing field, where no one group is advantaged or disadvantaged, and where the algorithmic results are as effective as possible.

References

Citron, Danielle Keats and Pasquale, Frank A., The Scored Society: Due Process for Automated Predictions (2014). Washington Law Review, Vol. 89, 2014, p. 1-; U of Maryland Legal Studies Research Paper No. 2014-8. Available at SSRN: https://ssrn.com/abstract=2376209

DeBrusk, Chris (March 26, 2018). The Risk of Machine-Learning Bias (and How to Prevent It), MIT Sloan Management Review, Accessed at https://sloanreview.mit.edu/article/the-risk-of-machine-learning-bias-and-how-to-prevent-it/.

Duhigg, Charles. The Case Against Google, New York Times. (February 20, 2018). (Accessed at: https://www.nytimes.com/2018/02/20/magazine/the-case-against-google.html)

Dwork, Cynthia, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. Fairness through awareness. In Proc. ACM ITCS, pages 214–226, 2012.Accessed at: https://arxiv.org/abs/1104.3913

Engemann, Krista. N., and Kurt. J. Engemann, "Risk Attitude Chain: Safety Climate, Risk Attitude and Risk Decisions," *International Journal of Business Continuity and Risk Management*, V7N (3), pp. 211–221, 2017.

Engemann, Kurt J. and Holmes E. Miller, 'Risk Strategy and Attitude Sensitivity,' *Cybernetics and Systems*, Vol. 46, No. 3, pp. 188–206, 2015.

Friedman, Batya, and Helen Nissenbaum, "Bias in Computer Systems," ACM Transactions on Information Systems, Vol. 14, No. 3, July 1996, pp. 330–347.

Fussell, Sidney. Algorithms Are People, The Atlantic. (September 18, 2019). Accessed at: https://www.theatlantic.com/technology/archive/2019/09/is-amazons-search-algorithm-biased-its-hard-to-prove/598264/)

Kleinberg, Jon and Jens Ludwig, Sendhil Mullainathan, and Cass R. Sunstein, Discrimination in the Age of Algorithms, Journal of Legal Analysis, Volume 10, 2018, Pages 113–174, https://doi.org/10.1093/jla/laz001

Loftus, Joshua R, Chris Russell, Matt J. Kusner, Ricardo Silva. Causal Reasoning for Algorithmic Fairness. May, 2018. Accessed at: https://arxiv.org/pdf/1805.05859.pdf

Mehrabi, Ninareh, and Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. (2019). A survey on bias and fairness in machine learning. CoRR, https://arxiv.org/abs/1908.09635.

Miller, Claire C. (Aug. 10, 2015). Algorithms and Bias: Q. and A. With Cynthia Dwork, New York Times. P. A3.

Miller, Holmes and Kurt Engemann. A Methodology for Managing Information-based Risk, Information Resources Management Journal, (1996), 9:2; p. 17–24.

Mozur, Paul. One Month, 500,000 Face Scans: How China Is Using A.I. to Profile a Minority, The New York Times, (May 6, 2019). Assessed at https://www.nytimes.com/2019/04/14/technology/china-surveillance-artificial-intelligence-racial-profiling.html

Mozur, Paul and Aaron Krolik. (December 17, 2019). A Surveillance Net Blankets China's Cities, Giving Police Vast Powers, New York Times. Accessed at: https://www.nytimes.com/2019/12/17/technology/china-surveillance.html

Nicas, Jack. How YouTube Drives People to the Internet's Darkest Corners. The Wall Street Journal. (February 7, 2018). Accessed at: https://www.wsj.com/articles/how-youtube-drives-viewers-to-the-internets-darkest-corners-1518020478

Osoba, Osonde A. and William Welser IV, An Intelligence in Our Image: The Risks of Bias and Errors in Artificial Intelligence. Santa Monica, CA: RAND Corporation, 2017. https://www.rand.org/pubs/research_reports/RR1744.html. Also available in print form.

Pearl, Judea, Causality, Cambridge: Cambridge University Press, 2009.

Sandvig, Christian, Kevin Hamilton, Karrie Karahalios, and Cedric Langbort, “Auditing Algorithms: Research Methods for Detecting Discrimination on Internet Platforms,” paper presented to the Data and Discrimination: Converting Critical Concerns into Productive Inquiry preconference of the 64th Annual Meeting of the International Communication Association, Seattle, Wash., May 22, 2014.

Silberg, Jake and James Manyika. Notes from the AI frontier: Tackling bias in AI (and in humans), McKinsey Global Institute, (June 2019). Accessed at: MGI-Tackling-bias-in-AI-June-2019.ashx

Singer, Natasha and Cade Metz. “Many Facial-Recognition Systems Are Biased, Says U.S. Study”, New York Times. (December, 2019). Accessed at: https://www.nytimes.com/2019/12/19/technology/facial-recognition-bias.html

Smith, Craig. Dealing with Bias in Artificial Intelligence. (January 2, 2020). The New York Times. Accessed at: https://www.nytimes.com/2019/11/19/technology/artificial-intelligence-bias.html

Part II: **Applications**

Subhajit Datta, Amrita Bhattacharjee and Subhashis Majumder

7 Interactional motifs: Leveraging risks in large and distributed software development teams

7.1 Introduction

7.1.1 Software development is about people

DeMarco and Lister begin their classic *Peopleware* with an air of ominous inevitability "somewhere today, a project is failing" (DeMarco and Lister, 2013). They are talking about software projects, and as the book so brilliantly establishes, software is *peopleware*. A failed project is the dreaded culmination of all the perceptible and imperceptible risks that are associated with the project. For software projects, a large majority of such risks originate in the interactions of people who are involved in the project. People who build the software are the most valued and the most vulnerable asset of any software project, something that has been recognized ever since software became a large scale industrial enterprise (Brooks, 1995; Weinberg, 2011; Meyer, 2019). However, over past decade and half, global teams have become the primary vehicle for large scale software development. This has elevated the importance of developer interaction in the understanding and mitigation of software development risks.

In this chapter, we present a perspective of developer interaction using the lens of *motifs*. Through a case study using development data from a large real-word system involving 2000+ individuals and 150000+ units of work, we demonstrate how a motif based view can endow a deeper sense of two of the critical drivers of software development risk – workload and task completion time.

7.1.2 People and their interactions

Large scale software development ecosystems offer an interesting context for studying interactions between individuals engaged in a collective enterprise. As has been investigated across disciplines, such interactions have many nuances, each with its own implications on outcomes at individual and team levels (Guimera et al., 2005). With globally distributed teams now being the norm in large scale software development, it is imperative that we have a thorough understanding of how members of such teams fulfil collective responsibilities such as peer review of code. Developers engage in such

https://doi.org/10.1515/9783110652321-007

an activity in a *networked* context, with information and insights being shared across communication channels.

Networks – as a form of representation of connected entities – have been used in several domains such as social networking sites (Adamic et al., 2003), (Boyd and Ellison, 2007), passenger traffic along flight pathways (Rosvall et al., 2014), (Salnikov et al., 2016), protein-protein interaction(PPI) models (Pržulj et al., 2004) and transcription networks (Mangan and Alon, 2003). The ability to encode interactions between agents into the structure and dynamics of the network, abstracting out the domain-specific complexities, has made it possible to apply strong graph theoretic concepts, approaches and algorithms in the analysis of these complex data.

An interactive network for a group of individuals can be generated where pairs of them are linked by their co-participation in some activity. Characteristics of such interaction networks are then expected to offer insights on interaction dynamics of interest. Of all the different types of real world interactions modelled as networks, we focus on developer social networks (DSNs). Nowadays, with the proliferation of highly collaborative open source software(OSS) projects, representing and understanding developer interactions and relationships as a network can improve development outcomes, reduce costs and make quality control of software projects more effective. The concept of local structure in DSNs has the potential to reveal more subtle patterns in the way developers collaborate in a certain context or setting.

7.1.3 Characteristics of interaction

In the study of these networks, clustering is often taken to reflect on an important characteristic of interaction. The *clustering coefficient* (Newman, 2003) essentially measures the extent of triadic closure in the network, and highlights the criticality of triads – sets of three vertices some or all of which may be connected to one another – in the flow of information in the network. Clustering has been found to be related to several consequential aspects of the systems underlying the networks represent (Barabasi et al., 2001; Guimera et al., 2005; Datta, 2018).

Researchers over the years have studied and explained how social networks differ from other types of networks. Usually social networks are known to be scale-free, have high transitivity and preserve the "small-world" property (Watts and Strogatz, 1998). Transitivity refers to the property by which, if vertices A and B are connected by an edge, and if B and C are connected by another edge, A and C will also be connected, thus forming a triad. Social networks are usually highly clustered – an outcome of high transitivity – as these triangles are the building blocks of the clusters. Wasserman & Faust further explained in (Wasserman and Faust, 1994) how positive and negative ties among participants in such triads affect the balance of the network and, in turn, the "clusterability" of the network. Thus, keeping in mind that these triads are the fundamental units of larger structures of collaboration, we examine them with a deeper focus, with a view to a better understanding of developer interaction characteristics.

7.1.4 Interactive patterns

In this chapter, we posit that a closer examination of triadic relationships – beyond merely measuring the extent of triadic closure – can be useful in a software development ecosystem. The very nature of the software development enterprise, with its inherent dichotomy and challenges (Weinberg, 1971; Brooks, 1995; DeMarco and Lister, 1987) leads to a variety of patterns in the interaction between developers. We observe that the notion of network motifs (Milo et al., 2002) has been used to explain the behavioural characteristics of interaction networks in many domains. Motifs are patterns of interaction among a small number of nodes that occur much more frequently in certain types of real-world networks than in a collection of randomly generated networks. Several types of motifs have been identified and enumerated by researchers over the years; Benson *et al.* (Benson et al., 2016) identified 13 different types of 3-node motifs and the 4-node "bifan" motif in directed networks. Yaveroğlu et al. (2014) identifies 30 different 2 to 5 node "graphlets" which are induced subgraphs with undirected edges. According to Milo et al. (2002), these network motifs may be seen as *"structures that arise because of the special constraints under which the network has evolved"* (Callaway et al., 2001). For example, in information- processing networks, the motifs may function as elementary computational circuits (Shen-Orr et al., 2002). Thus motifs can illuminate local structures in interaction networks that more coarse grained measures may miss. Examination of local structures has the potential to reveal subtle patterns in the way developers collaborate in a particular project setting.

7.2 How do motifs matter?

The variety comes from the diverse set of activities that collectively define the software development life cycle (Jacobson et al., 1999). Thus it is natural that there will be different patterns of connection between triads of developers. As outlined in Section 7.1, motifs offer a framework to isolate and study these patterns. Figure 7.1 shows the motifs identified by Benson et al. in (Benson et al., 2016). As we observe,

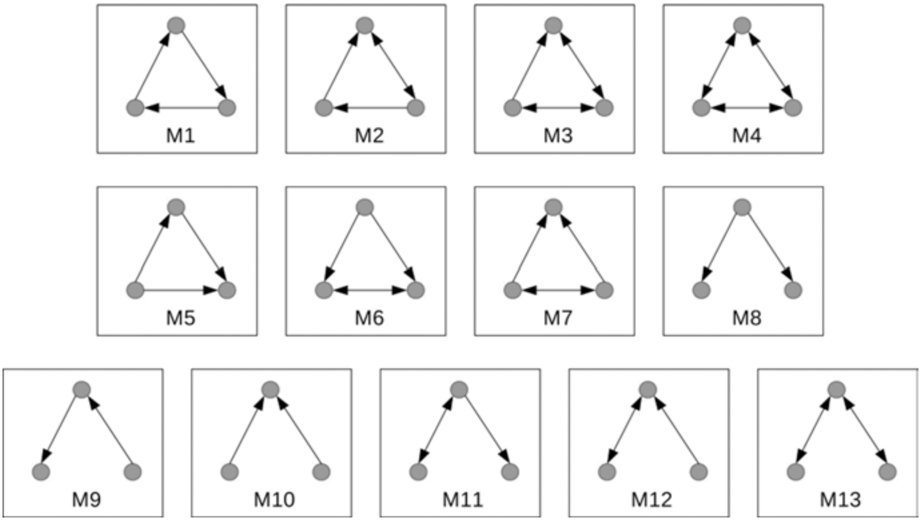


Figure 7.1: The 13 types of motifs present in our developer social network; motif names as given by Benson et al. (2016).

each motif is a particular patterns involving three vertices, some or all of which are joined by links that can be unidirectional (arcs) or bi-directional (edges) in nature.

In a software development context, with the vertices as developers, each motif represents a particular interaction pattern involving the triad. For example, M1 is an instance of *circular dependency* where members of the triad are connected in a cycle; M8 typifies *fan out*, where information flows out from one vertex to the other two vertices; M10 is a reflection of *fan in*, with information flowing in to one vertex from the other two vertices. The 13 motifs shown in Figure 7.1 cover all possible combinations of connections between members of a triad. Each motif is thus a template of a specific type of interactional relationship between trios of developers.

In a software project, developers fulfil different roles as they are deployed on various tasks. A developer's role in a project may be defined by her general level of seniority or experience; or by specific expertise. Given the diversity of developer roles and the variety of development tasks, it is quite likely that developers' interactions with their peers will also be varied. Each developer's pattern of interaction is reflected in her level of involvement in different motifs across the set of motifs identified earlier; we call this a developer's *motif profile* (as defined formally in Section 7.4). We believe motif profiles can be an important indicator of individual outcomes in a large scale software project, which leads us to the question:

In a large scale software project, how do developer motif profiles relate to their performance?

Developer performance has many dimensions; in this study we focus on two aspects: *workload* and *task completion time*. As explained in detail in Section 7.4, the former is measured in terms of the number of work items a developer is responsible for, and the latter, in terms of the time required to complete those work items.

As discussed earlier, each motif represents a specific pattern of triadic interaction among developers. In a given interaction network, the number of times a developer participates in a specific motif indicates her level of involvement with the corresponding pattern. Developers participate in different motifs to different extents. Across the entire set of 13 motifs, the distribution of the number of motifs a developer is involved in, will vary from one developer to the other. In the context of this study, we define the *uniformity of interaction* of a developer to denote how evenly spread out her involvement across the different motifs are. This reflects whether a developer's interactions are predominantly concentrated around few motifs, vis-a-vis being spread out across the entire set of motifs. As uniformity of interaction across the motifs is a key indicator of how a developer interacts with her peers, we believe this can also inform how developers perform. Accordingly, the question introduced earlier is distilled into the following hypotheses, which we address in our case study.

H1: For developers, higher uniformity of interaction relates to higher workload. The null hypothesis corresponding to H1 is that there is no relationship between uniformity of interaction and workload.

H2: For developers, higher uniformity of interaction relates to higher task completion time. The null hypothesis corresponding to H2 is that there is no relationship between uniformity of interaction and task completion time.

7.3 Existing studies

7.3.1 Network motifs

In recent times, open source software development has become increasingly pervasive, leading to a lot of research being conducted to understand the structure and behaviour of developer social networks. Hong et al. (Hong et al., 2011) compares the developer social networks (DSNs) with general social networks (GSNs) and also studies how these networks evolve over time. Zhang et al. (2014) elucidates the construction, analysis and application of DSNs in detail. The authors explain the different methods of constructing DSNs based on project participation, version control system, email archives, bug tracking systems, etc. Apart from these there are also some hybrid DSNs which may have heterogeneous types of entities as its nodes. For example, there can be networks where the links can be from module to module, based on software dependencies, or developer to module, based on ownership, etc. Interactions in collaborative networks (Hahn et al., 2008; Datta et al., 2015) of software developers can be defined in many ways – two developers may have a link between them if they have been involved in the development of the same module, or if they have co-edited or co-commented on reviews, etc. Ever since Milo et al. (2002) introduced the concept of motifs, there has been extensive research on the implication of presence or absence of different motifs in various kinds of networks. Several algorithms for efficient detection of motifs in networks have been developed and proposed (Romijn et al., 2015), (Wernicke, 2006). Motif based approaches to clustering and community detection have been tried out by researchers over the years with varying degrees of success. Benson et al. (Benson et al., 2016) put forth a generalized framework for spectral clustering of networks based on local structure in the form of motifs with *“mathematical guarantees on the optimality of obtained clusters”*. This might further imply the possibility of these motifs being the building blocks of real-world complex networks, hence governing the behaviour and dynamics of the network. At the far end of the spectrum, in the context of transcription regulation networks that control gene expression, Alon (Alon, 2007) showed the significance of the 3-node feed-forward motif, among several others. The function of different types of feed-forward loops have been further studied and worked on by Mangan and Alon (Mangan and Alon, 2003) and also by Shen-Orr et al. (2002) based on *E. Coli* transcriptional regulation network. Each of these studies reiterated the importance of looking at motifs as individual functional units that form the basic building blocks of complex networks, including neuronal and other biological networks. In the context of this study, we shift our focus to the domain of social networks. Networks encode a significant amount of information which can be retrieved by graph mining. Most social networks, in particular, are known to have scale-free properties arising out of gradual growth by addition of new nodes and preferential attachment (Barabási and Bonabeau, 2003; Barabási et al., 2000), and they also exhibit closely-knit cluster or community structure (Girvan and Newman, 2002). These community structures involving subsets of nodes in a network, come into being as a result of certain attributes that are shared among these nodes, i.e. the concept of homophily (McPherson et al., 2001). For example, in a network representing connections among people in a social networking site, one can expect denser interconnections among employees working at a particular organization, or among people from a specific geographical area, etc., thus giving rise to communities in the network. Efficient community detection in large social networks has been an active research problem and several algorithms have been proposed, many of which are in use today.

7.3.2 Software code review

With increasing incidence of global software teams, the complexities of code review activities in such teams have received research attention. Rigby, German and Storey examined the peer review techniques of review-then-commit, and commit-then-review that are leveraged in the Apache server project, and offer a set of observations on the dynamics of peer review processes in that project (Rigby et al., 2008).

Rigby and Storey performed an empirical study to find out how developers discern units of code change for their review, and how stakeholders in the review process interact (Rigby and Storey, 2011). In addition to identifying defects, reviews are found to facilitate heightened awareness among team members and knowledge transfer within a team, in a study of software development teams at Microsoft by Bacchelli and Bird (2013). Baysal et al. report that non-technical factors related to organizational and personal dimensions notably influence code review outcomes in a study of code review processes of a large, open source project (Baysal et al., 2013). Rigby et al. have concluded that "that conducting peer review increases the number of distinct files a developer knows about by 66% to 150% depending on the project" (Rigby and Bird, 2013) The relationships between software quality, code review coverage and code review participation in Qt, VTK, and ITK projects have been studied by McIntosh et al.; they found that that poorly reviewed code has a detrimental effect on software quality in large systems (McIntosh et al., 2014). There is evidence that bug-fixing activities relate to fewer changes, and activities with more modified files and higher levels of code churn have more modifications (Beller et al., 2014). On the basis of examining 25 OSS projects and the historical records of six large, mature, and successful OSS projects, it has been concluded that peer review in open source software is "drastically" different from code inspection in traditional settings (Rigby et al., 2014). As evident from the studies outlined above, understanding peer review processes offer diverse insights. Our current work complements existing results by presenting a network motif based perspective of developer interaction in the review process.

7.4 Interaction motifs and developer outcomes: A case study

With reference to Figure 7.2 we highlight the major steps in the methodology of our study. Starting from the top left corner of the figure, the following sequence of activities are performed: accessing review data from online repositories, filtering and cleaning the data, constructing developer interaction networks on the basis of co-commenting on review items, identifying the motifs of interest from the network, computing model variables, developing statistical models, using the model outputs to validated the hypotheses, examining the results in the light of existing literature, and deriving insights from the study.

7.4.1 Study setting

To understand our study setting, a brief background of the typical review life cycle will be helpful (Datta et al., 2015). A unit of code that needs to be reviewed is first identified, followed by its review by one or more developers who have not been involved in its development. While reviewing a code unit, developers discuss by exchanging comments, and through this process, an agreement is hopefully reached on what needs to be changed. Accordingly, the changes are implemented in the code unit and tested for residual bugs. Subsequently, the code unit is either approved to be merged with the main body of code, or abandoned (if no consensus emerged during the review process, or the suggested changes could not be satisfactorily implemented); followed by the review being closed. As evident, peer reviewing offers valuable scope for developers to share their perspectives on the development ecosystem (Rigby and Bird, 2013). It is in the best interest of all stakeholders to have

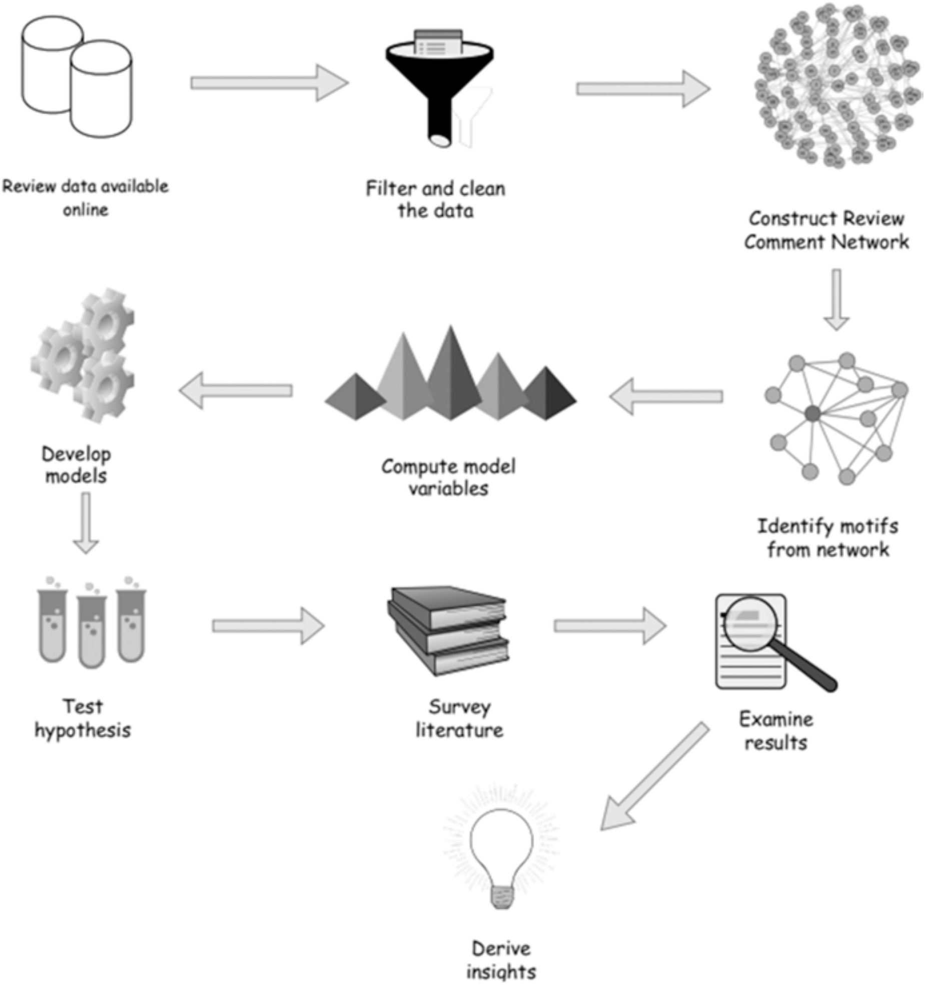

Review data available online
Filter and clean the data
Construct Review Comment Network
Develop models
Compute model variables
Identify motifs from network
Test hypothesis
Survey literature
Examine results
Derive insights

Figure 7.2: Study Setting.

reviews closed quickly. In the event they cannot be worked upon and closed due to factors outside the project's purview, they need to be parked aside or abandoned.

7.4.2 Review comment network

We constructed a review comment network (RCN) using a protocol described in one of our earlier studies (Datta et al., 2015). The original code review data from the Chromium[1] development project was curated and made available by Hamasaki et al. (2013), which was used for constructing the RCN. Drawing on co-commenting data from 159625 reviews, the network is composed of 2208 developers who map to the vertices (nodes), and an arc (unidirectional link) exists between two developers *u* and *v* if *u* has been the sender of at least one message whose receiver has been *v*, during the review process. We removed zero-degree or singleton vertices from the network, as these vertices cannot be part of any motifs. After this pruning, our network consisted of 2139 vertices and 95951 arcs.

7.4.3 Motif identification

To get the frequency of occurrence of each of the 13 types of motifs(Benson et al., 2016), we used code from the SNAP API (http://snap.stanford.edu/). The SNAP code uses an efficient and scalable algorithm (Wernicke, 2006) to find counts of each type of motif in a given network. We then found out how many times each node participated in a particular type of motif, and repeated this for all the 13 motifs. We also calculated standard network metrics for all the vertices in our network and compiled all these results into a data-set, which was used for subsequent analysis.

7.4.4 Model variables

To examine our hypotheses in the light of real world data, we need to build models that can identify statistically significant relationships between our parameters of interest. In the context of this study, we consider uniformity of developer interaction as the *independent variable* (IV) whose effect on the respective *dependent variable* (DV) for each hypothesis – workload, and task completion time – we wish to study. However, the relationship between independent variable and the dependent variable(s) can be influenced by the presence of peripheral factors, as identified from experience and

[1] https://www.chromium.org/.

literature survey these are included in our models as *control variables* (CV). Let us now describe how these model variables are calculated in the context of this study.

7.4.4.1 Independent variable

The *motif profile* P_i for a developer D_i is defined as a *tuple* containing the counts of the different motifs that developer is involved in, in the review comment network (*RCN*).

$$Pi(RCN) = (CM1,\ CM2,\ CM3,\ \ldots,\ CM13)$$

So, each P_i captures the spread of the corresponding developer's motif counts. Figure 7.3 shows the distribution of the number developers participating across the

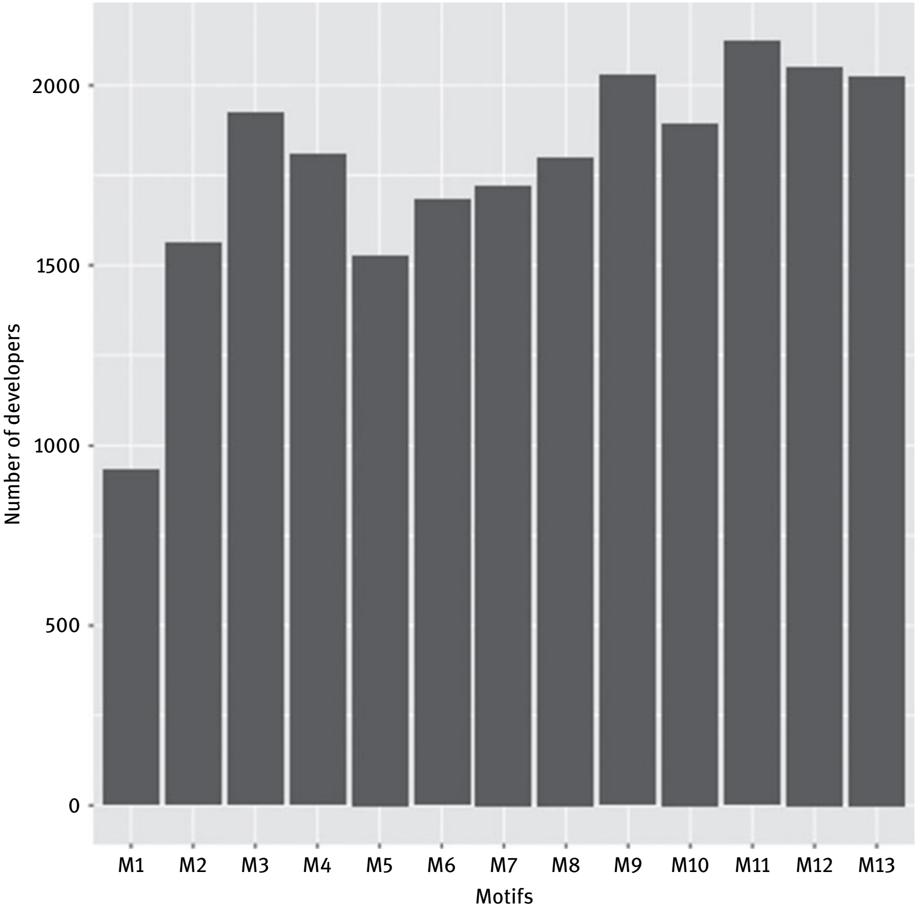


Figure 7.3: Developer Distribution across Motifs.

set of motifs. Barring very few, motifs appear to have attracted developer participation to reasonably similar extents. Thus no motif seems to have a preponderance in the network we are studying.

A developer's **Uniformity of interaction** (referred to as *Uniformity* in subsequent discussion) is defined as $1 - G(P_i)$, where $G(P_i)$ is the Gini coefficient of the motif counts in the developer's interaction profile P_i. The Gini coefficient is an established measure of inequality and has been widely used in different contexts (Gini, 1921).

7.4.4.2 Dependent variable

As mentioned earlier, we seek to examine two aspects of developer performance in this study: workload and task completion time. In the context of hypothesis *H*1, the dependent variable **Workload** for a developer is defined as the total number of reviews owned by the developer in our study period. Developer workload has been identified as a key parameter influencing team outcomes in similar studies (Cataldo et al., 2006; Cataldo et al., 2008; Wagstrom et al., 2010). For testing hypothesis *H*2, we define the dependent variable **TaskCompletionTime** for a developer as the median of the elapsed times between the opening and closure of all reviews owned by that developer. As in any other industrial activity, in software development also how quickly tasks are completed is of essence (Datta, 1998). However, given the nature of our study, the distribution of the elapsed times is expectedly not close to a normal distribution. Thus the median, rather than the mean is a more effective measure of central tendency. We recognize that workload and task completion time can be measured in other ways.

7.4.4.3 Control variables

While examining the relationship between the independent variable and the dependent variable for each hypothesis, we account for the effects of the following control variables:

- **Interest**: As identified in existing literature, the extent to which a developer is interested in a particular task, influences the outcome of the task (Wagstrom et al., 2010). In the context of this study, the level of interest of a developer in the review process is defined as the number of reviews commented upon by the developer.
- **Engagement**: How closely a developer is engaged in a particular activity can be measured in different ways (Wagstrom et al., 2010). For our study, the extent to which a developer is engaged in the review process is taken to be reflected in the number of reviews approved by the developer.

- **Experience**: Many studies on software development ecosystems have established that developer experience is related to the quality of work products (Espinosa et al., 2007; Faraj and Sproull, 2000; Boh et al., 2007; Cataldo et al., 2006; Cataldo et al., 2008; Wagstrom et al., 2010). In this study, the period of time between the earliest and the latest comment by a developer in the set of reviews we are studying is taken as a proxy for that developer's level of experience.
- **Dissemination**: The influence of team structure – often abstracted as a network – on information sharing within the team has been studied for long (Curtis et al., 1988; Wolf et al., 2009; Meneely et al., 2011; Grewal et al., 2006). Drawing from some of the existing findings we consider a developer's out-degree in the RCN as a measure of the information disseminated by the developer to her peers.
- **Reception**: On the basis of similar considerations, a developer's in-degree in the RCN is taken as a measure of the information received by the developer from her peers.
- **Authority**: The *pagerank* of a vertex is often taken as a proxy for the measure of the vertice's importance or authority in a network. especially when the network represents an ecosystem of interacting individuals (Qiu et al., 2009). Thus in our context, the level of authority of a developer is defined as her pagerank in the RCN.

7.4.5 Modelling approach

Having identified the independent variable, control variables, and dependent variables for the hypotheses we are testing, we seek to develop statistical models that will help isolate the relation between the independent variables and the dependent variable, after accounting for the effects of the control variables. In the next section, we highlight the considerations in the choice of modelling paradigms, describe the models, and discuss the implications of the results from the models.

7.5 Results and perspectives

7.5.1 Choice of modelling paradigms

We develop two separate sets of models, each of which will test one of the hypotheses. We have two models in each set. The first model in a set includes the dependent variable and the control variables; whereas the second model additionally includes the independent variable. For each hypothesis, by comparing the results of the two models – without and with the independent variable – we will be able to identify how the independent variable relates to the respective dependent variable after accounting for the control variables. We describe below how the modelling paradigm for each model is selected.

7.5.1.1 Regression model for hypothesis H1

For hypothesis H1, we examine the relationship between *Uniformity* and *Work- load* after controlling for the effects of known influences. The correlations of the independent and control variables with the dependent variable are shown in Table 7.2. As *Workload* is measured in terms of the number of review items owned by a developer, this dependent variable is a *count*. We thus first considered Poisson regression as the modelling paradigm. In a Poisson distribution, the mean is equal to the variance of the random variable. A strong assumption underlying Poisson regression is the requirement for the mean and variance to be reasonably close to one another. Thus, overdispersion – violation of this assumption – can be a major threat to the validity of Poisson regression results (Barron, 1992). As evident from Table 7.1, this threat is present in this study to a major extent; hence Poisson regression is not suitable. Negative binomial regression is useful for over dispersed count data. It can be considered as a generalization of Poisson regression, as it includes an additional parameter to address overdispersion. Thus, negative binomial regression is used to develop the models used for testing hypothesis H1 (Long, 1997). In Table 7.4 we present parameters describing the negative binomial regression models such as the Akaike information criterion (Akaike, 1974).

7.5.1.2 Regression model for hypothesis H2

For hypothesis H2, we examine the relation between *Uniformity* and *TaskCompletionTime* after controlling for the effects of known influences. The correlations of the independent and control variables with the dependent variable are shown in Table 7.3. In this study, *TaskCompletionTime* is measured as the median of the elapsed times between the opening and closure of all reviews owned by a developer; it is evidently a *continuous* variable. As the dependent variable is continuous in nature, multiple linear regression can be considered as a modelling paradigm (Tabachnick and Fidell, 2007). The assumptions underlying multiple linear regression include linearity, normality, and homoscedasticity of the residuals, and absence of multicollinearity between the independent variables. We used histograms, Q-Q plots and scatter plots of the standardized residuals to verify the residual properties, and concluded that the assumptions are reasonably met in our context.

