Jan Bosch · Jan Carlson
Helena Holmström Olsson
Kristian Sandahl
Miroslaw Staron *Editors*

Accelerating Digital Transformation

10 Years of Software Center

Springer

Accelerating Digital Transformation

Jan Bosch • Jan Carlson • Helena Holmström Olsson • Kristian Sandahl • Miroslaw Staron
Editors

Accelerating Digital Transformation

10 Years of Software Center

Springer

Editors
Jan Bosch
Department of Computer Science
and Engineering
Chalmers University of Technology
Gothenburg, Sweden

Jan Carlson
Division of Computer Science and Networks
Mälardalen University
Västerås, Sweden

Helena Holmström Olsson
Department of Computer Science
Malmö University
Malmö, Sweden

Kristian Sandahl
Department of Computer and Information
Science (IDA)
Linköping University
Linköping, Sweden

Miroslaw Staron
Department of Computer Science
and Engineering
Chalmers University of Technology
Gothenburg, Sweden

ISBN 978-3-031-10872-3 ISBN 978-3-031-10873-0 (eBook)
https://doi.org/10.1007/978-3-031-10873-0

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword

Effective collaboration between industry and academia is hard. This is the case because of a mismatch in incentives between researchers and industry practitioners that is quite significant and causes natural tension points. Industry practitioners, in the end, want to turn the fruits of the collaboration into benefits for the business. Researchers are interested in high-profile publications. Industry often seeks to achieve these benefits as fast as possible whereas researchers often work on a time line that is much more relaxed. Industry wants access to experts in the field whereas researchers often use collaboration as an education and training context for PhD students and early career staff.

For collaboration between multiple companies and universities to be successful, a carefully designed set of mechanisms needs to be put in place to ensure that the incentives for companies and researchers are as aligned as possible. Over the last decade, Software Center has evolved a set of mechanisms that allow the collaboration between the member companies and the universities to not only survive but positively thrive. These mechanisms include the sprint model where all activities have a 6-month sprint heartbeat, a mechanism where it is the companies, rather than the researchers, that prioritize and fund the research projects, an active and engaged steering committee, and several types of recurring events that allow for frequent dissemination of results as well as constructive engagement around topics of interest.

Also, Software Center has been able to predict and proactively address topics that have become important for the member companies before the need at the companies was there. At the start, the center predominantly focused on software engineering challenges, and over the last 10 years, data and artificial intelligence as well business models and product management challenges have also been addressed.

It is the partners in the center that fund the center with the companies providing cash and the universities in-kind funding. This allows the decision processes as well as the collaboration between researchers and companies to be free from the distractions that publicly funded projects typically bring to a collaboration. Software Center has a broad and mature experience with collaborations between companies and universities. These collaborations have long been a mainstay of research and development – from creating the knowledge foundations for the next generation of

solutions, to serving as an extended "workbench" to solve short-term, incremental problems. This provides a flow of newly minted talent and allows for investigation of long-term research challenges.

This book that you are holding and for which I have the pleasure to provide the foreword captures some of the most impactful and relevant research work conducted over the last decade. However, it can truly not be viewed as a fair representation as hundreds of research articles have been published by Software Center researchers. For a full overview, I refer the reader to the Software Center website.

Over the last decade, I have had the honor to act as the chair for the Software Center steering committee. Software Center has become an essential innovation and research partner and to my understanding, the involved partners have significantly benefited from the collaborations. It has been a joy to contribute to building up such an effective collaboration between 17 companies and 5 universities, together with all the others involved in Software Center. And although the last decade was great, I expect even more from the next decade of Software Center!

Chair Software Center Steering Committee Anders Cáspar
Stockholm, April 2022

Preface

William Gibson, the famous science fiction writer, once stated that the future is already here; it just is not evenly distributed. This basically sums up the state of digitalization and digital transformation in society in one sentence. There are companies that were born digital and that have incorporated digital technologies at the very core of their business, infusing business models, ways of working, and culture with fast iterations, data-driven decision making, and machine learning models.

Most companies, however, were born in a different age, and their initial differentiation was enabled by mechanical engineering, manufacturing, or other physical capabilities. Over the years of their existence, these companies have had to adopt new technologies, mindsets, ways of working, and organizational approaches in response to the latest insights and innovations in order to stay competitive and to drive growth. For instance, many mechanical companies have adopted digital design tools and advanced simulation approaches to analyze the properties of physical designs before the first instance of the design is created. This allows for a much more systematic and structured exploration of the design space, resulting in better outcomes.

Although the majority of companies were not born digital, all of these companies have spent their entire existence focusing on incorporating and embracing new technologies and innovations. This raises the question as to why digital transformation would be different from the continuous flow of change that has existed for decades, if not centuries. Our research shows that the main reason is the scope and magnitude of the change caused by digital transformation. It literally affects everything the company is concerned with, and it represents such a significant paradigm shift that it causes stress and tension in any organization that goes through the transformation as many of the beliefs held by the employees are fundamentally broken and need to be replaced.

