TOWARDS R&D AS INNOVATION EXPERIMENT SYSTEMS: A FRAMEWORK FOR MOVING BEYOND AGILE SOFTWARE DEVELOPMENT

Helena Holmström Olsson
Department of Computer Science
Malmö University
Malmö, Sweden
helena.holmstrom.olsson@mah.se

Jan Bosch
Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden
jan.bosch@chalmers.se

Hiva Alahyari
Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden
hiva.alahyari@chalmers.se

ABSTRACT
In focusing on flexibility, efficiency and speed, agile development practices have lead to a paradigm shift in how software is developed. However, while agile practices have indeed proven to be successful these are not the final step of software development. There is a “beyond agile” in which software development companies can capitalize even more on customer contributions and where customer feedback is the main driver for innovation. In this paper, we present a multiple-case study where we explore five software development companies moving from agile towards continuous deployment of software and a future in which R&D works as an ‘innovation experiment system’. Based on a qualitative interview study, we present benefits and barriers when moving towards R&D as an innovation experiment system. Also, we present a framework in which we identify key initiatives that companies deploy in order to evolve their software development practices.

KEY WORDS
Innovation experiment system, continuous deployment, agile software development.

1. Introduction
Market uncertainties, competitive pressures, and the constant need for shortened development cycles call for software development practices that are flexible, responsive and adaptive. In recent years, a variety of agile methods designed to enhance development teams’ ability to respond to changes have emerged. Typically, agile methods advocate flexibility, efficiency and speed [1]. In emphasizing the use of iterations and development of small features, agile practices have indeed 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 order to improve responsiveness to customers, agile development is not the end, not the final step, of software development. As can be seen in recent studies [3, 4], there are a number of new and innovative approaches that support continuous customer feedback and mechanisms to better capitalize on this. As can be seen in these studies, the opportunity to have research and development activities (R&D) as an innovation experiment system, where users test parts of functionality on a continuous basis, is gaining momentum. In this way, software development companies move beyond the concept of agile development and towards a situation in which software functionality is continuously deployed and where customer input is the main driver for innovation.

In this paper, we present a multiple-case study in which we explore five software development companies moving towards R&D as an innovation experiment system. The contribution of this paper is twofold. First, we identify the benefits and the barriers that the companies experience when moving towards R&D as an innovation experiment system. Second, we present a framework in which we identify the key initiatives that the companies deploy in order to evolve their software development practices. 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 and analysis of these. Finally, section V presents our conclusions.

2. The “Stairway to heaven”: The typical evolution path for software development
All companies evolve their software development practices over time. As we reported in earlier work [5], 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.

The phases of the “stairway to heaven” model 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) 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). At this point, however, agile practices are not fully deployed, i.e. organizations usually adopt only parts of the methods and techniques but not the entire range of agile practices. Usually, most organizations start the transition...
towards agile R&D by adopting smaller, cross-functional teams, i.e. feature teams, and by giving these teams an end-to-end responsibility for a particular feature. At this stage, 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). Once continuous integration is working internally, lead customers often express an interest to receive software functionality earlier than through the planned release cycles. What they want is to be able to deploy software functionality continuously (step D) and as soon as there is a change to the code base. The final step (step E) is where the software development company not only releases software continuously, but also develops mechanisms to collect data from its installed customer base. This data is then used to drive an innovation experiment system, in which new ideas are tested continuously on segments of the installed customer base and the data collected from these customers is used to steer the direction of all R&D efforts, i.e. as a basis for the R&D roadmap and requirements [3].

![Figure 1. “The stairway to heaven”, i.e. the typical evolution path for companies moving towards R&D as an innovation experiment system.](image)

### 2.1 Traditional development

We refer to traditional development software development characterized by long development cycles. Typically, project teams are large and the different competences are separated into disciplines such as system architecture, design and test [6]. Development is sequential with a rigorous planning phase in the beginning of each project. A major characteristic of traditional development is that delivery to customers 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. As a result of this, changes in requirements often cause major re-work resulting in a tiresome and very expensive process.

