Chapter 2
Climbing the "Stairway to Heaven": Evolving From Agile Development to Continuous Deployment of Software

Helena Holmström Olsson and Jan Bosch

Abstract Software-intensive systems companies need to evolve their practices continuously in response to competitive pressures. Based on a conceptual model presented as the "Stairway to Heaven," we present the transition process when moving from traditional development towards continuous deployment of software. Our research confirms that the transition towards agile development requires a careful introduction of agile practices into the organization, a shift to small development teams, and a focus on features rather than components. The transition towards continuous integration requires an automated test suite, a main branch to which code is continually delivered, and a modularized architecture. The move towards continuous deployment requires internal and external stakeholders to be fully involved and a proactive customer with whom to explore the concept. Finally, the transition towards R&D as an innovation system requires careful ecosystem management in order to align internal business strategies with the dynamics of a competitive business ecosystem. Characteristic for all transitions is the critical alignment of internal and external processes in order to fully maximize the benefits as provided by the business ecosystem of which a company is part.

2.1 Introduction

Today, software development is conducted in increasingly dynamic environments characterized by inter-organizational relationships and dependencies. From being an activity defined by an organization's internal processes and practices, software

H.H. Olsson (✉)
Department of Computer Science, Malmö University, Malmö, Sweden
e-mail: helena.holmstrom.olsson@mah.se

J. Bosch
Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden
e-mail: Jan@JanBosch.com

J. Bosch (ed.), *Continuous Software Engineering*,
DOI 10.1007/978-3-319-11283-1_2

development is transitioning towards becoming an activity characterized by open innovation and co-creation of value in which an ecosystem of stakeholders co-evolves capabilities around new innovations [1, 2]. Typically, fast-changing and unpredictable market needs, complex and changing customer requirements, and pressures of shorter time-to-market are challenges that face most business ecosystems. To address this situation, the vast majority of companies have started adopting agile development methods with the intention to enhance the organization's ability to respond to change. In emphasizing flexibility, efficiency, and speed, agile practices have led to a paradigm shift in how software is developed [3–5].

However, while the adoption of agile practices has enabled companies to shorten development cycles and increase customer collaboration, there is an urgent need to learn from customers also after deployment of the software product. The concept of continuous deployment, i.e., the ability to deliver software more frequently to customers and benefit from frequent customer feedback, has become attractive to companies. If this is achieved, companies could benefit from even shorter feedback loops, more frequent customer feedback, and the ability to more accurately validate whether the functionality that is developed corresponds to customer needs and behaviors. While the ability to continuously deploy new software functionality creates new business opportunities and extends the concept of agile practices, it presents a number of challenges.

In this chapter, we present a multiple-case study in which we explore four software development companies that are currently moving towards continuous deployment of software. We present the challenges these companies face when transitioning towards continuous deployment and beyond. Also, we identify actions companies need to take to address, and overcome, these challenges. As our theoretical framework for analysis, we use the ESAO model [6]. The model provides dimensions for assessing the end-to-end dimensions of business, technology, and organization with considerations especially taken to external stakeholders and the business ecosystem of which a company is part.

2.2 Background

2.2.1 The "Stairway to Heaven"

Companies evolve their software development practices over time. Typically, there is a pattern that most companies follow as their evolution path and that we have reported on in previous studies [5]. We refer to this evolution as the "Stairway to Heaven," and it is presented in Fig. 2.1. The phases of the "Stairway to Heaven" are discussed in more detail in the remainder of this section. As a summary, companies evolving from traditional development start by experimenting with one or a few agile teams. Once these teams are successful, agile practices are adopted by the R&D organization. As the R&D organization starts showing the benefits of working

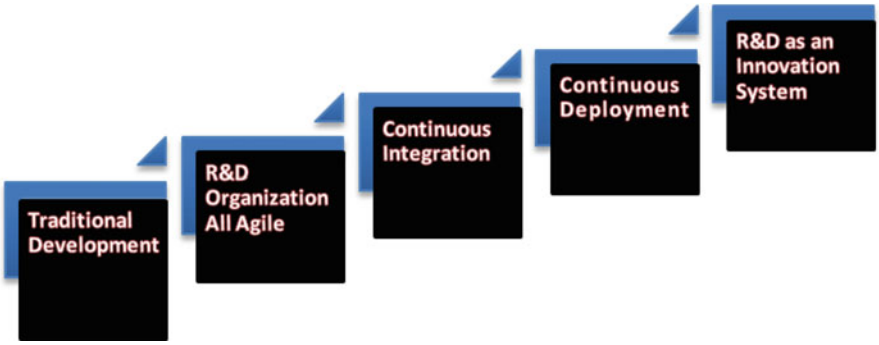

Fig. 2.1 The "Stairway to Heaven" evolution model

agile, system integration and verification becomes involved and the company adopts continuous integration. Once continuous integration runs internally, lead customers often express an interest to receive software functionality earlier than through the normal release cycle. They want continuous deployment of software. The final step is where the software development company collects data from its customers and uses the installed customer base to run frequent feature experiments.