Although it is hard to structure a multifaceted transformation such as a digital transformation, using the BAPO model [465] we can discuss several aspects:

- **Business**: One of the key changes during a digital transformation is the change in business model. Traditional business models rooted in "atoms" tend to be transactional in nature. This means that the user buys a product, uses it for a

period of time, and then replaces the product with a new instance of the product. Digital companies tend to have continuous business models, such as subscription- or usage-based models, that automatically cause the company and the user to have a continuous relationship as well. This continuous relationship then opens up for continuous improvements to the offering which is initially a delighter but soon becomes an expectation. Once it is an expectation, companies that do not offer continuous improvements are at a distinct disadvantage.

- **Architecture**: The continuous business model, especially the ability to continuously improve the offering to the customer, has significant implications on the architecture and technology used in the product. Traditionally, many products were built with deep integration between different subsystems to optimize for bill of materials. The result was a highly interconnected design that exhibits high coupling between the various parts, which complicates updating of individual components and subsystems. This does not only affect upgradability but also other quality attributes such as testability, as it is virtually impossible to test individual components and subsystems independently. This then requires all testing to take place at the system level, which is complex, expensive, and resource consuming.
- **Process**: Although many companies have adopted agile practices at the team level, it is interesting to observe that beyond the team level, many companies still live in a waterfall world where various activities are performed sequentially rather than in parallel. As these activities are performed occasionally and not very frequently, there is little pressure to automate, and consequently significant manual effort is expended on what basically are repetitive tasks. Digital transformation and the associated continuous improvement models require much higher degrees of automation as tasks that were performed once or twice per year are now conducted every few weeks.
- **Organization**: The changes to the business model, architecture, and technology as well as the process and ways of working also has significant impact on the way companies organize themselves. Rather than functional units organized in a hierarchical organization, digitalization requires cross-functional teams operating in a highly autonomous fashion, guided by quantitative output metrics that they are asked to accomplish, rather than by first and second level line managers that tell people what to do.

The impact of digital transformations as described above only starts to scratch the surface of the required changes in organizations, but it provides an illustration of the breadth and depth of the required changes due to digitalization. Everything has to change and basically at the same time as a continuous business model and a static, immutable product do not go well together. Similarly, agile practices and yearly releases do not mix either, and organizations drop agile ways of working when there is no need to push functionality out to customers. The careful sequencing and building of capabilities as well as the transitioning from the traditional to new ways of working are challenging and many companies struggle with this.

Accelerating Digital Transformation

One of the key questions that many organizations ask themselves is how their digital transformation could be accelerated. Many organizations face competitive pressures not only from their peers in industry, but also from new entrants who are digitally born and embrace all the advantages that this offers. Although the incumbents have the customer base and the new entrants have to overcome this inertia, the threat of being disrupted by pure digital players is real in many industries. The best way to meet the threat is to undo the main advantage that new entrants have and to operate in the same way toward customers as the new entrants.

Although this may seem like a change on the interface between your organization and the customer, the effect is of course much more involved than just that. The interface with the customer includes the business model as well as the way the organization services its customers. As we discussed in the previous section, the main shift in business model typically is to move from a transactional to a continuous model so that the customer experiences a continuously improving offering. Especially for embedded systems, the business model may involve a change in ownership of the physical product itself. Instead of the customer owning the product, the customer has the access and use but is in fact not the owner of the product. These changes go far beyond the interface between the company and its customers.

It is clear that change is needed, and it needs to occur rapidly in order to protect the competitive position of the organization. Our experience is that all companies change, but the pace of change tends to be relatively slow. Organizations are inherently built to resist change, and accelerating the pace of change requires typically external forces.

The history of Software Center starts with the above realization by key R&D leaders at four large software development organizations in Gothenburg, Sweden, i.e., Ericsson, Saab Surveillance, Volvo Car Corporation, and AB Volvo. These leaders requested from the president of Chalmers University of Technology at that time that the university start a center, in part funded by these companies, to help them collaborate in building software development expertise in their respective organizations. Software Center started in 2011 with the four aforementioned companies, with Gothenburg University and Chalmers University of Technology as founding members. Over the last decade, the center has grown to 17 companies and 5 universities, and the scope of research has evolved and expanded dramatically.

Initially, the primary focus of the companies was the adoption of agile software development practices, but over the years, the research has expanded to include work on software architecture, metrics, customer data, business ecosystems, artificial intelligence, business models, digitalization, etc. As you will read in the remainder of the book, there is a multitude of topics addressed in the center—all in response to the digital transformation that the companies are increasingly involved in and pursuing.

The companies use Software Center not only for the research that inspires specific changes and improvements at the company. The other primary mechanism is that companies work on exchanging knowledge between each other, without the in-

volvement of the researchers at the universities. Over the years, we have developed a variety of mechanisms to support the knowledge exchange, research, and adoption for the center members, including our reporting workshop that runs twice per year, our YouTube channel, industrial PhD candidates, weekly brownbag seminars, a monthly newsletter, and our website to mention a few.

This Book

This book aims to celebrate the 10-year anniversary of Software Center by presenting some of the most impactful and relevant publications that the participants in the center have contributed over the last decade.

The book is organized into the five themes that research in Software Center is woven around, i.e., Continuous Delivery, Continuous Architecture, Metrics, Customer Data and Ecosystem-Driven Development, and, finally, AI Engineering.