In Table 7.5 we present results from the multiple linear regression model to examine hypothesis H1. The H2.I model is described on the left, which only considers the control variables and the dependent variable; whereas the H2.II model on the right additionally includes the independent variable. As specified in the table caption, superscripts of the coefficients denote ranges of their respective *p* values. Calculation of the *p* value for each coefficient is based on the t-statistic – the ratio of each coefficient to its standard error – and the Student's t-distribution.

After the coefficients for each model variable, we describe the overall model in terms of the following: *N* as the number of data points in the model, in this case the number of developers; R^2 as the coefficient of determination, expressing the ratio of the regression sum of squares to the total sum of squares as an indication of the goodness-of-fit; *df* as the degrees of freedom; *F* as the Fisher F-statistic – the ratio of the variance in the data explained by the linear model divided by the variance unexplained by the model. The *p* value for the overall model is calculated using the F-statistic and the F-distribution. It indicates whether the overall model is statistically significant. On the basis of null hypothesis significance testing, if *p level of significance* – for the coefficients as well as the overall regression models – we conclude that the corresponding result is statistically significant.

7.5.2 Examining the hypotheses

With reference to Table 7.4, we now address hypothesis H1. As we observe from the AIC values, the introduction of Uniformity as the independent in model H1.II vis-a-vis only the control variables in model H1.I has led to H1.II being a more effective model than H1.I. The coefficient of Uniformity in H1.II indicates that higher levels of Uniformity relates to higher Workload, and the relation is statistically significant. Thus the null hypothesis corresponding to H1, that there is no relationship between uniformity of interaction and workload – can be rejected in favour of H1.

The results presented in Table 7.5 allow us to examine hypothesis H2. Com- paring models H2.I and H2.II we observe that introduction of Uniformity as an independent variable over and above the control variables has increased the goodness of fit of the regression models by a factor of two ($R^2 = 0.04$ in model H2.I versus $R^2 = 0.08$ in Model H2.II). From the coefficient of Uniformity in H2.II, we see that higher Uniformity relates to lower median elapsed time and this relation is statistically significant. Thus the null hypothesis corresponding to H2, that there is no relationship between uniformity of interaction and task completion time, can be rejected in favour of H2.

So, we find statistically significant evidence that, across the developer pool, higher levels of uniformity relates to higher workload, as well as quicker closure of reviews, after controlling for the peripheral factors considered in this study. The mores of developer interaction as reflected in their motif profiles thus relates to how much work they handle, and how quickly such work is completed. In the context of this study, higher uniformity of interaction for a particular developer denotes that her participation in different motifs is relatively more even. As each motif captures a distinct pattern of triadic communication, an evenly distributed motif profile is an indicator of a developer's versatility in sustaining – with similar levels of involvement – a wide range of interactive relations with her peers. Such versatility can engender deeper levels of familiarity with the project ecosystem, leading to assignment of higher responsibilities – as reflected in higher workload – as well as the knowledge and expertise to complete the assigned activities faster – as reflected in shorter closure time.

In conventional wisdom, the amount of work assigned to a developer and the time it takes to complete the work are believed to be directly proportional; more work entailing more time, and vice versa. From that perspective, our results are counter-intuitive. We find higher uniformity of developer interaction to relate to *higher* workload as well as *lower* closure time. This dichotomy can illuminate an interesting dynamic of large software development ecosystems. As developers interact with their peers in different ways – giving and getting advice, sharing expertise, using and augmenting the "tribal memory" (Booch, 2008) of their peer group – they are enriched by myriad channels of information flow. Accordingly, we see that developers who participate in many such motifs – rather than being confined to just a few – are more likely to effectively manage large volumes of work.

These findings have a number of implications.

7.5.3 Insights and implications

Our results address some key concerns around risks arising from individual developers' nature of engagement in software projects. In large scale open source development ecosystems like the one we are studying, developers at all levels of seniority – veterans and tyros alike – frequently face the *depth versus breadth* dilemma: whether to aim for deep expertise in a particular development activity, or strive to establish one's familiarity with a wide array of tasks. This is a common quandary across disciplines, as brilliantly highlighted in Berlin's *Hedgehog and the Fox* (Berlin, 1993). Whichever path is chosen by a particular developer, we see evidence that engaging with peers across the whole spectrum of motifs, vis-a-vis remaining confined to just a few, makes it more likely for the developer to manage larger workloads *and* complete assigned tasks quickly. Conway's Law, the mirroring hypothesis or the notion of socio-technical congruence all point to an interesting phenomenon: the communication structure of developers being reflected in the structure of the software system being developed (Conway, 1968), (Cataldo et al., 2006), (Kwan et al., 2012). This leads to the possibility that predominance of particular motif(s) in a development ecosystem may be reflected in similar connections between code

Table 7.1: Descriptive statistics of the developer details.

Variable	Mean	SD	Median
Interest	193.282	561.5	13
Engagement	96.86	240.887	9
Experience	365.338	406.979	204.5
Dissemination	44.748	81.522	14
Reception	44.789	79.425	14
Eigen Centrality	0.009	0.042	4.05×10^{-4}
Authority	4.66×10^{-4}	0.001	1.111×10^{-4}
Workload	97.586	241.011	9
TaskCompletionTime	186.07	253.261	31
Uniformity	0.299	0.085	0.303

components. The relations between motif profiles of developers and their workload and task completion times as identified in this study can inform a deeper understanding of the influence of developer interaction on code structure.

A key challenge in the governance of large and distributed teams is to determine whether developers are able to harness communication resources at their disposal towards effective individual and team outcomes. While it is unreasonably expensive – and intrusive – to monitor individual interactions, our results offer insights on how patterns of interactions, as expressed in motifs, can indicate the effectiveness of developers in coping with their work and meeting deadlines. Unusually non-uniform motif profiles of particular developers can also indicate hitherto unidentified risks that can be mitigated by intervention in terms of guidance and skill enhancement. The benefits of developers interacting in various ways with their peers can also inform the tuning of tools and processes at organizational levels, to encourage such interaction.

As we study code review data from a large scale open source process, our results can illuminate some of the characteristics of the code review process. We see evidence that higher levels of involvement across the entire range of interaction motifs relates to developers completing higher volumes of tasks in quicker time. For peer review outcomes, this points to the value of developer participation – with even emphasis – in various information exchange scenarios. Effective peer review of code calls for a deep understanding of the overall development context, as well as close familiarity with the interaction of code modules. Thus developers with a more uniform motif profile can be expected to be better positioned to complete reviews more effectively, as highlighted by our results.

Table 7.2: Pearson correlations coefficients of model variables with the variable – *Workload*.

Variable	Correlation
Interest	0.936
Engagement	0.843
Experience	0.609
Dissemination	0.811
Reception	0.868
Authority	0.649
TaskCompletionTime	−0.003
Uniformity	0.36

Table 7.3: Pearson correlations coefficients of model variables with the variable – *TaskCompletionTime*.

Variable	Correlation
Interest	0.017
Engagement	0.013
Experience	0.089
Dissemination	0.002
Reception	−0.029
Authority	0.039
Workload	−0.003
Uniformity	−0.127

7.6 Ode to humility

As evident from the discussion in Section 7.4, we report results from an observational study. Thus the relations between variables as identified in our statistical models are correlational only, and they do not imply causation.

However, as controlled experiments are almost impossible to perform in a real world software development setting, insights from correlational studies can be useful.

7.6.1 Construct validity

Threats to construct validity arise from potential measurement errors in the variables of interest in a study. As discussed in Section 7.2, using the notion of motifs to closely examine triadic relationships in networks is an established approach. Our definition of motif profiles of developers draws upon this notion. To quantify the uniformity of

Table 7.4: Statistical models to examine hypothesis H1.

	Model H1.I	**Model H1.II**
Intercept	2.082 **** (0.034)	−0.638 **** (0.097)
Interest	4.183×10^{-4} **** (1.071×10^{-4})	2.9×10^{-4} *** (9.567×10^{-5})
Engagement	0.002 **** (1.573×10^{-4})	0.002 **** (1.38×10^{-4})
Experience	0.001 **** (8.727×10^{-5})	8.414×10^{-4} **** (8.255×10^{-5})
Dissemination	0.002 –(0.001)	−0.007 **** 0.001
Reception	0.012 **** (0.001)	0.011 **** 0.001
Authority	−199.361 **** (38.292)	57.31 * 34.337
Uniformity		10.294 **** (0.358)
Model parameters		
N	2130	2130
AIC	1.868×10^{4}	1.803×10^{4}
residual – df	2123	2122
2 * *log – likelihood*	-1.866×10^{4}	-1.801×10^{4}

Note: Significance levels "****", "***", "**", "*", "-", denote corresponding *p*-value ≤ 0.001, ≤ 0.01, ≤ 0.05, ≤ 0.1, and ≥ 0.1 respectively.

interaction in the context of our study, we have used the widely applied Gini coefficient metric. Our model variables are also based on standard network measures or calculated by querying our data-set. Thus, these aspects are unlikely to introduce notable threats to con- struct validity. Our review comment network has been constructed on the basis of available data. Although we have used a similar construct in an earlier study (Datta et al., 2015), we recognize that there may be other ways to generate interaction networks between developers engaged in SDLC activities (Datta et al., 2012), and choice of network construction protocol, and/or considering edges weights in the analysis can lead to different results.

Table 7.5: Statistical models to examine hypothesis H2.

	Model H2.I	**Model H2.II**
Intercept	139.245 **** (16.108)	350.479 **** (26.495)
Interest	−5.082 ** (1.986)	−2.467 −(1.96)
Engagement	−2.44 * (1.311)	−1.311 −(1.287)
Experience	0.159 **** (0.021)	0.222 **** (0.022)
Dissemination	−0.307 −(0.442)	0.221 −0.436
Reception	−0.209 −(0.344)	−0.687 ** 0.339
Authority	4319.028 ** (1831.354)	2680.83 −1798.381
Uniformity		−800.891 **** (80.777)
Model parameters		
N	2130	2130
R^2	0.036	0.078
df	2123	2122
F	13.099	25.786
Sig level	****	****

Note: Significance levels "****", "***", "**", "*", "-", denote corresponding *p*-value ≤ 0.001, ≤ 0.01, ≤ 0.05, ≤ 0.1, and ≥ 0.1 respectively.

7.6.2 Internal validity

Internal validity is concerned with establishing whether a study is free from systematic errors and biases. As we have used a single source of historical data, common concerns around internal validity such as mortality and maturation of the subjects being studied, do not pose any threats in this study. However, the extent to which the data set captured developer interaction relevant to our study, can be a threat to internal validity. As the collection and curation of the data set used in this study has been peer reviewed (Hamasaki et al., 2013), we trust this threat to be minimal. Open source development ecosystems have some typical mores of interaction, and our results may have been influenced by them. Replicating our study on a proprietary system will allow us to confirm whether our results are agnostic of development paradigms.

7.6.3 External validity

Threats to external validity of a study come from lack of generalizability of the results. In this study, we present results from studying a single system. Although the data-set we examine is significantly large in terms of depth and breadth, it only includes information from Chromium development. As observational studies of single systems can be useful, we believe our results offer interesting insights (Wolf et al., 2009), (Bird et al., 2009). However, given the nature of our study design, we do not claim our results to be generalizable as yet.

7.6.4 Reliability

The reliability of a study is concerned with the reproducibility of the results. Given access to the data set, our results are fully reproducible.

7.7 Towards a motif based evaluation of risks

In this chapter, we have examined developer interaction in a software project using the notion of motifs, which identifies all the variations of triadic relationships in a network. We defined the motif profile of developers to capture the extent of their participation in various motifs in a network where two developers are joined by a directed edge if the former has sent a message to the latter, the message being related to some common unit of work both developers are working on. With this background, we presented a case study around how developer motif profiles can relate to their performance. Using development data from a large open-source project, we tested two hypotheses, each pertaining to a particular aspect of developer performance. After controlling for known effects on developer performance, we find statistically significant evidence that higher uniformity of developer interaction – measured in terms of evenly distributed participation across different motifs – relates to higher workload, as well as quicker completion of assigned tasks. These results counter conventional wisdom around more work requiring more time to complete; and offer a number of interesting insights for developers and organizations.

Critical risks in large scale software development arise from how much work is assigned to developers, as well as how quickly the work gets completed. While the quanta of work is related to project scope, completion time is associated with project deadlines. The so-called "software crisis" of the 1970s – which never really ended, by some reckoning (Gibbs, 1994) – was characterized by projects that do not effectively deliver what was defined in their scope, and/or overshoot their schedules. Tales from failed software projects usually have one common narrative – ineffective (or outright detrimental) interactions between stakeholders (Hoare, 1981). Thus, awareness of foundational software development risks and resilience towards them, both need to involve a deep understanding of how developers interact. The motif-based perspective and case study presented in this chapter can offer a useful starting point for further examination of how interaction motifs and development risks relate to one another.

Acknowledgments: The authors thank Mr. Aniruddha Mysore for his help with processing some parts of the data-set used in this study.

References

Adamic, L., Buyukkokten, O., and Adar, E. (2003). A social network caught in the web. *First monday*, 8(6).

Akaike, H. (1974). A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723.

Alon, U. (2007). Network motifs: theory and experimental approaches. *Nature Reviews Genetics*, 8(6):450.

Bacchelli, A. and Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 712–721, Piscataway, NJ, USA. IEEE Press.

Barab´asi, A.-L., Albert, R., and Jeong, H. (2000). Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: statistical mechanics and its applications*, 281(1–4):69–77.

Barab´asi, A.-L. and Bonabeau, E. (2003). Scale-free net- works. *Scientific American*, 288(5):60–69.

Barab´asi, A. L., Jeong, H., Neda, Z., Ravasz, E., Schubert, A., and Vicsek, T. (2001). Evolution of the social network of scientific collaborations. Physica A 311, (3–4) (2002), pp. 590–614.

Barron, D. (1992). The analysis of count data: Overdispersion and autocorrelation. *Sociological methodology*, 22:179–220.

Baysal, O., Kononenko, O., Holmes, R., and Godfrey, M. (2013). The influence of non-technical factors on code review. In *Reverse Engineering (WCRE), 2013 20th Working Conference on*, pages 122–131.

Beller, M., Bacchelli, A., Zaidman, A., and Juergens, E. (2014). Modern code reviews in open-source projects: Which problems do they fix? In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 202–211, New York, NY, USA. ACM.

Benson, A. R., Gleich, D. F., and Leskovec, J. (2016). Higher-order organization of complex networks. *Science*, 353(6295):163–166.

Berlin, I. (1993). *The hedgehog and the fox: an essay on Tolstoy's view of history*. Ivan R. Dee, Publisher, Chicago.

Bird, C., Nagappan, N., Devanbu, P., Gall, H., and Murphy, B. (2009). Putting it All Together: Using Socio-Technical Networks to Predict Failures. In *Proceedings of the 17th International Symposium on Software Reliability Engineering*. IEEE Computer Society.

Boh, W. F., Slaughter, S. A., and Espinosa, J. A. (2007). Learning from experience in software development: A multilevel analysis. *Management Science*, 53(8):1315–1331.

Booch, G. (2008). Tribal memory. *IEEE Software*, 25(2):16–17.

Boyd, D. M. and Ellison, N. B. (2007). Social network sites: Definition, history, and scholarship. *Journal of computer-mediated Communication*, 13(1):210–230.

Brooks, F. P. (1995). *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition*. Addison-Wesley.

Callaway, D. S., Hopcroft, J. E., Kleinberg, J. M., Newman, M. E., and Strogatz, S. H. (2001). Are randomly grown graphs really random? *Physical Review E*, 64(4):041902.

Cataldo, M., Herbsleb, J. D., and Carley, K. M. (2008). Socio- technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, ESEM '08, page 2–11, New York, NY, USA. ACM.

Cataldo, M., Wagstrom, P. A., Herbsleb, J. D., and Carley, K. M. (2006). Identification of coordination requirements: implications for the design of collab- oration and awareness tools. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, CSCW '06, page 353–362, New York, NY, USA. ACM.

Conway, M. (1968). How do committees invent? *Datamation Journal*, pages 28–31.

Curtis, B., Krasner, H., and Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM*, 31: 1268–1287.

Datta, J. P. (1998). Iso 9000: A roadmap for design, installation and implementation of quality management systems. Manuscript.

Datta, S. (2018). How does developer interaction relate to software quality? an examination of product development data. *Empirical Software Engineering*, 23(3):1153–1187.

Datta, S., Bhatt, D., Jain, M., Sarkar, P., and Sarkar, S. (2015). The importance of being isolated: An empirical study on chromium reviews. In *Empirical Software Engineering and Measurement (ESEM), 2015, ACM/IEEE International Symposium on* pages 1–4. IEEE.

Datta, S., Sindhgatta, R., and Sengupta, B. (2012). Talk versus work: characteristics of developer collaboration on the jazz platform. In *Proceedings of the ACM international conference on Object oriented programming systems languages and applications*, OOPSLA '12, pages 655–668, New York, NY, USA. ACM.

DeMarco, T. and Lister, T. (1987). *Peopleware: Productive Projects and Teams*. Dorset House Pub. Co.

DeMarco, T. and Lister, T. (2013). *Peopleware: Productive Projects and Teams*. Addison Wesley, Upper Saddle River, NJ, 3rd edition.

Espinosa, J. A., Slaughter, S. A., Kraut, R. E., and Herbsleb, J. D. (2007). Familiarity, complexity, and team performance in geographically distributed soft- ware development. *Organization Science*, 18(4):613–630.

Faraj, S. and Sproull, L. (2000). Coordinating expertise in soft- ware development teams. *Management Science*, 46(12):1554–1568.

Gibbs, W. W. (1994). Software's chronic crisis. Scientific American September 1994.

Gini, C. (1921). Measurement of Inequality of Incomes. *The Economic Journal*, 31(121):124–126.

Girvan, M. and Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826.

Grewal, R., Lilien, G. L., and Mallapragada, G. (2006). Location, location, location: How network embeddedness affects project success in open source systems. *Manage. Sci.*, 52(7): 1043–1056.

Guimera, R., Uzzi, B., Spiro, J., and Amaral, L. A. N. (2005). Team assembly mechanisms determine collaboration network structure and team performance.*Science (New York, N.Y.)*, 308(5722): 697–702. PMID: 15860629.

Hahn, J., Moon, J. Y., and Zhang, C. (2008). Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties. *Information Systems Research*, 19(3):369–391.

Hamasaki, K., Kula, R. G., Yoshida, N., Cruz, A. E. C., Fujiwara, K., and Iida, H. (2013). Who does what during a code review? datasets of OSS peer review repositories. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 49–52, Piscataway, NJ, USA. IEEE Press.

Hoare, C. A. R. (1981). The emperor's old clothes. *Communications of the ACM*, 24(2):75–83.

Hong, Q., Kim, S., Cheung, S. C., and Bird, C. (2011). Understanding a developer social network and its evolution.

Jacobson, I., Booch, G., and Rumbaugh, J. (1999). *The Unified Software Development Process*. Addison-Wesley.

Kwan, I., Cataldo, M., and Damian, D. (2012). Conway's law revisited: The evidence for a task-based perspective. *IEEE Software*, 29(1):90–93.

Long, J. S. (1997). *Regression Models for Categorical and Limited Dependent Variables – Vol. 7*. SAGE Publications Inc, Thousand Oaks, 1 edition.

Mangan, S. and Alon, U. (2003). Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences*, 100(21):11980–11985.

McIntosh, S., Kamei, Y., Adams, B., and Hassan, A. E. (2014). The impact of code review coverage and code review participation on software quality: A case study of the Qt, Vtk, and Itk projects. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 192–201, New York, NY, USA. ACM.

McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444.

Meneely, A., Rotella, P., and Williams, L. (2011). Does adding man- power also affect quality?: an empirical, longitudinal analysis. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering,* ESEC/FSE '11, page 81–90, New York, NY, USA. ACM.

Meyer, B. (2019). In search of the shortest possible schedule. *Communications of the ACM*, 63(1): 8–9.

9Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. (2002). Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827.

Newman, M. E. J. (2003). The structure and function of complex networks. SIAM Review 45, 167–256

Prˇzulj, N., Corneil, D. G., and Jurisica, I. (2004). Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515.

Qiu, J., Lin, Z., Tang, C., and Qiao, S. (2009). Discovering organizational structure in dynamic social network. In *Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on*, pages 932–937.

Rigby, P. C. and Bird, C. (2013). Convergent contemporary software peer review practices. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pages 202–212, New York, NY, USA. ACM.

Rigby, P. C., German, D. M., Cowen, L., and Storey, M.-A. (2014). Peer review on open-source software projects: Parameters, statistical models, and theory. *ACM Trans. Softw. Eng. Methodol.*, 23 (4):35:1–35:33.

Rigby, P. C., German, D. M., and Storey, M.-A. (2008). Open source software peer review practices: A case study of the apache server. In *Proceedings of the 30th International Conference on Software Engineering*, ICSE ’08, pages 541–550, New York, NY, USA. ACM.

Rigby, P. C. and Storey, M.-A. (2011). Understanding broadcast based peer review on open source software projects. In *Proceedings of the 33rd Inter- national Conference on Software Engineering*, ICSE ’11, pages 541–550, New York, NY, USA. ACM.

Romijn, L., Nuall´ain, B. O., and Torenvliet, L. (2015). Discovering motifs in real-world social networks. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 463–474. Springer.

Rosvall, M., Esquivel, A. V., Lancichinetti, A., West, J. D., and Lambiotte, R. (2014). Memory in network flows and its effects on spreading dynamics and community detection. *Nature communications*, 5:4630.

Salnikov, V., Schaub, M. T., and Lambiotte, R. (2016). Using higher- order markov models to reveal flow-based communities in networks. *Scientific reports*, 6:23194.

Shen-Orr, S. S., Milo, R., Mangan, S., and Alon, U. (2002). Net- work motifs in the transcriptional regulation network of escherichia coli. *Nature genetics*, 31(1):64.

Tabachnick, B. and Fidell, L. (2007). *Using Multivariate Statistics*. Boston: Pearson Education.

Wagstrom, P., Herbsleb, J. D., and Carley, K. M. (2010). Communication, team performance, and the individual: Bridging technical dependencies. *Academy of Management Proceedings*, 2010(1):1–7.

Wasserman, S. and Faust, K. (1994). *Social network analysis: Methods and applications*, volume 8. Cambridge university press.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of ‘small-world’ networks. *nature*, 393(6684):440.

Weinberg, G. (2011). *The Psychology of Computer Programming: Silver Anniversary eBook Edition*. Weinberg & Weinberg.

Weinberg, G. M. (1971). *The Psychology of Computer Programming*. Van Nostrand Reinhold.

Wernicke, S. (2006). Efficient detection of network motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 3(4):347–359.

Wolf, T., Schroter, A., Damian, D., and Nguyen, T. (2009). Predicting build failures using social network analysis on developer communication. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE ’09, page 1–11, Washington, DC, USA. IEEE Computer Society.

Yavero˘glu, O¨. N., Malod-Dognin, N., Davis, D., Levnajic, Z., Janjic, V., Karapandza, R., Stojmirovic, A., and Pr˘zulj, N. (2014). Revealing the hidden language of complex networks. *Scientific reports*, 4:4547.

Zhang, W., Nie, L., Jiang, H., Chen, Z., and Liu, J. (2014). Developer social networks in software engineering: construction, analysis, and applications. *Science China Information Sciences*, 57(12):1–23.

Christof Ebert

8 Risk-based security engineering

8.1 Introduction

Night drive on the highway. The display suddenly flashes, and the loudspeakers transmit a loud and painful sound. The driver is highly disturbed and tries to stop this annoyance. In doing so he is losing control over the car and causes an accident.

Mere fiction? Not really. Cybersecurity today is the biggest risk in most critical systems, across industries (Ebert 2020a, Ebert 2017a, Ebert 2017b). Continuously growing complexity, interconnections by a variety of bus systems, wireless connectivity, and the use of standard components with open interfaces make networked systems increasingly vulnerable. Such risks demand strong protection on various levels along the entire life-cycle of components and of the system. Looking to past experiences with insufficient security across industry domains, it is obvious that cybersecurity will determine which suppliers and electronic platform will capture the market for standard components. And it will determine how fast further communication systems (e.g., telematics with internet access) will be accepted by customers and policy makers.

Risks always create opportunities – and vice versa. Let us look on this relationship from an IT perspective. The growing connectivity of any IT and electronic system has created huge potential for new functions and fast updates. Our society needs connectivity of devices and systems to connect knowledge and people. This opportunity creates risks with respect to cybersecurity. This risk in turn creates opportunities towards more resilient architectures.

Innovative products need to be pushed to markets covering the major trends of connectivity on one side representing the more embedded industries such as mobile services, automotive and transport and digital transformation on the other side, representing IT services. Both are converging towards one new software and service industry covering both IT and embedded Internet of Things (IoT) systems, which are connected by cloud solutions (Ebert 2020a).

Vector Consulting, a globally active consulting firm with focus on product development, has performed an industry survey to capture major engineering trends (Ebert 2020a). Figure 8.1 provides the survey results of this industry survey. The horizontal axis provides perceived short-term challenges, and the vertical axis shows more mid-term challenges. Since each reply allowed up to five challenges in both dimensions, the sum is more than 100 percent. The validity is given with a response rate of four percent covering different industries. It thus represents different B2B business models, as well as regions in the world.

The real challenge obviously is quality, in the survey emphasized with two major quality drivers, namely safety and security. We have chosen these two because they

https://doi.org/10.1515/9783110652321-008

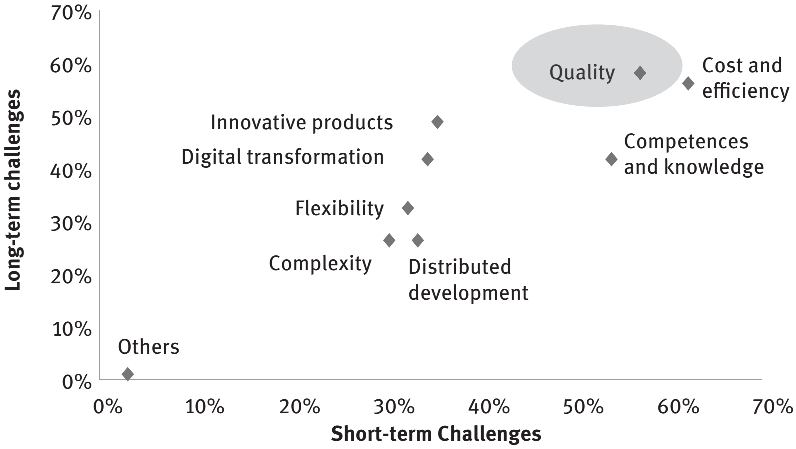


Figure 8.1: Industry challenges in product development. Safety and security (here labeled as quality) rank high for both short- and midterm challenges.

are pivotal in this converging software industry. Cybersecurity is mandatory to ensure trusted connectivity. It also safeguards the reliability of related mobile and distributed services. Functional safety represents the growing awareness of product liability, where specifically embedded devices must ensure absence of hazards to users and environment. Recent growth of lawsuits in medical, transport and industry show that functional safety is fast growing in its relevance. Understanding that there is no safety in distributed IT systems without cybersecurity makes this pair of qualities indispensable. Looking at its positioning in our survey shows that it is key soon – but currently there is not enough time.

We currently face an increasing amount of companies setting up task forces and spinning the hamster wheels at highest speed. This might work for a short time, but not with sustainable productivity and quality. The cure is reducing technical debt and refactoring legacy. For instance, many clients have abandoned their process improvement programs during past years, if a CMMI or ASPICE maturity will prevail. Even worse, some have established bottom-up agility without a clear focus on value and sustainability. Wrong! The results are increasing difficulties to integrate software, trace safety and security, and keep consistency across artifacts and variants. The cost of task forces to repair this missing process focus is by far higher than the usual 2–3 percent of R&D for maintaining engineering maturity. This immediate effect is overarched by reduced motivation of engineers with dramatic productivity impacts.

In the software-driven industries we realize that despite all these warning signs, necessary changes and investments, such as for digital transformation, had been avoided too long. Investments towards efficient processes had been stagnating. In fact, we have worked with big international companies who had reduced their focus on process maturity over the past few years. Process maturity as observed in project management or configuration management has been declining with an increasing amount of rework and quality problems. A heated software and IT business climate made many to just deliver innovations, rather than stabilizing their assets. Business "went well" and teams had more than enough to do with daily operations. To mitigate business risks, product portfolios should have been adjusted counter-cyclic. This has opened a gap of complexity and competence.

To better illustrate practical cybersecurity, we will look in this article towards automotive cybersecurity. This domain is currently most challenging as it advances the convergence of IT and embedded systems. Also, the results can be easily transferred to other industries such as transport, medical etc. Developing automotive software is challenging for several reasons, because it connects embedded software with big IT systems, it is developed in a global context in distributed teams, and it has one of the shortest cycle times of all industries. A modern car has 50–120 embedded microcontrollers on board and is connected over various external interfaces to a variety of cloud and infotainment technologies. Onboard software is in the hundred MLOC range, and still exponentially growing. Automotive software product lines and variants are some of the largest and most complex in all industries. It is said that the automobile is rapidly becoming a "computer on wheels." For more detail on specific challenges and solutions in automotive software we recommend reading below mentioned references.

Security and related measures demand well-founded concepts all along the lifecycle of both components and the car itself, especially if their effectiveness must be proven at a later point due to legal actions. We will in this book chapter elaborate risk-based security engineering. Hands-on case studies will show how to transfer results into different industry domains. With this chapter we strive not only to provide guidance for specific misuse cases but to change the mentality of engineers towards designing for security – rather than only for functionality. Some parts have been published in a different context (Ebert 2017a) and are further elaborated here.

8.2 Cybersecurity, safety and product liability

Cybercriminals can break into any connected system. If hackers get access to a critical infrastructure of industrial systems, the potential damage is no longer restricted to a stolen vehicle or abused social security numbers. Hackers have already proved that they can sabotage water and power utilities, government systems such as election systems, hospitals, and banks. Famously, in 2016, a group named the Shadow Brokers even succeeded in breaking into the highly protected systems at the National Security Agency to steal much of its highly secret arsenal of hacking and protection tools (Greenberg 2019). Of course, they would be able hacking any other system.

Traditionally, embedded systems were perceived too difficult to hack into and not worth the amount of time and energy required. But as systems have added Ethernet, WLAN, USB, Bluetooth, GPS, and other connectivity features, the amount of attack surfaces has increased. The most popular method involves attacking a diagnosis port, or open interface, which can give a hacker access to functions or at least to corrupt data and prohibit performance with Denial of Service (DoS) attacks.

What is cybersecurity? Cybersecurity, or briefly security as we will call it throughout this article, is a quality attribute which heavily interacts with other such attributes, such as availability, safety or robustness. Information security is the sum of all attributes of an information system or product which contribute towards ensuring that processing, storing and communicating of information sufficiently protects integrity, availability and trust. Information security implies that the product will not do anything with the processed or managed information which is not explicitly intended by its specification. If for instance the classic definition of a functional requirement meant that the car can be started by turning the key but would also allow a variety of mechanisms to start it otherwise, maybe for diagnostic or repair services. After all, who was not in such situation that he needed support on the road and the person would open the trunk and start the engine directly?

Recently safety and security have turned out to be the major engineering challenges as we can see in above figure. There is a big difference when we contrast safety and security. Safety is built upon reliability theory and investigates statistical malfunctions of components with small probabilities and how they will impact functionality. Security on the other hand must deal with the worst cases with a probability of one because once known, they will be exploited. One might argue that safety is about criticality for the life and health of the system’s user, while security is only about annoyances. It is however obvious that within a safety-critical system, such as a car, security meets safety because malfunctions can interact and cause disturbances that can result in accidents, as described in our introduction.

The pressure to deliver products as fast as possible combined with increasingly open architectures and overwhelming complexity has further weakened the quality and security of IoT systems, across industries. Today, medical devices such as insulin pumps and pacemakers are equally at risk as are cars, industry production facilities and the wide and distributed system of utilities. The more our society depends on connectivity the more we are at risk of being substantially hit by major attacks – with the potential to not only damage single systems, but rather entire cities and countries. Imagine a major break-down of electric utilities. They would immediately stop water suppliers and thus stop live in the impacted area.

Risk-oriented security with dedicated test methods and appropriate tools is the call of the day (Ebert 2020b). Security test must start with static code analysis, proceed with unit test and further advance with dedicated methods such as fuzzing and robustness tests up to the level of penetration testing (PenTest). Let us briefly introduce to risk-oriented security engineering and then dive into appropriate test methods and tools. We focus on novel grey-box penetration techniques which build a bridge from threat analysis towards more efficient and effective testing.