### 2.2 Agile R&D organization

The second step in the evolution is where product development, i.e. the R&D organization, adopts agile practices, but where product management and system verification still work according to the traditional approach. Typically, at this stage, agile practices are characterized by developers working in smaller, cross-functional teams, short time-boxed iterations, adaptive refinements of plans and goals, and lightweight processes that rely on peoples’ tacit knowledge to a greater extent than traditional document-driven development approaches [1, 7]. In this second step of the evolution path, the R&D organization works in shorter cycles with more frequent deliveries internally. R&D is organized in smaller teams and while the different disciplines, e.g. design, development and test, were traditionally separated they become integrated in cross-functional team arrangements. Customers, however, still receive bigger releases within planned, and less frequent, intervals.

### 2.3 Continuous integration

A company employing continuous integration has succeeded in establishing more advanced agile practices that allow for frequent integration of work, daily builds and fast commit of changes, e.g. automated builds and automated test, internally. Humble and Farley [8] 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 R&D and system validation are working according to agile practices.

### 2.4 Continuous deployment

Continuous deployment is the idea that you push out changes to the code all the time instead of doing large builds and having planned releases of large chunks of functionality. This allows for continuous customer feedback, the ability to learn from customer usage data, i.e. real-time customer behaviour, and to eliminate any work that doesn’t produce value for the customer. At this point, R&D, product management and customers are all involved in a rapid, agile development cycle in which response time is short [3]. While the concept is still to gain momentum, the introduction of Web 2.0 technologies, social network systems and especially, Software-as-a-Service (SaaS) solutions (sometimes referred to as ‘on-demand software’), has proven to be conducive to continuous deployment of new functionality [3]. One of the key characteristics is that the cost of deploying new versions of software is negligible. Typically, deployment starts by one of the many servers starting to run the new version under close monitoring. Gradually more servers are configured with a new version of the software until all servers are successfully running the new version of the software. At any point in time, the deployment can be rolled back in response to potential problems.
2.5 R&D as an innovation experiment system

The final step in the “stairway to heaven” model 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 functionality. The intention is that the R&D roadmap and requirements are driven by exposing customers to partial implementation of functionality in order to use their instant feedback for determining the value of a particular functionality. Recently, the concept of R&D as innovation experiment systems has been defined as an experiment-centric approach to product development with the purpose of accelerating innovation through systematic and continuous collection of user feedback [3, 4]. Common for innovation experiment systems, such as for example SaaS software, is that requirements evolve in real-time based on data collected from real-time customer use instead of being frozen early based on the opinions of product management about future customer needs [3]. Another characteristic is the avoidance of versioning of software. Traditionally, for many companies, every customer would have a unique configuration of the product with different versions of the software. This adds a layer of complexity to the already costly process of deploying new versions. In an innovation experiment system, there is only one version, i.e. the currently deployed one. All other versions have been retired and play no role.

2.6 Summary

In the table below, we summarize our discussion and show the involvement of each function in the organization, as well as the customer’s role in the different steps (A-E) in the “stairway to heaven” model. The ‘approach’ column refers to the different steps in the model (Figure 1). Related to the organizational functions and the way in which they operate, ‘T’ stands for traditional, ‘A’ for agile and ‘SC’ for short cycle. The table illustrates the evolution of software development practices, and a situation, if looking at the final step (E), in which all functions involved enjoy short feedback cycles and the opportunity to experiment with, and learn from, immediate customer feedback.

<table>
<thead>
<tr>
<th>Approach</th>
<th>PM</th>
<th>R&amp;D</th>
<th>Validation</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>B</td>
<td>T</td>
<td>A</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>C</td>
<td>T</td>
<td>SC</td>
<td>SC</td>
<td>T</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>SC</td>
<td>SC</td>
<td>A</td>
</tr>
<tr>
<td>E</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
</tr>
</tbody>
</table>

Table 1. Summary of each step in the “stairway to heaven” model (A-E) and how each organizational function (Product Management, R&D, Validation and Customer) work at this particular step.

3. Research approach

3.1 Research method

This study builds on a multiple-case study and adopts an interpretive research approach [9]. 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 [10]. In particular, case study research is appropriate to investigate real-life contexts where control over the context is not possible [11] and where there is an interest in accessing people’s experiences in order to create a rich understanding of a particular context [9]. In our study, the case companies represent interesting examples that well reflect our attempt to better understand the benefits and barriers that software development companies face, as well as the key initiatives they need to deploy, in order to evolve their software development practices.