The focus of the Continuous Delivery theme is to help companies to continuously build high-quality products with the right degree of automation. The researchers of the theme work with improving large-scale utilization of CI/CD (Continuous Integration/Continuous Delivery) in areas such as optimizing automated testing and improving the quality of the test cases and the test environment. Work is also done in finding the role of different manual testing activities and formal verification in CI/CD. Most companies already have a CI/CD tool chain in place that performs its individual functions well, but to get an overview of the process, for instance to assess the confidence about the product quality, is not trivial. One approach investigated in the theme is to use visualization techniques as a means for fulfilling the information needs. The Continuous Architecture theme addresses challenges that arise when balancing the need for architectural quality and more agile ways of working with shorter development cycles. We have, for example, investigated how architectural technical debt can be identified, managed, and prioritized and how companies can keep development artifacts consistent as the system evolves, such as code and models at different levels of abstraction. The research has also addressed how to extend agile principles from pure software to fit the development of mechatronic products, including more incremental processes for safety and security assurance.

The Metrics theme studies and provides insight to understand, monitor, and improve software processes, products, and organizations. It addresses challenges related to finding the right metrics for the right organization (e.g., quantifying productivity during organizational transformations), right product (e.g., finding the optimal feature set for a release), and process (e.g., finding the right metrics to monitor the evolution, or decay, of software architectures). The researchers in the metrics theme utilize the newest methods from machine learning, artificial intelligence, and mining software repositories to optimize test selection, identify cyber-security risks, support agile transformations, and support automotive software development.

The fourth theme, Customer Data and Ecosystem-Driven Development, helps companies make sense of the vast amounts of data that are continuously collected

from products in the field. The projects in the theme study how to effectively make use of this data, how it provides the basis for new digital offerings, and how it allows for fundamentally new ways-of-working and of delivering value to customers. The research in the theme studies how companies evolve from traditional toward digital companies and provides companies with methods, techniques, and tools that support the journey that they take when moving beyond agile ways-of-working toward continuous practices involving, e.g., agile requirements engineering, A/B testing and feature experimentation, and continuous validation of customer value.

Finally, the fifth theme, AI Engineering, addresses the challenge that many companies struggle with in terms of deploying machine- and deep-learning models in industrial contexts with production quality. This requires solutions in a variety of dimensions, ranging from monitoring and logging to building dependable data pipelines and from federated learning architectures to heterogeneous hardware.

Each theme has its own part in the book, and each part has an introduction chapter and then a carefully selected reprint of the most important papers from that theme. For the AI Engineering theme, only one chapter was included as the theme was formed quite recently.

As editors, we would like to thank all the partner companies and researchers that, over the last decade, have contributed to building up this collaboration and the amazing results. We are grateful for the last 10 years and yet want to also look forward to what Software Center has to offer going forward.

To you as a reader, we hope you enjoy and appreciate the content and that you are able to use it to your advantage!

Contents

Part I

Continuous Delivery

Introduction to the Continuous Delivery Theme

Kristian Sandahl (Theme Leader 2015–)

Even though there were many forerunners, the most widespread reference to Continuous Integration as a method was put forward by Grady Booch in 1990 [44] (page 209). In the 1995 book *Microsoft Secrets*, Cusumano and Selby interviewed 38 Microsoft employees to document how the world's leading software provider was managing its own software development [80]. One of the key practices found was the Daily Build concept. In the popular literature, this was described as everyone needed to check in their code and build the product at the end of the workday. In tight connection to the build, some smoke tests were run to ensure that the individual contributions could be integrated. If the build was broken, the person who broke it had to fix his/her code before going home. There was also modern folklore mentioning that the breaker of the build had to wear a funny hat for the remainder of the day.

The elegant marketing of daily build raised an enormous interest from industry; everyone wanted to try. I worked at Ericsson at the time and saw how daily builds were enthusiastically adopted in the teams and resources were spent to create effective test environments, some of them with SMS notifications of the outcome (which was hot those days). Many master thesis projects were initiated to make pilot studies.

The interest was amplified in the year 2000 with Beck's book on extreme programming, which further emphasized the use of automated testing and frequent updates and releases [32]. The large uptake of agile methods was started in combination with ideas about feature-oriented programming, which increased the independence and customer focus of the teams. This development put large requirements on the tool chain for compilation, build, test, and test feed-back, which was evolved into the CI/CD (Continuous Integration/Continuous Delivery) flow as we know it today. The more the teams are relieved from a one-to-one connection to individual system components, the more complicated the CI/CD flow becomes, and with that came research challenges that are the focus of research in Theme 1.

In this collection, we will start with the most-cited paper, "Climbing the Stairway to Heaven – A Multiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software" by H. H. Olsson, H. Alahyari, and J. Bosch. The chapter describes the state of continuous practices of four

J. Bosch et al. (eds.), *Accelerating Digital Transformation*,
https://doi.org/10.1007/978-3-031-10873-0_1

different companies that lead to formulating the "Stairway to Heaven" model. The model shows a typical evolution path for a company from traditional development to a state where every piece of software is continually deployed at the customer installation of the system and data from everyday use is fed back to guide the R&D organization. Barriers against climbing the stairway are identified from the case companies.

From this article, the CIViT model was created to guide companies about where in their organizations to focus short-term as well as long-term efforts to be successful in automating the CI/CD flow [338].