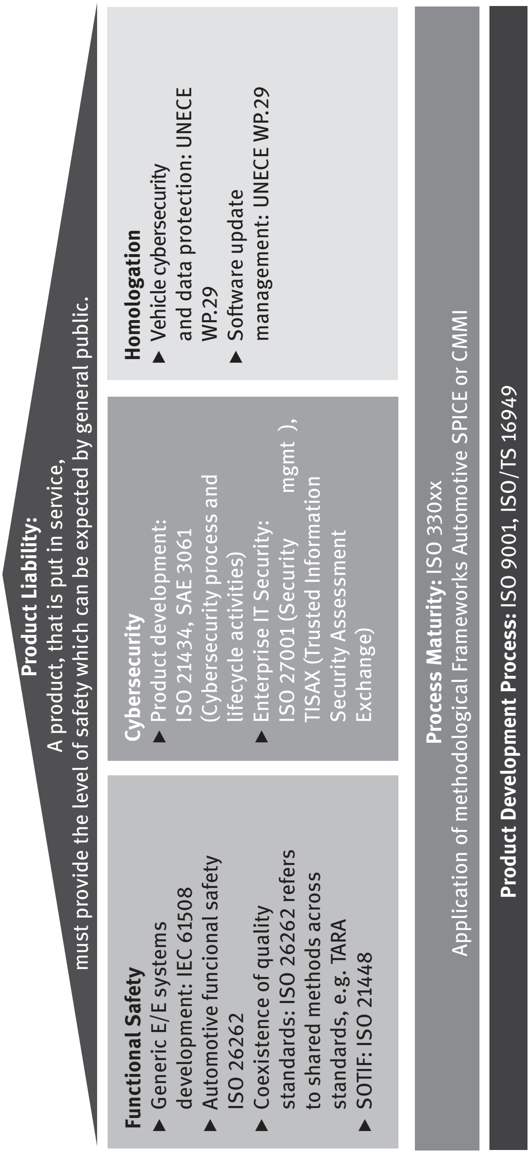
Product liability demands that companies ensure that risks are adequately mitigated when the bring a product to the market. This translates into applying the state of the art, as it is defined by standards (Figure 8.2)

Vulnerability scenarios within cars have been changing fast over the past years. The increasing interconnection on different architectural layers (e.g., electrical control units, software components, configurations and their changes, communication inside and outside the car, diagnosis, telematics) has caused a level of complexity that was unknown so far. It is a mere question of time until the resulting loopholes and weaknesses are identified and abused. It was showed already that widely used bus systems such as CAN and Ethernet can be brought from the outside – by connecting a device to any point of such bus systems – into conditions which will eventually cause malfunctions.

State of the art communication systems increasingly offer open interfaces (e.g., DVDs, E-Mails, USB, Bluetooth, IP-based diagnostics) that allow to inject viruses and Trojan horses to the respective embedded operating systems (Ebert 2017a, Ebert 2017b, Ebert 1997, Islam 2014). Also, defective code and configuration settings can create new and unknown vulnerabilities as we are used to from many information systems (Ebert 2017a, Islam 2014, ISO 2020a, ISO 2020b).

Most security attacks *happen like* in our illustrative case study from the beginning of this article. Figure 8.3 illustrates the cause-effect chain in our small introductory example. As so often in security attacks, the first step was just a normal upgrade of the multimedia equipment with a better device. Needless to say, that it was not delivered and installed by the OEM but came through an internet delivery for perceived cost reasons and enhanced functionality. In a second step infected software came into the multimedia devices, probably via an infected USB stick or from a media file. It could well be that software upgrades to one of the media devices also brought this infected software into the system. From here onwards it was just normal cause and effect, namely transmission of valid but dangerous signals on the infotainment bus which were triggered by listening to signals with speed and outside illumination. These signals are almost omnipresent in car networks due to many dependencies on these factors.

Automotive electronics and IT are changing at a fast pace. Multimodal mobility will connect previously separated domains like cars and public transportation. Mobility oriented services such as car sharing creates fully new eco-systems and business models far away from the classic buy your own car approach. Autonomous driving demands highly interactive services with multi-sensor fusion – far away from currently deployed functionally isolated control units. Connectivity and infotainment have transformed the car into a distributed IT system with cloud access, over-the-air functional upgrades, and high-band-width access to map services, media content, other vehicles and surrounding infrastructure. Energy efficiency evolves the classic



Product Liability:
A product, that is put in service,
must provide the level of safety which can be expected by general public.

Functional Safety
- ▲ Generic E/E systems development: IEC 61508
- ▲ Automotive funcional safety ISO 26262
- ▲ Coexistence of quality standards: ISO 26262 refers to shared methods across standards, e.g. TARA
- ▲ SOTIF: ISO 21448

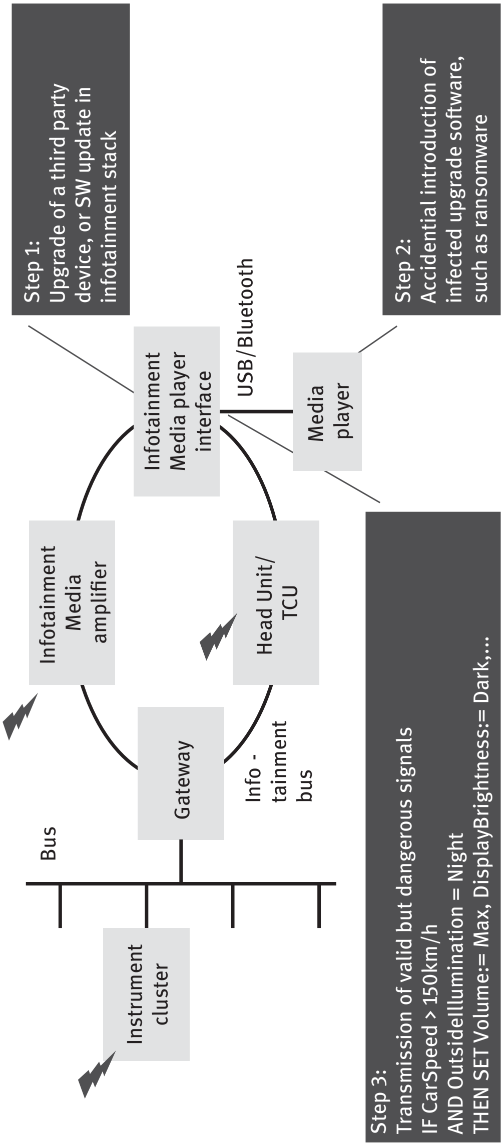
Cybersecurity
- ▲ Product development: ISO 21434, SAE 3061 (Cybersecurity process and lifecycle activities)
- ▲ Enterprise IT Security: ISO 27001 (Security mgmt), TISAX (Trusted Information Security Assessment Exchange)

Homologation
- ▲ Vehicle cybersecurity and data protection: UNECE WP.29
- ▲ Software update management: UNECE WP.29

Process Maturity: ISO 330xx
Application of methodological Frameworks Automotive SPICE or CMMI

Product Development Process: ISO 9001, ISO/TS 16949

Figure 8.2: Legal Situation: Product Liability Demands Using Standards.



Step 1:
Upgrade of a third party device, or SW update in infotainment stack

Step 2:
Accidential introduction of infected upgrade software, such as ransomware

Step 3:
Transmission of valid but dangerous signals
IF CarSpeed > 150km/h
AND OutsideIllumination = Night
THEN SET Volume:= Max, DisplayBrightness:= Dark,...

Figure 8.3: „Night Drive“ – How could it happen?

powertrain towards high voltage hybrid and electric engines. All these innovations make automotive systems the prime target for security attacks in the twenties.

The different reasons for insufficient cybersecurity can be rooted back to product (e.g., functionality, architecture, configurations), process (e.g., design, development, validation, stakeholder communication), and service (e.g., maintenance, enhancements, diagnosis).

All these scenarios result from unawareness of security needs and security technology, be it by ignorance ("this won't matter in cars because all our critical electrical units are protected by cryptography"), arrogance ("security matters only in information systems") or naivety ("we have verified all requirements and components according to our established test strategy"). An overall security strategy is mostly missing.

Typically, components are individually protected such as encrypted flashware for an engine controller. Critical functionality such as engine management, theft protection or engine diagnosis is hardened and verified (Ebert 1997). Increasingly secure networks and architectures are discussed and will certainly influence the design of cars ten years from now (Ebert 2017b). Safety has received a lot of focus recently in automotive engineering and qualification of components and systems, such as processes to ensure proper handling and engineering according to SIL-levels (Iso 2020a). But safety and associated design rules are insufficient as we have learned before. They look to faults and their probabilities, while security must deal with the worst case in scenarios where a probability is replaced by the willingness of the attacker to cause the worst possible damage.

Two aspects related to security in embedded systems must be considered: (1) Attack scenarios go well beyond individual components and functions. (2) While safety deals with avoiding critical failure modes, security has to cope with intelligently introduced causes of faults, which is far more difficult, given that the attackers' intelligence, willingness, determinedness, and creativity often exceed that of the engineers looking to a problem from the – different – perspective of how to solve it, and not how to find loopholes and strange feature correlations.

Telecommunication and information systems have realized several years ago that isolated mechanisms (e.g., distributed functionality in proprietary subsystems, protection on component-level, gateways and firewalls between components, validation of critical functions) are insufficient. This article underlines together with concrete examples how automotive security can be achieved. We will take the three different perspectives that were introduced in Figure 8.2, namely (1) the product and its architecture (e.g., specification of security requirements, misuse and abuse cases, vulnerability analysis, inherently secure architectures); (2) engineering for security during the development process (e.g., FMEA and hazard analysis as a basis for security, protection on component- and on system-level, systematic verification, code analysis, validation on product-level); and (3) the relevant after-sales activities in the field (e.g., fault analysis, patch and correction handling, emergency response and handling, distribution of corrections and protective mechanisms).

With the introduction of connectivity and increasing amount of IT in vehicles, precautions must be taken to increase the reliability and to reduce the vulnerability to the system. The basic principle is simple to understand, yet difficult to achieve: Functional safety needs security. Based on the specific challenges of automotive security, OEMs and suppliers must realize an effective protection against manipulations of automotive electric and electronic (E/E) systems.

Automotive security has gained huge relevance in short time-frame. Attacks are reported today almost continuously and therefore systems must be protected and hardened. Safety needs security as a mandatory condition, which means that any safety-critical system as a minimum must also be protected for cybersecurity. Security must be integrated early in the design phase of a vehicle to understand the threats and risks to car functions.

Key points in the development of protected E/E systems are the proper identification of security requirements, the systematic realization of security functions, and a security validation to demonstrate that security requirements have been met. The following items need to be considered to achieve security in the car development process:

- Standardized process models for a systematic approach which is anchored in the complete development process. This starts on the requirements analysis through the design and development down to the test of components and the network.
- Quick software updates to close vulnerabilities in electronic control unit (ECU) software.
- Reliable protocols that are state-of-the-art and meet long-term security demands. Related to security this is often combined with cryptographic keys. So, a key management over the lifecycle of the vehicle must be maintained.
- In-vehicle networks and system architecture that provides flexibility and scalability and are designed under consideration of security aspects.

Risk-oriented security helps to balance growing security threats with increasing complexity over the entire life-cycle. Unlike many previous attempts our research and many practice projects indicate that while design for security is good, it is not good enough. Effective security must handle the entire life-cycle. For covering the major risks related to safety hazards caused by security misuse and abuse scenarios, we combine for automotive risk-oriented security engineering activities of both safety and security engineering (Figure 8.4). While safety and security are very different methodologies, we realize that at present, both organizational infrastructures and governance are much better in many companies for functional safety than what they are for cybersecurity. So, we see this path as an evolution towards full (independent) cybersecurity organization in the life-cycle.

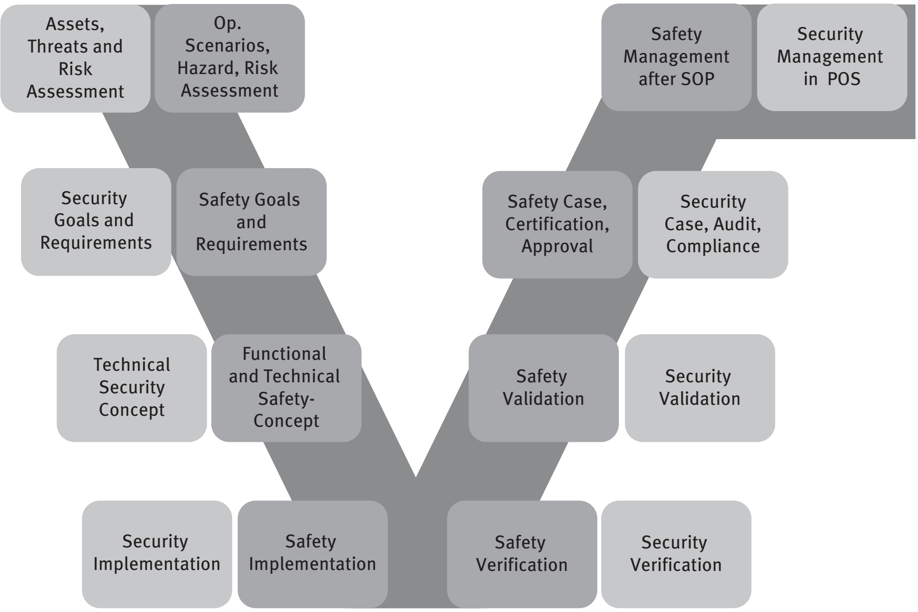

Assets, Threats and Risk Assessment
Op. Scenarios, Hazard, Risk Assessment
Safety Management after SOP
Security Management in POS
Security Goals and Requirements
Safety Goals and Requirements
Safety Case, Certification, Approval
Security Case, Audit, Compliance
Technical Security Concept
Functional and Technical Safety-Concept
Safety Validation
Security Validation
Security Implementation
Safety Implementation
Safety Verification
Security Verification

Figure 8.4: Risk Mitigation with Combined Safety and Security Engineering.

8.3 Risk-oriented security

Developing secure software is challenging for several reasons, namely because increasingly systems are connected, most software is developed in a global context in heterogeneous teams with various skills, systems complexity is exploding with embedded and IT systems converging such as IoT, and both budget and cycle times are continuously decreasing. For instance, a modern car has almost hundred embedded microcontrollers on board and is connected over several external interfaces to a variety of cloud technologies. At the same time cyberattacks and vulnerabilities are increasing. Therefore, software technology and the underlying security engineering must be constantly improved.

While there is a movement towards better understanding security from the ground up, many of the existing approaches in managing security have been focused around encryption, developing malware software, and to detect attacks to networks and systems. Existing methods and tools are limited by large number of false positives and inability to consistently trace such issues to the root causes. In this article we will draw attention to all aspects of security from specification to design and life-cycle support.

Over the past decade trends like connected car and driver assistance systems among others have led to software and connectivity playing an increasingly important part in developing vehicles and for business models of OEMs and suppliers likewise.

Devastating impact of security issues is already known from industrial sectors like IT-infrastructure, aviation, information technology and telecommunications, industrial control systems and energy and financial payments. Virtually every connected system will be attacked sooner or later. A 100% secure solution is not feasible. Therefore, advanced risk assessment and mitigation is necessary to protect assets. Consequently, the typical solution to security in these industries relies on suitable risk assessment that projects threats on assets of interests. Thereby cost of implementing specific security measures can be compared with the probability of a threat that they counter.

Asset-based risk assessment is a suitable tool for companies to steer efforts for security engineering in a systematic and comprehensive way and thereby involve all relevant stakeholders in the organization. For example, a CEO may not find it very helpful to have a long exhaustive list with every attack vector or potential threat – they need to be provided with a ranked listing and useful decision-support tools which clearly shows alternatives and consequences. From the view of an automotive system developer, a flat listing of potential threats might not help to improve the system. To really help, they need to be able to map security threats, countermeasures and requirements to system/architecture elements in their scope of the project.

The systematic management of security threats and associated security goals is essential to providing safe and competitive products, and to protect valuable assets and business models.

But what makes security engineering so complex? Automotive developers face the challenge of securing a system against attackers whose capabilities and intentions are at best partially known. Some attacks might today appear infeasible, but today’s impossible attacks might become more likely soon. An example of this is attacking a vehicle simply by exploiting wireless interfaces, 20 years ago would have been extremely unlikely, however today a cheap software defined radio and accomplish these types of attacks with little effort. On the other hand, an attacker might invest more effort into launching an attack the more valuable a successful attack is to him. Some attacks represent more effort to the attacker than others given the specific potential of the attacker. It is this risk/reward payoff that is analyzed in security engineering. Likewise, during testing and verification, suitable methods to verify that the vehicle has the required security level and process goals like, test strategy and coverage, need to be chosen.

Furthermore, the assets to be protected from attacks are decided by stakeholders involved, e.g. drivers would indicate different assets of their vehicle to be protected compared with what an automotive developer considers an asset. However, customers/drivers need to be satisfied with their vehicle in order to buy another one from the same company. Consequently, security engineering must seek tradeoffs between cost of security measures and benefit to assets in order to make sustainable decisions.

Security concepts must balance the cost of not having enough security and thus being successful attacked with all damaging consequences and the cost spent to implement appropriate security mechanisms and keep them updated along the life-cycle of the car – well beyond end of production. We therefore introduce here a strict risk-oriented approach to security (Figure 8.5).

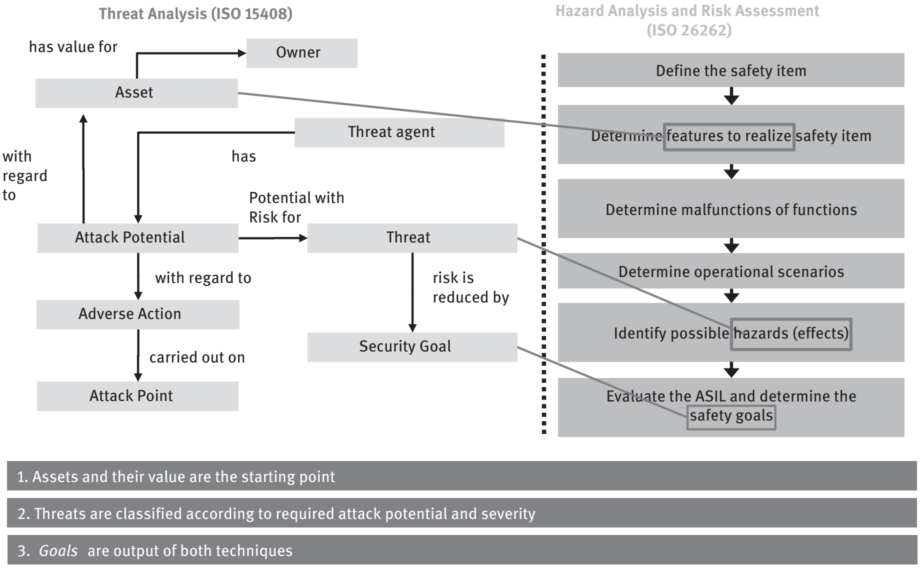

Threat Analysis (ISO 15408)
has value for
Owner
Asset
Threat agent
with regard to
has
Potential with Risk for
Attack Potential
Threat
with regard to
risk is reduced by
Adverse Action
carried out on
Security Goal
Attack Point
Hazard Analysis and Risk Assessment (ISO 26262)
Define the safety item
Determine features to realize safety item
Determine malfunctions of functions
Determine operational scenarios
Identify possible hazards (effects)
Evaluate the ASIL and determine the safety goals
1. Assets and their value are the starting point
2. Threats are classified according to required attack potential and severity
3. *Goals* are output of both techniques

Figure 8.5: Overview of risk-oriented cybersecurity analysis process with the major steps of asset determination, attack potentials and security goals derived from threats.

The relationship between assets, attackers and threats is complex and dynamic (e.g. attacks are more probable the less effort is required and the more value successful attacks represent; attack vectors and effort change over time). Furthermore, common understanding of assets among all stakeholders of security engineering is mandatory in order to provide information for steering the security engineering. Choosing the right set of security engineering methods for analysis, concept and testing is challenging but required in order to enable goal-oriented and manageable security engineering.

Risk-based Security Engineering combines state-of-the-art methods for automotive security risk assessment in a practical framework and supports all involved stakeholders to develop “secure-enough” products. The method and our approach for proposing a concrete technical security concept is based upon security best practices such as:

– ISO 15408 (Evaluation criteria for IT security) with its focus on IT systems, specifically the 7 evaluation assurance levels (EAL) for security requirements and

guidance on common criteria a standardized practice translated by Vector to automotive common criteria.
- ISO 27001 (Information security management systems) with its governance requirements for security engineering across the entire value chain.
- IEC 62443 (Industrial communication network security) with its strong view on distributed systems and necessary security technologies and governance.
- IEC 61508 (Functional safety for electronic systems) while being aware that it is only a high-level functional safety guidance for electronic systems.
- ISO 21434 (Automotive cybersecurity) being the first standard on the topic of automotive cybersecurity (ISO 2020b).
- ISO 26262 (Automotive functional safety) using its clear focus on automotive electronic systems with good coverage of entire life-cycle (ISO 2020a).

Our Vector SecurityCheck and underlying security engineering methods have adopted the state of the practice in security evaluation and proposed mitigation. It is using significant research work from our worldwide security projects. It also uses external best practices, such as HEAVENS (Islam 2014), and other proposed methods for security risk assessment in automotive development (ISO 2020b).

We will further on show by examples how to use the risk-oriented security concept covering the entire security life-cycle with focus on the upper left activities, namely
- Asset Definition and Threat and Risk analysis
- Security Goals
- Security Concept

Two aspects related to security in embedded systems must be considered: (1) Attack scenarios go well beyond individual components and functions. (2) While safety deals with avoiding critical failure modes, security has to cope with intelligently introduced causes of faults, which is far more difficult, given that the attackers' intelligence, willingness, determinedness, and creativity often exceed that of the engineers looking to a problem from the – different – perspective of how to solve it, and not how to find loopholes and strange feature correlations.

Risk-oriented cybersecurity connects security requirements directly with design decisions and test – following the triple peak model (Ebert 2017a). This ensures full traceability from the initial Threat Analysis and Risk Assessment (TARA) and definition of security requirements (Figure 8.6). From a compliance and governance perspective we see this approach helpful as it instruments the necessity to prove that security requirements and decisions have been adequately verified – with each single regression. For instance, we have introduced this risk-oriented cybersecurity to a leading tier-1 supplier in a high safety-critical, e.g. ASIL-D (Automotive Safety Integrity Level D) environment and ensured that road tests cycle time could be dramatically reduced.

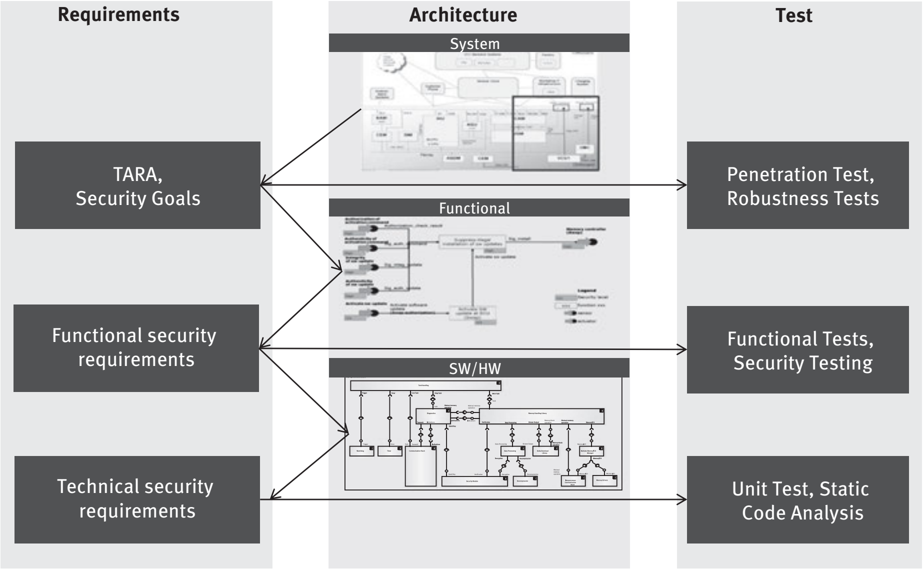
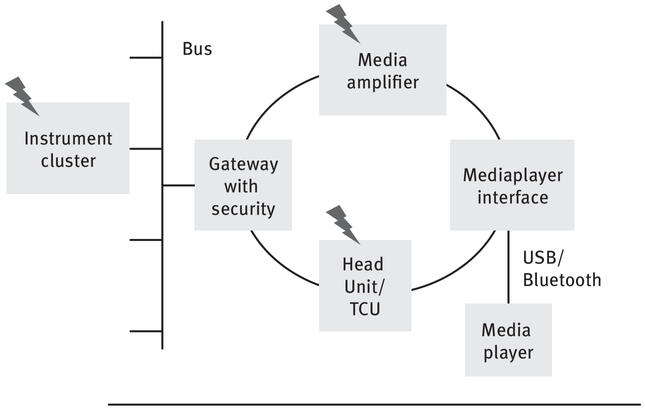

Requirements
Architecture
Test
System
Functional
SW/HW
TARA,
Security Goals
Functional security requirements
Technical security requirements
Penetration Test, Robustness Tests
Functional Tests, Security Testing
Unit Test, Static Code Analysis

Figure 8.6: Risk-oriented cybersecurity connects security requirements directly with design and test.

Let us go back to our initial case study with the night dive. Figure 8.7 shows the lessons learned when going back to our initial case study. It is a combination of different techniques starting with architectural decisions (i.e., encrypting bus messages, determining valid configurations of control units, software signatures, and bus topology), then systematically implementing these techniques during the development process (i.e., defining misuse cases and attack scenarios that are further elaborated to specific functional requirements which are reviewed during design), and certainly also impacting after-sales where system configurations can be hardened even during service and maintenance upgrades and checked for potential violations. After all, the automotive OEM still is the intellectual owner of the entire communication architecture and thus has more power and influence on design and security than in open systems such as IT or telecommunication.

Architectures, systems and protocols must be developed with security in mind (i.e., design for security). Competences must be developed around security engineering, and employees must be trained how to design, verify and sustain security throughout the product's life-cycle. Most important it is that all decided methods and processes are implemented consistently, systematically and rigorously with measurable effects. It is of not much use to invest in security but do it opportunistically and have no tangible measurements that show how security has been effectively improved. It is only by continuous measurements on effectiveness that the value of security measures improves.



	Techniques to use	Effect
Product/ architecture	Encryption of messages, authentication of upgrades	Critical information not accessible, unsafe devices cannot be plugged in
Develop-ment process	Attack scenarios, Misuse cases	Exchange of a device on the bus considered a misuse and handled by dedicated functional requirements.
After-sales/ field	Assurance of a consistent system configuration at any time after sales	Media player interface not permitted in the actual car configuration and thus no access to the infotainment bus.

Figure 8.7: „Night Drive“ – How it can be avoided.

8.4 Case study: Automotive security engineering

With vehicle dynamics being a major safety risk, automotive cybersecurity must also consider powertrain. Different attacks can be imagined, such as getting access to critical intellectual property up to manipulations to enhance vehicle functions, and not the least to impact the driver and thus directly the passenger safety. Considering security aspects in the basic structure from the very beginning can provide essential advantages for the flexibility and scalability of such a network. It can also reduce the risk for attacks. The major attack scenario sin automotive vehicles are described in Figure 8.8.

Security components like firewall and router are also important parts and are useful to separate networks. For example, account computers will not be connected to the same network as that from marketing or development. Instead, computers are grouped to separate networks depending on their use case and traffic. A router interconnects the networks and provides data exchange between them. It passes only the relevant and allowed data from one network to the other. The access to computers is already restricted by the structure of the network. The router with an integrated firewall also manages the access to the internet. This device observes the incoming and outgoing traffic and can be configured that only allowed traffic will pass. Additionally, maintenance can be done easily on the central part by applying patches or re-configure the device if needed.

Let’s consider a car network under the aspects shown above. At the very beginning, a safety and security analysis has been performed and the networks are partitioned so that the connected items are grouped under functional, safety and security aspects. The different networks are interconnected by a gateway. The ECU with a remote connection is considered particularly as unsafe. Even if great care was taken during the software implementation, a failure cannot be excluded and the potential risk is too high that someone could capture and take control of the ECU from outside. To minimize the risk, this ECU should not have access to any other internal networks. We locate this function into a separate ECU (inter comm. module) and connect it through the gateway. The gateway can now contain a firewall that has separate filter rules for each subnet. Only those messages are passed to other networks that are allowed. The traffic inside a network is not restricted and affected by the firewall.

We now take advantage of our threat analysis and risk assessment that has provided a detailed security analysis of our system. In an initial data flow analysis, we saw the interaction of signals between partitioned functions separated in ECUs. This has already helped us to separate the ECUs to different networks, respectively partitioned the networks according to safety, security and functional aspects. We can now classify networks into security zones according to the safety and security

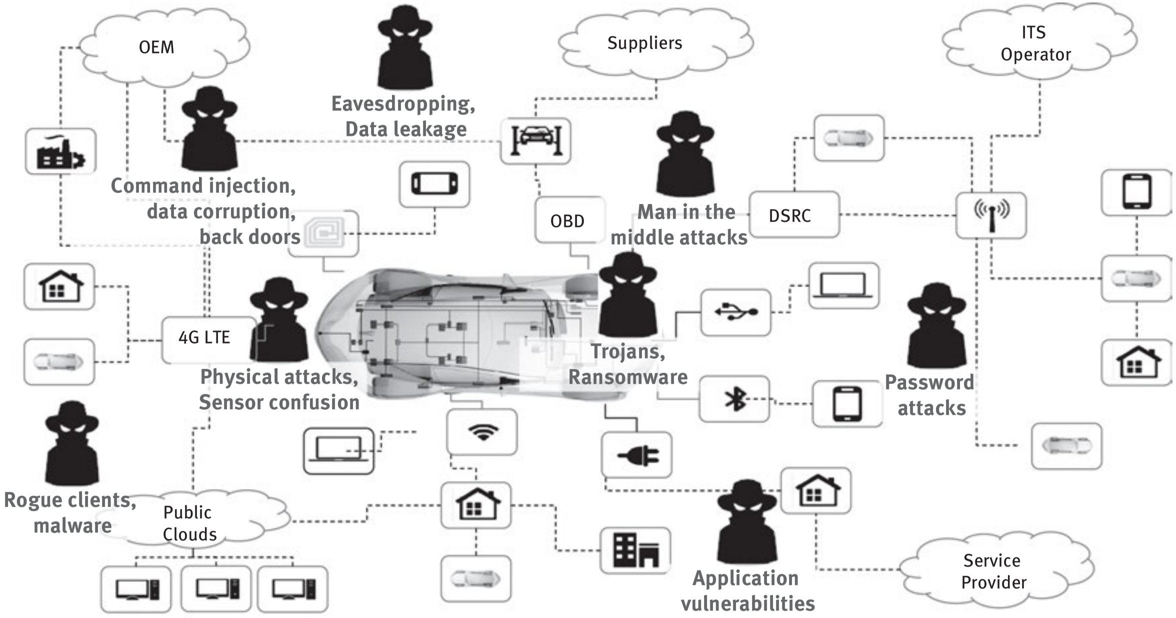


Figure 8.8: Increasing connectivity drives complexity and enables multiple attack paths.

requirements of the transmitted signals. If a network mainly contains signals with high security and safety requirements, that is classified as high security zone. The network with intermediate safety and security data is classified as a medium security zone. The network that contains just a few signals with safety and security requirements and many signals from remote connections is a low security zone. A network that contains an ECU with a remote connectivity must be treated as unsafe in principle and is therefore in a low security zone or even completely isolated (Figures 8.9 and 8.10).

The origin and distribution of the signals influences the settings of the firewall in the gateway. The presence of signals from other security zones gives an indication to the security measures for the internal signals. If a network is physically isolated and signals from other networks are rarely used, the threat potentials are low. Unless other threats from adversaries are identified, reduced measures can be applied for signals in such a security zone. This reduces efforts and costs for security measures of these ECUs. It shows how partitioning provides advantages. The signal flow from high to low security zones is not critical. However, if threats for data manipulation on the network are given, security measures like authentication or confidentiality can be added to the data. Greater care must be taken in the other direction. The risk potentials for these signals must be observed carefully. Also, if filter rules of the firewall can influence the complete security zone settings. For counter measures, authentication on signal may be required.

Security analysis provides requirements and test vectors and adequate measures can be derived for balanced costs and efforts. The results are useful in the partitioning phase when functionality is distributed to ECUs and networks. Networks isolated under security aspects helps to reduce the risks and efforts. Security key management will become an important part and requires a key infrastructure (PKI) managed by the OEM over the production and maintenance phase of the vehicle. Additionally, the secure key handling inside an ECU and the usage in development, production and maintenance phase must be considered. The PKI must be online to allow access by the workshops. Additionally, an OEM backend is needed that allows flash programming over the air, at least to provide hot fixes and patches. Such a backend can provide additional security and features to the car owners, but can also open new business divisions for OEMs.

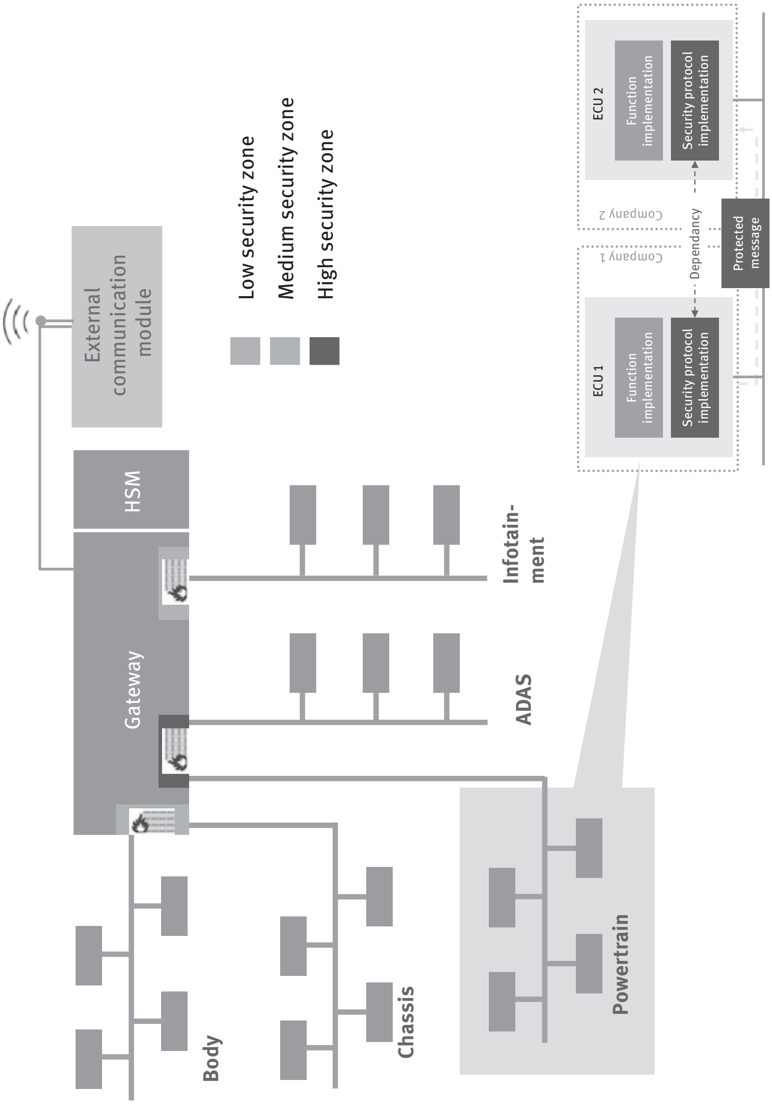


Figure 8.9: Security zones in automotive networks.