3.2 Data collection and analysis

The main data collection method used in this study is semi-structured interviews with open-ended questions [12]. In our study, we chose to have an interview protocol organized in four pre-defined themes, and allow for openness and flexibility within these themes. The themes were (1) organization and ways-of-working, (2) customer interaction mechanisms/models, (3) strengths/weaknesses in ways-of-working, and (4) benefits and barriers as experienced when moving towards R&D as an innovation experiment system. In total, 32 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. Finally, in company E we collected 14 interviews from managers, architects and engineers directly involved in on-going SaaS initiatives. All interviews were in English and each interview lasted for one hour. During the interviews, we were two researchers sharing the responsibility, i.e. one of us asked the questions and one took notes (the only exception to this were the interviews at company E where only one of the researchers was present). In addition to the interview notes, all interviews were recorded in order for the research team to have a full description of what was said [9]. Each interview was transcribed and the transcriptions were shared among the researchers to allow for further elaboration on the empirical material. In addition to the interviews, documentation review and field notes were complementary data collection methods, including software development documents, project management documents, and corporate websites and brochures. During analysis, all transcribed interviews were carefully read with the intention to identify recurring
elements and concepts as is common in open coding techniques [13]. In similar with grounded theory [13], we let coding categories and relationships emerge from the data. In our analysis, all interviews were read several times by the involved researchers and we continuously compared our emerging categories. After going back and forth in the empirical data, and applying the process of constant comparison, we reached what is referred to as 'theoretical saturation' [13], i.e. a state in which no new significant categories or concepts emerge. At this point, we had concepts related to agile ways of working, challenges when transitioning between different development approaches, visions and strategies in moving towards R&D as innovation experiment systems, and different key initiatives that were considered important for advancing current software development practices. All these concepts were the result of our data analysis and they reflect perceptions, experiences and beliefs as held by the interviewees involved in our study.

3.3 Validity and generalizability of results

As noted by Maxwell [14], 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 [12]. To address this aspect, we started each interview with sharing our understanding of concepts such as 'continuous deployment' and 'R&D as innovation experiment systems' with the interviewee. In this way, the researchers and the interviewee had a common view of the concept already before the interview. With respect to external validity, our contribution is related to (1) the drawing of specific implications and (2) the contribution of rich insight [9]. Based on our interviews, we present findings and implications in a particular domain of interest. While these implications should be regarded as tendencies rather than predictions [9], they might indeed prove useful for other companies with similar intentions.

3.4 Research sites

3.4.1 Company A

Company A develops systems for military defense and civil security. The systems focus on surveillance, threat detection, force protection and avionics systems. 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.

3.4.2 Company B

Company B is an equipment manufacturer developing, manufacturing and selling a variety of products within the embedded systems domain. The company structure is highly distributed with globally distributed development teams. Also, suppliers do a large extent of the development. In relation to the model presented in Figure 1, this company 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 with continuous integration mechanisms in place.

3.4.3 Company C

Company C is a manufacturer and supplier of transport solutions for commercial use. The R&D organization involves coordination of a large number of distributed teams. Similar to company B, the R&D 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.

3.4.4 Company D

Company D is a provider of telecommunication systems and equipment, communications networks and multimedia solutions for mobile and fixed network operators. The organization is highly distributed with globally distributed development teams. In relation to the model presented in Figure 1, this company is a company with established practices for continuous integration and with initiatives for 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 advance their development practices.

3.4.5 Company E

Company E is a software product and services company that serves consumers, small businesses, accountants, financial institutions and healthcare providers with primarily finance and accounting related solutions. The company is organized in business units that address different customer segments or provide solutions in specific domains. Although traditionally a licensed software company, over the last few years, more than half of the revenue has shifted to Software-as-a-Service (SaaS)
solutions. In relation to the model presented in Figure 1, company E represents a company at which R&D is working as an innovation experiment system.

4. Case study findings and analysis

In this section we present our case study findings and the analysis of these. In our presentation, we outline the benefits and the barriers that our interviewees experience when transitioning between the steps in our “stairway to heaven” model (Figure 1). Moreover, we present a framework in which we identify ‘key initiatives’ that support the transition between each step (see Table 1). The key initiatives are organized in four categories and emerged as a result of our data analysis. The four categories are (1) ecosystem, (2) business strategy, (3) architecture, and (4) organizing.