The second chapter in this collection, "Modeling Continuous Integration Practice Differences in Industry Software Development," by D. Ståhl and J. Bosch, is a systematic literature review of the practical implementation of CI in industry. By analyzing 46 publications, the authors formulated 22 cluster statements that can be thought of as variation points in implementing CI in practice. This contribution is an academic achievement but also a practical guide of decisions to take for the implementer of CI – a typical example of successful research that we are proud of in Software Center.

Recent research in Software Center Theme 1 helps stakeholders to find the information they need from the complex CI/CD process by utilizing data, for instance, from implementations of the Eiffel protocol [106, 423]. These implementations are centered around a unified event bus, where the different tools in the tool chain of a CI/CD pipeline produce and listen for events, thereby creating traceability between events in real time. So far, various types of visualization techniques have been the most prominent approach.

In spring 2017, Software Center companies SAAB Aeronautics, Axis Communication, Grundfos, and Ericsson identified automated testing as a critical bottleneck and joined forces to create a research agenda that was turned into the first company-initiated project proposal complemented with working packages from four universities. This has led to several research contributions, such as diversity-base testing, combinatorial testing, detection and avoidance of flaky tests, cognitive aspects of testing, scheduling of parallel test execution, and static analysis to improve test code quality.

In this collection, this line of research is represented by a recent chapter that got very high appreciation from the publisher: "Efficient and Effective Exploratory Testing of Large-Scale Software Systems" by T. Mårtensson, D. Ståhl, A. Martini, and J. Bosch. The authors make the point that the high degree of automation in CI/CD needs to be complemented with human testing, drawing on the human testers' curiosity and their knowledge of the end users' true needs. The chapter presents and validates the ExET model, which is an empirically founded model used to guide practitioners into the deployment of exploratory testing fulfilling nine key factors found in successful testing processes. On the method pages, the chapter also describes cross-company workshops, which is one of the most common ways of working in the Software Center to gather data and exchange experience between companies.

The future is the only thing we do not know anything about, but we have probably not seen the end of data-driven validation, where the software components are

deployed to end-customers and are subject to real usage behavior from which data can be fed back to the development and test. The deployment environment is built with a fallback to earlier releases if the new components fail. This is an alternative to build a testing environment that mimics real usage, which in many cases can be difficult and expensive and can slow down the time to market.

Another ongoing, but not finished, trend is to transfer CI/CD to systems engineering where hardware and environmental components are simulated. A parallel, continuous development of models, simulators, and software might give opportunities for increased collaboration between disciplines and to harvest the benefits from agile methods outside the software world.

For the testing part, many researchers are working on how to test AI/ML software as this type of component is becoming more integrated in industrial software. We would like to welcome well-accepted, industrially validated methods suitable for iterative development and CI/CD.

As the reader realizes, the CI/CD processes consume much computational resources and thus electrical energy that in turn has an environmental impact – in the worst case, through more emission of CO2. The contribution of information and communication technologies (ICTs) to a low carbon economy is still unclear [417]. This alone would motivate a more purposeful and exact utilization of computational resources in CI/CD and automated testing, not to mention the positive contributions in time and quality.

Chapter 1
Climbing the Stairway to Heaven

Helena Holmström Olsson, Hiva Alahyari, and Jan Bosch

Abstract Agile software development is well-known for its focus on close customer collaboration and customer feedback. In emphasizing flexibility, efficiency and speed, agile practices have led to a paradigm shift in how software is developed. However, while agile practices have succeeded in involving the customer in the development cycle, there is an urgent need to learn from customer usage of software also after delivering and deployment of the software product. The concept of continuous deployment, i.e. the ability to deliver software functionality frequently to customers and subsequently, the ability to continuously learn from real-time customer usage of software, has become attractive to companies realizing the potential in having even shorter feedback loops. However, the transition towards continuous deployment involves a number of barriers. This paper presents a multiple-case study in which we explore barriers associated with the transition towards continuous deployment. Based on interviews at four different software development companies we present key barriers in this transition as well as actions that need to be taken to address these.

1.1 Introduction

Today, software development is conducted in increasingly turbulent business environments. Typically, fast-changing and unpredictable markets, complex and changing customer requirements, pressures of shorter time-to-market, and rapidly advancing information technologies are characteristics found in most software development projects. To address this situation, agile practices advocating flexibility, efficiency and speed are seen as increasingly attractive by software development companies [102]. In emphasizing the use of iterations and development of small features,

Reprinted with permission from IEEE. Originally published in 38th Euromicro Conference on Software Engineering and Advanced Applications, 2012 DOI: https://doi.org/10.1109/SEAA.2012.54

J. Bosch et al. (eds.), *Accelerating Digital Transformation*,
https://doi.org/10.1007/978-3-031-10873-0_2

agile practices have increased the ability for software development companies to accommodate fast changing customer requirements and fluctuating market needs [102, 480].

However, while many software development companies have indeed succeeded in adopting agile practices in parts of their organization, there are few examples of companies that have succeeded in implementing agile practices to such an extent that software functionality can be continuously deployed at customer sites so that customer feedback and customer usage data can be efficiently utilized throughout development, delivery and deployment of software [404]. In order to advance the concept of agile development and move towards continuous deployment of software there are several steps that need to be taken.

