Building Products as Innovation Experiment Systems

Jan Bosch

Chalmers University of Technology, Department of Computer Science,
Gothenburg, Sweden
Jan@JanBosch.com

Abstract. Traditional software development focuses on specifying and freezing requirements early in the, typically yearly, product development lifecycle. The requirements are defined based on product management's best understanding. The adoption of SaaS and cloud computing has shown a different approach to managing requirements, adding frequent and rigorous experimentation to the development process with the intent of minimizing R&D investment between customer proof points. This offers several benefits including increased customer satisfaction, improved and quantified business goals and the transformation to a continuous rather than waterfall development process. In this paper, we present our learnings from studying software companies applying an innovation experiment system approach to product development. The approach is illustrated with three cases from Intuit, the case study company.

Keywords: Product development approach, experiment systems, case study.

1 Introduction

The Internet has brought many changes and, as a society, we have barely started to exploit the possibilities that it brings. Over the last two decades, developments ranging from online commerce to open-source software to social networks have either been created or accelerated with orders of magnitude. In addition to the changes to products, services and communities, there is also a quite fundamental shift underway in how products and services are developed and deployed. This shift is driven by several fundamental differences between traditional licenses, on-premise software or embedded software and software-as-a-service solutions in a cloud-computing environment. These differences are driven by several factors, including the prevalent business models on the web, the deployment infrastructure, customer expectations, the network connectivity of increasingly many products and the real-time nature of online solutions.

The aforementioned factors have lead to the evolution of a new software development model that is different from development approaches for traditional software. First, it is focused on continuously evolving the software by frequently deploying new versions. Second, customers and customer usage data play a central role throughout the development process. Third, development is focused on innovation and testing as many ideas as possible with customers to drive customer satisfaction and, consequently, revenue growth. Stepping back, we can recognize that R&D in this context is best described as an "innovation experiment system" approach where the development organization constantly develops new hypothesis (ideas) and tests these with groups of customers.

M.A. Cusumano, B. Iyer, and N. Venkatraman (Eds.): ICSOB 2012, LNBIP 114, pp. 27–39, 2012.

Although the first area of application of this approach can be found in SaaS and cloud computing, the techniques can be applied to any product that is able to collect and provide data about its usage and performance to the R&D organization. This includes software-intensive embedded systems.

In the management literature, apply experimentation to business is a well-established concept, e.g. Davenport [4], though not practiced consistently and broadly in industry. Also, in innovation, the role of experimentation is broadly recognized, e.g. [9]. Finally, the notion of experimentation in online software is not novel. For instance, in response to Ray Ozzie's call to arms [6] to the Microsoft organization, Kohavi et al. [5] have developed an experimentation platform.

In practice, however, experimentation in online software is often limited to optimizing narrow aspects of the front-end of the website through A/B testing and in connected, software-intensive systems experimentation, if applied at all, is ad-hoc and not systematically applied.

In this paper, we take a broader perspective and address the scope ranging from optimization to new features and products. Consequently, the contribution of this paper is as follows. First, we present a first systematization of this innovation experiment system approach to software development for connected systems. Second, we illustrate the model using an industrial case study, Intuit.

The remainder of the paper is organized as follows. The next section discusses the key differences between traditional software development and the "innovation experiment system" (IES) approach increasingly found in cloud-based SaaS solutions as well as in other connected solutions. Subsequently, we present a first systematization of the "innovation experiment system". Then, we present the Intuit case study. Finally, we conclude the paper with a discussion of related work, conclusion and exploration of future work.

2 Traditional versus IES Development Approach

For decades, software development was a very structured and sequenced activity. First, the organization would figure out what to build, then the product would be developed and finally the software would be deployed. For embedded systems, the software would be deployed as part of the overall product, including mechanics and hardware. In the case of licensed software, the product is installed at the customer site on hardware owned and provided by the customer. The traditional software development process applies to both embedded and licensed software.

Over the last years, cloud computing and Software-As-A-Service (SaaS) solutions are rapidly becoming the norm and enjoy enormously rapid growth. The key reasons for this include the lower cost for the customer, the simplicity associated with not having to own hardware and the freedom from long-term constraints associated with most licensed software solutions.

Interestingly, these benefits extend in part also to software-intensive embedded systems and increasingly companies building connected embedded systems, from mobile phones to cars, are starting to exploit the advantages of frequent, post-deployment updating of software and the collection of usage and other performance data from systems in the field.