Step 1: As the first step in our model, and the starting point for most embedded systems development companies, there is traditional development. We refer to traditional development as an approach to software development characterized by slow development cycles, sequential phases, and a rigorous planning phase in which requirements are frozen upfront [7]. Typically, the approach is characterized by a waterfall-style interaction between product management, product development, system test, and the customer. Projects adopting this development approach suffer from long feedback cycles and difficulties to integrate customer feedback into the product development process [5, 7]. Typically, delivery to the customer takes place in the end of the project life cycle, and it is not until then that customers can provide feedback on the software functionality they have received.

Step 2: As a way to address the many challenges experienced in the traditional development approaches, most companies start adopting agile development practices. Agile practices are characterized by small cross-functional development teams, short development sprints resulting in working software, and continuous planning in which the customer is involved to allow for continuous customer feedback [4, 8]. In agile organizations, however, product management and system verification still work according to the traditional development approach [5].

Step 3: The third step in our model is where a company succeeds in establishing practices that allow for frequent integration of work, daily builds, and fast commit of changes, e.g., automated builds and automated test. At this point, both product development organization and test and verification organization work according to agile practices with short feedback cycles and continuous

integration of work. Humble and Farley [9] define continuous integration as a software development practice where members of a team integrate their work frequently, leading to multiple integrations per day.

Step 4: Continuous deployment implies that you continuously push out changes to the code instead of doing large builds and having planned releases of large chunks of functionality. This allows for continuous customer feedback, the ability to learn from customer usage data, and to eliminate work that doesn't produce value for the customer. At this point, R&D, product management, and customers are all involved in a rapid, agile development cycle in which response time is short [10].

Step 5: The fifth and final step in the "Stairway to Heaven" evolution model is where the development organization responds based on instant customer feedback and where actual deployment of software functionality is seen as a way of validating functionality. The intention is to expose customers to partial implementation of functionality and use their feedback for determining the value of that particular functionality [11].

2.2.2 The ESAO Model

One of the most common models to provide a holistic perspective of the dimensions of business, technology, and organization is the BAPO model [12, 13]. The BAPO model defines four concerns, i.e., business, architecture, process, and organization, which can be used for the mapping of roles and responsibilities and for understanding organizational structures [13]. The model is frequently applied for analysis and assessment in both academia and industry.

More recently, the ESAO model has been proposed as an extension to the BAPO model [6]. The ESAO (Ecosystem, Strategy, Architecture and Organizing) model consists of six interdependent dimensions that are important to take into account for software development companies when assessing the alignment between their business, their technologies, their processes, and their internal and external interactions. The six dimensions concern an internal and an external company perspective, and it is helpful for understanding the interdependency between a company and the ecosystem of which it is part. In this study, we apply the external dimensions, i.e., the ecosystem dimensions, in our analysis of the challenges and the actions involved in the transition between each of the steps in the "Stairway to Heaven" model.

In Fig. 2.2, we present the ESAO model. The model emphasizes that the internal and external perspective on each dimension need to be aligned with each other, i.e., ecosystem strategy and internal strategy, ecosystem architecture and internal architecture, and ecosystem organizing and internal organizing. The ESAO model focuses on achieving and maintaining alignment between internal and external dimensions, and therefore, we find it useful in our analysis of how actions required by a company interplay with the ecosystem of which the company is part.

Fig. 2.2 The ESAO model [6]

2.2.2.1 The Ecosystem Perspective

In the ESAO model, the ecosystem perspective emphasizes the role of an organization in a larger context and the implications this has on business strategy. As illustrated in the model, it is critical to align internal processes with external processes and to align internal change initiatives with strategic options that are within the business ecosystem of which the organization is part. Below, we describe each dimension as presented in [6]:

Ecosystem Strategy: The strategy dimension of a company is related to the business ecosystem of the organization and the strategic options that it has available in its role in this ecosystem. Depending on the strategic choices made by the company, there are significant implications on the system and software development practices of the organization.