In this paper, we present a multiple-case study in which we explore four companies within the IT industry moving towards continuous deployment of software. While the ability to continuously deploy new functionality at the customer site creates new business opportunities and indeed extends the concept of agile development, it also presents challenges related to current customer collaboration models. As a result of our study, we identify barriers in moving towards continuous deployment of software – as well as actions that need to be taken to overcome these.

1.2 From Agile Development to Continuous Deployment of Software

Companies evolve their software development practices over time. Typically, there is a pattern that most companies follow as their evolution path. We refer to this evolution as the "stairway to heaven" and it is presented in Fig. 1.1.

The phases of the "stairway to heaven" are discussed in more detail in the remainder of this section. As a summary, however, we see that companies evolving from traditional waterfall development (step A in Fig. 1.1) start by experimenting with one or a few agile teams. Once these teams are successful and there is positive momentum, agile practices are adopted by the R&D organization (step B in Fig. 1.1). At this point, product management and system integration and verification are still using traditional work practices. As the R&D organization starts to show the benefits of working agile, system integration and verification becomes involved and the company can adopt continuous integration where system test takes place continuously and where there is always a shippable product (step C in Fig. 1.1). Once continuous integration is up and running internally, lead customers often express an interest to receive software functionality earlier than through the normal release cycle. What they want is to be able to deploy software functionality continuously (step D in Fig. 1.1). The final step is where the software development company not only releases software continuously, but also collects data from its installed base and uses this data to drive an experiment system where new ideas are tested in segments of the installed base and the data collected from these customers is used to steer the direction of R&D efforts [45].

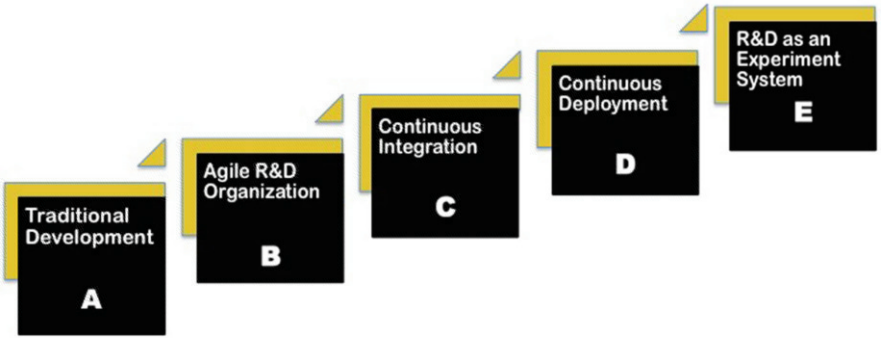

Fig. 1.1 "The stairway to heaven", i.e. the typical evolution path for companies moving towards continuous deployment of software

1.2.1 Traditional Development

In our discussion, we refer to traditional development as an approach to software development characterized by slow development cycles, e.g. yearly, waterfall-style interaction between product management, product development, system test and the customer and, finally, customer feedback processes that are not necessarily well integrated with the product development process [415]. Usually, project teams are large and competences are divided into disciplines such as system architecture, design and test. Development is sequential with a rigorous planning phase in the very beginning of each project. Typically, delivery to the customer takes place in the very end of the project and it is not until then that customers can provide feedback on the software functionality they have received.

1.2.2 Agile R&D Organization

The next step in the evolution is where product development, i.e. the R&D organization, has adopted agile practices, but where product management and system verification still work according to the traditional development model. Although some of the benefits with agile practices are reaped, there is not necessarily short feedback loops with the customer. As defined by Highsmith and Cockburn [181], a team should not be considered agile if the feedback loop with customers and management is six months or more.

1.2.3 Continuous Integration

A company employing continuous integration has succeeded in establishing practices that allow for frequent integration of work, daily builds and fast commit of changes, e.g. automated builds and automated test. Humble and Farley [196] define continuous integration as a software development practice where members of a team integrate their work frequently, leading to multiple integrations per day. The idea of automating test cases, builds, compilation, code coverage etc. allows teams to test and integrate their code on a daily basis which minimizes the time it takes from having an idea to actually implement the idea in software. At this point, both product development and system validation are working according to agile practices.

1.2.4 Continuous Deployment

At this stage, software functionality is deployed continuously, or at least more frequently, at customer site. This allows for continuous customer feedback, the ability to learn from customer usage data, i.e. real usage data, and to eliminate any work that doesn't produce value for the customer. At this point, R&D, product management as well as the customers are all involved in a rapid, agile cycle of product development.

1.2.5 R&D as an 'Experiment System'

The final step on the "stairway to heaven" is where the entire R&D system responds and acts based on instant customer feedback and where actual deployment of software functionality is seen as a way of experimenting and testing what the customer needs. At this step, deployment of software is seen more as a starting point for further 'tuning' of functionality rather than delivery of the final product.

1.2.6 Summary

In this section, we presented the typical evolution path for companies adopting agile practices. It is important to realize, however, that there are different levels at which adoption of agile practices can take place. First, and as can be seen in our study, agile practices are adopted at a team level. As will show, we found that teams are often significantly ahead of the organization as a whole, in particular up to the stage of continuous integration. Second, agile practices are adopted at an organizational level where they evolve into an institutionalized approach to software development. In our paper, we focus on the organizational level in order to identify barriers that need to be addressed by the organization to further accelerate. Finally, there is an

ecosystem perspective on the adoption of agile practices. Many companies, and as can be seen in our study, interact closely with suppliers. Although an organization may operate at a high level in the presented model, achieving the same level of agility with suppliers requires a significant effort.

