Climbing the "stairway to heaven" - A mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software

Article · January 2012

CITATIONS
74

READS
844

3 authors:

Helena Holmstrom Olsson
Malmö University
83 PUBLICATIONS **1,458** CITATIONS
SEE PROFILE

Hiva Alahyari
Chalmers University of Technology
10 PUBLICATIONS **225** CITATIONS
SEE PROFILE

Jan Bosch
Chalmers University of Technology
421 PUBLICATIONS **12,475** CITATIONS
SEE PROFILE

2012 38th Euromicro Conference on Software Engineering and Advanced Applications

Climbing the "Stairway to Heaven"

A multiple-case study exploring barriers in the transition from agile development towards continuous deployment of software

Helena Holmström Olsson, Hiva Alahyari and Jan Bosch
Dept. of Computer Science and Engineering
University of Gothenburg/Chalmers
Gothenburg, Sweden
e-mail: [helena.holmstrom.olsson@gu.se], [hiva.alahyari | jan.bosch]@chalmers.se

***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 lead 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.**

Keywords-agile software development; customer collaboration; continuous integration; continuous deployment

I. 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 [1]. In emphasizing the use of iterations and development of small features, agile practices have increased the ability for software development companies to accommodate fast changing customer requirements and fluctuating market needs [1,2].

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 [3]. 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.

The remainder of this paper is organized as follows. In the next section, we present the conceptual model that we use as a basis for our analysis. Section III discusses our research approach followed by a section presenting our case study findings. Section V presents our key lessons learned, followed by the conclusions.

II. 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 Figure 1 below.

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 Figure 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 Figure 1). At this point, product management and system integration and verification are still using traditional work

978-0-7695-4790-9/12 $26.00 © 2012 IEEE
DOI 10.1109/SEAA.2012.54

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 Figure 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 Figure 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 [4].

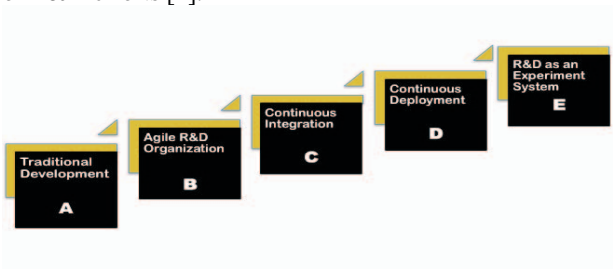

Figure 1. "The stairway to heaven", i.e. the typical evolution path for companies moving towards continuous deployment of software.

A. 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 [5]. 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.

B. 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 [6], a team should not be considered agile if the feedback loop with customers and management is six months or more.

C. 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 [7] 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.

D. 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.

E. 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.

F. 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 the table below, 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" (Figure 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.

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

Table 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.

III. RESEARCH APPROACH

A. 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 (Figure 1) we place them.

1) Company A

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 Figure 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.

2) Company B

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 Figure 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.

3) Company C

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 Figure 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.

4) Company D

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 Figure 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.

B. Research method

This study builds on a 6 months (July 2011 – December 2011) multiple-case study and adopts an interpretive research approach [8]. 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 [9]. These frames act as filters enabling people to perceive some things but ignore others [10]. 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 [12] and where there is an interest in accessing people's interpretations and expectations in order to create a rich understanding of a particular context [8]. 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" [12]. 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.

C. Data collection and analysis

The main data collection method used in this study is semi-structured interviews with open-ended questions [13]. When doing interviews, there is the delicate task in balancing between passivity and over-direction of interviews [8]. 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 [8]. 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.

D. Validity and generalizability of results

As noted by Maxwell (2005), 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 [13], 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 [8]. 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 [8], 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.

IV. 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 figure 1).

A. 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 figure 1) towards agile development (step B in figure 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. These barriers will be further elaborated on in section V where we also present actions to address these.

B. 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 figure 1) towards continuous integration (step C in figure 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. These barriers will be further elaborated on in section V where we also present actions to address these.

C. 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 figure 1) towards continuous integration (step C in figure 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. These barriers will be further elaborated on in section V where we also present actions to address these.

D. 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 figure 1) towards continuous deployment (step D in figure 1). These barriers are (1) network configuration and upgrade complexity, (2) internal verification loop, and (3) quality status of builds/systems. These barriers will be further elaborated on in section V where we also present actions to address these.

V. CLIMBING THE STAIRWAY TO HEAVEN

In this section the multiple-case study findings are discussed, leading to an understanding for key barriers that companies experience when moving towards continuous deployment. We use the model in Figure 1 to identify barriers in between the different steps, i.e. from traditional development to agile, from agile development to continuous integration and, finally, from continuous integration to continuous deployment of software. Furthermore, and based on the case study findings, we identify actions that need to be taken to address these barriers and succeed in the transition towards continuous deployment of software.