Ecosystem Architecture: The architecture dimension defines the interface between an organization's internal architecture and the solutions that are provided by suppliers, firms that build software on top of a product or platform, and other roles that have an impact on the product and its architecture. In addition to the focus on interfaces between different stakeholders, focus is also on the architecture strategy. In being part of a business ecosystem, commoditization and innovation of new functionality are ongoing processes that have an impact on long-term architectural planning.

Ecosystem Organizing: The organizing dimension deals with how firms work with their customers, suppliers, and ecosystem partners in terms of processes, tools, and ways of organizing the communication and collaboration. For example, many companies have internally adopted agile ways for working while they have suppliers that still use traditional development, causing a disruption in the internal

development processes. In the ESAO model, the organizing dimension emphasizes the importance of aligning internal and external development and release processes.

2.3 Research Method and Sites

This study builds on a six months multiple-case study [14], involving four software development companies. All four companies are moving towards continuous deployment of software. In representing different stages in this transition, we find these companies of particular interest for understanding the challenges involved, as well as the actions to take, in this process.

The main data collection method used is semi-structured interviews with open-ended questions [15]. In total, 18 interviews were conducted. In companies A and B, we conducted five interviews in each company, involving software and function developers, software architects, system engineers, configuration managers, and project leaders. In companies C and D, we conducted four interviews in each company, involving software developers, component/system integrators, project/release managers, product line maintenance, and a product owner. All 18 interviews were conducted in English and each interview lasted for about 1 h. In addition to summarizing notes, all interviews were recorded in order for the research team to have a full description of what was said [14]. Each interview was transcribed and the transcriptions were shared among the researchers to allow for further elaboration on the empirical material. In addition to the interviews, documentation review and field notes were complementary data collection methods, including software development documents, project management documents, and corporate websites and brochures. The four companies involved in our study are described below:

Company A is involved in developing systems for military defense and civil security. The systems focus on surveillance, threat detection, force protection, and avionics systems. Internally, the company is organized in different departments with systems engineering (SE) and quality assurance (QA) being the two departments included in this study. In relation to the model presented in Fig. 2.1, this company is best described as a company doing traditional development but moving towards an agile R&D organization.

Company B is an equipment manufacturer developing, manufacturing, and selling a variety of products within the embedded systems domain. The company structure is highly distributed with globally distributed development teams. Also, suppliers do a significant part of the development. In relation to the model presented in Fig. 2.1, this company is described as a company close to continuous integration. Still, parts of the organization are traditional, but there are a number of teams that operate in a highly agile manner and that have continuous integration in place.

Company C is a manufacturer and supplier of transport solutions for commercial use. Similar to company B, the development organization is largely dependent on supplier organizations. In relation to the model presented in Fig. 2.1, this company can be described as a company with parts of its development organization

being traditional and parts of it being highly agile and with continuous integration practices in place.

Company D is a provider of telecommunication systems and equipment, communications networks, and multimedia solutions for mobile and fixed network operators. The organization is highly distributed with globally distributed development. In relation to the model presented in Fig. 2.1, this company is described as a company with established practices for continuous integration and with initiatives to continuous deployment in place.

2.4 Findings and Analysis

In this section we present our case study findings and analysis. In our presentation, we outline the challenges our interviewees experience when transitioning between each of the steps in the "Stairway to Heaven" (Fig. 2.1). In our analysis, we use the ESAO framework (Fig. 2.2) and the dimensions of (1) ecosystem strategy, (2) ecosystem architecture, and (3) ecosystem organizing, to describe each transition and what actions the companies need to take to address these challenges.

2.4.1 From Traditional to Agile R&D

All companies involved in our study are transitioning towards agile development practices. Either small parts of the organization are agile or, as in some companies, the majority of the development teams work according to agile practices. One of the challenges that the companies experience when transitioning towards agile practices is that current processes and ways of working are often difficult to align with the agile parts of the organization. Also, while internal processes and structures are agile, it is not necessarily so that external stakeholders, such as suppliers and customers, transition towards agile practices. To be part of a larger business ecosystem has implications on processes as well as tools, and this is something that all companies experience as a true challenge when going agile. Another challenge is the business model that is usually traditional in character and based on yearly releases and long development cycles. This gives a conservative impression with expectations set upfront rather than being flexible and responsive towards evolving customer needs.