In Table 1.1, we focus on the organizational level. We summarize the involvement of each function in the organization, as well as the customer's role. The 'approach' column refers to the different steps in the "stairway to heaven" (Fig. 1.1).'T' stands for traditional, 'A' for agile and 'SC' for short cycle. The table illustrates the transition towards continuous deployment and a situation, if looking at the final step (E), in which all functions involved enjoy short feedback cycles and hence, the opportunity to experiment with an aim to continuously learn and improve.

Table 1.1 Summary of each step in the "stairway to heaven" (A-E) and how each organizational function (PM, R&D, Validation and Customer) work at this particular step

Approach	PM	R&D	Validation	Customer
A	T	T	T	T
B	T	A	T	T
C	T	SC	SC	T
D	A	SC	SC	A
E	SC	SC	SC	SC

1.3 Research Approach

1.3.1 Research Sites

This paper reports on a multiple-case study involving four software development companies. All four companies are moving towards continuous deployment of software. In representing different stages in this process, we find these companies of particular interest for understanding the barriers that need to be addressed when moving towards continuous deployment of software. Below, the characteristics of each company are presented as well as their current mode of development, i.e. at what step in the model (Fig. 1.1) we place them.

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. Often, the system solutions are developed through the use of microwave and antenna technology. 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. 1.1, this company is best described as a company doing traditional development but

moving towards an agile R&D organization. Already, there are a few pro-active agile teams that work as inspiration for the rest of the organization and the attitude towards agile practices and continuous deployment of software is positive.

Company B is an equipment manufacturer developing, manufacturing and selling a variety of products within the embedded systems domain. The products contain software running on micro-controllers that are connected via computer networks. The company structure is highly distributed with globally distributed development teams. Also, much of the development is done by suppliers. In relation to the model presented in Fig. 1.1, this company can be described as a company that is close to continuous integration. While parts of the organization are still to a large extent traditional and plan-driven, there are a number of pro-active teams that operate in a highly agile manner and continuous integration is already in place in between these teams.

Company C is a manufacturer and supplier of transport solutions for commercial use. The development organization involves coordination of a large number of teams distributed both nationally and internationally. Similar to company B, the development organization is largely dependent on supplier organizations. In relation to the model presented in Fig. 1.1, this company can be described as a company with parts of its R&D organization being traditional and parts of it being highly agile. In similar with company B, this company has continuous integration in place for some of the teams and the experience from these is used to pro-actively coach other parts of the organization in its transition towards continuous deployment.

Company D is a provider of telecommunication systems and equipment, communications networks and multimedia solutions for mobile and fixed network operators. They offer end-to-end solutions for mobile communication and they develop telecommunication infrastructure components for a global market. The organization is highly distributed with globally distributed development and customer teams. In relation to the model presented in Fig. 1.1, this company can be described as a company with established practices for continuous integration and with existing attempts to continuous deployment in place. During our study we could see that this company is very close to continuous deployment of software, making their experiences valuable for other companies trying to address the barriers that are present when moving towards continuous deployment of software.

1.3.2 Research Method

This study builds on a 6 months (July 2011–December 2011) multiple-case study and adopts an interpretive research approach [472]. It emphasizes software development as enacted by people with different values, different expectations and different strategies, as a result of their different frames of interpretation [470]. These frames act as filters enabling people to perceive some things but ignore others [317]. In particular, case study research is considered appropriate to investigate real-life contexts, such as for example software development, where control over the context is

not possible [491] and where there is an interest in accessing people's interpretations and expectations in order to create a rich understanding of a particular context [472]. In our study, a multiple-case design is used to ensure that "the events and processes in one well-described setting are not wholly idiosyncratic" [325]. In our study, the four case companies all represent different contexts with different prerequisites. However, they all share the same vision i.e. to enable customer feedback and customer usage data to feed into frequent delivery of software functionality and hence, move closer towards continuous deployment of software. In this way, they all represent interesting examples that well reflect our attempt to better understand the barriers that software development companies face when moving towards continuous deployment.

1.4 Data Collection and Analysis

The main data collection method used in this study is semi-structured interviews with open-ended questions [384]. When doing interviews, there is the delicate task in balancing between passivity and over-direction of interviews [472]. In our study, we chose to have an interview protocol organized in four pre-defined themes, but allow for openness and flexibility within these themes. The themes were (1) current way of working, (2) current customer interaction mechanisms/models, (3) strengths/weaknesses in current way of working, and finally a theme related to (4) an imagined future of continuous deployment and the barriers to get there. In total, 18 interviews were conducted. In company 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 company 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 one hour. During the interviews, we were two researchers sharing the responsibility, i.e. one of us asked the questions and one took notes. In this way, we had the opportunity to discuss the data after each interview and to compare our different insights. In addition to the notes, all interviews were recorded in order for the research team to have a full description of what was said [472]. Each interview was transcribed and the transcriptions were shared among the three 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. Also, e-mail correspondence was used as a follow-up to all interviews in order to clarify any misunderstandings in the transcription of interviews.