A. From Traditional to Agile R&D

As can be seen in our study, there are several barriers that need to be addressed when moving from traditional development to an agile R&D organization. For example, in company A, teams struggle with an insufficient process in which collaboration and information exchange is poorly supported. This situation, in combination with the lack of a product on which improvements can be continually made and a conservative business model, in which requirements are defined upfront, makes it difficult for teams to act agile. Moreover, old tools make development unnecessary complex and the developers express a situation in which the tools restrict rather than release activity.

To address these barriers, there are a number of actions that need to be taken. As can be seen in literature, a first and challenging step is to introduce agile working practices into the organization and to get managerial support for such an initiative [14]. Company A shows on several initiatives that supports agile working practices and the general attitude is positive. Having a few agile teams that work as inspiration for the larger organization is one action that will have a positive effect on other teams. Furthermore, and as can be

seen in our study, one of the major barriers is the difficulty in collaboration and information exchange, and as an action to address this barrier company A is organizing its development organization in smaller teams. As can be seen in previous research [15], cross-functional teams are critical to overcome barriers associated with collaboration and communication issues. In company A, cross-functional teams are being introduced and they will be one important step towards an agile R&D organization. Finally, companies strive towards having feature teams rather than component teams to shorten lead time and improve release frequency [15]. . Feature teams work with only a small part of the functionality and hence, respond quickly to changing requirements. . In introducing agile working practices and re-organizing the development team organization, company A is addressing several of the barriers associated with the transition from traditional to agile software development.

To summarize, the key focus area in the transition from traditional to agile R&D is the adoption of small and, cross-functional teams.

B. From Agile R&D to Continuous Integration

In our study, we have two companies moving from an agile R&D organization to continuous integration. In doing so, there are several barriers to address. As can be seen in both company B and C, they struggle with being dependent on suppliers which means that a lot of effort is put on communication and coordination with these. In addition, different interpretations of the current process model at different sites make the development process complex. Both companies are hardware oriented, meaning that a cultural shift is necessary in order to move further towards continuous integration. In both these companies the mechanical part has been the major part of the process so far and while it will always be critical it will have to successfully co-exist with important software functionality in development of future products. As can be seen in our interviews, this cultural shift is challenging and it requires a transformation of previous traditions and values. Furthermore, the companies find the variety of tools and the lack of transparency problematic. While company B notes that the many tools make learning and efficiency difficult, company C finds the lack of transparency problematic. Finally, and as noted in literature [7], both companies emphasize the test activities as one of the major challenges when moving towards continuous integration. The need for automated test processes is well understood but still difficult to fully implement.

To address the barriers mentioned above, the companies in our study mention a number of actions. What is most important, and something that is also noted in previous research [7,16], is the need to develop complete test suites including automated tests that are well integrated with system validation. In both company B and C development of automated tests is an on-going activity and the aim is to increase the number of automated tests significantly within a short period of time. Both companies find this the most important activity in order to consolidate the concept of continuous integration and make possible for a culture shift among developers who believe things "only when they see it happen" as mentioned by one of the developers in company C. Furthermore, and in order to reduce complexity, code needs to be checked into a main development branch, i.e. the production line. If so, companies can avoid having several branches that will only add to the complexity and lead-time described by our interviewees. As a result of this, the transparency between teams will also improve and the barrier with lack of transparency can be reduced. Finally, company B and C both touch upon the fact that lead-time is long and that the dependency on suppliers as well as mechanical parts is problematic. In order to at least to some extent address this situation, 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 and hence, quality can be reviewed at an earlier stage. This will allow for more frequent deliveries to customers and an opportunity to learn from customer feedback earlier.

To summarize, the key focus area in this transition stage is to develop a fully automated testing infrastructure that continuously verifies the product as it evolves during development.

C. From Continuous Integration to Continuous Deployment

Our last company, i.e. company D, is the company closest to continuous deployment of software. In this company, continuous integration is a well established practice and there are already attempts to involve pro-active customers in continuous deployment of certain software functionality. However, there are a number of barriers that need to be addressed also for company D. 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 cannot be fully counted for. Furthermore, the internal verification loop needs to be shortened in order to not only develop functionality fast but too also deploy it fast at customer site. Third, several of the companies mentioned lack of transparency as a barrier when moving closer towards continuous deployment of software. Depending on company, this barrier takes different forms but in common is the need to get an overview of the current status of development projects. This is evident in company D where one of the product line maintenance managers expressed a need to get better status reporting from teams in order to increase transparency and further improve speed.