Common for SaaS software and software in connected embedded systems is that allows for an approach where instead of freezing the requirements before starting product development, the requirements constantly evolve and also affect already deployed systems that are actively used by customers. Consequently, requirements evolve in real-time based on data collected from systems in actual use with customers instead of being frozen early based on the opinions of product management about the likely customer needs 12, 18 or 24 months from now.

There are several important differences between traditional and this alternative approach to development. Below we discuss the most important differences and use these as context for a first systematization of the IES model of software development in the next section.

Business Model of SaaS and Other Frequently Updating Products Tends to Be Conducive to Continuous Deployment of New Functionality

Most SaaS offerings use a subscription model or a freemium model, where the basic solution is free, but the more advanced solution is paid for by the customer. Also in software-intensive embedded systems, the trend is to transition to providing products in a service-model. This means that the customer does not acquire the product, but rather engages in a subscription-model based service agreement. A subscription model, which typically allows a customer to leave after a brief notification period, requires a constant monitoring of customer satisfaction. Also, it typically requires a continuous flow of new functionality and redesign of existing functionality. Contrast this with licensed software, where customers are typically resistant against taking in new solutions because of the cost of integration in their IT infrastructure, or traditional embedded software, where upgrading is hard if not impossible for unconnected systems, and the difference in business drivers becomes obvious.

Customer Expectation of Continuous Evolution of the System

Due to the approach that companies like Google have taking concerning "perpetual beta", customers expect a continuous evolution of product functionality. A similar example is provide by Apple's frequent updating of its iOS software for its phone and tablets products. This is quite different from traditional licensed software where customers expected upgrades with a low frequency, perhaps once per year, in return for their maintenance fee. Customers are becoming increasingly accustomed to frequent, trouble-free updates that provide relevant additional value and consequently this is increasingly an expectation.

Interestingly, cases exist where SaaS product companies failed to provide continuous evolution of the system that created the impression of the product being stale. In combination with a subscription model that allows customers to leave the franchise at any moment, is a dangerous situation.

Cost of Deployment Much Lower for Connected, Prepared Products

Traditional on-premise software and unconnected embedded software is exceptionally expensive to upgrade and brings all kinds of risks, ranging from incompatibilities in the unique configuration at some customers to the complexities of rolling back new versions once deployed.

One of the key characteristics especially in SaaS environments but also in well designed connected embedded systems is that the cost of deploying a new version of the software in negligible. In a SaaS environment, a typical deployment starts by one of the many servers starting to run the new version under close monitoring by the deployment team. If this is successful, 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 problems starting to appear. The highly controlled environment lowers the cost of deployment with several orders of magnitude. In connected, embedded systems a similar approach is employed, but instead of servers, the deployed products are the destination point for new software.

A final aspect is the avoidance of versioning of software. Traditionally, for many companies, every customer would ultimately have a unique configuration of the product with different versions of components making up the system. This adds a whole new layer of complexity to the already costly process of deploying new versions. In an IES environment, there is only one version: the currently deployed one. All other versions have been retired and play no role.

Cost of Active and Passive Customer Feedback and Usage Data Collection
A perhaps less obvious but very important advantage of connected products is that the cost of collecting active and passive information from and about the customer is much lower. Active customer feedback is concerned with surveys and other mechanisms where the customer is aware that he or she is providing feedback. Passive feedback and usage data is collected while the customer is using the system. Examples include the amount of time a user spends using a feature, the relative frequency of feature selections, the path that the user takes through the product functionality, etc. The low cost and ease of data collection leads to the next major difference between IES-based and traditional software.

In connected, embedded systems, in addition to usage data, several kinds of other performance data can be collected. For example, connected cars can collect fuel consumption data whereas telecom equipment can collect real-time bandwidth data. In many systems, this data is already collected for operational management purposes, but hardly used in evolution of already deployed systems.

Real-Time Connection between Business Goals and Operational Metrics
The ease of collecting customer feedback and usage and other performance data leads to another important benefit of connected and SaaS products: it becomes feasible to build a real-time connection between the quantified business goals of the organization and the operational metrics collected from the installed base. This allows the organization to respond rapidly and dynamically to any changes to the use of its deployed products. For example, SaaS offerings driven by advertising revenue often use the metric of unique user growth as the driver for operational metrics. This gives the team working on the SaaS offering a real-time goal and metric to strive for and provides focus for the work.

From Opinion- to Data-Based Decision-Making
Everyone involved with a software product has many ideas about how to make it better. Conventionally, these ideas were collected and prioritized during the

roadmapping and requirement management process as part of the yearly release cycle. The selection of the ideas to include was based on opinions by leaders in the organization with significant weight put on the opinions of those higher in the organizational hierarchy and often turned into a rather politicized process. These opinions formed the basis of dozens or hundreds of person years of R&D effort and the confirmation of the correctness of these opinions would only take place after the finalized product has been deployed, if at all.