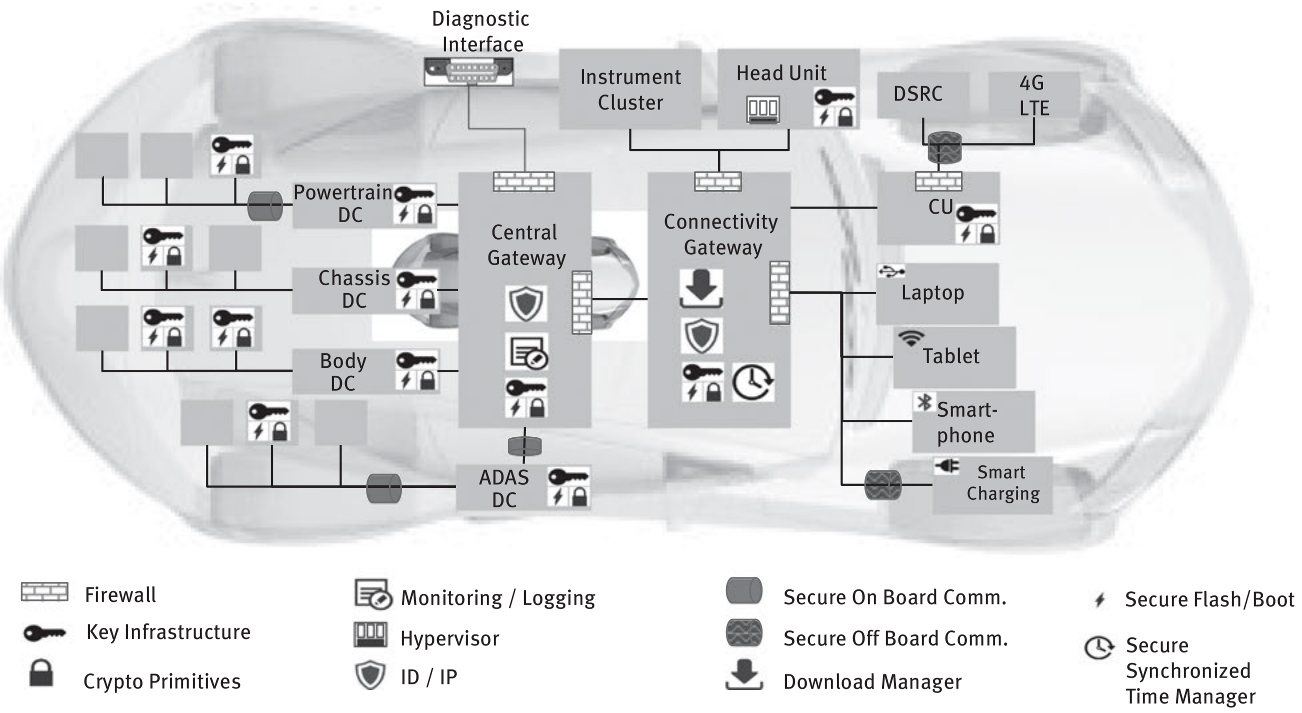


Figure 8.10: Complete reference architecture with security zones for automotive networks.

8.5 Conclusions

Modern risk management can be best portrayed with the challenges of cybersecurity. Modern society depends on connected systems, and exactly this connectivity is attacked to weaken our society. This article sheds a light on cybersecurity and related risk mitigation. We have used examples and a case study from the field with major cybersecurity threats, namely automotive. Automotive unlike other fields is the most innovative field in IT. Given the safety-criticality of these systems, it also shows actual risks with concrete hazards in case of manipulation.

Cybersecurity for connected systems has gained huge relevance with the convergence of IT and embedded systems. With the convergence of IT and embedded along all industries cybersecurity is a major requirement in any system. Isolated mechanisms such as distributed functionality in proprietary subsystems, protection on component-level, gateways and firewalls between components, validation of critical functions is insufficient.

The risks are obvious. Safety needs security as a mandatory condition, which means that any safety-critical system as a minimum must also be protected for cybersecurity. With introducing classic IT attack surfaces and vulnerabilities to critical infrastructures, the amount of attacks is fast growing. Being used across industries with high relevance on our society, such systems must be thoroughly protected and hardened.

Managing security risks is mandatory not only due to its safety-impact but also due to product liability. It is of no excuse anymore saying that hacking is inevitable. We must best possibly protect connected systems and prove that we have taken the necessary actions in terms of processes, education, management and technology. Testing plays a critical role in this process. Companies urgently need to build up necessary basic security expertise and obtain adequate external support, specifically where security meets safety. Mature development processes provide a good basis but need to be amended with dedicated security engineering activities as we have showed in this article.

In this article we have looked risk-oriented cybersecurity. Risk management is a life-cycle task. Risk-oriented security helps to balance growing security threats with increasing complexity over the entire life-cycle. Unlike many previous attempts our research and many practice projects indicate that while design for security is good, it is not good enough. Effective security must handle the entire life-cycle. Security must be integrated early in the design phase to understand the threats and risks to embedded functions. Grey-box test methodology evaluates risks and builds upon a TARA and vulnerabilities (Ebert 2020b). We showed how an initial security analysis and technical concept based on a given reference architecture first shows threats and risks and then is used to guide risk-oriented mitigation.

Each single engineer must manage risks – rather than just adding more functionality. Software process evangelist Tom Gilb once observed "*If you don't actively attack risks, they will actively attack you.*" That should by our guidance towards security. It is never comprehensive but can be vastly improved with risk-oriented security engineering.

References

Ebert, 2020a: Ebert, C., A. Kim, and J. van Genuchten: Technology Trends: Winning with ACES. IEEE Software, ISSN: 0740-7459, vol. 37, no. 2, pp. 6–13, May 2020.

Ebert, 2017a: Ebert, C.: Cyber Security Requirements Engineering. In: M. Ramachandran and Z. Mahmood (Eds): Requirements Engineering for Service and Cloud Computing. pp. 209–229. ISBN: 978-3-319-51309-6. Springer, 2017. http://www.springer.com/de/book/9783319513096

Ebert, 2017b: Ebert, C. and J.Favarro: Automotive Software. IEEE Software, ISSN: 0740-7459, vol. 34, no. 3, May 2017.

Ebert, 1997: Ebert, C.: Dealing with Nonfunctional Requirements in Large Software Systems. N.R. Mead, ed.: Annals of Software Engineering, Vol.3, pp. 367–395, Aug. 1997.

Islam, 2014: Islam, M. et al.: Project overview HEAVENS – Healing Vulnerabilities to Enhance Software Security and Safety, 2014.

ISO, 2020a: ISO 26262: Road vehicles – Functional safety, ed.2. Last accessed on 18. May 2020, www.iso.org

ISO, 2020b: ISO 21434: Road vehicles – Cybersecurity engineering. Last accessed on 18. May 2020, www.iso.org

Greenberg 2019: Greenberg, A.: Sandworm – A New Era of Cyberwar and the Hunt for the Kremlin's Most Dangerous Hackers. New York: Doubleday, 2019.

Ebert, 2020b: Ebert, C., Y.Rekik, and R.Karade: Security Test. IEEE Software, ISSN: 0740-7459, vol. 37, no. 2, pp. 13–20, Mrc 2020.

Kristina Kozakevitch, John Collins, C. Christopher Lee and Heechang Shin

9 Location-based access control enforcement

9.1 Introduction

Access control policies prevent unauthorized access to an organization's resources. In a mobile environment, physical location plays an important role in determining whether to grant or to deny access to resources. For example, a business can restrict access to certain resources to a group of selected employees. Incorporating location into the policy can strengthen security. One example of this would be a policy allowing access to important resources to managers only if they are currently located in the office. This would minimize the risks of those resources being accessible to unauthorized personnel. Decker (2008) specified that location-aware security enforcement is one of the most significant security requirements.

In a mobile environment, we assume that a location server enforces location-based access control policies. It is assumed that the location server tracks current locations of mobile entities. Indoor positioning systems locate moving entities in closed structures such as office buildings, hospitals, stores, factories, and warehouses with installed sensors. If the location server cannot track location information, the network service provider's network infrastructure can provide that information.

Currently no available technology can determine exact locations because of two main reasons: (1) measurement error and (2) sampling error. In terms of measurement error, accuracy of indoor positioning systems such as Active Badge (Want et al., 1992) is within 2 meters. The accuracy of network-based location sensing technologies is 50 meters in urban areas (Laitinen et al., 2001). Recent work by Han et al. (2007) and Koutsou et al. (2007) has improved to a rate of less than one meter of average measurement error, but the error can be more than one meter in the worst case. The problem of sampling error comes from the fact that it is unrealistic to obtain the current location of a moving object continuously under the existing location sensing and database technologies. These technologies measure positions at discrete intervals such as a few seconds. The system does not know the location of a moving object during these intervals. Most current moving object databases maintain an approximate value of the location in order to minimize updates for efficient query processing (Marsit et al., 2005).

Considering the inherent uncertainty of location measures, a location-based access control system must consider uncertainty when evaluating an access control request. Otherwise, access control decisions are based on imprecise location estimates. Consider a case in which access to certain important resources can be restricted to employees who are currently located within the office area. What if the

https://doi.org/10.1515/9783110652321-009

claimed location is incorrect and access is given? Most security systems assume the accuracy of provided location data. In GEO-RBAC (Damiani et al., 2007), a spatial role activates based on the location of the user; however, since location measures stored in the database do not capture the exact location, this uncertainty causes security concerns because the role activation in GEO-RBAC cannot guarantee the desired level of security (Garcia-Alfaro, 2010). Due to such risk, it is essential that all location-based access control systems incorporate the concept of uncertainty within their models (Decker, 2008).

9.2 Location-based access control model

Other academics have previously proposed a few models to support the idea of spatially aware access control systems. Atluri and Chun (2004) introduced an access control model for geographical data for satellite image maps. Bertino et al. (2004) proposed an access-control system for geometric- and vector-based spatial data, but it does not support multi-granularity of spatial data. In order to overcome such issues, Damiani et al. (2007) proposed GEO-RBAC, an extension of the RBAC model enhanced with spatial-and location-based information. Data regarding the position of the user activates spatial roles (Damiani et al., 2007); however, GEO-RBAC cannot judge the correctness of an access control request because it does not consider uncertainty within the claimed location. It assumes that all the claimed locations are precise. Thus, it bases access control decisions on imprecise location estimates. It does not guarantee the correctness of the access control system. For this reason, all location-based access control systems must incorporate the concept of uncertainty within their models.

In order to address location uncertainty, one can model location as an uncertainty region represented as a circle with radius *r*, which is called as the uncertainty threshold. Location sensing technology's measurement errors and sampling errors determines *r* (Shin and Atluri, 2009). The size of *r* is specified as the maximum possible distance away from a certain point that a mobile entity can be since the last location update. Figure 9.1 illustrates location uncertainty. The center is the location of the last location update, and the radius *r* specifies the maximum distance that the mobile entity can move since the last update.

Since the probability density function (PDF) of uncertainty region is usually unavailable, one can use a set of samples to approximate the PDF (Shin and Atluri, 2009). In some other applications, it would make sense to use the specific PDF for specifying the uncertainty region. For example, Wolfson and Yin (2003) proposed that location follows the Gaussian distribution over the uncertainty region. Also, many application scenarios (Ardagna et al., 2006) use uniform distribution to represent an uncertainty region.

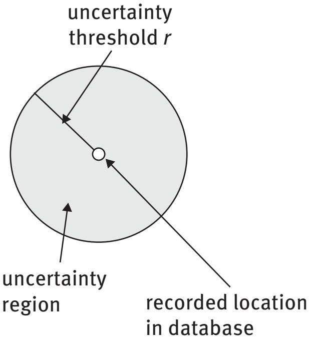


Figure 9.1: Uncertainty Model.

Traditional location-based services usually query using position-based conditions based on the location of the mobile entity in the form of:

in_area (*mobile entity, authorized region, threshold level*) → *Boolean value*

This states whether or not (depending on whether the Boolean value is *true* or *false*) the evaluation of the location predicate, *in_area* (i.e., check whether a *mobile entity* is located in the area of the *authorized region)* over parameters satisfies certain conditions discussed below. The following are assumed:

- *mobile entity* is used to uniquely identify each mobile entity known to the location server. Mobile entity could be either a user of the system (who requests access to resources) or a resource that a user would like to get information on.
- *authorized region* can be specified by geometric modeling as in the case of using geocoordinates to represent rectangular shape or symbolic modeling as in the case of references to real world entities such as cells, streets, cities, zip codes, buildings, and so on (Marsit et al., 2005).
- *threshold level* expresses the level of reliability that the location server can guarantee, according to the accuracy of the estimated location.

The confidence level of a location predicate's satisfiability is associated with measurement errors and sampling errors, as we discussed earlier. This, in turn, depends on the technology used for localizing the requester, the location server quality, and so forth. The confidence level is computed by the integration of PDF over the overlapped region between the authorized region and the uncertainty region (Shin and Atluri, 2009). More specifically, given an authorized region R and a mobile entity's uncertainty region denoted as ur, its confidence level (p) is computed as:

$$p = \int_{ur \cap R} f(x)dx \tag{9.1}$$

where x is the location of the mobile entity, f is the probability density function, and $ur \cap^{R}$ is the intersection of ur and R. In the case of location-based conditions, satisfiability of subject (or object) expression's location predicate is determined based on the predetermined location predicate threshold level, i.e., it satisfies only if $p \geq p_c$ where p the confidence level of the given location predicate and p_c is the location predicate threshold level. If $p \geq p_c$ satisfies, the result of the location predicate is true. Otherwise, it is false.

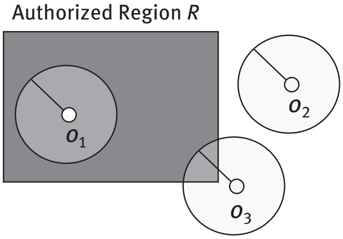


Figure 9.2: Access Control Decision Examples.

Figure 9.2 illustrates examples of mobile users' access requests. In Figure 9.2, o_2's request will be rejected since there is no overlap which implies zero confidence level; however, since the authorized region completely encloses o_1's uncertainty region, access is granted because the confidence level is 100%. When any overlap exists between the uncertainty region and the authorized region (i.e., o_3), an evaluation of the Equation (9.1) is necessary to check if the confidence is greater than or equals to the location predicate threshold level. This can be computationally expensive.

The remainder of the section will discuss how location-based access control policies are specified. We assume that each mobile entity (each assigned with a unique identifier) has registered with the location server. Registered users are assumed to have other user profile information such as name, address, and so on stored in the server. Objects are the data or services for which users can make requests. They also are associated with a set of properties through an object profile (Ardagna et al., 2006). Abstractions can also be defined within the domain of objects, allowing to group together objects with common characteristics and to refer to the group with a single name (Ardagna et al., 2006).

In general, an access control rule is specified with <*s, o, p*>. This specifies that the subject *s* has authorization to exercise privilege *p* on the object *o* (Shin and Atluri, 2009); however, this basic access control rule lacks specification power to include mobile data since an access control rule should be capable of specifying based on spatiotemporal attributes of both subjects and objects that are functions of time (Shin and Atluri, 2009). To address this, Shin and Atluri (2009) introduced an access control rule that is a triple of the form ‹*se*, *oe*, *pr*›. In this rule, where *se* is a subject expression that denotes a set of authorized subjects, *oe* is an object expression denoting a set of authorized resources, and *pr* is a set of privilege modes denoting the set of allowed operations.

Definition 1. Location-based access control policy. A location-based access control policy is a triple of the form ‹*subject expression*, *object expression*, *pr*› where

- *subject expression* includes (1) the spatial role of the subject and (2) the location predicate that links to the given time and space and its uncertainty threshold level;
- *object expression* includes (1) an object type specifying the membership of the object and (2) the location predicate that links to the given time and space and its uncertainty threshold level;
- *pr* is a set of privilege modes denoting the set of allowed operations.

One can classify the conditions specified in the subject or object expressions into two categories: generic conditions and location-based conditions (Ardagna et al., 2006). Generic conditions evaluate membership of subjects or objects into roles or attribute values in their profiles. Suppose the following three location-based access control policies:

- α_1 = NCO Unit can track the locations of dispatched police squad cars currently located within New York City within over 80% of the location predicate threshold.
- α_2 = *S*ecurity managers currently located within the server room within 90% of the location predicate threshold who can read the mobile network data.
- α_3 = *S*ecurity personnel currently located within the retail outlet within over 80% of the location predicate threshold who can track their employees' current locations if they are located within the retail outlet with greater than 60 % of location predicate threshold.

Each access control policy has its own threshold level specified for location-based condition, i.e., α_1's 80% in object expression, α_2's *90*% in subject expression, 80% in subject expression and 60% in object expressions of α_3. Table 9.1 shows the examples of access control policies discussed above.

Refer to Shin and Atluri (2009) for more detailed information on how to specify location-based access control policies.

9.3 Efficient processing of location predicate satisfiability

Shin and Atluri (2009) introduced the upper and lower bounds of the region being evaluated to address the issue of efficient location predicate satisfiability processing. More specifically, Shin and Atluri (2009) introduced two spatial filters, $R_{\min}$ and $R_{\max}$. Here, $R_{\min}$ is the region that guarantees the correctness of the location

Table 9.1: Examples of Access Control Policies.

Location-based access control policies	Subject Expression		Object Expression		Privileges
	Generic conditions	Location-based conditions	Generic conditions	Location-based conditions	
1	user.role = NCO unit		object.type = police squad car	in_area (car id, New York City, 80%)	track
2	user.role = security manager	in_area (employee id, server room, 90%)	object.type = mobile network data		read
3	user.role = security personnel	in_area (employee id, retail outlet, 80%)	object.type = employees' current locations	in_area (employee id, retail outlet, 60%)	read

predicate evaluation if the claimed location is contained in this region, and $R_{\max}$ is the region where any claimed location exterior of this region is guaranteed not to satisfy the given location predicate. Once these filters are found, the system overhead for evaluating location predicate condition significantly reduces because it requires simple location containment tests to evaluate the predicate correctness for most of the location measures instead of expensive computation of Equation (9.1) for all the candidate mobile entities (Shin and Atluri, 2009). The following steps summarize this evaluation process:

1. Those mobile entities located outside of R_{max} are filtered from the candidate set.
2. Those mobile entities located within the interior and boundary of R_{min} satisfy the location predicate condition ($p \geq p_c$ is true).
3. One needs to compute p in Equation (9.1) only for those entities located in the region of the intersection of the exterior of R_{min} and the interior and the boundary of R_{max}.

Figure 9.3 illustrates the use of $R_{\min}$ and $R_{\max}$ for the authorized region, R. In Figure 9.3, the center of o_1 is located in the interior of $R_{min.}$ This implies that o_1's confidence level is greater than or equals the threshold level. Since o_1 is located inside $R_{min,}$ $p_{o1} \geq p_c$ is true. This implies that the location predicate test result is true for o_1, thus it is not necessary to compute Equation (9.1) for o_1. Since o_2 is located exterior of R_{max}, $p_{o2} < p_c$ holds. Thus, this does not satisfy the location predicate. We do not know how the result of the location predicate evaluation for any location measure located in the intersection of the exterior of R_{min} and the interior and the boundary of R_{max}. Thus, in Figure 9.3, we should compute o_3's confidence level using Equation (9.1) in order to evaluate whether $p_{o3} \geq p_c$ holds. This example illustrates the importance of keeping the intersection as small as possible to reduce the computational overhead. Without the loss of generality, we restrict our discussion on 2-dimensional space. This provides an easier illustration and allows for a simpler extension to a higher dimensional space.

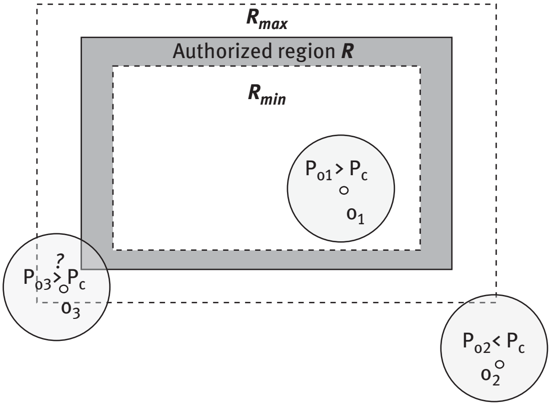


Figure 9.3: Use of $R_{\min}$ and $R_{\max}$.

Uniform Distribution Case: Given an authorized region $R = [a_1, b_1] \times [a_2, b_2]$ where $a_1 < b_1$ and $a_2 < b_2$, we want to find c_{in} and c_{out} such that $R_{\min} = [a_1 + c_{in}, b_1 - c_{in}] \times [a_2 + c_{in}, b_2 - c_{in}]$ where $a_i + c_{in} < b_i - c_{in}$ for $i = 1, 2$ and $R_{\max} = [a_1 - c_{out}, b_1 + c_{out}] \times [a_2 - c_{out}, b_2 + c_{out}]$ where $a_i + c_{in} < b_i - c_{in}$ for $i = 1, 2$. Under the uniform distribution assumption of PDF, p in Equation (9.1) is computed as $p = \frac{area(ur \cap R)}{area(ur)}$ where *area*() returns the area of the given region. Shin and Atluri (2009) found the following results by approximating the uncertainty region using squares:

- $cin = \max\left(r\left(\sqrt{\pi p_c} - 1\right), 0\right)$
- $c_{out} = max\left(\left(1 - \frac{1}{2}\pi p_c\right)r,\ 0\right)$

Once c_{in} and c_{out} are computed, finding the corresponding R_{min} and R_{max} for a given authorized region R, is straightforward. Refer to Shin and Atluri (2009) for more details and examples.

General Probability Distribution Case: In practice, it is important to have $R_{\min}$ and $R_{\max}$ without assuming any specific probability distribution. Shin and Atluri (2009) addressed this issue by introducing probabilistically constrained lines (PCL). Suppose x is the mobile entity's location and f is the probability density function. Given a mobile entity's uncertainty region *ur* and a threshold level p_c, L_i and H_i are lines that are perpendicular to the i_{th} dimension. The probability that a mobile entity is located within the overlapping area between *ur* and L_{i+} equals to the specified uncertainty threshold level ($p = \int_{o.ur \cap L_2^+} f(x)dx = p_c$), and in the same way, the probability that a mobile entity is located within the overlapping area between *ur* and H_{i-} equals to the specified location predicate threshold level, i.e., $p = \int_{o.ur \cap H_2^-} f(x)dx = p_c$. Figure 9.4 illustrates two PCLs, L_2 and H_2.

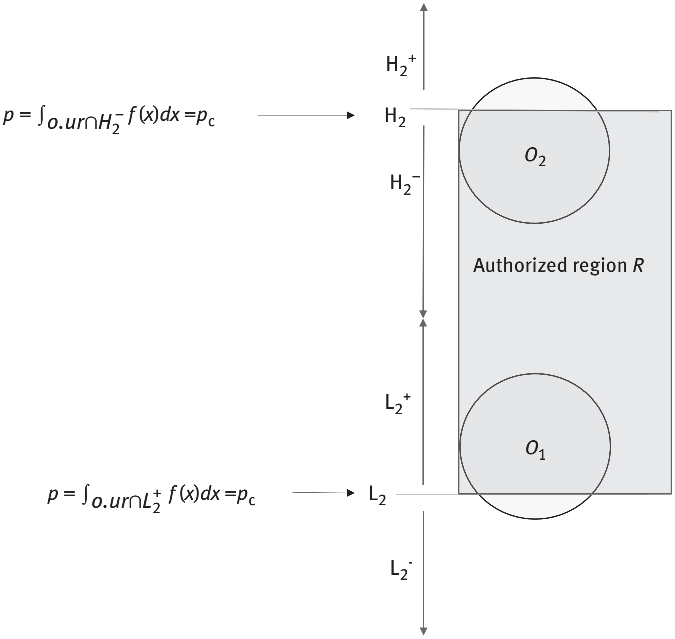

$p = \int_{o.ur \cap H_2^-} f(x)dx = p_c$
H_2^+
H_2
O_2
H_2^-
Authorized region *R*
L_2^+
O_1
$p = \int_{o.ur \cap L_2^+} f(x)dx = p_c$
L_2
L_2^-

Figure 9.4: Probability Constraint Lines L_2 and H_2.

One can find PCLs with small overhead because they can be computed by considering each individual dimension in turn (i.e., marginal distribution of each dimension). In Figure 9.4, *o_1's* L_2 touches the south boundary of the authorized region *R* and *o_1's ur* touches the west boundary of *R*, and *o_2's* H_2 touches the north boundary of *R*, and *o_2's ur* touches the west boundary of *R*. Observations are as follows:

- **Observation 1**: Any mobile entity *o* located vertically higher than o_1 will in turn have higher *p* until its uncertain region *ur* touches *R*'s north boundary, thus satisfying $p_o > p_{o1} = p_c$ *where* p_o is the confidence level of *o*, p_{o1} is the confidence level of o_1, and p_c is the location predicate threshold level.
- **Observation 2**: Any mobile entity *o* located vertically lower than o_2 will in turn have higher *p* until its uncertain region *ur* touches *R*'s south boundary, thus satisfying $p_o > p_{o2} = p_c$ where p_o is the confidence level of *o, p_{o2}* is the confidence level of *o2*, and p_c is the location predicate threshold level.

- **Observation 3**: Any object *o* located east of o_1 will have the same confidence level with that of o_1 until its uncertainty area touches *R*'s east boundary, thus satisfying $p = p_{o1} = p_c$ where p_o is the confidence level of *o*, p_{o1} is the confidence level of *o1*, and p_c is the location predicate threshold level.
- **Observation 4**: After finding objects such as o_1 (i.e., o_1*'s* L_2 touches the south boundary the authorized region *R* and it touches the west boundary of *R*) and o_2, any mobile entity *o* located vertically lower than o_1 or located vertically higher than o_2 will have lower *p*, thus satisfying $p_o < p_{o1} = p_{o2} = p_c$ where p_o is the confidence level of *o*, p_{o1} (or p_{o2}) is the confidence level of o_1 (or o_2), and p_c is the location predicate threshold level.

Based on the above observations, Figure 9.5 illustrates a boundary of the rectangle that can be used as a filter. Once o_1, o_2, o_3, o_4 are identified where (i) o_1*'s* L_2 touches the south boundary of the authorized region *R* and o_1*'s ur* touches the west boundary of *R*, (ii) o_2*'s* H_2 touches the north boundary of *R* and o_2*'s ur* touches the west boundary of *R*, (iii) o_3*'s* L_2 touches the south boundary the authorized region *R* and o_3*'s ur* touches the east boundary of *R*, and (iv) o_4*'s* H_2 touches the north boundary the authorized region *R* and o_4*'s ur* touches the east boundary of *R*, a rectangle represented with four corners by using centers of o_1, o_2, o_3, o_4 can be generated.

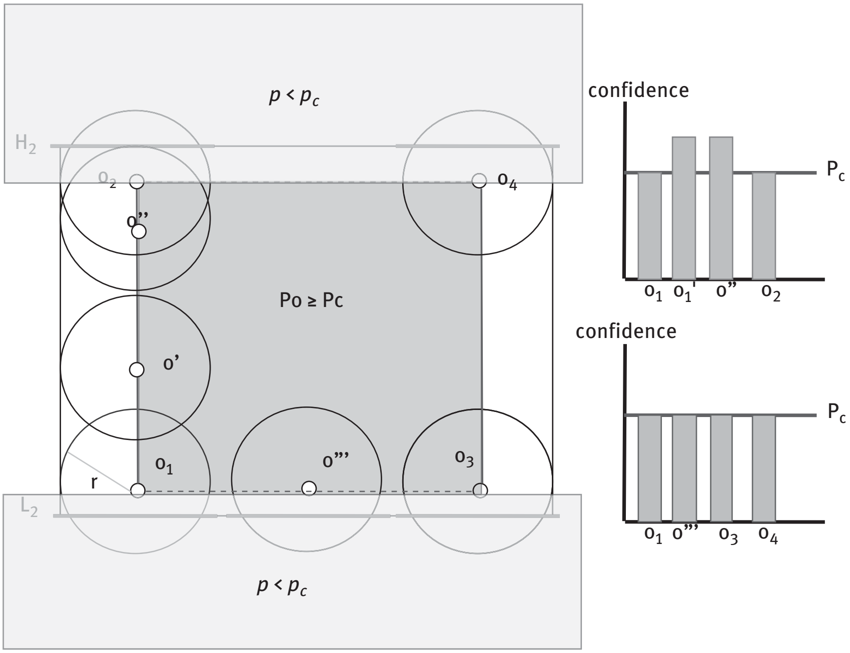


Figure 9.5: Illustration of Finding R_{min} and R_{max}.

Figure 9.5 illustrates this with dashed lines. Properties of such a rectangle are as follows:

- Any mobile entity *o* located in the interior and within the boundary of the rectangle satisfies $p_o \geq p_{c,}$ where p_o is the confidence level of *o*, and p_c is the location predicate threshold level.
- Any mobile entity *o'* located in north and south of the rectangle satisfies $p_{o'} < p_c$ where $p_{o'}$ is the confidence level of *o'*, and p_c is the location predicate threshold level.

The same steps can be followed for the east and west sides of the authorized region, and R_{min} and R_{max} can be identified as illustrated in Figure 9.6. For more details, refer to Shin and Atluri (2009).

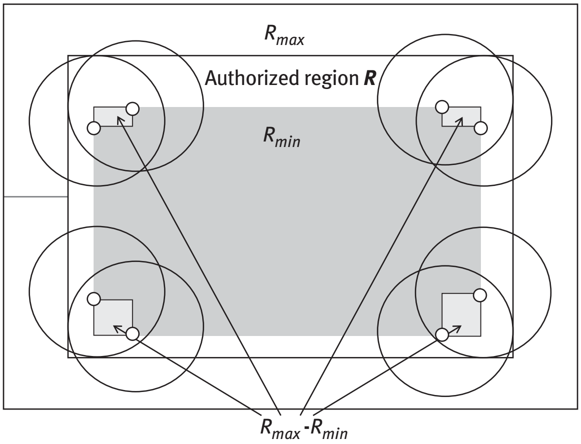


Figure 9.6: Illustration of R_{min} and R_{max} for Arbitrary PDF.

9.4 Evaluation of access requests

We assume that a location server enforces location-based access control policies. We also assume that the location server maintains current locations of mobile entities. The location server maintains the authorization base (the collection of the access control policies) and the moving object database that stores current locations of mobile entities as illustrated in Figure 9.7. We are now ready to discuss the evaluation process of access control policies, starting with the access requests submitted to the location server.

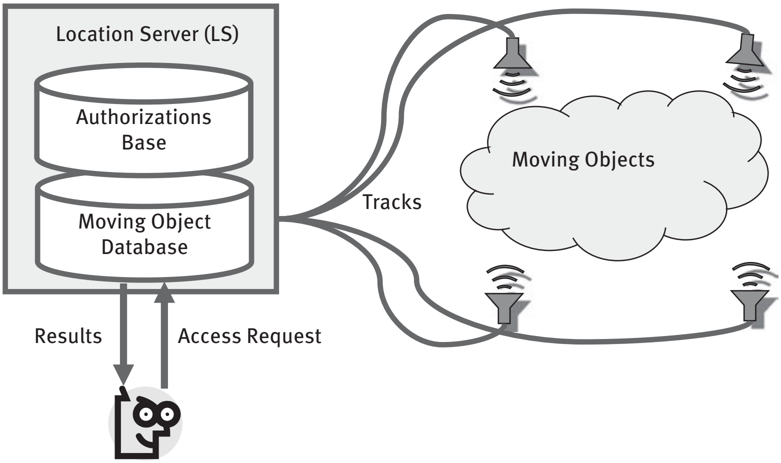


Figure 9.7: Evaluation of Access Requests with Uncertain Location Estimates.

Definition 2. Access request. An access request is a 3-tuple of the form ‹*user id*, *object expression*, *action*› where

- *user id* is the identifier of the user who makes the request;
- *object expression* is the object type that specifies the membership of the object and the optional location predicate linked to the given time and space and its uncertainty threshold level;
- *action* is the requested action

An access request results in allowing the accessibility of a set of resources. Given an access request, the location server first evaluates the authorization base and collects the access control policies applicable to the requester. Then, the subject expressions and object expressions of the access control policies are compared to the access request.