To address the barriers mentioned above, there are a number of actions that can be taken. In company D, one major action has been to involve not only the R&D units but also the product management units in the vision of delivering smaller features more frequently to customers. In 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. In company D, we learnt from product managers that their involvement was important and that their insight into the R&D organization

had increased when working towards continuous deployment. This is in line with previous research, which emphasizes the importance of having the R&D organization and the product management organization sharing the same goal when moving towards continuous deployment [3]. Finally, finding a pro-active customer who is willing to explore the concept of continuous deployment is critical. Here, the concept of 'lead customer' is useful and what is important is to find mechanisms to facilitate for fast customer feedback and mechanisms for translating this feedback into improved software functionality.

To summarize, in transitioning towards continuous deployment the internal action is to involve product management in the short, agile cycle of product development. The external action is to develop a new engagement model with lead customers to facilitate for continuous deployment

D. From Continuous Deployment to Innovation System

In this study, we have not been able to explore the final step in the "stairway to heaven", i.e. the move from continuous deployment to R&D as an 'experiment system'. Based on earlier research with other companies, we anticipate that the key actions in this transition are twofold. First, the product needs to be instrumented so that usage and other data can be automatically collected from the installed product base. Second, the overall R&D organization needs to develop the capability to effectively use the collected data to test new ideas with customers.

In our future research, we see the transition between the final steps in the "stairway to heaven" as our main interest to explore further.

VI. CONCLUSIONS

In this study, we explored how software development companies evolve their practices over time. Based on a conceptual model presented as the "stairway to heaven" we presented the transition process when moving towards continuous deployment of software. In doing so, we identified the key barriers for such a transition as well as actions that address these.

While the details in our study relate to each specific company, there are a number of implications that we think apply to more companies than those we studied. First, 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. Second, the transition towards continuous integration requires an automated test suite, a main branch to which code is continually delivered and modularized development. Finally, the move towards continuous deployment requires organizational units such as product management to be fully involved and a pro-active lead customer to work closely with when exploring the concept further.

For the last transition, i.e. from continuous deployment towards R&D as an 'experiment system' the barriers are still to be explored. While our study does not cover this final transition, we believe that the empirical insights presented will be important for companies with an attempt to reach also this final and highly innovative step.

ACKNOWLEDGMENT

This study was funded by the Software Center at Chalmers University of Technology and University of Gothenburg.

REFERENCES

[1] N. D. Fogelström, T. Gorschek, M. Svahnberg, and P. Olsson, The Impact of Agile Principles on Market-Driven Software Product Development, Journal of Software Maintenance and Evolution: Research and Practice, 2010, Vol: 22, pp. 53-80.

[2] L. Williams and A. Cockburn, Agile Software Development: It's about feedback and change, Computer, 2003,Vol: 36, Nr: 6, pp. 39-43.

[3] A. Shalloway, J. R. Trott, and G. Beaver, Lean-Agile Software Development: Achieving Enterprise Agility, Addison-Wesley, 2009

[4] J. Bosch, Building Products As Innovation Experiment Systems, Proceedings of the 2012 International Conference on Software Business (ICSOB 2012), to appear, 2012.

[5] I. Sommerville, Software Engineering, 6th edition, Pearson Education: Essex, England, 2001.

[6] J. Highsmith and A. Cockburn, Agile software development: the business of innovation, *Computer*, vol. 34, no. 9, pp. 120-127, Sep. 2001.

[7] J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test and Deplyment Automation, Addison-Wesley: Boston, 2011.

[8] G. Walsham, Interpretive case studies in IS research: nature and method. European Journal of Information Systems, 1995, Vol: 4, p. 74-81.

[9] R. Vidgen and X. Wang, Coevolving systems and the organization of agile software development, Information Systems Research,2009, Vol: 20, Nr: 3, pp. 355-376.

[10] N. Melao and M. Pidd, A Conceptual Framework for Understanding Business Processes and Business Process Modelling, Information Systems Journal, 2000,Vol: 10, Nr: 2, pp. 105-129.

[11] R. K. Yin, Case study research, design and methods, 3rd edition, Sage Publications: Newbury Park, 2003.

[12] M. B. Miles and A. M. Huberman, Qualitative Data Analysis, Sage Publications, 1994.

[13] P. Runesson and M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering, 2009, Vol: 14, p. 131-164.

[14] P. Abrahamsson, Is Management Commitment a Necessity After All in III: Software Process Improvement?, In Proceedings of EUROMICRO 2000, Maastricht, The Netherlands, IEEE Computer Society, 2000, pp. 246-253.

[15] C. Larman and B. Vodde, Scaling Lean and Agile Development: Thinking and Organizational Tools for Large-Scale Scrum, Pearson Education: Boston, 2009.

[16] P. Agarwal, Continuous Scrum: Agile management of SAAS Products, In Proceedings of the 4th India Software Engineering Conference (ISEC), Thiruvananthapuram, February 32-26, 2011.