4.1 From traditional to agile R&D organization

In our study, several companies emphasize the many benefits in moving from traditional to agile development. In our interviews, we learnt that company A has a number of agile teams that use methods such as Scrum and XP and that the short cycles with which they develop functionality has already started to reap benefits visible to other teams in the organization. Likewise, company B has a number of teams that have started to practice the more lightweight approaches advocated by agile methods. While still struggling inside a hierarchical company structure with a large supplier network, these teams have succeeded in shortening the internal development lead-time as well as allowed for a more flexible management of adjustments and changes in requirements.

When it comes to barriers, the companies mention the complexity of having a few teams work in an agile fashion while others do not. Even though there are benefits to reap, and even though the agile teams inspire other teams, there are sometimes difficulties when interacting with the surrounding organization and especially, with a large supplier network. In looking back at this transition, company D remembers the complexity of having an agile R&D organization while at the same time having a traditional release organization. In company B and C, the complexity in interacting with other parts of the organization, in this case the many internal and external suppliers, is evident. “...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”. In particular, the dependency between components and component interfaces is highlighted as complex. This makes development teams highly dependent on each other and hence, less prone to adopt agile ways of working. Another common barrier is the feeling of reduced control by R&D management. In particular, this is due to the fact that teams are encouraged to be self-selected and self-directed to a larger extent than before. In company D where the transition to agile R&D took place a few years ago, the feelings among former project managers were fragmented. While some appreciated the new and smaller team formations, others felt overlooked by the new organization. Over time though, the benefits have outweighed this problem and the former project managers now appreciate the smaller teams and their more informal role as team leaders.

Key initiatives

Ecosystem: Renegotiating supplier contracts to facilitate agile development need to be a highly prioritized activity. To have all stakeholders in the ecosystem agree on the way forward is a challenge and the interviewees emphasize that having internal and external processes align is critical in order to increase speed in delivery.

Business: The business strategy is of great importance in establishing a culture and support change towards a new development approach. As can be seen in previous research on process improvement [14], management commitment is critical and a clear vision is a powerful tool. Our interviewees emphasize the importance of building awareness of agile development as an approach that allows for frequent adjustment of feature content.

Architecture: An important initiative is to make sure that feature teams are supported by an architect who “safeguards” the team and the integrity of the architecture. In company D, the transition to agile development took place several years ago and one of their most important decisions was to have appointed architects working closely with the teams in protecting the architecture.

Organizing: One important initiative when moving to agile R&D is to have mechanisms for team formation and for teams to empower and self-direct themselves. Similarly to previous studies, our research shows on the importance of adopting feature teams [16]. In all companies, team formation is challenging, but experiences reveal that while empowerment and self-direction is stressful in the beginning, it is appreciated after being practiced for a while.

4.2 From agile R&D to continuous integration

Although at different steps in the “stairway to heaven” model, all of our companies have to some extent experienced the move from agile R&D to continuous integration of software. In company A, there is a first attempt to continuous integration and the teams involved are the system engineering team and the quality assurance team. Even though the company is still in it’s beginning towards having continuous integration fully in place, the attitude among the interviewees is positive and there is anticipation on the benefits that this will bring. This is reflected on by the configuration manager: “I like to think about each build as a release candidate meaning that the notion of continuously integrating functionality brings with it the opportunity to also continuously improve quality”. In company B, continuous integration is about to
start between a few teams and the ambition is to extend the use of agile practices to other parts of the organization. When asking the interviewees about what they perceive as beneficial, they mention ‘quality improvement’ and the idea of always having a shippable product. The appreciation of continuous quality improvement is shared among all companies in our study, and they mention continuous integration as an important step towards closer interaction with customers.

Besides benefits, our study reveals a number of barriers when moving to continuous integration. Several interviewees mention testing as critical when talking about this transition, and it is clear that a fully automated test process is desirable, but difficult to achieve. For example, one developer at company C emphasizes the need for automatic testing while at the same time realizing that this is difficult in an embedded system involving hardware components with slow development cycles. Another concern is the broad variety of tools and the difficulties they introduce. According to one interviewee the tool issue might affect team discipline around test cases since the tools introduce so much problems that people get uncertain and lose confidence in the test suite.