A connected or SaaS product provides the invaluable ability to run experiments with the installed base and customer population. Rather than committing many person years of effort, an idea can be translated into a hypothesis, the hypothesis can be tested by investing a few weeks of R&D effort and deploying the solution to some segment of the customer population. Data comparing the base product and the product extended with experimental software can be collected and if the hypothesis holds, i.e. there is a positive correlation to the business goals, more R&D effort can be invested to fully develop the concept that tested successfully. This approach allows for a much more effective investment of R&D resources as one has confirmation that the resources are spent on value for customers.

3 Connected Products as Innovation Experiment Systems

Innovation is lifeblood of any organization, but notoriously hard to get right in many companies. Innovation in large organization is often characterized by an enormous imbalance between the number of ideas that, informally or formally, exist in the organization and the number of concepts that are in fact tested with customers. The ratio, depending on the attention the organization puts towards idea generation by its employees, can range from one in a hundred to one in thousands. With that strict a selection process and the high cost associated with testing, the importance of selecting the most promising ideas, turning these into concepts and then designing a (prototype) product to test the concept with customers becomes such that it receives significant attention by senior management and many other functions and layers in the organization.

The selection process is, unavoidably, driven by the earlier experiences and beliefs of the people in the selection process. In most organizations, it is the opinions of the more senior persons in the organization that tend to weigh the heaviest. The challenge with this approach is twofold. First, opinions are a very poor substitute for real customer data and the innovation literature has many examples of successful innovations that were resisted for years inside the organization before made successful by a small “skunk works” team working under the radar. Second, even if the organization is sufficiently open minded to explore more innovative ideas and concepts, there is a natural risk avoidance that causes organizations to settle on the safe bets. Human psychology, as has been studied extensively in behavioral economics, experiences a loss much more strongly than it experiences a win, causing a selection process where losses are as much as possible avoided, resulting in mundane innovations.

The solution is, obviously, to find ways to decrease the dependence on opinions and to increase reliance on real customer or other data. Traditional metrics such as the

Net Promoter Score [8] have been used for the last decade or more, but often fail to provide timely feedback during the development process as these are backward looking and focus on the entire product. To collect customer or performance data early in the innovation process, the organization needs to find mechanisms to test more ideas and concepts with customers and in the installed base in real-time and obviously at a much lower cost than earlier. This requires new behaviors at the business level, i.e. involving customers in feature and product concept validation without an, initially clear, business model. Also, it requires changes to the R&D processes as customers need to be involved much earlier and deeper in the R&D process. Finally, this requires changes to the architecture of the products and platforms to facilitate testing versions of screens, components, subsystems and entire products in order to determine customer preference and interest. The mechanisms used for achieving customer involvement and the efficient execution of experiments on the deployed product base depends heavily on the type of experiments, system, stage and purpose.

Cloud-based SaaS products as well as connected, software-intensive embedded systems offer a particularly well-suited context for building an innovation experiment system. As we discussed in the previous section, connected systems allow for the rapid and low-cost deployment of new functionality. In addition, the collection of customer feedback as well as usage and other performance metrics is simple and the connection to business goals is virtually real-time.

In figure 1, we present the concept of innovation experiment systems in R&D graphically. The loop between deploying new functionality, measuring usage and other performance metrics and subsequently using the collected data to drive development is the main process. The goal of an innovative product is to maximize the number of iterations that can be executed per time unit, e.g. per quarter. The rationale is that the faster the organization learns about the customer and the real world operation of the system, the more value it will provide and consequently the more successful it will be compared to its competitors.

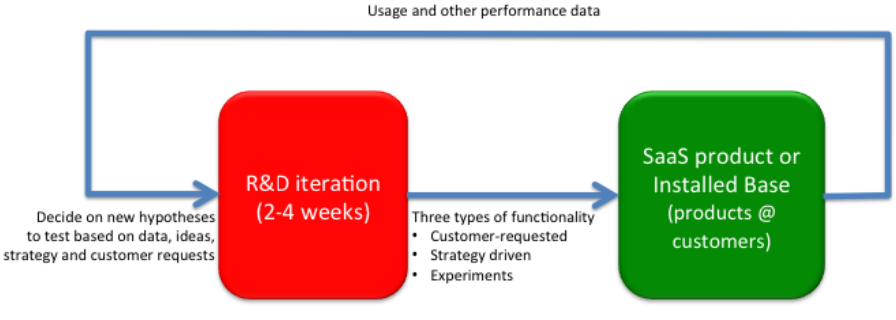

Fig. 1. Overview of R&D as an Innovation Experiment System

Development Approach

In this paper, we take the innovation experiment system as the basis for development. The approach recognizes three stages, i.e. pre-deployment, non-commercial

deployment and commercial deployment. In addition, we distinguish between three scopes for experimentation, i.e. optimization, features and new products. Below, in figure 2, we provide an overview of these two dimensions and, for each combination, we mention one or more example techniques that can be applied to collect customer or product performance data.