In the first phase, subject expressions of those applicable policies are evaluated. If the subject expressions of those applicable policies contain generic conditions only (evaluating membership of subjects into roles or attribute values in their profiles), their resource expressions are added to the authorized resource expressions. For example, if the subject expression of an applicable policy is of the form, *user.role = employee,* and the requestor is an employee, the subject expression is evaluated as true. If the subject expression contains location predicates, it is first simplified by evaluating all the generic conditions (Ardagna et al., 2006). For example, if the subject expression is (*user.role = security manager* ∧ *in_area*(*employee id*, *server room*, 90%) and the requester is a security manager, the simplified subject expression is (*True* ∧ *in_area* (*employee id*, *server room*, 90%)), that is, *in_area* (*employee id*, *server room*, 90%). This condition cannot be further simplified, and the location server begins to evaluate location-based conditions. For each applicable access control policy's subject expression:

1. If the requester is located exterior of R_{max}, this subject expression is not applicable.
2. If the requester is located interior and boundary of R_{min}, its resource expression is added to the authorized resource expressions.
3. If the requester is located within the intersection of exterior of R_{min} and interior and boundary of R_{max}, the requester's confidence specified in Equation (9.1) is computed. If it is greater than or equals to the threshold level, its resource expression is added to the authorized object expressions.
4. Move to the next applicable access control policy and repeat from the step 1.

In the second phase, the authorized resource expression is evaluated. Similar to evaluating subject expressions, if the authorized resource expression contains generic conditions only (i.e., evaluating membership of objects into roles or attribute values in their profiles) and if it is evaluated to be *true*, its *privilege modes* are added to the authorized privilege modes. If the action of the access control request is included in the authorized privilege modes, access is granted. If the resource expression contains location predicates, it is first simplified by evaluating all the generic conditions, and the location server begins to evaluate location-based conditions. The authorized resource expression is now compared with the access control request's resource expression. If the resource expression of the access request includes generic conditions only, this access control policy is not applicable, thus rejecting the request. If the resource expression of the access request includes location-based conditions, evaluating all the generic conditions first and searching all the applicable moving entities, denoted as O_q, then, for the given authorized resource expression:

1. Search all the objects located within R_{min}, denoted as O_{c1}.
2. Search all the objects located in the intersection of the exterior of R_{min} and the interior and the boundary of R_{max} denoted as O_{c2}.
3. For each resource o in O_{c2}, if $p_o \geq p_c$ (where p_o is the confidence level of o, and p_c is the location predicate threshold level), add o to O_{c3}.
4. Compute $O_q \cap (O_{c1} \cup O_{c3})$ and add to the set of authorized resources.
5. Add privilege modes of the given access control policy to the authorized privilege modes.

If the action of the access control request is included in the authorized privilege modes, access is granted for the set of authorized resources. Refer to Shin and Atluri (2009) for more details.

9.5 Conclusion

In this chapter, we raised the issue of uncertain location measures regarding security in a mobile environment and presented a solution to enforce access control policies by considering location uncertainty. We have discussed the uncertainty-embedded authorization model, efficient enforcement algorithms, and how to handle user requests. As a future work, one could propose an index structure for authorizations considering uncertainty to improve the evaluation process. Then, it would be beneficial to develop enforcement algorithms based on the proposed index structure.

References

Ardagna, Claudio A., et al. "Supporting location-based conditions in access control policies." *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. 2006.

Atluri, Vijayalakshmi, and Soon Ae Chun. "An authorization model for geospatial data." *IEEE Transactions on Dependable and Secure Computing* 1.4 (2004): 238–254.

Bertino, Elisa, Maria Luisa Damiani, and Davide Momini. "An access control system for a web map management service." *14th International Workshop Research Issues on Data Engineering: Web Services for e-Commerce and e-Government Applications, 2004. Proceedings*. IEEE, 2004.

Damiani, Maria Luisa, et al. "Geo-rbac: A spatially aware rbac." *ACM Transactions on Information and System Security (TISSEC)* 10.1 (2007a): 2-es.

Decker, Michael. "Modelling location-aware access control constraints for mobile workflows with UML activity diagrams." *2009 Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*. IEEE, 2009.

Decker, Michael. "Requirements for a location-based access control model." *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia*. 2008.

Garcia-Alfaro, Joaquin, et al., eds. *Data Privacy Management and Autonomous Spontaneous Security: 4th International Workshop, DPM 2009 and Second International Workshop, SETOP 2009, St. Malo, France, September 24–25, 2009, Revised Selected Papers*. Vol. 5939. Springer, 2010.

Han, Soonshin, HyungSoo Lim, and JangMyung Lee. "An efficient localization scheme for a differential-driving mobile robot based on RFID system." *IEEE Transactions on Industrial Electronics* 54.6 (2007): 3362–3369.

Koutsou, Aikaterini D., et al. "Preliminary localization results with an RFID based indoor guiding system." *2007 IEEE International Symposium on Intelligent Signal Processing*. IEEE, 2007.

Laitinen, Heikki, Jaakko Lahteenmaki, and Tero Nordstrom. "Database correlation method for GSM location." *IEEE VTS 53rd Vehicular Technology Conference, Spring 2001. Proceedings (Cat. No. 01CH37202)*. Vol. 4. IEEE, 2001.

Marsit, Nadhem, et al. "Query processing in mobile environments: A survey and open problems." *First International Conference on Distributed Frameworks for Multimedia Applications*. IEEE, 2005.

Shin, Heechang, and Vijayalakshmi Atluri. "Spatiotemporal access control enforcement under uncertain location estimates." *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, Berlin, Heidelberg, 2009.

Subramanian, Suguna P., et al. “RIL – reliable RFID based indoor localization for pedestrians.” *2008 16th International Conference on Software, Telecommunications and Computer Networks*. IEEE, 2008.

Want, Roy, et al. “The active badge location system.” *ACM Transactions on Information Systems (TOIS)* 10.1 (1992): 91–102.

Wolfson, Ouri, and Huabei Yin. “Accuracy and resource consumption in tracking and location prediction.” *International Symposium on Spatial and Temporal Databases*. Springer, Berlin, Heidelberg, 2003.

Sungjin Yoo and He Li

10 Information security management in the cloud computing environment

10.1 Introduction

Cloud computing is defined by the US National Institute for Standards and Technology (NIST) as “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.” The NIST definition of cloud computing is mainly from the technical aspect and emphasizes the platform running applications. While the technical platform is certainly an important element to understand cloud computing, researchers have extended the original technical-oriented definition to incorporate the types of applications running on the cloud platform. For example, Boss et al. (2007) view cloud computing as a platform that dynamically provisions, configures, reconfigures, and de-provisions computing resources as needed, enabling applications to be accessible through the Internet and scalable using powerful servers in large hosting data centers. Cloud computing supports a wide range of access devices, including PCs, laptops, smartphones, tablets, Internet-enabled sensors, and other forms of mobile computing and smart devices (Cubitt et al., 2011; Iyer & Henderson, 2010; Pritchard, 2012).

There are several different types of cloud computing models that organizations can use (Kavis, 2014). Vertically, depending on the level of computing resources, customers have the choices of three different cloud service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Horizontally, depending on the exclusiveness of computing resources made to customers (Liu et al., 2012), a cloud computing customer can deploy cloud computing in one of the following four models: private, community, public, and hybrid clouds (Reddy & Reddy, 2011). Figure 10.1 illustrates these different cloud computing options.

10.1.1 Cloud computing service models

10.1.1.1 Infrastructure as a service

In an IaaS model, cloud computing vendors develop physical computing resources, including networks, storage, servers, and virtualization. Cloud vendors rent these physical computing resources to their clients by providing and maintaining interfaces and resource abstractions such as virtual networks/machines. IaaS cloud clients

https://doi.org/10.1515/9783110652321-010

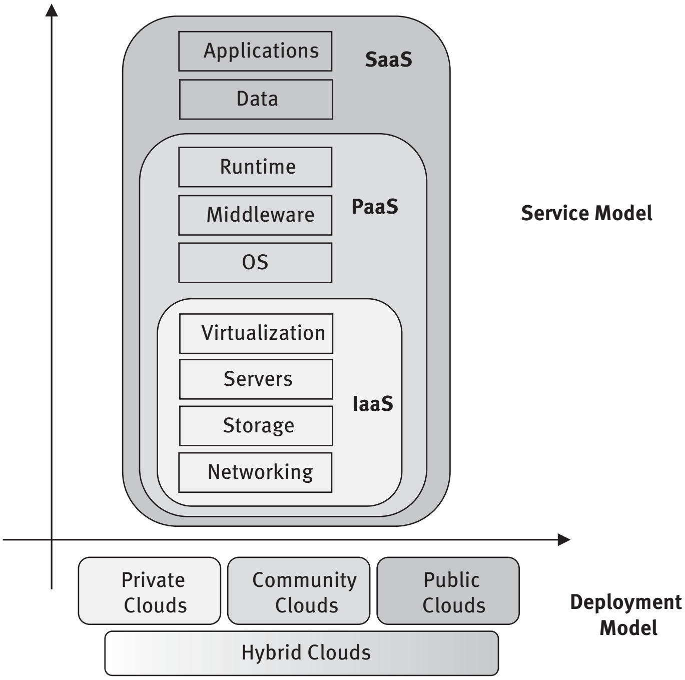


Figure 10.1: Cloud Computing Models.

only purchase access to the computing resources from the vendor and develop their own arbitrary software such as operating systems and applications running on the IaaS resources. In the IaaS model, cloud vendors control the underlying cloud infrastructure. In contrast, cloud customers retain full control over the operating systems and applications and possibly some control of certain networking components such as host firewalls. Major public companies, including Amazon, IBM, and Google, are utilizing the IaaS model. IaaS cloud computing also has several benefits for startups or firms with limited IT resources, such as low-cost, high efficiency, and faster development (Liu et al., 2012; Kavis, 2014).

10.1.1.2 Platform as a service

The PaaS model adds the platform components, including operating systems, databases, runtime software execution stake, and other middleware components, into the physical computing resources available in an IaaS. The PaaS cloud vendors often provide development tools such as software development kits (SDKs) and integrated development environments (IDEs). Clients can develop, deploy, and manage their applications using these development tools, along with high-level programming languages, services, and libraries offered by the cloud vendors. In the PaaS model, customers retain control over the applications and sometimes the limited hosting environment specifications, while PaaS cloud vendors manage the infrastructure and platform resources. PaaS offers customers the capabilities of executing applications without installing the hardware and operating systems (Liu et al., 2012; Kavis, 2014).

10.1.1.3 Software as a service

Cloud computing vendors with a SaaS model develop, deploy, configure, and maintain the full stack of infrastructure, platform, and applications to provide the expected service levels to their customers. SaaS cloud customers can access applications directly from various client devices through interfaces such as Web browsers (e.g., Gmail and Yahoo! Mail) with an Internet connection. The SaaS model is also known as on-demand software services. SaaS is relatively cheap for customers as they only need to pay fees associated with licensing, installation, and maintenance. SaaS cloud customers lose their capabilities of controlling the cloud infrastructure, platform, and even individual applications that have minimal administrative control of the applications (Liu et al., 2012; Kavis, 2014).

As organizations migrate from an in-house data center to cloud computing, the cloud vendors are responsible for the security of using a cloud, including infrastructure, operating systems, and applications. Depending on which cloud service models cloud customers employ, the level of security responsibility for customers vary. For example, in an IaaS model, the infrastructure of cloud computing, such as physical facilities, data centers, network interfaces are managed by cloud vendors. PaaS models add a few more factors to cloud vendors' duties, such as operating systems and middleware. However, despite the cloud service models, cloud customers still maintain responsibility for the security of their own. Furthermore, the responsibility to verify the security requirements complying with the regulations or policies in the industries always lies with the customer (Chou, 2013).

10.1.2 Cloud computing deployment models

10.1.2.1 Private cloud

With a private cloud, an individual cloud consumer has the exclusive right of accessing to and using the computing resources. The computing resources in a private cloud are managed by cloud consumers themselves or by a third-party manager and be hosted by their own premises (i.e., on-side private clouds) or outsourced to an external vendor (i.e., outsourced private clouds). Multiples consumers inside the organization (such as different business units) can share the private cloud provisioned for the single use of the organization (Reddy & Reddy, 2011).

10.1.2.2 Community cloud

A community cloud often refers to the scenario when a group of customers from organizations use the same provisioned cloud resources. The community formed by more than one customer often have shared concerns and similar interests such as goal, mission, security requirements, and compliance policy. Similar to a private cloud, the community cloud may be implemented on-premise (i.e., on-site community cloud) or outsourced to a third-party host (i.e., outsourced community cloud); and could be managed by the communities or by an external vendor. Compared to the private cloud, deploying organizations can save costs in a community cloud model since the services are shared among various organizations (Garg et al., 2017; Reddy & Reddy, 2011).

10.1.2.3 Public cloud

A public cloud is often owned and managed by a cloud computing vendor and serves many client organizations. The cloud infrastructure and computing resources are open to the general public over a public network. The public cloud usually has a lower cost for customers than other types of cloud deployment models (Garg et al., 2017; Reddy & Reddy, 2011).

10.1.2.4 Hybrid cloud

A hybrid cloud refers to the deployment of two or more different types of clouds (i.e., any combination of on-site or outsource, and private, community, or public). The different types of clouds operate as distinct entities but are coupled to technologies (i.e., either standardized or proprietary) enabling data and application portability. It is easier to move applications from one cloud to another in a hybrid cloud (Reddy & Reddy, 2011).

10.2 Characteristics and benefits of cloud computing

Cloud computing's unique service models, deployment models, and their possible combinations indicate several attributes that characterize the cloud computing technology, which enhances its capabilities of delivering competitive advantages for adopting organizations. Major characteristics of cloud computing are summarized in Table 10.1.

Table 10.1: Major Characteristics of Cloud Computing.

Characteristic	Description
On-Demand Self-Service	Cloud computing users can access the server capacities, uptime, and allocated network storage in real-time, enabling users to monitor the computing capabilities (Takabi et al., 2010)
Ubiquitous Network Access	Cloud computing enables users to access and upload data to the cloud using a proper device along with the Internet connection, which overcomes the geographical location constraints (Ali et al., 2015).
Resource Pooling	Cloud computing providers supply computing resources to offer services to several customers (Takabi et al., 2010). The computing resources are facilitated by the multi-tenant model, which allows cloud vendors to manage and control resources efficiently through the partition of a virtualized infrastructure shared by multiple customers. The multi-tenant model is particularly relevant for public clouds. Cloud computing users often have no information or control about the actual physical locations of the infrastructure and computing resources but can specify location at a higher abstraction level.
Rapid Elasticity	Cloud computing increases users' scalability of resources, which refers to the capabilities of quickly adding or removing resources in different granularity to efficiently match resources to workload (Venters & Whitley, 2012). The rapidity of the resource scalability is denoted as elasticity. The granular elements in traditional on-premise computing are servers that have slow processing speed and are expensive to acquire. Given that cloud computing can dynamically provision, configure, reconfigure, and de-provision services in an on-demand manner (Boss et al., 2007), it provides the capabilities of elastically scaling.
Measured Service with Pay-Per-Use	Cloud computing providers retain control and can optimize the use of computing resources on the cloud infrastructure through methods and tools such as load balancing, automated resource allocation, and metering (Ryoo et al., 2013). This automatic analysis feature of cloud computing also increases transparency for both customers and service providers. The resource utilization monitoring systems allow cloud computing service providers to adopt a pay-per-use business model, in which the prices for customers depend on their actual usage of the cloud computing resources. Cloud computing customers can save costs by avoiding hidden or extra fees.

Through the combination of its major characteristics, as summarized in Table 10.1 (Mell & Grance, 2011), cloud computing increases the ease of access of services and business continuity, reduces investments in IT infrastructure, and provides infinite and expandable resources and capacity (Naldi & Mastroeni, 2016). Organizations' effective use of cloud computing also delivers strategic and transformative impacts such as strengthening organizational agility and adaptability, improving functional competencies (i.e., operational capabilities), and enhancing the dynamic capabilities for organizations (Battleson et al., 2016; Benlian et al., 2018).

Cloud computing has great potentials for enhancing organizational performance. From a technical perspective, users can access infinite computing resources and eliminate an up-front commitment to implementing their own physical data center (Kaufman, 2009). Organizations thus can save costs for data center implementation and achieve the faster speed of operating systems by implementing cloud computing In addition, cloud computing achieves economy of scale by decreasing cost of electricity, operation, network bandwidth, software, and hardware resources as well as providing better utility compared to traditional in-house data centers (Armbrust et al., 2010, Son et al., 2014). From a strategic perspective, cloud computing enhances organizations' operational capabilities and dynamic capabilities (Battleson et al., 2016). The ubiquitous of cloud computing can improve business continuity and facilitate collaboration (McAfee, 2011), leading to an increased level of operating efficiency (Choudhary & Vithayathil, 2013). In addition, the adaptability and flexibility of cloud computing can improve organizations' abilities to reconfigure their resources in response to a turbulence environment. Thereby, organizations are increasingly starting to implement a small size of cloud computing technology and increase computing resources only with the increase in their needs (Armbrust et al., 2010).

10.3 Cloud security risks

We undertake the conceptualization of a socio-technical perspective on understanding IT-related phenomenon. In this way, the IT security risks of cloud computing originate from both the technical and social aspects. From the technical perspective, we analyze the security vulnerabilities due to the unique characteristics of cloud computing. Given that cloud computing is used by social actors, we analyze additional security risks caused by user behaviors.

10.3.1 Technical aspects of security risks

10.3.1.1 Lack of control

Implementing cloud computing indicates that the organization outsources some extent of computing resources, operating systems, data, or applications to the cloud vendors, depending on their service models and deployment models. In this way, cloud computing customers give up some level of control over IT resources. Compared to the on-premise IT infrastructure where organizations retain full control over the whole stake of computing resources, cloud vendors and customers jointly design, develop, deploy, and manage IT systems running on a cloud platform. Cloud computing offers access to resources (e.g., data), which in turn creates the challenge of ensuring that only authorized users gain access. In a cloud environment, customers depend on external cloud vendors to make decisions about their internal data and the cloud platform. Beyond the technical notions of the risks of experiencing security incidents on the cloud, the outsourcing relationship in a cloud usage contract increases the risks of cloud vendors' abuse of customers' data in all cases (Khan & Malluhi, 2010).

10.3.1.2 Shared security responsibilities

The partition of computing resources control infers that the security responsibilities are shared among major players in a cloud environment. The responsibility allocation is typically determined by the level of service offered, mostly depending on the service model of cloud computing (Takabi et al., 2010). For instance, IaaS providers are responsible for the account management controls for the privileged users of the initial system, while the responsibilities of securing application deployment on the IaaS are often assigned to the cloud users. IaaS offers a very limited number of application-like features. Thus, cloud users are expected to be responsible for securing operating systems, applications, and context. At the same time, the IaaS provider needs to offer a basic level of data protection on the infrastructure. In a PaaS model, cloud customers' primary responsibilities are securing the applications they developed running on the cloud platform. In contrast, cloud vendors should make efforts to isolate users' applications and workspaces. SaaS providers are supposed to take more responsibilities to safeguard the whole cloud stack, including infrastructure, platform, and applications, which is more prevalent for public clouds than private clouds that provide customers with more extensibility to make customizations (Liu et al., 2012).

10.3.1.3 Independent security risks

Cloud computing uses a shared infrastructure to pool resources, which introduces data isolation and privacy concerns and triggers the security risks originated from external users that affiliate to the same cloud stack (Liu et al., 2012). Cloud applications also integrate with other resources such as services, databases, and applications through standard Application Programming Interfaces (APIs). The vertical integration of infrastructure and applications also increases the interconnectedness of cloud customers and the transmission of data. While the shared resources model help achieves the economy of scale, the sharing of inter-organizational resources increases the security externalities, which refers to the increased security risks caused by the organization itself as well as other users of software running on the same or interconnected network (August et al., 2014). Cloud computing services optimize the automation and resource scalability, which can increase the speed and scale of security incidents. Users can use cloud administrator consoles to provision, configure, manage, and delete service on a large scale. However, each virtual machine has its unique privileges and authorized accounts that require proper onboarding and management. The fast-charging and high-automation dramatically increase security vulnerabilities.

10.3.1.4 Low visibility

Cloud computing is convenient for users to subscribe to SaaS applications and sometimes doesn’t retain sufficient information of user identification. Thus, there is a high risk of unauthorized access in a cloud environment, and cloud providers and users need to establish a stronger authentication and authorization process. In addition, IT applications and tools developed in an on-premise environment or a specific type of cloud usually are incompatible with other clouds. This incompatibility problem can be translated into the lack of visibility and controls that raise security risks such as misconfigurations, data incidents, compliance issues, and excessive privileged access (Kaufman, 2010).

10.3.2 Social aspects of cloud security risks

10.3.2.1 Service-level agreements and contract breach

Cloud computing offers on-demand services and pay-per-use pricing model, requiring a well-developed service-level agreement. As a part of the service contract between cloud users and providers, the service-level agreement formally defines the scope and level of services, as well as the security responsibilities. Although the service-level agreement is vital to the contract negotiation and enforcement between cloud service customers and providers, security remains as a major challenge due to its unique attributes such as non-quantitative and hard to bargain (Takabi et al., 2010). In addition, even though some cloud service providers have adequately considered the security issues in their service-level agreements, how much consumers trust their security measurements as well as how well consumers compliant to the security agreements are uncertain. Hence, security risks also arise if consumers do not have stronger security compliance behaviors. Furthermore, the non-compliance concern not only occurs at the consumer side but also applies to cloud service providers, leading to the well-known risks of psychological contract breaches. In other words, given the established service-level agreements, how well cloud service providers and customers perform "good" security behaviors as specified in the agreement also determines the risks of cloud computing (Kalaiprasath et al., 2017).

10.3.2.2 User unsecure behavior

In addition to the security risks caused by technical vulnerabilities, organizations in a cloud environment face the potential loss due to internal employees with and without malicious intent. The usage of cloud computing can increase the risks of malicious and inadvertent security behavior due to its unique characteristics. For example, cloud computing increases the channels of data and information sharing. The ubiquitous feature of cloud computing enables users to access to the data and information on the cloud from anywhere with the Internet, imposing the risks of invalid access. In addition, cloud narrative IT applications often use account and authentication approaches to access the data, which increases the risks of security risks if employees forget to log out of the system. Therefore, organizations should govern employees' security policy compliance to reduce the internal security risks (Reddy & Reddy, 2011).

10.4 Countermeasures of cloud security risks

Based on the identified cloud security risks due to the unique characteristics of cloud computing, we summarize several countermeasures that organizations can take to mitigate safeguard cloud computing usage. We identified three major categories of organizational cloud security management strategies: (1) organizational awareness, policy, and training; (2) technical control; and (3) physical security management. Below we elaborate on each of these strategies and provide a comprehensive list of actionable strategies.

10.4.1 Organizational awareness, policy, and training

Organizations need to have a comprehensive and holistic cloud security program to account for data ownership, security responsibility accountability, both internally and externally, clearly define cloud security policies and compliance behaviors, and identify effective controls.[1] Table 10.2 summarizes the cloud security policies commonly used along with exemplar items that need to be specified in the policy.

[1] https://www.beyondtrust.com/resources/glossary/cloud-security-cloud-computing-security

Table 10.2: Organizational Cloud Security Policies.

Policy	Objective	Example
ICT Acceptable Use	How organizational computers and other ICT devices should be accessed	– Installation of applications – Using company supplied software – Restriction to loading unauthorized software – Handling of sensitive information – Limit to forwarding emails with confidential information to external parties – Communication in connection with company business – Personal use – Storing limited amounts of personal data – Restricting to overuse of services for non-business-related communication
Password	How often passwords need to be updated and the complexity of password to enhance authentication checks	– Password management best practices – Password update frequency – Password strength (length, age, and complexity) – No shared passwords – Two-factor authentication
User Access Management	Principle of least privilege – i.e., users should be granted the minimum amount of access required to do their routine jobs	– How to allocate/restrict access for users – Discontinuity of access when an employee turnover – Individual actions should be traceable – Limit membership for the privileged accounts and shared accounts
Bring Your Own Device (BYOD)	How to control the use of employees' own devices in the workplace	– Use of external devices – Access to internal network using external devices – Access to the cloud services using employees' devices

In addition to security policies, organizations should provide sufficient training and create a cloud security awareness program to increase employees' security compliance behavior. Practical security training can help prevent social engineering attacks such as phishing emails that are not well protected through technical measures. When safeguarding organizations' cloud computing resources, performing employee background checks is also necessary to prevent security threats from malicious internal users (Kalaiprasath et al., 2017; Hutchings et al., 2013).

Furthermore, there are several notable governmental and industrial regulations that are related to the security aspects of cloud computing. For example, the Health Insurance Portability and Accountability Act (HIPPA) Privacy and Security Rules were established to improve health care efficiency and patients care outcomes by implementing the health data and information systems. HIPAA compliance requires hospitals and healthcare institutions in the United States to mandate the national security standard about the privacy of personal health information. The General Data Protection Regulation (GDPR) is the personal data protection regulation enacted by the European Union (EU). GDPR applies to all companies that collect and process customers' data on EU residents regardless of companies' nationalities. Payment Card Industry Data Security Standards (PCI-DSS) includes a set of security standards focusing on securing sensitive cardholder data. Different from GDPR and HIPAA that are government regulations, PCI-DSS compliance requirements are mandated by the major credit card companies and enforced by the Payment Card Industry Security Standards Council (PCI-SSC). The California Consumer Privacy Act (CCPA) emphasizes on consumer privacy rights and regulates consumers' individual data such as their Internet activities, cookies, IP addresses, and biometric information. Given the growing popularity of the Internet of Things (IoT) devices, CCPA will be relevant to IoT afforded "household data". Gramma-Leach-Bliley Act (GLBA) requires financial institutions to disclose the plans concerning how they share and protect their customers' personal information.

Organizations also should be aware of the importance of service level agreements and adequately address security and data control. When negotiating the cloud computing contract through service level agreements, cloud computing providers and customers should have a mutual and clear understanding of the allocation of security responsibilities. More importantly, both providers and customers in the cloud environment should have appropriate incentive mechanisms to ensure that all parties complain about the agreements. In other words, security responsibilities need to be clearly specified in the service level agreements and well-executed by both parties in the cloud environment on a continuous timespan (Hussain et al., 2017).

10.4.2 Technical control

Like many other IT contexts, cloud security risks can be mitigated by establishing effective resource control. Cloud security control can be categorized into four major types: preventive, deterrent, detective, and corrective control mechanisms (Krutz et al., 2010; Paul & Aithal, 2019). *Preventive control* aims at reducing the external attacks on the cloud. Preventive control systems focus on avoiding the cloud security problem but do not emphasize on the actual elimination of cloud security vulnerabilities. Preventive cloud security control mechanisms can prevent unauthorized access and avoid disrupting cloud privacy. Cloud users, therefore, can be correctly identified. *Deterrent control* will display a warning sign when detecting a scheduled attack on the cloud system, which can deter security risks by informing the authorized users about the potential violation. Also, the waring concerning adverse consequences if the attack is further exploited will be provided. *Detective control* helps detect the occurred security attacks. With a detective control, users will be informed to perform corrective behaviors to address the risks. Detective control also includes prevention arrangements and intrusion detection systems, which support the communication infrastructure so that to detect the attacks. *Corrective control* typically mitigates the negative impacts of a security accident by putting a halt on the damage. Corrective control will use the backup server and files and rebuild a recovery system to ensure the system operations. With these different types of controls in place, cloud security risks and incidents should be addressed quickly.

Both cloud users and cloud service providers can adopt a number of technical prevention measures to enhance different types of control mechanisms. Some examples of technical cloud security measures include:

- Operating system patching
- Preventing vulnerabilities and malware using Internet browsers and software applications
- Anti-virus and malware tools
- Firewalls to prevent unauthorized access
- Multifactor authentication to enhance authentication checks
- Encryption: the data transmission between cloud and browsers and data stored in the cloud need to be encrypted.[2] As an important part of cloud security, the cloud security providers offer encryption as a service in which the data will be encrypted and stored in the cloud after the encryption.
- Intrusion detection and prevention systems
- Network traffic monitoring
- Network segmentation: given the multi-tenant characteristic of cloud computing, cloud providers and users should be cautious about the partition of valid access to the networks among organizations using the same shared cloud platform. Cloud computing users can use a “zone” approach to make their instances, applications, containers, and systems isolated from others as much as possible.[3]
- Vulnerability scans, security audits, and patching known vulnerabilities
- Disaster recovery: be prepared to the backup, retention, and recovery of data

[2] https://www.mcafee.com/enterprise/en-us/security-awareness/cloud/how-does-cloud-encryption-work.html

[3] https://www.beyondtrust.com/resources/glossary/cloud-security-cloud-computing-security

- Cyber threat and security attacks analytics: with the development of data analytics techniques, organizations can leverage the analytics tools to better understand and prevent security threats

10.4.3 Physical security management

Physical security refers to cloud computing providers' offering of a secure data storage that only authorized users can access so that to prevent physical infrastructure attacks and the abuse of access by malicious insiders. Physical security control also can play an important role in deterring security risks caused by employees' careless (Wells et al., 2014; Hutchings et al., 2013). Below is a list of examples for physical security that are relevant in the cloud security context.

- Shielded server rooms and cages, which prevents interference through electromagnetic radiation, external scanning, and eavesdropping
- Facility access logs
- Surveillance, e.g., security guards and CCTV
- Perimeter security, including gates, fences, and bunkers, etc.
- Access control, such as biometric authentication, identity cards, swipe cards, and turnstiles
- Fire management
- Backup power systems

Relating to the organizational security culture, technical control, and physical security management in the cloud environment, DevSecOps methodology has attracted growing public interest. DevSecOps refers to the concept of positioning and prioritizing software security in the software development life cycle. DevSecOps combines cultural philosophies, practices, and tools that bring operation and development together with security functions. Traditionally, developers are lack of compromising security operations when developing software. Developers write and deploy code with limited consideration of security issues. This approach has its limitations in terms of software security, especially in cloud computing environments, since the security testing such as potential errors and vulnerabilities is only available after the software is developed. If a security error was detected, the deployed systems need to be withdrawn. Unlike the traditional software development life cycle, DevSecOps addresses these problems by bringing security as requirements into all stages of the software development process, including the design, the code, and the deployment stage. DevSecOps requires developers to consider security concerns in the code development stage. Also, a formal security test is necessary before it is delivered and deployed. Cloud computing makes the implementation of DevSecOps easier. Through virtualizations and on-demand availability, cloud allows the software to be built and tested by removing the need for physical machine tests, which usually takes a long time and has higher costs.

10.5 Toward an integrative cloud security management framework

In this section, we blend all together to derive an integrative framework aiming at navigating organizational cloud security management. The framework, as shown in Figure 10.2, focuses on how cloud computing's unique characteristics that are manifested by different service and deployment models cause inherent security risks, which may lead to security failures originated from different sources. Accordingly, organizations can place security countermeasures to weaken the occurrence of security failures caused by cloud security risk factors. As displayed in the box of cloud security countermeasures, organizations should place a multi-level security program involving organizational awareness, policy, and training, technical control, and physical security management. The different security strategies should be fused and aligned together to solve the identified cloud security problems. The intensity of investing in cloud security countermeasures depends on the level of cloud security risks and anticipated loss from security failures, which are signified by the level of cloud security characteristics. Furthermore, organizational cloud security management is not a one-time static decision. If an organization experiences a security failure, they need responsive strategies by revisiting their cloud security countermeasures.

10.6 Conclusion

With the tremendous potentials of cloud computing's distributed computation framework, organizations can achieve significant benefits from using cloud computing. In this data-rich digital age, organizations are pursuing business value from data using more sophisticated algorithms such as artificial intelligence. Cloud computing continues to offer promises for the growing demand for big data and analytics. Major artificial intelligence providers are migrating their products and services to the cloud environment. Organizations in other sectors are also adopting cloud computing to facilitate improved agility and operational efficiencies. However, security plays a significant role in deterring the full actualization of cloud computing's potentials.

An efficiency cloud security management would require organizations to understand better how cloud computing is different from security risks in the traditional

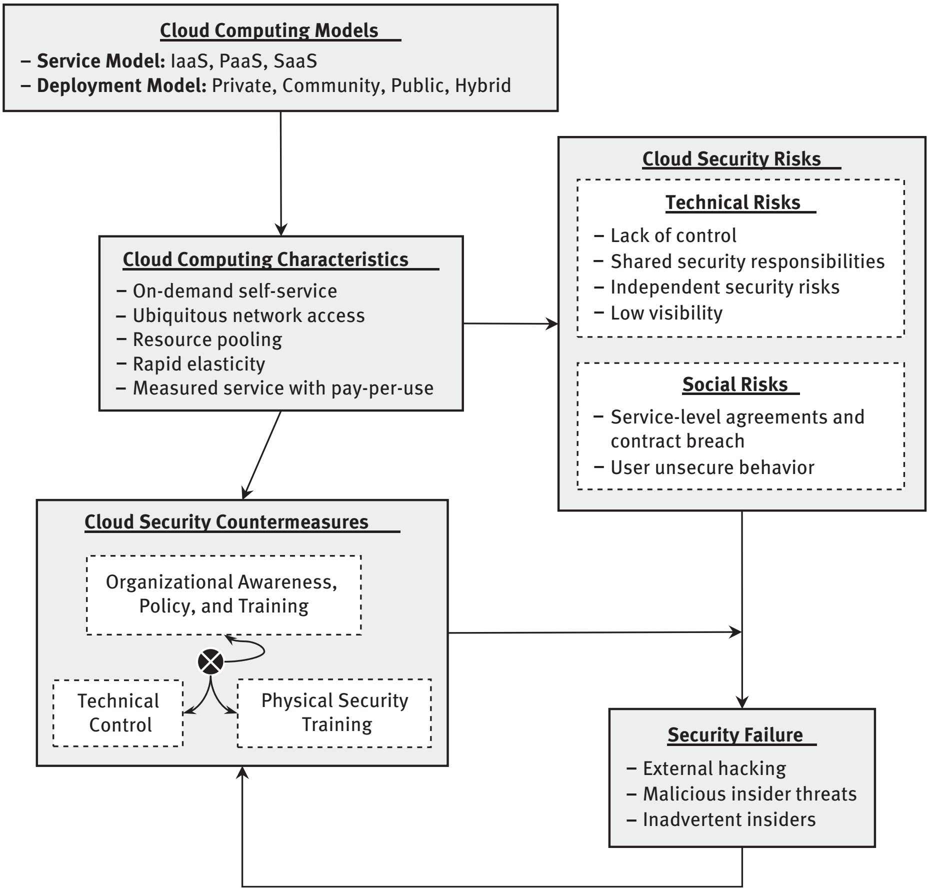

Cloud Computing Models
– Service Model: IaaS, PaaS, SaaS
– Deployment Model: Private, Community, Public, Hybrid
Cloud Computing Characteristics
– On-demand self-service
– Ubiquitous network access
– Resource pooling
– Rapid elasticity
– Measured service with pay-per-use
Cloud Security Risks
Technical Risks
– Lack of control
– Shared security responsibilities
– Independent security risks
– Low visibility
Social Risks
– Service-level agreements and contract breach
– User unsecure behavior
Cloud Security Countermeasures
Organizational Awareness, Policy, and Training
Technical Control
Physical Security Training
Security Failure
– External hacking
– Malicious insider threats
– Inadvertent insiders

Figure 10.2: An Integrative Framework of Organizational Cloud Security Management.

on-premises computing environment. In this chapter, we focus on identifying the unique cloud security risks based on a comprehensive synthesis of the cloud's unique characteristics. We also possess that cloud characteristics vary at the levels of prevalence due to the different deployment and service models organizations adopted. Based on the identified unique characteristics, we summarized the major cloud security risks from both technical and social perspectives. Accordingly, we identify several organizational cloud security management strategies. This logic and all relevant factors are finally organized into an integrative framework of organizational cloud security management. We hope that the integrative framework can help organizations effectively navigate their cloud security management.