Based on the experiences from our study, we outline actions companies need to take when transitioning from traditional to agile development. To do this, we use the ESAO ecosystem dimensions [6] emphasizing the importance of aligning internal company dimensions with external ecosystem dimensions.

Ecosystem Strategy: The business strategy is critical when establishing a culture and support for a new development approach. Our study emphasizes the importance of introducing agile development as an approach that allows for new business

opportunities in terms of frequent releases, shorter time-to-market, and close collaboration to customers. While this creates new opportunities in terms of business value, it might upset other stakeholders in the ecosystem. Therefore, companies need to strategically position themselves to maximize revenue while at the same time contribute to the business ecosystem.

Ecosystem Architecture: When transitioning towards agile development, an important initiative is to make sure that feature teams are supported by an architect who "safeguards" the team and the integrity of the architecture. Our study emphasizes the importance of having appointed architects that work closely with the development teams and help in protecting the architecture. Also, these architects play an important role in aligning the internal architecture with the dynamics of an external ecosystem.

Ecosystem Organizing: The introduction of agile working practices is important and requires strong managerial support. Agile development implies small, cross-functional teams working on parts of the functionality. One important initiative when moving towards agile development is therefore to have mechanisms for team formation and for teams to empower and self-direct themselves. Also, renegotiating supplier contracts to facilitate for agile development might be necessary and, if so, needs to be a highly prioritized activity.

2.4.2 From Agile to Continuous Integration

Several of the companies involved in this study have continuous integration practices in place in at least parts of the organization. While the adoption of automated tests is diverse, it is seen as the way forward and as one highly prioritized activity to achieve more frequent delivery of software. According to our interviewees, there are a number of challenges to address when transitioning from agile to continuous integration. First, the dependency to other ecosystem stakeholders such as suppliers makes development complex, and most interviewees experience this dependency as having a negative effect on development speed. In addition, several of the interviewees mention that fitting different components from different suppliers takes time, so it is not only the development lead-time that is long but also the integration of components that is time-consuming and costly. One challenge that is often mentioned is the dependency between components and component interfaces. This makes modularization difficult and hence, development teams are highly dependent on each other. Another challenge is the testing activities. All companies emphasize the need for automatic testing and daily builds, while at the same time finding this is difficult in an embedded system involving hardware with slow development cycles.

Based on the experiences from our study, we outline actions companies need to take when transitioning from agile to continuous integration. To do this, we use the ESAO ecosystem dimensions [6] emphasizing the importance of aligning internal company dimensions with external ecosystem dimensions.

Ecosystem Strategy: When transitioning to continuous integration, the business perspective needs to shift from the "milestone perspective" with yearly releases to a perspective in which delivery and release are viewed as continuous activities and where there is always a shippable product. This requires support, not only from the internal business models but also from other stakeholders in the business ecosystem, such as external suppliers as well as customers.

Ecosystem Architecture: To reap the benefits of continuous integration, development needs to be modularized into smaller units, i.e., the build process needs to be shortened so that tests can be run more frequently on particular parts of the system. Our study shows that development lead-time is efficiently reduced once the architectural concerns are modularized. This allows for more frequent deliveries, the opportunity to learn more frequently from feedback, but also for efficient decoupling of dependencies to other ecosystem stakeholders.

Ecosystem Organizing: The transition towards continuous integration requires a cultural shift and a transformation of previous traditions and values. Companies need to organize themselves to allow for short cycles between the development organization and validation and verification. To align these two is critical to benefit from daily builds and frequent tests. Also, companies need to adopt test-driven development practices and processes that support automated tests. In order to reduce complexity, companies should strive for a process in which code is checked into one main development branch, i.e., the production line, to avoid having several branches since that will only add to complexity and lead-time.

2.4.3 From Continuous Integration to Continuous Deployment

In several of the companies involved in this study, continuous integration is a well-established practice, and there are attempts to involve proactive customers in continuous deployment of software functionality. However, there are a number of barriers that need to be addressed to succeed in this endeavor. What is most evident from our interviews is the complexity that arises in different network configurations at customer sites. While the product has its standard configurations, there are always customized solutions as well as local configurations that add complexity. In a business ecosystem involving a large number of stakeholders, this becomes a major task to manage for the company deploying the software. A common challenge is when a customer wants a new feature but has an old version of the product to which this new feature has to be configured. From the interviews it is clear that customers still regard upgrades and new features as a challenge due to the risk of interfering with legacy. Furthermore, the internal verification loop needs to be significantly shortened to not only develop functionality fast but to also deploy it as fast at customer site.