1.5 Validity and Generalizability of Results

As noted by [228], qualitative researchers rarely have the benefit of previously planned comparisons, sampling strategies, or statistical manipulations that control for possible threats. Instead, qualitative researchers must try to rule out validity threats after the research has begun by using evidence collected during the research itself to make alternative hypotheses or interpretations implausible. One important aspect of validity is construct validity [384], which reflects to what extent the operational measures that are studied represent what the researchers have in mind, and what is reflected in the interview questions. To address this aspect, we started each interview with an introduction in which we shared our understanding of 'continuous deployment' with the interviewee. In this way, the researchers and the interviewee had a shared understanding of the concept already before the interview. With respect to external validity, i.e. to what extent it is possible to generalize the findings, our contribution is related to (1) the drawing of specific implications and (2) the contribution of rich insight [472]. Based on our interviews, we present findings and implications in a particular domain of action. While these implications should be regarded as tendencies rather than predictions [472], they might indeed prove useful for other similar organizations and contexts. In relation to rich insight, our study brings together four empirical contexts that allow for a broad understanding of the concept of 'continuous deployment'. Our study aims at capturing the typical evolution path for software companies moving towards continuous deployment, and the findings we present should be regarded as insights valuable for other companies interested in this evolution.

1.6 Case Study Findings

In this section we present the interview findings from each company that was involved in our study. In particular, we present the barriers that each company experiences when climbing from one step to the next in the "stairway to heaven" (see Fig. 1.1).

1.6.1 Company A

As mentioned earlier in this paper, company A is characterized as a company doing traditional development but with a strong interest in agile practices. In our interviews, we learnt that company A has teams that are agile in nature and that these are used as inspiration for the larger organization. In our study, company A represents a company starting its journey way towards more agile practices and there are still many steps to take. While the attitude among the interviewees is positive, and there is anticipation on the benefits that agile practices will bring, there are a number of barriers that need to be addressed. One of the major barriers is the lack of a base

product on which improvements can be continually done. This is reflected in the following quote by one of the software developers: *"I think we could deliver better products if we had a better way of working with our products...we do not continually work with improving the products. We only work with the products when we have a customer and a customer project"*.

Another barrier is the current way of working which is sometimes insufficient when it comes to process. This is expressed by one of the configuration managers: *"Sometimes we lack a proper process on how to deploy the builds to our internal systems...we do not really have a process on how to collaborate and exchange information in between teams"*. The common opinion is also that there is a need for automating the process to a larger extent than is done today. In addition to difficulties associated with the current way of working, old tools make work unnecessarily difficult and while people agree on that writing commands sometimes has its advantages, a nice graphical interface would make work easier and more efficient. The barrier with having old tools is described by one of the software developers: *"It is the software, the tools, the installation engine...it is 15 years or so...we don't have any fancy tools"*. Finally, the business model is seen as a barrier itself as it gives a conservative impression with expectations set up front rather than being flexible and responsive towards emerging needs.

To summarize, company A experiences several barriers when moving from traditional development (step A in Fig. 1.1) towards agile development (step B in Fig. 1.1). These barriers are (1) the lack of a base product prohibiting continual improvement, (2) an insufficient process, (3) old tools, and (4) the current conservative business model.

1.6.2 Company B

Company B has a number of agile teams and the organization is moving towards continuous integration. While our interviewees describe a number of initiatives supporting this move, they agree that there is much to be done in order to enhance flexibility and encourage more frequent delivery. One of the software architects reflects on this when saying: *"If you want to change something it is difficult...you need to go through several steps involving several people and this will have a long lead time"*. According to our interviewees, there are a number of barriers that need to be addressed when transitioning from having agile teams to also have continuous integration in place. First, the company is depending on its many suppliers, a situation that makes development complex. *" Everything gets harder to do when you buy it from a supplier...our process depends on their process...some suppliers are fast and some are slow, but in general they have a negative effect on development speed"* (software developer). Accordingly, one of the project leaders mentions that: *"One thing that could indeed be improved is the communication with suppliers...it is sometimes slow"*. 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.

Another barrier which is mentioned by all interviewees is the fact that the company is still very hardware oriented in character and profile. There is a great deal of experience in hardware but not so much in software. One of the project leaders reflects on this when saying: *"We have experience in hardware, but our experience in software is not so rigorous. To some extent we are still a mechanic company and not a software company but we are slowly changing due to all software that is needed in our mechanical products"*. In similar, one of the software developers touches upon this: *"We are moving away from being a mechanical company to being more of a software company, but we still have many of the systems and processes from the mechanical part. In the mechanical part you do not update hardware each day...therefore, the lead-time that we are used to is slow and our background systems and processes are slow"*. In relation to the traditions and the experience within the company, the current ways of working are sometimes seen as problematic. One barrier that is often mentioned is the dependency between components and the dependency between component interfaces. This makes separation difficult and hence, development teams are highly dependent on each other. Furthermore, the interpretation of the current development process is different at different sites – a situation that makes it even more difficult to separate deliveries in a way that all involved parties agree with.