**Key Initiatives**

*Ecosystem:* Suppliers need to support the notion of continuous integration. While our case companies experience continuous integration as a minor hurdle to establish internally, they all report on challenges when having external suppliers involved in this loop.

*Business:* The business perspective needs to transition from the “milestone perspective” to a perspective in which delivery and release are viewed as continuous activities.

*Architecture:* Modularization needs to be increased. Our interviewees emphasize that development lead-time is efficiently reduced once the architectural concerns are modularized.

*Organizing:* Test-driven development and automated tests needs to be developed [17]. Furthermore, to align the R&D organization with the validation and verification organization, as well as avoid branching is critical.

### 4.3 From continuous integration to continuous deployment

While company D and E are the only companies in our study that is currently practicing continuous deployment, interviewees in the other companies see benefits that make them push for such a transition. Primarily, the benefits are related to the opportunity to learn about customers based on their actual use of the software, i.e. to quickly learn what functionality the customer values the most. According to one of the release program managers at company D, faster feedback means cheaper development since the R&D organization can then spend time developing the “right things” rather than correcting mistakes in functionality that is not necessarily what the customer wants. According to this reasoning, time is better spent when having the opportunity to receive feedback earlier and as a result, adapt functionality faster than is done today. Among developers, the feeling is that continuous deployment of parts of functionality means less effort spent on functionality that nobody will use. One of the developers notes this when saying: “I think we would get the functionality that the customers want with minimum effort...instead of spending time doing things they do not actually ask for”. In company E, real-time customer data is collected through ‘lead customer’ use of particular features that are released as ‘initial versions’, i.e. latest stable release of the product extended with one new feature. While these features are not fully developed they can be used in day-to-day operations and customers can provide feedback on the features in the context of their real business. The feedback provided by these lead customers is critical for further improvement of the feature, as well as for the direction of the R&D efforts.

Besides the benefits, our interviewees identify a number of barriers when moving to continuous deployment. In particular, the barriers are associated with the lack of trust in software quality. As mentioned by one of the product line maintenance managers in company D, the development teams would benefit from knowing more about the 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. Also, the interface towards customers is considered a barrier since adjustments of business models are needed. In our study, a number of interviewees mention the difficulty in acting agile and promote continuous deployment of functionality at the same time as the business model gives a conservative impression in promoting fixed releases and fixed price models assuming that requirements are frozen upfront.

**Key Activities**

*Ecosystem:* Lead customers need to be identified with whom the R&D organization can start building a continuous deployment culture and capability. These lead customers serve as role models to other customers.

*Business:* The business model needs to be reviewed so that there are mechanisms that support continuous deployment of functionality.

*Architecture:* Mechanisms to roll back unsuccessful deployments, and to deploy components of the system rather than the entire system, are needed.

*Organizing:* All corporate functions, such as for example the release organization, need to be aligned with the R&D organization in order to facilitate continuous deployment.

### 4.4 From continuous deployment to R&D as an innovation experiment system

In our study, company E represents the final step in the “stairway to heaven” model. This company has several successful SaaS products that have been commercially deployed for several years. Over the last years, the team responsible for one of these products has adopted a vigorous experimentation approach using A/B testing.
The team adopted a weekly cycle of experimentation where it decides which experiments to run early in the week, take one or two days to develop alternative implementations of aspects of the system, deploy the solutions and start the experiment towards the end of the week, collect data over the weekend and decide early the week after which version (A or B) was more successful. Every week, the team runs dozens of experiments and constantly improves the performance and customer satisfaction of the product based on these. The customer feedback is collected passively when customers use the feature online and their click-behaviour is saved. When having R&D as an innovation experiment system, customer input is efficiently used for improving existing functionality, develop new functionality to existing products, and for developing new products. Moreover, software R&D is much more than working efficient with development and release – it is a way of experimenting to keep up with a new generation of users and the particular needs of new user groups.

Besides benefits, the barriers relate to the resistance for releasing “experimental functionality” to customers. In the companies, the tradition of rigorous test and validation processes is well anchored and the culture is that all testing and validation activities are to be performed before functionality is released to any customer. Also, from a customer perspective, the idea of having partially developed functionality released with the intention to “experiment” might be a challenging task to pursue.

**Key Initiatives**

**Ecosystem:** Customers need to be involved in providing early feedback on new functionality. To collect, analyze and capitalize on customer feedback is important and the establishment of mechanisms that allow for quick response to customers is the major initiative to undertake.