In the transition towards continuous deployment, all corporate functions need to be involved. Similar with introducing agile practices to different parts of the organization as the very first step when moving from traditional development to agile development, the transition towards continuous deployment requires involvement of different organizational units in order to fully succeed. Especially, product management needs to be involved as they are the interface towards customers. Finally, finding a proactive customer who is willing to explore the concept of continuous deployment is found critical.

Based on the experiences from our study, we outline actions companies need to take when transitioning from continuous integration towards continuous deployment. To do this, we use the ESAO ecosystem dimensions [6] emphasizing the importance of aligning internal company dimensions with external ecosystem dimensions.

Ecosystem Strategy: When moving towards continuous deployment of software, companies need to identify a proactive customer within the ecosystem who is willing to explore the concept. When having identified a "lead customer," the development organization can start building a continuous deployment culture and capability, in which fast customer feedback serves as direct input to improved software functionality. The lead customer serves as a role model to other customers in the ecosystem and benefits from the opportunity to get new software functionality as soon as the development organization has something to deliver. To achieve this, companies need to adapt their business model and strategy to support continuous deployment of functionality. Also, continuous deployment of software will allow companies to move closer to its customers, and therefore, current relationships and dependencies within the ecosystem need to be strategically reconfigured.

Ecosystem Architecture: When deploying software on a continuous basis, companies need efficient mechanisms to roll back unsuccessful deployments and mechanisms to deploy parts of the system rather than the entire system. In terms of architecture, complexity arises when different customers have different network configurations to which software is deployed. While the product has its standard configurations, there are always customized solutions as well as local configurations that cannot be fully accounted for. Therefore, companies need to carefully align the internal architecture with the ecosystem of which it is part.

Ecosystem Organizing: To succeed with continuous deployment, all corporate functions need to work in short cycles and with the intention to deliver smaller features more frequently to customers. This requires that not only the development and testing organizations work in short cycles but that product management and release, as well as customers, engage in this. Similar with introducing agile practices, the transition towards continuous deployment requires involvement of different organizational units in order to fully succeed. In addition, external ecosystem stakeholders are affected which requires companies to manage increased complexity and competitiveness.

2.4.4 From Continuous Deployment to R&D as an "Innovation System"

In one of the companies involved in this study, agile processes have been around for several years, and they have become widespread within the company. A large part of the organization is familiar with continuous integration, and the company is pushing towards continuous deployment for parts of the product and for a selected segment of its customers. The fast feedback loop to customers is regarded as the major benefit. According to the company, faster feedback means cheaper development since the development organization can spend time on developing things customers want instead of correcting mistakes in functionality that is not necessarily what the customer asked for. In advancing towards R&D as an "innovation system," the company foresees that their product needs to be instrumented so that customer data can be automatically collected from the installed product base. To identify relevant metrics is a highly prioritized activity in order to collect data that will work as a basis for product improvements and new feature development. Second, the company needs to develop the capability to effectively use the collected data to test new ideas with customers. In the transition towards R&D as an innovation system, the company will move closer to its customers. This might upset other stakeholders in the ecosystem who have previously been the ones interacting with customers. To carefully manage this tension, while at the same time generate business revenue, is a challenge identified as critical for successful co-creation of customer value.

Based on the experiences from our study, we outline actions companies need to take when transitioning from continuous deployment to R&D as an "innovation system." To do this, we use the ESAO ecosystem dimensions [6] emphasizing the importance of aligning internal company dimensions with external ecosystem dimensions.

Ecosystem Strategy: To have R&D as an innovation system that responds directly based on customer feedback, companies need to establish a strategy and culture in which they continuously innovate in synergy with its customers. Instead of having requirements frozen before development starts, this approach allows for evolving requirements once the system is taken into use by its customers. To achieve this, strategic collection and analysis of customer data are critical, as well as mechanisms that allow for quick response to customers. Also, business and pricing models need to support short-cycle and customer data-driven innovation processes in the ecosystem.

Ecosystem Architecture: To support a viable ecosystem architecture, infrastructures need to be established that support run-time variation of functionality. This is required to allow for continuous innovation and experimentation with customers. For example, the architecture needs to support A/B testing which is one technique that is applied by companies when experimenting with customers. Also, instrumentation of code for data collection purposes is necessary which usually requires an extension of current architectures.