Another common barrier is the testing activities. For example, one developer emphasizes the need for automatic testing while at the same time realizing that this is difficult in an embedded system involving hardware with slow development cycles. Finally, the broad variety of tools is a major barrier, a view that is shared among all the interviewees: *"I think one of the major weaknesses is that we have too many tools... we change tools all the time and learning a new tool is like learning a new language. If you have too many tools and updates it is like learning a new language with a new dictionary all the time"* (developer). The tool issue also brings problems that might affect the testing activities mentioned above. One of the developers stress this when saying: *"The tools are sometimes not mature enough to fill the purpose they were assigned to...sometimes they introduce so much problems that we have to deal with...you get uncertain with how they work and this means doing much more tests"*.

To summarize, company B experiences several barriers when moving from agile development (step B in Fig. 1.1) towards continuous integration (step C in Fig. 1.1). These barriers are (1) communication and coordination with suppliers, (2) company tradition of being hardware oriented, (3) component dependencies, (4) interpretation of the current process model, (5) testing activities, and (6) the broad variety of tools.

1.6.3 Company C

Several teams at company C work in an agile way and there are a number of teams that have established practices for continuous integration. As an organization, company C is moving towards continuous integration and there are several initiatives

supporting this. However, to make this transition there are a number of barriers to address. First, the dependency to suppliers is something that the interviewees find troublesome. *"The projects become complex due to the many suppliers...no development is really going on inside the company but instead we have to coordinate and integrate components from our suppliers"* (system integrator). The complexity is highlighted further by one of the software developers: *"...then of course if you want to come to a situation in which you work in shorter loops, then all suppliers must also embrace this and see that it is good for them to have short loops and to abandon the waterfall projects. But as long as one supplier remains in the old paradigm then...yeah, they are a big part of the challenge"*. Moreover, the interviewees find it difficult to be flexible and work more agile in a world which is predetermined by fixed price models and where the business model can be difficult to adjust. One of the developers reflects on this: *"People try to work agile and to stay flexible but this is very difficult in a world which is predetermined due to economics ruling"*. As it seems, the difficulty is not only to navigate in the network of suppliers but also to adjust to the current business model.

One major barrier at company C is the difficulty in getting an overview of the status of the projects. According to one of the software developers: *"...the projects are in our mailboxes and it is very hard to get a picture of the status of the different projects"*. In general, the feeling is that the daily work could be much more efficient if only people made use of the tools that are available for this. Also, there is the need for a connection in between the different systems so that information doesn't have to be pushed out as is done today. Rather, our interviewees would like to see a situation in which information is transparent and in which it was available for anyone that needs it.

In similar with company B, company C is dependent on hardware and many of its processes are adjusted in accordance with the hardware platforms. According to one of the developers this sometimes causes problems: *"I think a problem is that we are still focused on the hardware part of the product. We build our product [the hardware part] and for this we have some tools...this is a slow process compared to how fast you can change software"*. The interviewees agree that in order to change this they need a better hardware platform then what is available today. One of the developers emphasizes this when saying: *"We need a more stable and commoditized hardware platform in order to move towards a more 'software-way-of-working"*. What seems to be a common opinion is that when functionality of the product is distributed between both hardware and software, the hardware part cannot be ignored or viewed as a computing resource only. At the same time, they agree that the overall process cannot be too heavily tailored to fit the mechanical development process. Finally, all interviewees mention the test process as critical for the ability to move towards continuous deployment. To get more confident in the test suite and to be able to automate tests will be important as well as increasing the number of people that are dedicated to software testing.

To summarize, company C experiences several barriers when moving from agile (step B in Fig. 1.1) towards continuous integration (step C in Fig. 1.1). These barriers

are (1) dependency on suppliers, (2) current business model, (3) lack of transparency, (4) hardware oriented mindset and process and, (5) the test process.

1.6.4 Company D

At company D, 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 at least parts of the product and for a segment of its customers. Here, the faster feedback loop to customers is regarded the major benefit. With continuous deployment customers get releases more often and software features can be deployed at customer site on a more frequent basis than today. According to one of the release program managers faster feedback means cheaper development since the R&D organization can then spend time on developing the right things rather than correcting mistakes in functionality that is not necessarily what the customer wants.

However, in order to move further towards continuous deployment of software, there are a number of barriers to address. The interviewees at company D all mention the complexity of the network and the many different configurations that their customers have. A very 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. Similarly, an upgrade of any kind is considered stressful by customers, something that is highlighted by the release manager: *"it is more difficult to guarantee minimal network impact if the configuration of the product is complex"*. 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. Another barrier is the internal verification loop which needs to be shortened and automated in order to meet up with the requirements that continuous deployment raises. As mentioned by one of the product line maintenance managers, more automated tests are needed in order to increase speed and frequency of delivery. Also, several interviewees highlight the importance of improving the quality on each build and to increase awareness on what effect each build has on the overall software package. In this, the teams would benefit from knowing more about the quality status of the development projects, i.e. the current quality of features, the number of errors etc. If such knowledge could be better established, teams could respond faster and act more pro-actively towards customers.

To summarize, company D experiences several barriers when moving from continuous integration (step C in Fig. 1.1) towards continuous deployment (step D in Fig. 1.1). These barriers are (1) network configuration and upgrade complexity, (2) internal verification loop, and (3) quality status of builds/systems.