References

Ali, M., Khan, S. U., & Vasilakos, A. V. (2015). Security in Cloud Computing: Opportunities and Challenges. *Information Sciences*, 305, 357–383.

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. & Zaharia, M. (2010). A View of Cloud Computing. *Communications of the ACM*, 53(4), 50–58.

August, T., Niculescu, M. F., and Shin, H. (2014). Cloud Implications on Software Network Structure and Security Risks. *Information Systems Research* 25(3),489–510.

Battleson, D. A., West, B. C., Kim, J., Ramesh, B., & Robinson, P. S. (2016). Achieving Dynamic Capabilities with Cloud Computing: An Empirical Investigation. *European Journal of Information Systems*, 25(3), 209–230.

Benlian, A., Kettinger, W. J., Sunyaev, A., & Winkler, T. J. (2018). The Transformative Value of Cloud Computing: A Decoupling, Platformization, and Recombination Theoretical Framework. *Journal of Management Information Systems*, 35(3), 719–739.

Boss, G., Malladi, P., Quan, S., Legregni, L., & Hall, H. (2007). Cloud Computing. Technical Report. IBM High Performance on Demand Solutions, 10–08.

Chou, T. S. (2013). Security Threats on Cloud Computing Vulnerabilities. *International Journal of Computer Science & Information Technology*, *5*(3), 79.

Choudhary, V., & Vithayathil, J. (2013). The Impact of Cloud Computing: Should the IT Department be Organized as a Cost Center or a Profit Center?, *Journal of Management Information Systems*, 30(2), 67–100.

Cubitt, S., Hassan, R., & Volkmer, I. (2011). Does Cloud Computing Have a Silver Lining?. *Media, Culture & Society*, 33(1), 149–158.

Garg, P., Goel, S., & Garg, S. (2017). Investigation of Cloud Computing Security Issue. *International Journal of Advanced Research in Computer Science*, 8(5), 2117–2120.

Hussain, S. A., Fatima, M., Saeed, A., Raza, I., & Shahzad, R. K. (2017). Multilevel Classification of Security Concerns in Cloud Computing. *Applied Computing and Informatics*, 13(1), 57–65.

Hutchings, A., Smith, R. G., & James, L. (2013). Cloud computing for small business: Criminal and security threats and prevention measures. *Trends and issues in Crime and Criminal Justice*, (456), 1.

Iyer, B., & Henderson, J. C. (2010). Preparing for the Future: Understanding the Seven Capabilities Cloud Computing. *MISQ Executive*, *9*(2), 117–131.

Kaufman, L. M. (2009). Data Security in the World of Cloud Computing. *IEEE Security & Privacy*, 7(4), 61–64.

Kaufman, L. M. (2010). Can Public-Cloud Security Meet its Unique Challenges?. *IEEE Security & Privacy*, 8(4), 55–57.

Kalaiprasath, R., Elankavi, R., & Udayakumar, D. R. (2017). Cloud. Security and Compliance-a Semantic Approach in End to End Security. *International Journal of Mechanical Engineering and Technology*, 8(5), 987–994.

Kavis, M. J. (2014). Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS). John Wiley & Sons.

Khan, K. M., & Malluhi, Q. (2010). Establishing trust in cloud computing. *IT professional*, 12(5), 20–27.

Krutz, Ronald L., & Russell Dean Vines (2010). Cloud Security: A Comprehensive Guide to Secure Cloud Computing. Indianapolis, IN: Wiley, 2010. 179–80.

Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., & Leaf, D. (2012). NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology. (Special Publication 500–292).

McAfee, A. (2011). What Every CEO Needs to Know about the Cloud. *Harvard Business Review*, 89(11), 124–132.

Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing.

Naldi, M., & Mastroeni, L. (2016). Economic Decision Criteria for the Migration to Cloud Storage. *European Journal of Information Systems*, 25(1), 16–28.

Paul, P., & Aithal, P. S. (2019). Cloud Security: An Overview and Current Trend. *International Journal of Applied Engineering and Management Letters*, 3(2), 53–58.

Pritchard, S. J. (2012). Cloud Guarantees Bright Outlook. The Guardian, 13.

Reddy, V. K., & Reddy, L. S. S. (2011). Security Architecture of Cloud Computing. *International Journal of Engineering Science and Technology*, 3(9), 7149–7155.

Ryoo, J., Rizvi, S., Aiken, W., & Kissell, J. (2013). Cloud security auditing: challenges and emerging approaches. *IEEE Security & Privacy*, 12(6), 68–74.

Son, I., Lee, D., Lee, J. N., & Chang, Y. B. (2014). Market Perception on Cloud Computing Initiatives in Organizations: An Extended Resource-Based View. *Information & Management*, 51(6), 653–669.

Takabi, H., Joshi, J. B. D., & Ahn, G.-J. (2010). Security and Privacy Challenges in Cloud Computing Environments. *IEEE Security & Privacy*, 8(6),24–31.

Venter, W., & Whitley, E. A. (2012). A Critical Review of Cloud Computing: Researching Desires and Realities. *Journal of Information Technology*, 27(3),179–197.

Wells, L. J., Camelio, J. A., Williams, C. B., & White, J. (2014). Cyber-Physical Security Challenges in Manufacturing Systems. *Manufacturing Letters*, 2(2), 74–77.

Béatrix Barafort, Antonia Mas and Antoni-Lluís Mesquida

11 A process framework for integrated risk management in IT

11.1 Introduction

This chapter has foundations resulting from many years of applied research in the domain of process assessment and improvement with a framework branded TIPA® (TIPA is a registered trademark). It has been designed and populated with various artefacts with the following initial raised subject of research: "*Does the combined use of ITIL and IEC/ISO 15504 truly increases effectiveness and efficiency and can be adapted to the need of flexibility of today's organizations?*" (Barafort et al., 2002). In order to address this research question, a Process Reference Model (PRM) and a Process Assessment Model (PAM) based on ITIL (The Cabinet Office, 2011) was developed, fulfilling ISO/IEC 15504 (International Organization for Standardization, 2003b) (since this publication, the ISO/IEC 15504 series has been reviewed and is from now on the ISO/IEC 330xx series (International Organization for Standardization, 2015d)) requirements for designing process models enabling process assessments. A Design Science approach had been followed with various iterations (Barafort & Rousseau, 2009). It was in a context of research-action in the Service Science, with a multidisciplinary approach. For designing process models (PRMs and PAMs), a Goal-Oriented Requirements Engineering (GORE) technique was used. Back in 2008, the systematic way used to design PRMs and PAMs (Transformation process) was documented, with an illustration on the ISO/IEC 20000–1 PRM/PAM design (Barafort et al., 2008). Joint works in the domain of Project management with the design of the Project Management SPICE PRM and PAM (Mesquida et al., 2015) were also initiated.

For performing process assessment, a documented process assessment process is required. Then beyond the process models, the TIPA Framework includes a process assessment method. The TIPA method was formalized in a published handbook (Barafort et al., 2009). Extensive works have been performed in the context of the TIPA Framework (Barafort et al., 2014). Over time, the TIPA Framework has been applied to various domains and is composed of a set of artefacts, with customization to the targeted domain when necessary.

The TIPA Framework has been expanded and additional components were planned. A Software-as-a-Service tool development has been initiated in order to support the TIPA Framework (to embed process models and support the TIPA method and toolbox, to gain time and reduce costs, to improve assessability, effectiveness, and efficiency of process assessments) (Barafort, Shrestha, et al., 2018). From a TIPA factory perspective, additional process models were developed (whether developed

https://doi.org/10.1515/9783110652321-011

by LIST or by others such as in ISO) and populate the process model library; more connections and interoperability between models can be operated.

In this overall TIPA Framework context, an Integrated Risk Management process model was foreseen. It can be the opportunity to provide an integrated process view from a governance, risk management, and compliance (GRC) perspective for quality, project, IT services, and information security management aspects.

GRC activities are key challenges in organizations. With the era of digitalization, the governance of digital transformations is a critical topic, with many instruments and ways of maintaining operations with an adequate organization and in a growing regulation landscape. IT is pervasive and essential for any business. Risk management is addressing these challenges and is related to a multitude of domains, for IT and non-IT concerns. In IT settings, many activities are strongly related to risk management: project management, information security, and ITSM to quote the main domains. Risk is defined in (International Organization for Standardization, 2009) as "*effect of uncertainty on objectives*" and a Note to this definition mentions that "*Objectives can have different aspects (such as financial, health and safety, and environmental goals) and can apply at different levels (such as strategic, organization-wide, project, product and process)*". Because of its indispensable nature, risk management has also become vital and must be under control. It can be for dedicated risk management purposes or from a broader perspective in management systems that can address a single discipline or several disciplines, as defined by ISO (International Organization for Standardization, 2018c).

Depending on their strategic goals, competitive advantage on the market, regulation, and compliance constraints, IT companies or IT departments may need to be certified regarding management system standards (MSSs) such as the ISO/IEC 27001 (International Organization for Standardization, 2013b) for information security or the ISO/IEC 20000–1 (International Organization for Standardization, 2018a) for ITSM. They may also need to integrate these IT-related standards with more general ones such as the ISO 9001 (International Organization for Standardization, 2015e) for a quality management system (QMS).

Process performance is one of many ways of governance, with process improvement to enhance practices. Capability and Maturity Models (C&MM) support process improvement with process assessment facilities. They provide a guide and a structure for a process improvement roadmap. There are a plethora of process models for various business domains and sectors. At the ISO and on the market, there are several published Process Reference Models (PRM) and Process Assessment Models (PAM) in different kinds of domains (*Automotive SPICE*, 2016; *TIPA for ITIL*, 2015; International Organization for Standardization, 2012b; Lepmets et al., 2016); these various initiatives are based on the ISO Process assessment standard series concepts (International Organization for Standardization, 2003b, 2015a); they rely on a very structured and systematic approach for process assessment and guided process improvement.

International standards represent an international consensus, provide open access to structured technical domains as well as voluntary positioning towards certifications, and contribute to companies' benefits (Association Française de Normalisation, 2016). The ISO continuously promotes standardization benefits (International Organization for Standardization, 2017a) and MSSs (International Organization for Standardization, 2017c). Every year, ISO performs a survey (International Organization for Standardization, 2018e) of certifications to MSSs. The 2017 results show again that ISO 9001 is the leader of management system certification standards. This survey also indicates an increase in the certifications related to ISO/IEC 27001. In 2015, ISO added a "new" management system standard: ISO/IEC 20000–1 (Service management system requirements). ISO/IEC 20000–1 is of interest for its alignment in intent and structure as a management system, for being closely related to ITIL processes, and a relative impact on the market (Cots & Casadesús, 2015). Regarding Project management, we can quote that ISO 21500 (Guidance on Project management (International Organization for Standardization, 2012c)) provides a globally accepted guideline in Project management. So in intent and with a process-based approach, ISO 21500, ISO/IEC 27001 and ISO/IEC 20000–1 are closely related to the famous ISO 9001 standard for QMSs. These four ISO standards are of high interest for many practitioners in IT settings, interested in the integration of process-based activities, implementing mechanisms for making the link between IT and non-IT entities of their organization with risk management challenges to address. By IT settings, the authors mean IT companies and IT departments, covering both software development and operations sides, with projects and non-projects based activities. These standards are significant for many companies and were reported back to the authors by practitioners. Addressing risk management in some market-significant ISO standards from integration and improvement perspectives in IT organizations appeared as key challenges. The authors assumed that an integrated risk management approach for IT Organizations will benefit organizations by being based on ISO standards which represent an international consensus. This assumption is supported by market demand for ISO 9001, ISO/IEC 27001 and ISO 20000–1 as popular standards for certification of management systems, completed by ISO 21500 because project management is always a critical process in IT organizations. As ISO 31000 (International Organization for Standardization, 2018d) is the ISO international reference for Risk management, this standard is providing an Ariane's thread for these research works, as explained in the related work (see section 11.2). These standards are the ground material of the research.

In this context, the overall objective of this research is to propose means to improve risk management processes in IT organizations, with a structured, integrated, interoperable, assessable, effective, and efficient way, based on ISO standards.

Thus the main research question is: how to improve risk management processes in IT settings from an integrated and management system perspective in multiple ISO standards?

Several sub-questions (RQ) are investigated with their respective research objectives (RO):

- RQ1: How to identify risk management activities throughout various selected ISO standards targeting management systems?
  - RO1: to investigate and compare risk management activities throughout selected ISO standards targeting management systems
- RQ2: How to drive integration for risk management activities in IT settings?
  - RO2: to show that a centralized and management system approach based on processes contributes to integration in a process-centric risk management mindset in IT settings.
- RQ3: How to improve risk management processes?
  - RO3: to propose means to improve Risk management processes in IT settings, with a structured, integrated, interoperable, assessable, effective, and efficient way (these criteria guide our applied research).

The intended outcomes of this research are represented by two main artefacts according to a Design Science approach which supports the research approach: Integrated Risk Management in IT Settings (IRMIS) Process Reference Model (PRM) & Process Assessment Model (PAM).

The second section of this chapter presents related work, then the research contribution in section 11.3, a case study illustrating the use of the process assessment model in section 11.4 and finally the discussion and conclusion with future works.

11.2 Related work

Various facets of integrated risk management, management systems, significant ISO standards in IT settings, have been investigated in the next three paragraphs in order to address research questions.

11.2.1 Integration perspectives

Integrating management systems has been a topic of interest in research and industry for many years now (Casadesús et al., 2011; Simon et al., 2012). This has been particularly true for quality management, environmental management, and health and safety domains (International Organization for Standardization, 2018e). It has been more and more necessary to integrate these systems for cost reductions, efficiency, effectiveness, and market positioning. The integration of management systems, in particular from the ISO 9001 perspective, has been considered in many works. In the IT domain, with the first publication in 2005 of the ISO/IEC 20000–1 and ISO/IEC 27001, new MSSs appeared on the international scene, respectively for ITSM and Information Security. As MSSs interest increased, ISO published in its Directives an annex named "*High-level structure (HLS), identical core text, common terms and core definitions*" for MSS (International Organization for Standardization, 2018c). The goal was to standardize the core content of management systems and to impose the adoption of this structure to all management systems to the rhythm of their respective revision. The ISO/IEC 27001 standard is from now on aligned with the HLS since its second revision in 2013 (International Organization for Standardization, 2013b). The ISO 9001 has been upgraded in its last revision of 2015 (International Organization for Standardization, 2015e). The ISO/IEC 20000–1 standard is aligned in its revised version published in 2018 (International Organization for Standardization, 2018a).

Among the integrative aspects of management systems, risk management is a particular topic of great importance and interest for organizations. From the ISO standards perspective, the ISO 31000 standard on Risk management (International Organization for Standardization, 2018d) is the main reference, with a holistic view on risk management. Furthermore, in many domains there are dedicated risk management standards: i.e. for Information security, we can quote the ISO/IEC 27005 (Information security risk management) (International Organization for Standardization, 2018b). Several approaches target methodologies for implementing risk management; we can cite (Santos Olmo Parra et al., 2016) for Risk management in ISO/IEC 27001; we can also mention specific risks such as cloud computing ones (Chou, 2015). When related to methodologies, these researches target the "How to", and do not concentrate on the "What" which is addressed by processes and then not being prescriptive when seen from a generic perspective.

Last but not least, IT settings are commonly organized by projects and have to face project risks. From the ISO perspective, the ISO 21500 (International Organization for Standardization, 2012c) standard provides guidance for project management: processes, continual improvement, and risk management are important tackled concerns.

In the context of the problem of integrated management systems, risk management is a critical cornerstone that has not been addressed specifically from the IT organizations' point of view with a management system and process-based perspective. Integrated risk management addresses risks at very different levels in the organization, including strategy and tactics, and covering both opportunity and threat (Hillson, 2006). Diverse frameworks and approaches to support Integrated risk management in IT companies have been developed (Alberts & Dorofee, 2010; Bandyopadhyay et al., 1999; Chittister & Haimes, 1993; Kontio, 2001; Lyytinen et al., 1996; Roy, 2004). In the software domain, improvements are proposed in (Buglione et al., 2016) for the Risk management process of the PAM ISO/IEC 15504–5 (International Organization for Standardization, 2012a). Recently, a development of a maturity model for risk management has been performed (Proença et al., 2017), based on the ISO 31000 standard version of 2009. This maturity model addresses directly the ISO 31000 standard but is creating its own framework; it is not meeting ISO/IEC 330xx requirements for process capability and maturity assessment and does not address this chapter's research.

IT has become crucial in the digital era, and more and more threats are existing. Organizations have to face risks with appropriate approaches depending on their size. Despite the fact there are numerous risk management standards, few of them are integrated and adapted to small and medium sizes enterprises. From the project management perspective, a recent survey on ISO 21500 and PMBOK (Varajão et al., 2017) has shown that quality management and risk management are the last processes to be considered by project managers.

In the IT domain, Software Engineering plays a significant part where risk management is also considered from various perspectives: embedded in project management, included in SPI approaches or part of the software and/or system life cycle. The SPI Manifesto (Pries-Heje & Johansen, 2010) "*gives expression to state-of-the-art knowledge on SPI*" with three values (people, business, change), further elaborated into ten principles including risk management. Risk management must be a part of any SPI project and SPI risks must be managed as in any project. For software and system developments, risk management must be present.

In the years 2000, maturity models, process assessment, and improvement frameworks were very popular, such as CMMI (Software Engineering Institute, 2010) and ISO/IEC 15504–330xx series of standards (International Organization for Standardization, 2003a). From a complementary perspective compared to a management system certification, performance management approaches dealing with process assessment and process improvement raised. In ISO, development works have proposed PRMs and PAMs based on MSSs. It is the case for Information security management (ISO/IEC 33072 (International Organization for Standardization, 2016c)), and ITSM (ISO/IEC 15504–8 (International Organization for Standardization, 2012b)), but also for quality management based on ISO 9001 (ISO/IEC 33073 (International Organization for Standardization, 2017b)). These three domains are of particular interest, as they propose from a generic perspective, a common set of processes addressing the management system mechanisms, as stated in the HLS for management systems.

11.2.2 Overview of targeted ISO standards for comparing risk management

As stated previously, ISO 31000 has been selected for risk management as the international reference in the domain. The other standards were selected because of their relevance and their demand by the market. There are presented in Table 11.1.

Table 11.1: Main characteristics of ISO targeted standards.

Standard name	Main characteristics
ISO 31000:2018 Risk management – Principles and guidelines (International Organization for Standardization, 2018d)	Principles and generic guidelines on risk management. It is not for certification (does not provide requirements). It can be used whether for IT or non-IT applications, in public, private, associations or group, for any type of risk (Not specific to any industry or sector).
ISO Directives Part 1 Annex L:2018 – High-level structure for management system standards (International Organization for Standardization, 2018c)	Generic requirements for management systems to ensure consistency among various MSS and enable easier integration whatever the domain (information security, service management, quality, etc . . .). Reducing costs and providing the transversal approach via processes: fulfilled by integrated and interoperable management systems.
ISO 9001:2015 Quality management systems – Requirements (International Organization for Standardization, 2015e)	2015 version of ISO 9001 aligned with the changes that organizations have to face, focusing more on performance, combining the process approach with risk-based thinking and activating the Plan-Do-Check-Act cycle at all levels of the organization.
ISO 21500:2012 Guidance on project management (International Organization for Standardization, 2012c)	Guidance for project management. It can be used by any type of organization, for any type of project, irrespective of complexity, size, or duration. It provides a high-level description of concepts and processes that are considered to form good practice in project management. Identifies the recommended project management processes to be used during a project.
ISO 20000–1: 2018 IT Service Management – Service management system requirements (International Organization for Standardization, 2018a)	Service management system (SMS) standard. Specifies requirements for the service provider to plan, establish, implement, operate, monitor, review, maintain and improve an SMS (requirements including the design, transition, delivery, and improvement of services to fulfill agreed service requirements).

Table 11.1 (continued)

Standard name	Main characteristics
ISO 27001:2013 Information security management (International Organization for Standardization, 2013b)	Information security management system (ISMS): a systematic approach to managing sensitive company information so that it remains secure. It can be applied to small, medium, and large businesses in any sector. Includes people, processes, and IT systems by applying a risk management process and is aligned with the HLS.

11.2.3 Study of relevant standards with Risk management process(es) in PAMs

The authors have studied existing and available PRMs & PAMs related to Risk management in the C&MM context, based on publicly available ISO/IEC 15504/330xx. Table 11.2 lists these Risk management processes and their source.

Table 11.2: List of Risk management processes in existing Process models fulfilling ISO/IEC 15504–330xx requirements for PRM & PAM.

Process model	Name of the Risk management related process(es)
ISO/IEC 15504–5:2012 – Part 5: An exemplar software life cycle process assessment model (International Organization for Standardization, 2012a)	MAN.5 Risk management
ISO/IEC 15504–6:2013 – Part 6: An exemplar system life cycle process assessment model (International Organization for Standardization, 2013a)	PRJ.5 Risk management
ISO/IEC 15504–8:2012 – Part 8: An exemplar process assessment model for IT service management (International Organization for Standardization, 2012b)	SMS.6 Risk management
Enterprise SPICE (ISO/IEC 33071:2016 – An integrated process capability assessment model for Enterprise processes) (International Organization for Standardization, 2016b)	GVM.9 Risk management
ISO/IEC 33072:2016 – Process capability assessment model for information security management (International Organization for Standardization, 2016c)	COM.11 Risk and opportunity management

Table 11.2 (continued)

Process model	Name of the Risk management related process(es)
ISO/IEC 33073:2017 – Process capability assessment model for quality management (International Organization for Standardization, 2017b)	COM.11 Risk management
ISO/IEC 30105–2: 2016 – Information technology – IT Enabled Services-Business Process Outsourcing (ITES-BPO) lifecycle processes – Part 2: Process assessment model (PAM) (International Organization for Standardization, 2016a)	ENB1 Risk management
Automotive SPICE Process Assessment Model (*Automotive SPICE*, 2016)	MAN.5 Risk management
COBIT Process Assessment Model (PAM): Using COBIT 5 (ISACA, 2013)	EDM03 Ensure risk optimisation APO12 Manage risk

According to these processes, the risk management process, as addressed by the ISO 31000 standard, is very general. There is little difference among these processes, where risk identification is performed, and then analysis and evaluation, from the risk assessment perspective, and finally risk treatment. There is not much detail in each of these PAM.

11.3 Research contributions

In order to design and build process models providing a solution for the research questions, artefacts have been created for an Integrated Risk Management for IT Settings (IRMIS) PRM & PAM. A Design Science Research (DSR) method has been followed, as reported in (Peffers et al., 2007). For the Build part of the DSR method, the authors have met the requirements of ISO/IEC 33004 (International Organization for Standardization, 2015b) for designing PRMs and PAMs. They have also used a Transformation Process (Barafort et al., 2008). This process is a systematic approach, based on goal-oriented requirements engineering techniques, for designing PRMs and PAMs. It contains nine steps described in detail in (Barafort et al., 2008).

11.3.1 Contribution related to research sub-question RQ1: “How to identify risk management activities throughout various selected ISO standards targeting management systems?”

In order to compare risk management approaches in the various selected ISO standards previously mentioned, the following systematic method has been followed:

- Step 1: Identification of risk-based activities in all standards (search on the keyword “Risk”).
- Step 2: Mapping of the sections/requirements to some requirement in Clause 5 (Framework) or 6 (Process) of ISO 31000.
- Step 3: Description of relations or connection points among risk-based activities and the related requirements.

Table 11.3 summarizes the results of steps 1 and 2.

Table 11.3: Results from the comparison process.

	Sections/requirements of the Standard addressing “risks”	Sections mapped to some requirement in ISO 31000 clauses 5 or 6
Annex L	1	1
ISO 9001	14	12
ISO 21500	17	17
ISO/IEC 20000–1	12	12
ISO/IEC 27001	9	7

The relations detected during step 3 were analyzed according to the following classification:

- Context of risk management in all standards
  - ISO 31000 recommends that organizations develop, implement and continuously improve a framework (see Figure 11.1) whose purpose is to integrate the process for managing risk into the organization’s overall governance, strategy, and planning, management, reporting processes, policies, values, and culture.
- Leadership and commitment
  - The introduction of risk management and ensuring its ongoing effectiveness requires a strong and sustained commitment by the management of the organization, as well as strategic and rigorous planning to achieve commitment at all levels.

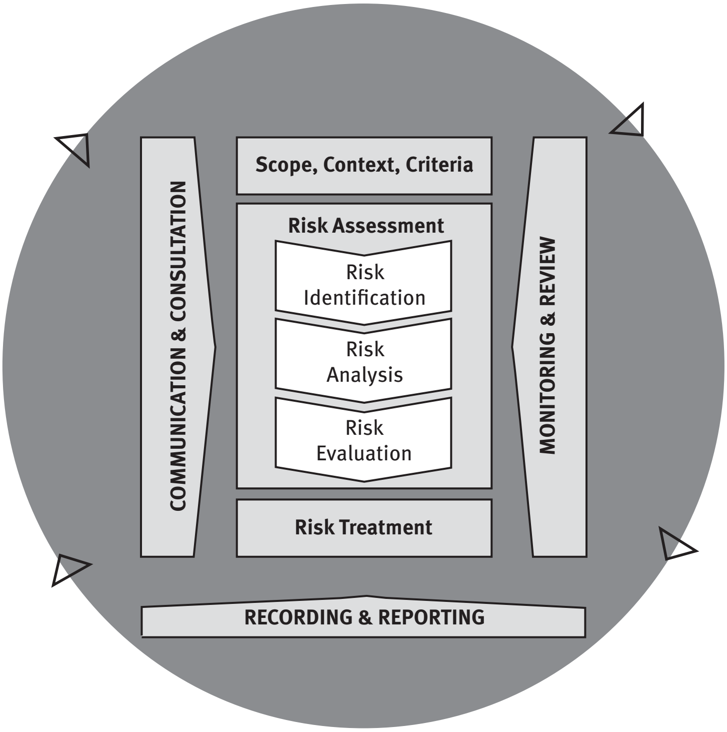


Figure 11.1: ISO 31000 Risk management process (source: ISO 31000).

- Plan-Do-Check-Act (PDCA) cycle
  - The organization’s objectives for, and commitment to, risk management should be clearly stated (Plan).
  - When implementing risk management (Do), an organization should implement the framework for managing risk and should ensure that the risk management process is applied through a risk management plan at all relevant levels and functions of the organization.
  - Communication and consultation with external and internal stakeholders should take place during all stages of the risk management process.
  - By establishing the context, the organization articulates its objectives, defines the external and internal parameters to be taken into account when managing risk, and sets the scope and risk criteria for the remaining process.
  - Risk assessment is the overall process of risk identification, risk analysis, and risk evaluation.
    - In Risk identification, the organization should identify sources of risk, areas of impacts, events, and their causes, and their potential consequences.

  - Risk analysis involves developing an understanding of the risk. Risk analysis provides input to risk evaluation and to decisions on whether risks need to be treated, and on the most appropriate risk treatment strategies and methods.
  - The purpose of risk evaluation is to assist in making decisions, based on the outcomes of risk analysis, about which risks need treatment and the priority for treatment implementation.
  - Risk treatment involves selecting one or more options for modifying risks and implementing those options. Once implemented, treatments provide or modify the controls.
- Both monitoring and review should be a planned part of the risk management process and involve regular checking or surveillance.
- To ensure that risk management is effective the organization should measure risk management performance against indicators, periodically measure progress against the risk management plan and review the effectiveness of the risk management framework, policy, and plan (Check).
- Based on the results of monitoring and reviews, decisions should be made on how the risk management framework, policy, and plan can be improved (Act).

11.3.2 Contribution related to research sub-question RQ2: "How to drive integration for risk management activities in IT settings?"

11.3.2.1 Elementary statements, requirements trees, and process elicitation

The Transformation Process has been used successfully several times and validated in the context of the TIPA Framework (Renault & Barafort, 2014).

In the next paragraphs, we explain how we went through the Transformation process and when needed additional mappings were performed in order to provide full process descriptions based on ISO 31000, and complementary views for ISO 21500 and ISO/IEC 27001 (completed with ISO/IEC 27005) as these standards provide inputs for specific risk management processes. ISO 9001 and ISO/IEC 20000–1 are not long-winded on risk management and are very aligned with Annex L.

Identification of elementary statements from ISO 31000

This step consisted of identifying all of the statements from ISO 31000 under the form of a collection of elementary items. The final list was composed of 281 elementary items made up of a subject, a verb and a complement, without coordination, conjunctions, or enumeration. Table 11.4 shows an example of decomposed elementary requirements. Then from this final list, the "should statements" (main statements) contained in the text of the ISO 31000 standard were easily identified (172 “should” statements). They are the basis for the next steps.

Table 11.4: Example of decomposed elementary statements.

4.3.2 Extract from ISO 31000	Example of decomposedelementary statements
The organization should continually improve the suitability, adequacy, and effectiveness of the risk management framework and the way the risk management process is integrated.	The organization should continually improve the suitability, adequacy, and effectiveness of the risk management framework The organization should continually improve the way the risk management process is integrated.

Organization and structure of the statements

A “mind map” for statement trees organized and structured the elementary “should” statements, completed by “info” statements (74), “may” statements (16), “can” statements (24), “purpose” statements (9), and other statements (7). A graphical view of the elementary items having the same object (or component) was provided. The requirements were then gathered around the objects they were relating to in order to build statement trees. For instance, elementary items targeting “context” aspects were grouped under an “External and internal context” label. This statement tree structuring was inspired by previous works on the Annex L for Management Systems Standards (Barafort, Mesquida, et al., 2018) (Cortina et al., 2014), where some groupings were similar, and by mappings performed on the Risk term in the various selected standards. Therefore, related to the statements establishing the overall framework of risk management, we identified a Statement tree named *Leadership*, which has the following nodes (each node comprising leaves where each leave is an elementary statement): Needs of the organization, Top management and oversight bodies commitment, Accountabilities-responsibilities-authorities, External and internal context, Risk management integration and Scope definition. The other following statement trees were developed: *Communication and reporting*, *Resources*, *Implementing risk management*, *Risk assessment*, *Risk treatment*, and *Monitoring and review*. Finally, with the integration criteria, the Statement trees developed by the authors for the HLS of management systems were superimposed for relevant similar items, guided by terminology and common meanings. For instance in the Leadership tree, the “Leadership and commitment” clause in ISO 31000, represented in a leaf was superimposed with the “Leadership and commitment” clause of the HLS.

Identification and organization of common purposes

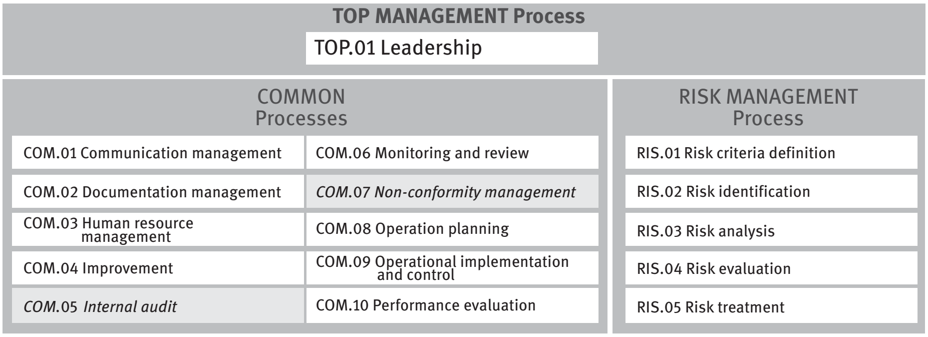
With the identification and organization of common purposes, the first list of elicited processes appeared, for an integrated risk management PRM. In addition to the Transformation Process, which has been followed for previous PRMs and PAMs development, we used low-level objectives resulting from the HLS and superimposed them with those from ISO 31000 to cover the common purposes of all the selected ISO MSSs for an integrated risk management PRM.

Based on Statement trees, supported by the terminological work and by previous works at the ISO for developing PRMs and PAMs based on ISO/IEC 20000–1, ISO/IEC 27001, and ISO 9001, a mapping was performed. It was between the subclauses of ISO 31000, and the process names of MSSs common processes related to the core processes of a management system. We insist here on the fact that the framework for risk management of the ISO 31000 shares the concepts of management systems (without seeking for a certification). This mapping also comprised the processes of ISO 21500. The mapping contributed to the identification of common purposes, which are formulated into Goal trees in the next paragraph and to derive a first list of processes, to be refined.