Ecosystem Organizing: For achieving short cycles and rapid response to customer feedback, requirements, development, validation, and release functions need to work together. Initiatives that facilitate for aligning and integrating internal and external corporate functions need to be prioritized. Companies need to develop the capability to effectively use data collected from other ecosystem stakeholders in order to improve current system versions, develop new functionality, and innovate entirely new products. This requires careful collaboration with other ecosystem stakeholders and an organization that allows external contributions.

Conclusions
In this study, we explore how software development companies evolve their practices over time. Based on a conceptual model presented as the "Stairway to Heaven," we present the transition process when moving from traditional development towards continuous deployment of software.

Based on our analysis, and in accordance with previous research, we see that the transition towards agile development requires a careful introduction of agile practices into the organization, a shift to small development teams, and a focus on features rather than components [16]. The transition towards continuous integration requires an automated test suite, a main branch to which code is continually delivered, and a modularized architecture [9]. The move towards continuous deployment requires internal and external stakeholders to be fully involved and a proactive customer with whom to explore the concept [9]. Finally, the transition towards R&D as an innovation system requires careful ecosystem management in order to align internal business strategies with the dynamics of a competitive business ecosystem [6, 11]. Characteristic for all transitions is the critical alignment of internal and external processes in order to fully maximize the benefits as provided by the business ecosystem of which a company is part.

References

1. Moore, J.F.: Predators and prey: a new ecology of competition. Harv. Bus. Rev. **71**(3), 75–86 (1993)
2. Iansiti, M., Levien, R.: Strategy as ecology. Harv. Bus. Rev. **82**(9), 69–78 (2004)
3. Fogelström, N.D., Gorschek, T., Svahnberg, M., Olsson, P.: The impact of agile principles on market-driven software product development. J. Softw. Maintenance Evol.: Res. Pract. **22**, 53–80 (2010)
4. Williams, L., Cockburn, A.: Agile software development: it's about feedback and change. Computer **36**(6), 39–43 (2003)
5. Olsson, H.H., Alahyari, H., Bosch, J.: Climbing the "Stairway to Heaven"—A multiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. Software Engineering and Advanced Applications (SEAA), 38th EUROMICRO conference, pp. 392–399. IEEE Press (2012)

6. Bosch, J., Bosch-Sijtsema, P.: ESAO: A holistic ecosystem-driven analysis model. In: Proceedings of the 5th International Conference on Software Business (ICSOB), Cyprus, 15–18 June 2014
7. Sommerville, I.: Software Engineering, 6th edn. Pearson Education, Essex (2001)
8. Highsmith, J., Cockburn, A.: Agile software development: the business of innovation. Computer **34**(9), 120–127 (2001)
9. Humble, J., Farley, D.: Continuous Delivery: Reliable Software Releases through Build, Test and Deployment Automation. Addison-Wesley, Boston (2011)
10. Shalloway, A., Trott, J.R., Beaver, G.: Lean-Agile Software Development: Achieving Enterprise Agility. Addison-Wesley, Boston, MA (2009)
11. Bosch, J.: Building products as innovation experiment systems. In: Proceedings of the 3rd International Conference on Software Business (ICSOB), MIT, Cambridge, 18–20 June 2012
12. Hofmeister, C., Kruchten, P., Nord, R.L., Obbink, H., Ran, A., America, P.: A general model of software architecture design derived from five industrial approaches. J. Syst. Softw. **80**(1), 106–126 (2007)
13. Van der Linden, F., Bosch, J., Kamsteries, E., Kansala, K., Obbink, H.: Software product family evaluation. In: 3rd International Conference of Software Product Lines, SPLC 2004, pp. 110–129. Springer, Boston (2004)
14. Walsham, G.: Interpretive case studies in IS research: nature and method. Eur. J. Inf. Syst. **4**, 74–81 (1995)
15. Runesson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. **14**, 131–164 (2009)
16. Larman, C., Vodde, B.: Scaling Lean and Agile Development: Thinking and Organizational Tools for Large-Scale Scrum. Pearson Education, Boston (2009)

http://www.springer.com/978-3-319-11282-4

Continuous Software Engineering
Bosch, J. (Ed.)
2014, XIV, 226 p. 50 illus., 30 illus. in color., Hardcover
ISBN: 978-3-319-11282-4