**Business:** Business models and pricing models need to support short-cycle innovation processes based on customer usage data.

**Architecture:** Infrastructures need to be established in order to support run-time variation of functionality that allow for innovation experiments with customers. Also, a variety of data collection mechanisms are necessary which means an extension of current architectures.

**Organizing:** Requirements, R&D, validation and release functions all need to work together, and to get managerial support for such a transition is as critical as when introducing agile practices in the first place [18]. Initiatives that facilitate for aligning and integrating different corporate functions need to be prioritized.

### 4.5 Summary

In this section we presented the main benefits and barriers involved in moving towards R&D as an innovation experiment system. We presented key initiatives related to (1) ecosystem, (2) business strategy, (3) architecture, and (4) organizing that need to be deployed to support the transition from one step to the next in our model. In Table 1, we present a framework in which we summarize the benefits and the barriers, and in which we outline the key initiatives that the companies deploy to evolve their software development practices.

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Barriers</th>
<th>Key activities</th>
</tr>
</thead>
</table>
| Traditional -> Agile R&D organization | Shorter development cycles allowing for flexible adjustment of direction and management of changing customer requirements | Ecosystem
Renegotiate supplier contracts |
| Agile R&D organization -> Continuous integration | Lack of test automation solutions and team discipline around test cases | Business strategy
Build awareness that agile development allows for structured, but frequent adjustment of feature content |
| Continuous integration-> Continuous deployment | Lack of trust in software quality and customer interfaces that are not conducive to continuous deployment | Architecture
Support feature teams with architects safeguarding the integrity of the architecture |
| Continuous deployment <-> R&D as an innovation experiment system | Resistance to releasing experimental functionality to customers and difficulty in getting customers willing to participate in experiments | |

- **Continuous Integration:** Always a shippable product, i.e. software with production quality
- **Continuous Deployment:** Continuous deployment of customer relevant functionality
- **R&D as an Innovation Experiment System:** Innovation and experimentation in R&D that allows for immediate customer feedback and improved R&D effectiveness

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Barriers</th>
<th>Key activities</th>
</tr>
</thead>
</table>
| Traditional -> Agile R&D organization | Shorter development cycles allowing for flexible adjustment of direction and management of changing customer requirements | Ecosystem
Renegotiate supplier contracts |
| Agile R&D organization -> Continuous integration | Lack of test automation solutions and team discipline around test cases | Business strategy
Build awareness that agile development allows for structured, but frequent adjustment of feature content |
| Continuous integration-> Continuous deployment | Lack of trust in software quality and customer interfaces that are not conducive to continuous deployment | Architecture
Support feature teams with architects safeguarding the integrity of the architecture |
| Continuous deployment <-> R&D as an innovation experiment system | Resistance to releasing experimental functionality to customers and difficulty in getting customers willing to participate in experiments | |
Organizing (Ways of Working & Organization) | Adopt feature teams instead of component teams | Adopt test driven development and test automation infrastructures with rapid cycles | Align R&D organization with R&D organization to facilitate frequent deployment of functionality | Extend architecture with data collection mechanisms
---|---|---|---|---
| Develop effective mechanisms for empowering teams and team formation | Align R&D and validation & verification organizations | Align product management and R&D organizations | Integrate innovation, product management, R&D and release organizations around driving requirements elicitation as innovation experiments

Table 1: Summary of benefits and barriers, as well as key initiatives that companies need to deploy when moving towards R&D as an innovation experiment system.

5. Conclusion

In this paper, we present a multiple-case study exploring software companies moving towards R&D as an innovation experiment system. As a result of our research, we see that evolving one’s software development practices is a sequential process. In order to advance in the “stairway to heaven” model (Figure 1), there are a number of initiatives related to (1) ecosystem, (2) business strategy, (3) architecture, and (4) organizing, that companies need to deploy. Also, our findings show that the further companies evolve, the more important it becomes to align and integrate R&D efforts with (1) surrounding corporate functions such as validation and verification, product management, and product release, and (2) external stakeholders such as suppliers and customers. While the findings of the paper specifically apply to the five case companies investigated, we believe that there are indeed valuable insights for any software company interested in evolving its practices and move beyond agile software development.

Acknowledgement

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

References