Considering the Risk Management process viewed from ISO 31000 perspective, the “*Risk and opportunity management*” process proposed by the PRM and PAM for Management Systems is not satisfactory. Indeed, it does not provide the necessary structure and details that we expect for a dedicated Risk Management PRM and PAM. ISO 21500 proposes a subject group dedicated to Risk management, with four processes: Identify risks, Assess risks, Treat risks, and Control risks. These four processes support our idea for having the overall Risk management process split into more detailed ones. In order to strengthen the approach, we used another ISO standard: the ISO/IEC 27005 Information security risk management (International Organization for Standardization, 2018b). This standard is fully aligned with ISO 31000 and provides a more detailed view for the Information security domain. A mapping was performed between the sub-clauses of ISO 31000 and clauses and sub-clauses of ISO/IEC 27005. It confirmed our view for targeting Risk identification, Risk analysis, Risk evaluation, and Risk treatment.

The IRMIS process model is composed of three groups of processes: Top Management, Common processes, and Risk management (see Figure 11.2). This structure with three groups is similar to the one of management systems including top management, and core common processes. Top management and common processes are mainly derived from the ISO/IEC 33073 standard (International Organization for Standardization, 2017b); only two processes are derived from ISO/IEC 33072 (International Organization for Standardization, 2016c) for COM.08 and COM.09 as there were too quality management dedicated; a more generic process description from ISO/IEC 33072 was then chosen. The Risk management group represents the specific processes for risk management, aligned with the overall risk management process proposed by ISO 31000.

Remark: the grey cells with italic texts show two processes which are not at all present in ISO 31000, but necessary in a management system context according to Annex L; we decided to leave them in the PRM and PAM for global integration purposes.



TOP MANAGEMENT Process		
TOP.01 Leadership		
COMMON Processes		RISK MANAGEMENT Process
COM.01 Communication management	COM.06 Monitoring and review	RIS.01 Risk criteria definition
COM.02 Documentation management	*COM.07 Non-conformity management*	RIS.02 Risk identification
COM.03 Human resource management	COM.08 Operation planning	RIS.03 Risk analysis
COM.04 Improvement	COM.09 Operational implementation and control	RIS.04 Risk evaluation
COM.05 Internal audit	COM.10 Performance evaluation	RIS.05 Risk treatment

Figure 11.2: IRMIS PRM proposed list of processes.

11.3.3 Contribution related to research sub-question RQ3: "How to improve risk management processes"

In parallel and in order to confirm the identification of processes, identifying common purposes and organizing them in Goal trees was performed. It was the step preceding the formalization of process descriptions themselves.

Figure 11.3 shows the goal tree for the *Leadership* process, containing six different objectives, resulting into five outcomes identified from the core common process Leadership of Annex L (for our ISO 31000 PRM & PAM design objective, "management system" and "quality management" have been respectively changed by "risk management framework" and "risk management").

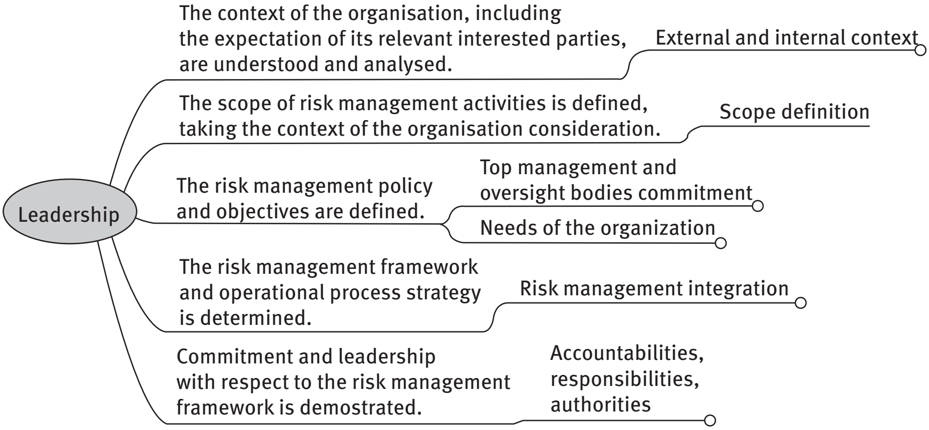

Leadership
The context of the organisation, including the expectation of its relevant interested parties, are understood and analysed.
External and internal context
The scope of risk management activities is defined, taking the context of the organisation consideration.
Scope definition
The risk management policy and objectives are defined.
Top management and oversight bodies commitment
Needs of the organization
The risk management framework and operational process strategy is determined.
Risk management integration
Commitment and leadership with respect to the risk management framework is demostrated.
Accountabilities, responsibilities, authorities

Figure 11.3: Goal tree for the Leadership process.

11.3.3.1 Identification and phrasing of outcomes and purpose

Common purposes were identified by grouping statements. Then it enabled to formulate outcomes according to ISO/IEC 33004 requirements (An outcome is an observable result of 1) *"the production of an artefact"*, 2) "*a significant change of state*", or 3) "*the meeting of specified constraints*".). For instance, for the Leadership process, this step was shortened by mapping the goal tree with the outcome of the core common Leadership process of the MSS (i.e. in ISO/IEC 33073). The process description is then simplified and straightforward as long as the grouping of elementary statements are mapped with outcomes of the MSS-based process. For Risk management specific processes, outcomes were identified and phrased from the grouping of elementary statements as common purposes with fulfilling ISO/IEC 33004 requirements above-mentioned. Then from the phrased outcomes, a purpose for each process has been formulated.

11.3.3.2 Determination of indicators such as base practices and work products

In ISO 31000, sometimes the statements are detailed enough and can be the source of information for phrasing base practices. In the case they are not detailed, practices are directly deduced from the outcomes and represent functional activities of the process, with the adequate phrasing starting with an action verb at the infinitive. Each base practice must contribute to at least one outcome and must not contribute to capability levels upper than 1; they are phrased as actions.

The artefacts associated with the execution of a process are work products. Input and output work products are indicative and not exhaustive.

The selected measurement framework of IRMIS PAM is based on the process measurement framework for process capability assessment proposed in ISO/IEC 33020 (International Organization for Standardization, 2015c).

For core common processes deduced from ISO 31000 and quite similar to core common MSS ones, a mapping has been performed between goal trees, and existing process descriptions in (i.e.) ISO/IEC 33073. The Management system terms are not reused as such but are replaced by ISO 31000 relevant ones: the main replacement concerns "management system", replaced by "risk management framework". For instance, for the Improvement process, the process description is directly inspired by the Improvement process of the core common processes for a management system. The improvement mechanisms are sufficiently generic and can be applied to a risk management framework without particular adaptations. In the case of this process, no dedicated view is provided for ISO 21500 and ISO/IEC 27001 as there are no detailed statements related to improvement in these respective standards.

In order to provide a process illustration dedicated to Risk management, the Risk treatment process is proposed below. As mentioned previously in the paper, the activities at the heart of risk management are specifically described in the IRMIS PRM and PAM. We are now presenting Risk treatment derived from ISO 31000, with additional views providing information coming from ISO 21500 and ISO/IEC 27001 (see Table 11.5). We have made this deliberate choice because ISO 9001 and ISO/IEC 20000–1 do not provide detailed information related to Risk treatment, contrary to ISO 21500 and ISO/IEC 27001 (as well as inputs from ISO/IEC 27005).

Risk treatment process description

Process **ID RIS.05**
Name Risk treatment
Purpose The purpose of risk treatment is to select and implement options for addressing risk.
Outcomes As a result of a successful implementation of this process:

1. Risk treatment options are selected by balancing potential benefits against the costs, effort, or disadvantages of implementation.
2. Selected risk treatment options are specified with appropriate information for justification, implementation, integration, and documentation.
3. Risk treatment plans for remaining risks and new risks are executed.
4. Remaining risks are communicated to decision-makers and other stakeholders.
5. Each risk change to consider is updated.

Comments on the Risk treatment process

This process is critical in the overall risk management loop. It is the process to modify risk (as defined in the ISO Guide 73). When treating risks, new risks can appear (and then, they have to be assessed), and existing risks are modified.

After designing the IRMIS PRM and PAM first drafts, the first level of validation has been performed by experts with knowledge in ISO/IEC 330xx, project management, ITSM, and Information security. A set of systematic review criteria has been used: an outcome is targeting capability level 1 only; an outcome can be identified as an artefact; the wording is clear and appropriate for all PAM components; the vocabulary used in the PAM is consistent; each process is defined with the characteristics: integration, assessability, interoperability, completeness, adoption, and applicability. Some improvements have been performed, particularly for the wording and the used terminology. All the processes of the PRM and PAM are reviewed in the same way.

Table 11.5: The Risk treatment process description and views in the IRMIS PAM.

	ISO 31000 view	ISO 21500 view	ISO/IEC 27001 view
Process ID	RIS.02		
Process Name	**Risk identification**		
BP1 (Out 1)	Select risk treatment options.For selecting risk treatment options, consider the organization's objectives, risk criteria, and available resources.	Insertion of resources and activities into the budget and schedule	Selection of appropriate information security treatment options, taking into account of the risk assessment results
BP2 (Out 2)	Specify selected risk treatment options with appropriate information for justification, implementation, integration, and documentation in risk treatment plans.	Risk treatment includes measures to avoid the risk, to mitigate the risk, to deflect the risk, or to develop contingency plans to be used if the risk occurs	Formulate an information security risk treatment plan
BP3 (Out 3)	Execute risk treatment plans for remaining risks and new risks.		Determine all controls that are necessary to implement the information security risk treatment options chosen
BP4 (Out 4)	Communicate remaining risks to decision-makers and other stakeholders.		Obtain risk owner's approval of the information security risk treatment plan and acceptance of the residual information security risks
BP5 (Out 5)	Update risk changes in the risk register.		The organization shall retain documented information about the information security risk treatment process.
Input Work Products	Risk register Risk criteria	Risk register Project plans	Information security risk treatment plan
Output Work Products	Risk treatment plans Remaining risks Risk register	Risk responses Change requests Risk register	

11.4 Case study: Use of the IRMIS PAM

In order to contribute to the validation of the IRMIS PAM, a case study has been performed. It occurred in a university of the Greater Region (the Great Region encompasses the Grand Duchy of Luxembourg, the French region Grand Est, Wallonia, the Federation Wallonia-Brussels and Ostbelgien in Belgium, Saarland and Rhineland-Palatinate in Germany). This university (we will name it University A), has performed a risk management campaign in 2018 and 2019 in order to address several domains in terms of risk management such as governance, ethics, human resources, finance, data management, IT service management, information security, project management, etc. According to the overall maturity of risk management practices in University A, and for the purpose of using the IRMIS PAM, two perspectives have been targeted: project management and IT service management. The project management context related to the ISO 21500 standard has been selected with a Software-as-a-Service development project. For IT service management, the overall IT service management system that is being implemented in University A has been selected for the context related to the ISO/IEC 20000–1 standard. One process has been selected from the common processes of management systems: the Leadership process (TOP.01) and all processes from the risk management group have been selected: Risk criteria definition (RIS.01), Risk identification (RIS.02), Risk analysis (RIS.03), Risk evaluation (RIS.04), Risk treatment (RIS.05). The Capability level target is 1 (Process Performance). A competent lead assessor has interviewed the project manager of the SaaS development project, and an IT engineer involved in ITIL implementation has been interviewed for the ITSM side.

In order to perform the process assessment in University A, the PAM has been completed with questions (Q) related to each base practice for all selected processes. Then based on interviews, answers (A) are provided. For risk management dedicated processes, answers are provided consecutively for each base practice for project management (PM) and for IT service management (ITSM). A rating for each base practice and Capability level 1 (Process Performance) have been determined. Table 11.6 provides an example for the Leadership process.

Table 11.7 provides an example for the Risk identification process.

Table 11.6: IRMIS Process assessment form and results for the Leadership process.

Process ID	TOP.01
Process Name	Leadership
Comment	Ref ISO/IEC 33073 TOP.01 Note 1: "Quality management system" has been replaced by "risk management framework"; "quality policy" has been replaced by "risk management policy"
Process Purpose	The purpose of Leadership is to direct the organization in the achievement of its vision, mission, strategy, and goals, through assuring the definition of a risk management framework, a risk management framework policy, and risk management framework objectives.
Process Outcomes	As a result of the successful implementation of the Leadership process: 1. The context of the organization, including the expectations of its relevant interested parties, are understood and analyzed; 2. The scope of risk management activities is defined, considering the context of the organization; 3. The risk management policy and objectives are defined; 4. The risk management framework and operational process strategy is determined; 5. Commitment and leadership concerning the risk management framework are demonstrated.
Base Practices	**TOP.1.BP1 Determine external and internal issues that are relevant to the organization and analyze their impacts.** Determine external and internal issues that are relevant to the purpose of the assessed organization and that affect its ability to achieve the intended outcome(s) of its risk management framework. [Outcome 1] Q – What are external and internal issues related to the risk management framework? What are the regulations? What are the compliance obligations? A – University A: national law and Internal regulations + GDPR + HSE regulations **TOP.1.BP1 -> F** **TOP.1.BP2 Determine the relevant interested parties and analyze their requirements.** Determine the relevant interested parties that are relevant to the risk management framework and establish appropriate contacts with them. [Outcome 1] Q – What are the relevant interested parties: Governance committee, Directors, Managers, Internal audit? What are their requirements regarding risk management? A – Board of governors, Rectorate, University Council, Committees -> requirements for risk management in terms of domains: Governance, Data management, IT Service Management, Information security, Project management, HR, HSE, Finance, Ethics, . . . **TOP.1.BP2 -> F**

Table 11.6 (continued)

	TOP.1.BP3 Determine the scope of the risk management framework. Determine the boundaries and applicability of the risk management framework, taking into consideration the context of the organization, the requirements of relevant interested parties and the interfaces and dependencies between activities performed by the organization, and those that are performed by other organizations. [Outcome 2] Q – What are the boundaries of the risk management framework? Are they domains which are not assessed? A – Depending on the years, all domains were not assessed; in 2019, all domains have been assessed **TOP.1.BP3 -> F**
	TOP.1.BP4 Define a risk management policy. Define a risk management policy that is appropriate to the purpose of the organization. [Outcome 3] Q – Is there a risk management policy? Is it adapted to the organization? Does it need to be tailored? A – At the moment, there is not yet a risk management policy but objectives for risk management are defined and clearly allocated for implementation and followed up (cf BP5) **TOP.1.BP4 -> P**
	TOP.1.BP5 Define risk management objectives. Define risk management objectives at relevant functions and levels, which are measurable, consistent with the risk management policy. [Outcome 3] Q – Are there risk management objectives defined per year? A – Every year, risk management objectives are determined by the Board of Governors and the Rectorate, and are clearly allocated for implementation and followed up. But this is in place only since 2018. **TOP.1.BP5 -> F** . . .

11.4.1 Process assessment results

Table 11.8 summarizes the results of the process assessment for Capability level 1. Level 1 is reached for all processes except for Risk treatment where practices need to be improved and institutionalized.

Table 11.7: IRMIS Process assessment form and results for the Risk identification process.

Process ID	**TOP.01**
Process Name	Risk identification
Comment	Specific risk management process
Process Purpose	The purpose of the Risk identification process is to find and describe risks that might help or prevent an organization from achieving its objectives.
Process Outcomes	As a result of the successful implementation of the risk identification process: 1. Relevant information and risk identification techniques are selected. 2. Factors of risks and their relationships are examined. 3. Risks are identified, based on factors of risks.
Base Practices	**RIS.02.BP1. Gather relevant and up-to-date information for the identification of risks** (appropriate background information where possible) [Outcome 1] Q – What is the background information for identifying risks? What is the relevant and up-to-date information for the identification of risks? *Input 21500 for RIS.02.BP1: information comes as the project progresses through its life cycle.* A – PM For identifying risks within the SaaS development project, all information related to the partnership with the client company was gathered. Meeting minutes were considered, in particular steering committees meeting minutes, which are the moment when the risks situation is considered. **RIS.02-PM-BP1 -> F** A – ITSM For identifying risks related to the ITSM system, IT committee meeting minutes were considered (the moment when the risks situation is considered), as well as Executive committees meeting minutes. It was not always easy to identify all relevant information for ITSM risks (scattered in the organization) **RIS.02-ITSM-BP1-> L** **RIS.02.BP2. Select context-relevant risk identification tools and techniques**. [Outcome 1] Q – Did you identify particular tools or techniques for risk identification? Did you select any of these tools or techniques? If yes, what were your selection parameters? A – PM There were no particular selected tools or techniques for identifying PM risks. IT was performed empirically, and based on the experience of the project team (sufficient in this context) **RIS.02-PM-BP2-> L** A – ITSM The technique used for identifying risk was based on 2 approaches: firstly a selection of risks made by the risk manager in the university, for the overall risk campaign. Then brainstorming sessions were organized for each domain and one of them was ITSM. It was sufficient for the context. **RIS.02-ITSM-BP2-> F** . . .

Table 11.8: IRMIS Process assessment results for University A.

Assessed processes	Capability Level 1 result
TOP.01 Leadership	L
RIS.01 Risk criteria definition	
– Project management	L
– IT service management	F
RIS.02 Risk identification	
– Project management	L
– IT service management	F
RIS.03 Risk analysis	
– Project management	F
– IT service management	F
RIS.04 Risk evaluation	
– Project management	L
– IT service management	F
RIS.05 Risk treatment	
– Project management	P
– IT service management	P

11.4.2 Lessons learned

A SWOT (Strengths / Weaknesses / Opportunities / Threats) analysis (see Table 11.9) has been performed in order to learn lessons from the case study and identify areas for improvement of the approach and framework.

Table 11.9: SWOT analysis.

Strengths	Weaknesses
Appropriate format for process assessment Usefulness and added value of questionnaires Didactic approach for risk management Common processes for management systems Integrated approach for various domains (quality, ITSM, Information security, and project management)	Not appropriate for immature organizations in terms of Risk management Too detailed for small organizations Vocabulary and questions unclear for risk criteria and risk factors
Opportunities	**Threats**
Use of the PAM for Information Security and Quality management systems (with associated questionnaires) Use of the IRMIS approach for Internal Control and Internal Audit purposes	Heaviness of the approach if all processes are assessed. Lack of example of risks, criteria, factors, . . . in the IT targeted domains

This case study has shown the usability of the integrated framework with some limits and improvements to perform to the approach. For a pragmatic use in industry, some engineering refinements are needed, but the overall framework can be used easily. Some examples of risks, criteria, factors, should be added for optimizing the approach.

11.5 Discussion

In this chapter, the integration aspect is paramount. This is the reason why integration based on terminology and structuring is essential. As ISO standards are developed based on international consensus, the terminology equipping these standards is proven and recognized. On top of that, ISO has performed a dedicated effort for harmonizing Management System Standards by imposing a common structure for all of them, with compulsory clauses and requirements. Even if our mainline is driven by ISO 31000 which is not identified "directly" as a management system it is admitted that the risk management framework advocated by ISO 31000 is similar to a management system as defined in Annex L. The various mappings performed by the authors confirmed this as well as the case study.

From assessability and adoption perspectives, it is necessary to keep an adapted number of processes for a pragmatic and operational implementation in organizations.

When developing a process reference model, as stated in ISO/IEC 33004: "*process descriptions shall not contain or imply aspects of the process quality characteristics beyond the basic level of any relevant process measurement framework conformant with ISO/IEC 33003*". The fact to deal with documentation and planning aspects could be linked to Capability Level 2. But for simplification, the authors decided to propose a dedicated process and to adopt the same documentation management mechanisms like the ones of this process in MSS PRM and PAM.

The IT organizations' specificities are not particularly visible in the elicitation of processes at the PRM level. Particular attention is paid on these aspects at the PAM level in particular with the view provided for project management with ISO 21500 and for Information security with ISO/IEC 27001.

Finally, the risk management dedicated processes of the PRM are finding most of their inputs in ISO 31000, and ISO 21500, ISOIEC 27001 and ISO/IEC 27005 as a complement in the IRMIS PAM. With the IT organization's mindset, specific concerns related to risk management remain connected with service management and information security respectively for ISO/IEC 20000–1 and ISO/IEC 27001.

11.6 Conclusion and future works

In current IT organizations, GRC activities play an important part in risk management as a key challenge in several areas in which the nature of risk differs (they can be related to quality, projects, IT services, Information security). Risk management has to be organized, to be part of and integrated within the management system(s) of the company. In order to address all these challenges, this chapter proposes an integrated risk management approach in IT settings based on ISO standards, with two main artefacts: a Process Reference Model and a Process Assessment Model. Both artefacts contribute to an Integrated Risk management Improvement Framework in IT settings with ISO standards and are the main research contributions of this work.

This research contributes to the literature of various domains as it associates them in several ways; the various contributions relate mainly to the following literature: risk management and integrated risk management, management systems, capability and maturity models, process assessment, and process improvement. The research also contributes to the literature on ISO standards, with a particular focus on management systems and process-based approach standards, including such main relevant IT-related management system standards in service management and information security management.

The research methodology used in this work involves a DSR approach with the problem stated from which the design of the main artefacts (IRMIS PRM and PAM) was triggered; additional intermediary artefacts (mappings, requirements trees, goal trees) were designed for supporting the overall approach: in particular, a Transformation process has been followed to derive the process purpose, outcomes and base practices from the elementary statements of the Ariane's thread of the research works: the ISO 31000 standard. A first complete iteration according to the DSR approach was performed; more iterations are to be performed to improve the main artefacts (IRMIS PRM and PAM). Academic researchers and industry practitioners' feedbacks are considered throughout scientific and professional communications, as well as with standardization works in the ISO community.

Future research avenues can progress along different lines. In particular:

- The consolidation of the results with more case studies and DSR iterations is foreseen. This will enable us to refine the PRM and PAM. These iterations will also enable us to present the research outcomes to the ISO community in expert groups and get more feedback.
- These ISO perspectives can be additional benefits of ISO standards for industry related to risk management, and also to quality management, project management, service management, and information security management in a management system context.
- Revisions of the selected standards for this Chapter can also be the opportunity for new DSR iterations and improvements of the IRMIS PRM and PAM.

- Integration is a key challenge in this Chapter and has been tackled from the management systems perspective with a systematic methodology (Transformation process), for considering statements from the ISO 31000 standard on risk management with a GORE technique. This enabled to align statements with the process elements of existing PAMs (International Organization for Standardization, 2016c, 2016b, 2017b). With the growing complexity and market demands for standards such as ISO ones targeting management systems certification, and regulations imposed by legislators, their combined translation into integrated C&MM (s) become more difficult. New research challenges appear with the integration of several domains where risk management is just one case. Then not only requirements engineering play an important part, but also other disciplines such as regulatory compliance; the demonstration of traceability is then an additional key challenge to tackle, beyond integration when there are multiple standards and regulations to address.
- Integration could also be tackled from a harmonization point of view, with an ontology to represent the knowledge. An ontology could clarify the risk management domain's structure of knowledge, and enable knowledge sharing; several ontologies could be developed to complete the generic risk management one in order to cover multiple domains. These ontologies could be the basis for formalizing processes of the IRMIS PRM and PAM and help to update them.
- Situational factors related to risk management may also be investigated in order to check the best way to apply this generic and integrated Risk management process reference model in IT organizations. Key situational elements affecting risk management in IT settings could be investigated and a reference framework with classifications and factors that inform the risk management processes could be proposed.
- Process assessments based on the IRMIS PAM in various sectors could be compared in order to investigate further its relevance and to determine potential adaptations in IT settings for each specific sector as the nature of risk varies.
- Finally, this Chapter work contribute to the enhancement of the TIPA Framework and can extend it in several ways:
  - Populate the TIPA library of process models with an additional PRM and PAM (IRMIS).
  - Contribute to the Transformation process on-going enhancement with an additional case of multiple sources.
  - Strengthen the overall TIPA approach by considering underpinning theories such as the Unified process, as explained by Scott (Scott, 2002).

Acknowledgment: This work has been supported by the Spanish Ministry of Science and Technology with ERDF funds under grants TIN2016-76956-C3-3-R and PID2019-105455GB-C33.

References

Alberts, C. J., & Dorofee, A. J. (2010). *Risk Management Framework. Technical Report CMU/SEI-2010-TR-017 ESC-TR-2010-017* (SEI (Ed.)).

Association Française de Normalisation. (2016). *Afnor normalisation, Standardization: a Genuine Advantage for the Economic Activity of Companies that get Involved in it*.

Automotive SPICE. (2016). http://www.automotivespice.com/fileadmin/software-download/Automotive_SPICE_PAM_30.pdf

Bandyopadhyay, K., Mykytyn, P. P., & Mykytyn, K. (1999). A framework for integrated risk management in information technology. *Management Decision*, *37*(5), 437–445. https://doi.org/10.1108/00251749910274216

Barafort, B., Di Renzo, B., & Merlan, O. (2002). Benefits Resulting from the Combined Use of ISO/IEC 15504 with the Information Technology Infrastructure Library (ITIL). In M. Oivo & S. Komi-Sirviö (Eds.), *Product Focused Software Process Improvement* (pp. 314–325). Springer Berlin Heidelberg.

Barafort, B., Mesquida, A., & Mas, A. (2018). Integrated risk management process assessment model for IT organizations based on ISO 31000 in an ISO multi-standards context. *Computer Standards & Interfaces*, *60*, 57–66. https://doi.org/10.1016/j.csi.2018.04.010

Barafort, B., Renault, A., Picard, M., & Cortina, S. (2008). A transformation process for building PRMs and PAMs based on a collection of requirements – Example with ISO/IEC 20000. *International Conference SPICE*.

Barafort, B., & Rousseau, A. (2009). Sustainable Service Innovation Model: A Standardized IT Service Management Process Assessment Framework. In R. V O'Connor, N. Baddoo, J. Cuadrago Gallego, R. Rejas Muslera, K. Smolander, & R. Messnarz (Eds.), *Software Process Improvement* (pp. 69–80). Springer Berlin Heidelberg.

Barafort, B., Rousseau, A., & Dubois, E. (2014). How to Design an Innovative Framework for Process Improvement? The TIPA for ITIL Case. In *Communications in Computer and Information Science* (pp. 48–59). https://doi.org/10.1007/978-3-662-43896-1_5

Barafort, B., Shrestha, A., Cortina, S., & Renault, A. (2018). A software artefact to support standard-based process assessment: Evolution of the TIPA® framework in a design science research project. *Computer Standards & Interfaces*, *60*, 37–47. https://doi.org/10.1016/j.csi.2018.04.009

Barafort, B., V., B., Cortina, S., Picard, M., St-Jean, M., Renault, A., & Valdés, O. (2009). *ITSM Process Assessment supporting ITIL® – Using TIPA to Assess and Improve your Processes with ISO 15504 and Prepare for ISO 20000 Certification* (V. H. Publishing (Ed.)).

Buglione, L., Abran, A., Gresse Von Wangenheim, C., McCaffery, F., & Rossa Hauck, J. C. (2016). Risk Management: Achieving Higher Maturity & Capability Levels through the LEGO Approach. *2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, 131–138. https://doi.org/10.1109/IWSM-Mensura.2016.028

Casadesús, M., Karapetrovic, S., & Heras, I. (2011). Synergies in standardized management systems: some empirical evidence. *The TQM Journal*, *23*(1), 73–86. https://doi.org/10.1108/17542731111097506

Chittister, C., & Haimes, Y. Y. (1993). Risk associated with software development: a holistic framework for assessment and management. *IEEE Transactions on Systems, Man, and Cybernetics*, *23*(3), 710–723. https://doi.org/10.1109/21.256544

Chou, D. C. (2015). Cloud computing risk and audit issues. *Computer Standards & Interfaces*, *42*, 137–142. https://doi.org/10.1016/j.csi.2015.06.005

Cortina, S., Mayer, N., Renault, A., & Barafort, B. (2014). Towards a process assessment model for management system standards. *Communications in Computer and Information Science*. https://doi.org/10.1007/978-3-319-13036-1_4

Cots, S., & Casadesús, M. (2015). Exploring the service management standard ISO 20000. *Total Quality Management & Business Excellence*, *26*(5–6), 515–533. https://doi.org/10.1080/14783363.2013.856544

Hillson, D. (2006). Integrated risk management as a framework for organisational success. *PMI Global Congress Proceedings*.

International Organization for Standardization. (2003a). *ISO/IEC 15504–2: Information Technology – Process assessment – Performing an assessment*.

International Organization for Standardization. (2003b). *ISO/IEC 15504: Information technology – Process assessment, Parts 1–10*.

International Organization for Standardization. (2009). *ISO Guide 73, Risk management – Vocabulary*.

International Organization for Standardization. (2012a). *ISO/IEC 15504–5: Information Technology – Process assessment – An exemplar software life cycle process assessment model*.

International Organization for Standardization. (2012b). *ISO/IEC 15504–8: Information Technology – Process assessment – An exemplar process assessment model for IT service management*.

International Organization for Standardization. (2012c). *ISO 21500: Guidance on project management*.

International Organization for Standardization. (2013a). *ISO/IEC 15504–6: Information Technology – Process assessment – An exemplar system life cycle process assessment model*.

International Organization for Standardization. (2013b). *ISO/IEC 27001: Information technology – Security techniques – Information security management systems – Requirements*.

International Organization for Standardization. (2015a). *ISO/IEC 33001: Information Technology – Process assessment – Concepts and terminology*.

International Organization for Standardization. (2015b). *ISO/IEC 33004: Information Technology – Process assessment – Requirements for process reference, process assessment and maturity models*.

International Organization for Standardization. (2015c). *ISO/IEC 33020: Information Technology – Process assessment – Process measurement framework for assessment of process capability*.

International Organization for Standardization. (2015d). *ISO/IEC 330xx: Information technology – Process Assessment*.

International Organization for Standardization. (2015e). *ISO 9001: Quality management systems – Requirements*.

International Organization for Standardization. (2016a). *ISO/IEC 30105–2: TS Information Technology – IT Enabled Services -Business Process Outsourcing (ITES-BPO) lifecycle processes – Process assessment model (PAM)*.

International Organization for Standardization. (2016b). *ISO/IEC 33071: TS Information Technology – Process Assessment – An integrated process capability assessment model for Enterprise processes*.

International Organization for Standardization. (2016c). *ISO/IEC 33072: TS Information Technology – Process Assessment – Process capability assessment model for information security management*.

International Organization for Standardization. (2017a). *Benefits of Standards*. http://www.iso.org/iso/home/standards/benefitsofstandards.htm

International Organization for Standardization. (2017b). *ISO/IEC 33073: TS Information Technology – Process Assessment – Process capability assessment model for quality management*.

International Organization for Standardization. (2017c). *Management System Standards*. http://www.iso.org/iso/home/standards/management-standards.htm

International Organization for Standardization. (2018a). *ISO/IEC 20000–1: Information technology – Service management – Part 1: Service management system requirements*.

International Organization for Standardization. (2018b). *ISO/IEC 27005: Information technology – Security techniques – Information security risk management – Requirements*.

International Organization for Standardization. (2018c). *ISO/IEC Directives, Part 1. Annex L Proposals for management system standards*.

International Organization for Standardization. (2018d). *ISO 31000: Risk management – Principles and guidelines*.

International Organization for Standardization. (2018e). *The ISO Survey*. http://www.iso.org/iso/iso-survey

ISACA. (2013). *COBIT 5: Process Assessment Model (PAM)*.

Kontio, J. (2001). Software Engineering Risk Management: A Method, Improvement Framework, and Empirical Evaluation. In *Risk Management*.

Lepmets, M., McCaffery, F., & Clarke, P. (2016). Development and benefits of MDevSPICE®, the medical device software process assessment framework. *Journal of Software: Evolution and Process*, *28*(9), 800–816. https://doi.org/10.1002/smr.1781

Lyytinen, K., Mathiassen, L., & Ropponen, J. (1996). A Framework for software risk management. *Journal of Information Technology*, *11*(4), 275–285. https://doi.org/10.1057/jit.1996.2

Mesquida, A.-L., Mas, A., & Barafort, B. (2015). The Project Management SPICE (PMSPICE) Process Reference Model: Towards a Process Assessment Model. In *Communications in Computer and Information Science* (Vol. 543, pp. 193–205). https://doi.org/10.1007/978-3-319-24647-5_16

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*. https://doi.org/10.2753/MIS0742-1222240302

Pries-Heje, J., & Johansen, J. (2010). *SPI Manifesto*. European System & Software Process Improvement and Innovation.

Proença, D., Estevens, J., Vieira, R., & Borbinha, J. (2017). Risk Management: A Maturity Model Based on ISO 31000. *2017 IEEE 19th Conference on Business Informatics (CBI)*, 99–108. https://doi.org/10.1109/CBI.2017.40

Renault, A., & Barafort, B. (2014). TIPA for ITIL – from genesis to maturity of SPICE applied to ITIL 2011. *Proceedings of the International Conference EuroSPI 2014*.

Roy, G. G. (2004). A risk management framework for software engineering practice. *2004 Australian Software Engineering Conference. Proceedings*., 60–67. https://doi.org/10.1109/ASWEC.2004.1290458

Santos Olmo Parra, A., Sanchez Crespo, L.-E., Alvarez, E., Huerta, M., & Fernandez Medina Paton, E. (2016). Methodology for Dynamic Analysis and Risk Management on ISO27001. *IEEE Latin America Transactions*. https://doi.org/10.1109/TLA.2016.7555273

Scott, K. (2002). *The unified process explained*. Addison-Wesley Longman Publishing Co., Inc.

Simon, A., Karapetrovic, S., & Casadesús, M. (2012). Difficulties and benefits of integrated management systems. *Industrial Management & Data Systems*, *112*(5), 828–846. https://doi.org/10.1108/02635571211232406

Software Engineering Institute. (2010). *CMMI for Development, Acquisition & Services, version 1.3*.

The Cabinet Office. (2011). *IT Infrastructure Library. The Stationery Office Edition*.

TIPA for ITIL. (2015). https://www.list.lu/fileadmin//files/projects/ TIPA_T10_ITIL_PAM_r2_v4.1.pdf

Varajão, J., Colomo-Palacios, R., & Silva, H. (2017). ISO 21500:2012 and PMBoK 5 processes in information systems project management. *Computer Standards & Interfaces*, *50*, 216–222. https://doi.org/10.1016/j.csi.2016.09.007

John Malara

12 Embracing risk to gain competitive advantage

12.1 Introduction

Planning for and mitigating risk is a central tenet to good software development practices. While this holds true for most development projects, avoiding *all* risk when developing software is not always a good thing. In this chapter, we will discuss why sometimes business situations make it critical that companies both embrace and encourage risk taking, especially if their business life is on the line.

Business conditions are constantly changing. Companies that can change and evolve are usually the most successful and that evolution requires some risk taking. Technology is often at the forefront of this risk taking, and in some cases can be utilized to rescue a failing business or develop groundbreaking innovation used to gain or keep competitive advantage. Whatever the reason, sometimes embracing risky software development can be crucial to the success or failure of a company.

Most large SW Development projects have some risk. But most of these risks are well known and documented. The riskiest projects are those that either use new, untested technology or are custom developed for a specific application and where failure has the potential to harm the business.

Managing this software development requires principles and practices unique to each situation making this a complicated topic with many implications. In this chapter the focus will be not on avoiding and reducing risks, but when and how to embrace those risks to increase the chance for project, and ultimately business success. We will discuss the business conditions that might force risk taking, how a strategy might be developed, and look at a case study where risk and contingency is examined.

12.2 Identifying a call to action: Why should companies embrace risk?

In this section, we discuss two risk scenarios. Businesses that were *forced* to change and businesses that *chose* to change.

https://doi.org/10.1515/9783110652321-012

12.2.1 Scenario 1: Taking risks because you have to

In the last two decades, the introduction of new technologies and new business models has caused massive disruption to existing companies. This disruption has changed the way brands are built, supply chains are run, and how products are marketed and sold. But disruption is not new. Ford disrupted the horse and carriage business more than 100 years ago with the introduction of the automobile (Leinward and Mainardi, 2017).

This scenario examines companies that found themselves in situations where they *needed* to embrace change because new business models led by technology transformed their industries.

The introduction of digital technology has profoundly changed the camera business by merging cameras with phones, thus making traditional cameras almost obsolete. By creating digital images that could be stored on personal devices, new technology removed the need to buy film. The camera giants, Kodak and Fujifilm needed to recognize these changes in technology and adapt. Both were leaders in their field who relied heavily on profits from the old business model and were reluctant to change. Kodak, who held on to their old model for as long as it could, quickly lost most of its camera and film sales as digital took over, and ultimately filed for bankruptcy. Kodak's indecision and reluctance to move quickly and change their technology almost cost them their business. Fuji on the other hand, proactively embraced the new technologies and thrived. In the same market conditions, Fuji found a way to innovate by producing more advanced digital cameras and printers and became a leader in their industry (Esvary, 2016).

Another example of companies forced into taking risks stemmed from the introduction of increased bandwidth. This led to a boom of streaming services that have changed the way we watch TV and listen to music. Blockbuster, who led the industry with numerous brick and mortar stores that rented video tapes, prospered by providing entertainment content to consumers. They were slow to recognize and adopt the opportunity that increased bandwidth brought. Netflix, who initially had a similar business model as Blockbuster, did embrace those changes and ultimately prospered, putting Blockbuster out of business (Leinward and Mainardi, 2017).

These are just two examples of companies in industries that were changed by the introduction of new technology and how their reactions to those changes influenced their success. This scenario described changes they were *forced* to make that were completely transformational in nature but were not at all strategic. Their embracing of the change was reactionary and defensive. These types of issues are rarely cured with software development alone, and often must result in complete technology conversions and business model changes. While it is interesting to look at these examples of companies that needed to embrace risk to survive, this chapter is about software development, so let's look at another scenario where SW Development plays a larger part.

12.2.2 Scenario 2: Taking risks because you want to

Some companies are built on Innovation. Apple has been an innovator from their first product introduction, the Apple 1. They embraced principles that their main competitor, IBM did not. Elegance, simplicity and a better user experience is what they thought would differentiate them. For companies like Apple, taking risks has become the norm, and as such, they are able to manage through risks that other companies find impossible. When you consider that most new company startups and new product introductions fail, the true innovators have figured out how to embrace and master risk. Apple's whole corporate culture touts and celebrates their innovative spirit. Risk taking is encouraged and rewarded. They hire only the most creative people and are willing to fail a few times before they find success. Consumers have come to expect a certain level of innovation from Apple and expect them to be on the leading edge of risk. Their product lines are always changing, with new hardware and software being released continuously. Apples SW Development teams have embraced risk and have been developing cutting edge software since the company's inception. IBM on the other hand, struggled with Innovation. Their stubborn insistence on staying with their cash cow, mainframe computer sales, almost brought them down. It was only their size and support of long time corporate customers that saved the company, and only when it reluctantly changed product lines to the PC (Mattera, 2018).

Amazon is another company that embraces risk. They started out with a strategy that building a customer base was more important than showing profits while they expanded their product line from Books only to millions of SKU's. They built their technology around their strategy that online retail with fast delivery would ultimately win over consumers. Their ability to add products to their lineup, even challenging grocery items, has shown to be a winning strategy that has completely disrupted retail. The development of their eCommerce platform was the core of their business model and they developed that software when the model was new. The scale they had to accommodate had never been tried and the model was extremely risky. They knew their success as a business relied on their ability to drive this new eCommerce model and took on the risk. Amazon is now leveraging these platforms they created to expand into areas of retail that were traditionally not considered possible for an on line sales company (Brang, 2017).

These risk scenarios tend to be more organic and constant versus reactive and defensive. Adopting new technologies often is at the forefront of innovation and a discussion of SW Development and risk taking is more appropriate for this scenario.

Most companies are somewhere in between these two scenarios. They don't have a "burning platform" like scenario one companies do and aren't innovators like scenario two companies. They might be established companies that need to try to increase their sales or cut costs. They might be companies that want to modernize their operations or improve communication. No matter the reason, choosing where to innovate becomes critically important as funding can be the limited and time short. Embracing risks may be difficult for them because they aren't forced into it and don't have it in their DNA. In some ways, these mainstream companies have a harder time choosing a risky path. These companies are where one or two critical SW development projects can make a huge difference. As you can see from some of the examples above, sometimes making that initial decision to change is the hardest part.

12.3 Developing an IT strategy

Many companies have a formal Strategy Development Process where they plan for the future. Most have a three to five year time horizon where they try to look ahead and predict where their industries can be so they can develop strategies to achieve their business goals. When companies are confronted with the reality that they must make massive changes like in scenario one above, it is rarely done inside of a formal Business Strategy Cycle. These companies may seem blindsided by adverse business environments, but it rarely sneaks right up. As an example, Amazon's march to own retail has been over ten years in the making. Smart companies are those that recognize they must change and choose to act sooner rather than later.

For the rest of the companies like scenario two above, or companies a step or two behind them, change is usually part of the strategy cycle. These new ideas for the strategy may come from many sources. Very often they surface in the Boardroom where experienced Board Members see market changes coming and advise their CEO's to develop specific plans for the future. Meetings can happen at the highest levels that discuss business results, competition, market conditions, etc. Discussions also happen down at the operational level. Field employees raise issues with supervisors or ideas spring up from the people that are actually doing the work. Good ideas can come from anywhere. Great companies listen to their employees and take action. Talks happen in Boardrooms, conference rooms, meetings, hallways and coffee rooms. Ideas may start at the top and trickle down, or start at ground level, make their way to the leaders, and trickle back down as formal plans.

Formal company strategies sometimes are developed using planks, or sections. There might be a plank for Growth, a plank for Productivity, and a plank for People or Organization. Usually at least one of the planks of a sound strategy will focus on business core competencies, defined as those that provide a competitive business advantage, and focus on that area. Companies should focus their risk on these strategic core competencies. For example, a global Food Service company like Aramark should not be taking much risk when implementing a payroll system. They are in the Food Service Business, not Payroll Business. For a Food Service Company, payroll is arguably not a strategic function so minimal investment, and risk should be taken in this area. Typically, the most critical areas are those that touch customers or consumers or provide a core service. At most companies, increasing sales is extremely difficult to get, so that will surely get strategic attention. Productivity, or cost cutting, runs a close second in priority as increasing margin is often a key strategic goal. Good strategies outline specific yearly plans to achieve those stated targets. Much of the work to develop a strategy is selecting the right planks and developing the specific goals for each plank.

When IT Departments develop a long-term strategy, it follows behind the overall Business Strategy. IT Departments, like all others in the company, have a role to play in these plans. They will not try to develop their own strategy in a vacuum. It should tie back to the business strategy. Increasingly, because of the rapid advancements in technology, IT has a major role in achieving these targets.

Each department will develop their own long-term strategy (3 to 5 years), with a roadmap that is calendarized by year. Every year there is usually some type of ANNUAL OPERATING PLANNING (AOP), that takes those roadmaps and maps out the current year's goals. Out of this comes the projects for that year, along with funding requirements and staffing levels. Each department's projects will have a technology dependency that IT must use to plan resources. A good strategic plan is updated every year and a good AOP process does not deviate from that strategy.

Many large companies will utilize a Project Management Office (PMO) to aid in the development of an IT Strategy and many times will lead the efforts to determine strategy. The PMO can also be there to aid in helping see the overlap in project schedules. Some methodologies employ a "Map Day" to calendarize projects by phases, to assess the impact on schedules and resources.

It should be noted that there is a significant difference in a company embarking on a packaged SW Implementation project, versus a custom SW development project in terms of risk. While most companies certainly will benefit from implementing standard package software, true Innovation usually means CUSTOM SW development. While the package implementation is rarely simple, the risk and contingency are well known. For example, at service companies that differentiate through a proprietary business process, custom software may be the only way to support the process. Custom development may be necessary to gain an edge.

For companies that have not yet implemented an Enterprise Resource Planning System (ERP), developing an Innovation Strategy is more difficult. Having competing priorities can be dangerous because the safest course is usually to implement the base ERP before embarking on an Innovation Strategy. Using a principal-based prioritization process that has input and buy in at the top is critical to making sure that priority is given to the most strategic projects. Many times, if the funding is available, management will try to get everything done at once. It is very difficult to convince senior management that fewer, more focused projects are safer bets and patience to stay the course on a plan is the best way forward. Care must be taken to ensure that change is kept to a minimum so the disruption can me managed.

Benchmarking is also a necessary step in developing a strategy. Among the many ways that companies can benchmark are:

- Comparing yourself to your peers, especially your direct competition, by scouring the internet and using your employees field level observations.
- Getting Insights from research companies like Gartner and Forrester that gather industry trends and rate leaders in the SW Packaged Application world.
- Watching the big consulting houses who build practices around hot technologies. Recently we have seen practices built around Cloud Migrations, Digital Transformations, Mobil Development, Usage of Big Data, Internet of Things, etc.

An example of one of the planks in a simple strategy for a CPG company might be:

Plank – Increase Sales

- Grow current portfolio 5 % for 3 years
  - Win shelf space in grocery
    - Develop new ordering systems to drive new sales

Shortly after that strategy was approved, the AOP Plan was developed, and a project to develop a new ordering system was proposed.

In 2005, Pepsi began development of this new Ordering and Inventory System for Large Stores with the goal of only stocking the products that sold the most, reducing inventory and maximizing shelf space. Previously, orders were created in the office using Historical Sales Data with Promotions factored in. The new system would develop novel algorithms that would use real time data from IN THE STORE to determine the next order. Now, shelf consumption, pricing and promotions, competitors pricing and promotions were all factored in to create a new, better order. The goal was to prove to the retailer that Pepsi could sell more product with less inventory, thus resulting in the retailer awarding more shelf space to Pepsi (Thornton, 2010).

This is a real example of an IT project that was spawned from a top down business strategy. Hopefully you have seen that strategy can be developed in many ways and can be through a formal or informal process. That strategy should always include plans to improve the business in core areas and to do that risks must be taken. Good strategies embrace that risk and recognize the necessity of it. Once the strategy is set and the projects chartered and funded, execution begins. In the next section we will examine some tactics and methods for managing a highly risky project by examining a case study.

12.4 Case study: Digital/mobile transformation at aramark uniform services

Aramark Uniform Services (AUS) is a provider of professional work uniforms, supplies, and services. They offer rental, leasing and purchase options across a wide selection of businesses that include food service, automotive, retail, hospitals and hospitality. Their services also include restroom and hygiene programs, first aid kits, and floor mats to help promote safety and cleaner work environments. They deliver these products and services through certified Route Sales Representatives who consistently serve valued and loyal clients across all industries. AUS manufactures its own uniforms globally. The Regional AUS locations are called Market Centers, with Laundry Plants, Route Management Operations, Fleet Operations, Store Rooms and Administrative Support in most big cities in the US and Canada.

In 2016, AUS found itself under pressure to improve its business processes and technology. In a historically competitive environment, with aged legacy systems and processes that were hampering growth, the company faced the fact that it had to modernize. The company also realized that to keep up with its competitors, the product lines had to be expanded to add “adjacent” products to its mix. Restroom Services and Supplies, First Aid kits and other products were planned to be added over the next few years, and systems and processes had to be streamlined to manage the complexity of these additions. Old Business Processes relied solely on paper and additional staff was needed to manage manual processes not automated by technology. The call to action was clear, change or risk falling farther behind its peers (Reuters, 2016).

The company embarked on a program designed to reengineer their Field Operations called “Market Center of the Future”. This multi-year program, comprised of many projects, resulted in a true Digital/Mobile Transformation that changed how core business was performed. This was an extremely risky program in that it touched many people in the field, modernized core business processes and introduced new technologies to the company. As outlined above, the company decided it MUST take on this total reengineering effort in order to stay competitive. There was no choice.

Components of the project were:

- Creation of a Shared Services Center in Lexington, Kentucky. The Accounts Receivable and Customer Service Functions were removed from field sites and relocated to the Shared Service Center. Oracle Software was implemented, and Cloud Based Call Center Technology used.
- A new Route Accounting System for all sites was piloted using a proprietary Laundry Package.
- A new, Mobile Device with custom software was rolled out to 3000+ Customer Service Reps (CSR) that digitized Invoices and used electronic signature capture.

Apple iPhones were implemented with cellular technology to capture data in real time as accounts were serviced.
- Advanced Selling tools were introduced to the Mobile devices which aided CSR in selling in new products to customers. Catalogues and route books added to aid in route service management. New Product video capability was added to streamline new product innovation. (Managed Restroom Services, First Aid).
- Oracle Content Management was implemented to digitally store invoices allowing Customer Service to access these new digital images from the Mobile App versus sending out to a third party imaging company for copies of paper invoices.
- Digitization of Customer Contracts in Oracle to make contract info available to Customer Service and CSR's while out on routes.

Because of the overall size and scope of the projects, each individual project was led by a Project Manager with a Program Manager responsible for them all.

The overall goals of the project were to Increase Sales by Introducing New Products. Cut Costs by eliminating unnecessary headcount, reducing paper and printers, and automating manual processes. Improve Customer Service by removing non value added work from Customer Service Reps and arm them with current tools so they could spend more time servicing their accounts.

One huge advantage was that AUS had already completed an ERP Implementation using Oracle's eBusiness Suite for Financials including General Ledger, Accounts Payable and Purchasing. There was a strong base of other Back Office applications already in place including ADP for Payroll and Oracle's Hyperion for Financial Reporting. Operations Systems were also in place including Manhattan Associates for Warehouse Management and Hybris for eCommerce. Oracle's Data Warehouse was in place and Oracle was also used for Identity and Access Management. What remained were the most strategic, Core front line systems. These systems were historically slow to change because of the massive training effort and huge risk to the business if it were disrupted by a complex change as well as the huge Change Management bottleneck that would have been caused by the previous ERP implementation.

This Program did not begin as a bundled package at the start. The genesis of this effort started as separate projects but AUS quickly saw the benefits of end-to-end reengineering through Integration. The benefits went far beyond monetary savings and resulted in huge productivity gains detailed in a section below. As with most projects this size there were inherent risks just because of the scale. Almost every business function was affected, and many were blown up and rebuilt. There was potential for huge business disruption which needed to be avoided at all costs.

One way the risk of business disruption was mitigated was to split the project into modules so if one module was delayed, it would not delay the others. This was in fact what happened, and splitting the projects resulted in avoiding these disruptive delays. The Shared Services Transition, Route Accounting, and Field Mobile

Refresh, the three main components of the program, were separated and allowed to roll out separately. While this created extra work to manage the rollouts separately, it minimized a huge risk and contributed to the overall success of the project.

Let's focus on the technology changes as we examine the SW development risks and discuss ways to mitigate or reduce it. The large SW components were:

- Implementation of Packages from Oracle for Advanced A/R and Customer Shared Service.
- Implementation of a Package Application for Route Accounting.
- Implementation of new Call Center Cloud Based Software.
- Development of a new mobile based SW for Customer Service Reps.
- Integration of these new programs.

While most of these components are large, complex and therefore risky, most of them are packaged SW with known implementation risks. The riskiest application was the Mobile App used Customer Service Reps which was a Custom Developed App. For the purposes of this case study, we will examine the development of the new mobile based SW that was completely custom and developed in house. While this isn't as profound a change as adding a product line or new brands, as far as software development is concerned, mobile SW is one of the most risky, custom development efforts that can be undertaken. Below we will examine a few areas where risks needed to be taken and principles and practices were used to mitigate them.

12.4.1 Priority

The decision to embark on a custom SW project to develop mobile applications is always complex. This is extremely complicated SW with historically long development time and a high failure rate. These types of applications are usually loaded with complex business rules for pricing and tax related financial calculations, and getting it wrong can have monumentally disastrous consequences. Combined with the fact that new devices would be rolled out at the same time added to the complexity. There is very little packaged mobile SW available to the Laundry Business, and no off the shelf SW that can handle the introduction of new products and services. Using custom SW in this area also gave AUS the ability to differentiate itself from its competition. Given AUS Business Strategy to add "Adjacent" products to its mix of offerings, the ability to sell in and service new product categories, as well as train CSR's using video, made this custom application would be proprietary to AUS and not available to its competitors. This application became a must have.

Based on these factors, the decision was made to embark on a custom SW development effort.

12.4.2 Sponsorship

When developing a Program this big and important, having the right sponsorship is critical to a successful Program. Ensuring that the whole organization is pulling together is an absolute must. Keeping Senior Leadership in the loop through regular communications and regular demos helps to make sure the organization gets excited and management support stays strong throughout the life of the projects. The Program Manager must keep these projects the top priority and not allow the organization to lose focus. Many outside things can happen to derail projects of this size. Sponsors leave, budgets change, priorities shift and these can profoundly impact the projects in any number of ways. Having the rest of the organization on board is also critical as dragging feet can certainly slow down progress. Creating a Communication and Change Management Team can make the difference between success and failure. Managing the multiple phases of rollouts and their impact on field teams must be carefully managed and having a separate team can be critical. Recruiting the right business people that are dedicated to the project team was also important in that it brough credibility to the solution. Field people can push back on an IT team at times, citing lack of business experience so a good mix of team members can boost credibility.

Because of the way this program evolved, and the criticality of introducing new streams of revenue to the business, sponsorship was strong. Everyone on the senior team easily bought in, especially when they saw other phases of the program successfully being implemented and the benefits it was bringing. Senior management was instrumental in understanding and delivering the message to other parts of the business to help support the projects and make them successful. This support may have been the most important factor of all.

Another need is to create some “pull” from the field organization that thirsts for modernization. It is the field who feels the most pain by having to use antiquated tools and cumbersome processes. Seeing their peer companies and competitors using new tools can be frustrating. Adding to that the fact that almost everyone now is using Smartphone technology in their personal life has taken away much of the fear of new technology as people can see how much easier these tools make their lives. Having a plan to communicate the coming changes and demo the products creates a buzz and can pull the support for the project from the front line employees.

The proper funding of the projects is also a key factor in the success. Running out of money before the projects are completed can completely derail the efforts and cause the projects to stop. Having a clear and achievable business case and documenting the benefits as they happen can also “Self-Fund” these projects and help to keep momentum. The PMO can play a critical part in making sure all business and financial benefits are reached and communicated as they happen. Making sure the funds are flowing behind the scenes is imperative. Partnering with the Finance Team to manage the proper use of Capital Funding is critical, as these

Investments in Technology are very similar to buying trucks and building plants. The management of depreciation and amortization is important to make sure you don’t mortgage the future and bury the company in debt that can never be offset with real benefits.

Sponsorship is also needed to reign in development work on legacy systems. This is important for a few reasons. One, adding new functionality that then must be added to the new application just stretches out the time for new development. As previously mentioned, speed is an important factor, and anything you can do to limit scope will help. Two, diverting resources to the legacy applications can also slow down the new development. Legacy Staff can be critical to developing requirements, testing and supporting pilot locations, so diverting them away from the new project can just create more delay. Business doesn’t stop, so having a plan to accommodate some change is reasonable and practical.

12.4.3 Development methodology

When developing any new application to replace one that has been in production for many years, limiting scope can be a big challenge. There is usually pent up demand for new functionality that has been delayed either because the old technology cannot support it and/or it is difficult to find developers that can work with the technology. When developing new applications, it is advisable to fight to keep the scope of the functionality close to what is being used today to both manage the size of the development effort and to make the training of the field easier. In this case the internal company battle was fought to limit the scope to existing functionality and add new features and functions in later releases (ex. Signature Capture). This allowed AUS to “Throw it over the wall” to the field in a relatively rapid deployment. The development effort was streamlined by using functional specification developed from the legacy application. This cut a huge amount of time, and risk from the development by jump starting what can be the most time consuming part of any development, the requirements gathering. The ability to make the development effort smaller, and thus shorter and less complicated, allowed the old system to be decommissioned faster, saving money and reducing the overall development risk.

Using Agile Development methodologies are extremely efficient when there is existing software in production and iterative, smaller sprints can be released. The first version of this Mobile Software was developed in about six months, with smaller, faster sprints used later to fix bugs and add additional features and functionality. Once the base software was developed and deployed location by location across all field users, sprints were quickly released to add New Product Capability, Video Training and Signature Capture. The backlog of additional functionality wanted after these first sprints will last for as long as the application is in production, typical with critical applications used at the front lines of the business.

12.4.4 Staffing

Creating the right project team is also a key to a successful project and this project was no exception. Having a Program Manager that was experienced in Mobility Projects was vital to avoiding pitfalls and navigating through issues as they happened. Most importantly, a good SW Development manager can keep the project moving ahead as obstacles are encountered, such as obtaining approvals to proceed through phases, requests for additional funding, approval to hire and priority in queue for technical services like needing additional storage space in the data center. An important reason to use an experience Project Manager is that most projects typically bog down at some point, usually in the beginning. A good Project Manager can recognize when a project gets "Sideways" and can either step in and get it back on track or escalate to senior management to apply some pressure to help. This is critical to helping stay on track.

Having an experienced development team is also a must. Outsourcing and/or offshoring can potentially save some money but it must be planned carefully. There is no substitute to having a development team that you trust and has experience with this type of development. If a company has little experience with offshoring, this application is typically not where you want to learn. There is a time and place where outsourcing and offshoring may be right for your organization, but the company's most strategic applications may not be that time, especially when time can be more important than cost.

Many companies use Consulting Partners when implementing packaged applications like Oracle or SAP. The big consulting houses have practices built around these vendors and can expedite the projects while your company ramps up its in-house experience. Consulting Company costs are high but can be recouped by getting the software deployed earlier than you might be able to in-house. Careful use of implementation partners can be beneficial but having an exit strategy is a must. Managing these partners closely is also critical as sometimes your goals can be different from those of the consultants.

For this project, AUS had a Program Manager who had implemented Mobile Software previously and had access to an experienced development team. This paid huge benefits in that much of the requirements were familiar to the team including building to an Apple device. This focus during the initial phase of development was critical to the project in that the first version was ready in about six months. This is extremely fast for this type of application.

12.4.5 Pilot selection

With mission critical software that is field based like this Mobile Application, testing is important to make sure it is ready to be deployed nationally and won't damage the business. The desire to go quickly must be tempered with caution. Picking a pilot location is important to making sure you can properly test with the right amount of user support. Many times, pilot sites are selected because they represent all the conditions likely to be encountered with a full rollout. The desire is to test every condition you may encounter in one the one pilot site so the rollout can blast off quickly. In this case, it was determined that finding a "Friendly" site, with leadership that understood the difficulty and importance of the new application, would be more important than anything else. Most users fear change, and sometimes when new software is introduced, users can push back and blame faulty software for everything that is wrong in the location. This makes it difficult to really assess if the software is having the intended affect or other factors are affecting results. AUS knew that these risky changes were somewhat new territory, and changes in business process were also high risk, so not everything would be perfect at the start. Having Leadership that understood that and was patient and supportive helped get through the rocky introduction of the new application. Getting Headquarters to provide some necessary plan relief also help the location sign up for the disruptive Pilot Program.

The project team also decided that a pilot site close to HQ in Burbank, Ca. was selected so there could always be on site support from the development team to make sure route trucks could always run the business. More resources were available as a result of this proximity and help keep the pilot site stable. The pilot did indeed introduce instability in the location. The initial pilot lasted four months with numerous releases to fix bugs and stabilize the SW.

12.4.6 System rollout

During the pilots, the Rollout Strategy was discussed, and many different options considered. A few implementation options were also tested in the field pilot sites to see the impact of the change, and just how much training and support were required. Because most people were already familiar with using a personal Smartphone, little to no training was needed on the device. Also, because the scope was limited to existing functionality, little training was needed for the new application. The most complicated aspect was the initial loading of the new software on the new devices which had to be done at the location to take advantage of the higher bandwidth.

Some advocated for a "Big Bang" rollout where all locations received the new software at the same time. There are benefits to this as sometimes "Ripping the band-aid off" can result in great pain but for a shorter amount of time. This method required a large amount of support as support calls will come in all at once. Because this project required the introduction of new Smart Phones as well, the logistics of this approach became difficult to plan and manage. The lack of enough dedicated resources made this approach risky with the stakes very high.

Ultimately, to mitigate "Big Bang" risk, AUS decided to do a location by location rollout. The Change Management team created a plan to rollout the phones just ahead of installing the software. Back end software that synched up data to and from the phones had to be modified to accommodate both new software as well as old while the rollout was in progress. Field teams were created to provide minimal on-site support while back end teams monitored data to make sure the business was not being affected. In the end the system took about six months to completely rollout with about a month break in between for the holidays.

12.4.7 Benefits

Specific to the Mobil Device and Application, the benefits were enormous. The ability to introduce new product lines increased sales. The reduction of physical paper invoices saved time and money. Removing field-based printers resulted in savings. Creating digital images of invoices saved imaging and outside storage costs and allowed Customer Service Agents in the Shared Service Center access to the digital Images. Using the devices for video training resulted in increased productivity and cost savings. Having product catalogs, pricing and contracts digitally available on the mobile devices increased productivity and improved customer service. Overall, the ability for CSR's to provide a higher level of service to customers was a big win and a morale boost for the field. While not every component of the overall program went off as planned, most of the benefits were achieved, with many parts still being worked out for future releases. In hindsight, this project introduced a tremendous amount of risk to the core of the business but as detailed above, care was taken to understand, embrace, and mitigate that risk by utilizing some very specific tactics.

12.5 Conclusion

To recap overall:

- Challenging Business Environments make risk taking an imperative.
- Severe Risks should only be taken where a clear competitive advantage can be gained or in the extreme to save the company.
- Custom SW Development can sometimes provide a strategic advantage but with serious risk.
- Mitigating these risks is critical to maintaining business continuity.
- Having strong principles and practices can help mitigate the risks.

Hopefully, the discussion of the Case Study reinforced the idea that taking some risk in the right areas is necessary and that having the right principles and practices can mitigate some of that risk. Strong alignment with an overall Business Strategy and complete organizational support should be in place before risky projects are committed to. Hopefully, a review of the case study will give some examples of tactics that enabled the project in the study to mitigate some of the larger risk. A careful examination will also show that while some risks were taken, the worst risks were still avoided. Even on the riskiest projects the principles and practices commonly used should never be completely discarded. In today's business environment where speed to market has become more important than ever, project management failures are unacceptable and make risk identification and management more important than ever.

References

Brang, M., "Amazon Changed the Face of Retailing, let us Count the Ways," SPS Commerce (July 11, 2017) Accessed at: https://www.spscommerce.com/blog/ways-amazon-changed-retail-spsa/

Esvary, P., "Kodak vs. Fujifilm: The Truth Behind their Success and Failure," Leaderonomics (July 1, 2016) Accessed at: https://leaderonomics.com/business/kodak-vs-fujifilm-success-failure

Leinward, P. and C. Mainardi, "Disruptors and the Disrupted: A Tale of Eight Companies in Pictures," Strategy & Business (Sept. 17, 2017) Accessed at:. https://www.strategy-business.com/pictures/Disruptors-and-the-Disrupted-A-Tale-of-Eight-Companies-in-Pictures

Mattera, S., "IBM Used to be Bigger than Apple, What Happened?" The Motley Fool, (Oct.16, 2018) Accessed at: https://www.fool.com/investing/general/2015/04/11/ibm-used-to-be-bigger-than-apple-what-happened.aspx

Reuters, "Facilities Manager Cintas to buy Rival G&K in $2.2 Billion Deal" (August 16, 2016) Accessed at: https://www.reuters.com/article/us-g-kservices-m-a-cintascorp-idUSKCN10R1A4

Thornton, T., "Pepsi's Direct Store Delivery Program," Supply Chain Brain (Nov. 23, 2010) Accessed at: https://www.supplychainbrain.com/articles/9272-pepsis-direct-store-delivery-program

Contributors

Hyggo Almeida
Federal University of Campina Grande, Brazil

Béatrix Barafort
Luxembourg Institute of Science and Technology, Luxembourg

Amrita Bhattacharjee
Arizona State University, USA

Francois Christophe
Häme University of Applied Sciences, Finland

John Collins
Iona College, USA

Ricardo Colomo-Palacios
Østfold University College, Norway

Subhajit Datta
Singapore Management University, Singapore

Torgeir Dingsøyr
SINTEF Digital, Norway

Christof Ebert
Vector, Germany

Kurt J. Engemann
Iona College, USA

Jorge Marx Gómez
University of Oldenburg, Germany

Kyller Gorgônio
Federal University of Campina Grande, Brazil

Petri Kettunen
University of Helsinki, Finland

Kristina Kozakevitch
Iona College, USA

C. Christopher Lee
Central Connecticut State University, USA

He Li
Clemson University, USA

Subhashis Majumder
Heritage Institute of Technology, India

John Malara
Iona College, USA

Tomi Männistö
University of Helsinki, Finland

Antònia Mas
University of the Balearic Islands, Spain

Antoni-Lluís Mesquida
University of the Balearic Islands, Spain

Tommi Mikkonen
University of Helsinki, Finland

Holmes E. Miller
Muhlenberg College, USA

Manuel Mora
Autonomous University of Aguascalientes, Mexico

Joao Nunes
Federal University of Campina Grande, Brazil

Angelo Perkusich
Federal University of Campina Grande, Brazil

Mirko Perkusich
Federal University of Campina Grande, Brazil

Yvan Petit
University of Quebec at Montreal, Canada

Gloria Phillips-Wren
Loyola University Maryland, USA

Mary Sanchez-Gordon
Østfold University College, Norway

https://doi.org/10.1515/9783110652321-013

Heechang Shin
Iona College, USA

Ademar Sousa Neto
Federal University of Campina Grande, Brazil

Antti-Pekka Tuovinen
University of Helsinki, Finland

Fen Wang
Central Washington University, USA

Sungjin Yoo
Iona College, USA

Index

https://doi.org/10.1515/9783110652321-014

Developments in Managing and Exploiting Risk

The objective of this multi-volume set is to offer a balanced view to enable the reader to better appreciate risk as a counterpart to reward, and to understand how to holistically manage both elements of this duality. Crises can challenge any organization, and with a seemingly endless stream of disruptive and even catastrophic events taking place, there is an increasing emphasis on preparing for the worst. However, being focused on the negative aspects of risk, without considering the positive attributes, may be shortsighted. Playing it safe may not always be the best policy, because great benefits may be missed.

Analyzing risk is difficult, in part because it often entails events that have never occurred. Organizations, being mindful of undesirable potential events, are often keenly averse to risk to the detriment of capitalizing on its potential opportunities. Risk is usually perceived as a negative or downside, however, a commensurate weight should also be given to the potential rewards or upside, when evaluating new ventures. Even so, too much of a good thing may create unintended consequences of risk, which is also an undesirable situation. *Developments in Managing and Exploiting Risk* provides a professional and scholarly venue in the critical field of risk in business with emphasis on decision-making using a comprehensive and inclusive approach.

Vol. 1: Safety Risk Management: Integrating Economic and Safety Perspectives. Edited by Kurt J. Engemann and Eirik B. Abrahamsen

Vol. 2: Project Risk Management: Software Development and Risk. Edited by Kurt J. Engemann and Rory V. O’Connor

Vol. 3: Organizational Risk Management: Managing for Uncertainty and Ambiguity. Edited by Krista N. Engemann, Kurt J. Engemann, and Cliff W. Scott

Vol. 4: Socio-Political Risk Management: Assessing and Managing Global Insecurity. Edited by Kurt J. Engemann, Cathryn Lavery, and Jeanne Zaino

https://doi.org/10.1515/9783110652321-015