Although there are obvious differences between product optimization, new feature development and new product creation, the basic principle is that we want to invest as little possible until deploying something to customers. For instance, for new product development, putting a minimal viable product as rapidly as possible in the hands of customers as quickly as possible is essential. Once we have a product that customers can use, the initial value is often such that it is not yet monetizable. However, it is still valuable to receive customer. Then the product development cycle starts and the company uses a non-commercial deployment approach. Once there is sufficient value, the product is commercially deployed using the selected business model. Even when the product reaches this stage, collecting customer feedback is still of immense importance to continue evolution of the product and to direct R&D investment to the most valuable features and functionality.

In the section below, we present each phase and describe the techniques that exist for obtaining customer feedback in each phase.

	Pre-Development	Non-commercial deployment	Commercial deployment
Optimization	Ethnographic studies	Independently deployed extensions	Random selection of versions (A/B testing)
New features	Solution jams	Feature alpha	Instrumentation/ collecting metrics
New Products	Advertising Mock-ups BASES testing	Product alpha Labs website	Surveys Performance metrics

Fig. 2. The scope of innovation experiments

Pre-deployment

The pre-deployment stage is concerned with, for a new product or a major new feature of an existing product, to involve the customer in the prioritization process to the extent possible. As there is no working implementation, only active customer feedback can be collected. Consequently, in this section we focus on techniques that can be applied to obtain customer feedback before development of a feature or new product is initiated. The purpose of these techniques is to collect customer feedback before any R&D effort has been expended. In figure 2, we mention several examples of techniques that are used during pre-development. In this section, we discuss three of the examples.

New product: BASES Testing
Originally introduced by Nielsen, BASES testing [7] is a technique for testing product concepts that do not have an associated implementation. Together with competing products or services, the concept is presented to a panel of customers. Often, detailed information is provided including pricing information and an assessment of the key differentiating features of the concept. The panel members are asked to rate the presented concept relative to competing offerings. For products addressing established markets and in mature product categories, BASES tests provide strong correlation with product success and hence are considered to be good predictors. In software product lines, BASES tests can be used to determine the viability of new members of the product family.

New product: Advertising
Several companies use online ads as a mechanism to test concepts. The ad is presented on sites visited by potential customers or inside existing products developed by the company. The goal of the advertisement is to evaluate the response (or click through rate) of customers. In addition, depending on the type of concept, customers can even be asked to complete an order form including payment information before being informed that the product currently not available. This provides significant information about the attractiveness of the concept and even allows for testing different presentations of the concept through A/B testing. Advertising can be used to test both new products and new (cross-platform) features.

New feature: Solution Jams
During the feature identification and selection process, a company can organize "solution jams" with engineers, designers, product managers and customers where employees can bring new ideas and work on turning these into concepts while getting feedback from customers. At the end of the jam, the most promising concepts can be selected for development. See [2] for more details.

Non-commercial Deployment

Once development has lead to an minimal viable SaaS or connected embedded product or implementation of a significant new feature of an existing product that can be put in the hands of customers, additional techniques for collecting customer feedback can be employed. The best techniques to use depend on the context. The team can be building an alternative implementation for an existing feature in the product, a significant new feature for an existing product, a new product adjacent to an existing product or set of products or a new product in an entirely new space for which no customer base exists inside the company.

In general, employing innovation experiment systems require extensive instrumentation of the product in order to be able to collect information about the usage and performance of the product. The collected and aggregated data is used by the development organization to determine whether the current implementation is optimal or, in the case of different alternatives being considered, which alternative is preferred. It also provides a feedback mechanism for those cases where during the

pre-deployment stage decisions were made based on opinions. The data can be used to reflect on the accuracy of the original opinions.

In figure 2, we presented several examples of techniques to collect customer feedback during the non-commercial deployment stage. In the remainder of this section, we describe a few example techniques.

New feature: In-Product Surveys
Especially for SaaS products, though also applicable for connected embedded systems, it is very easy to create pop-up surveys at strategic locations in the product that allows for obtaining active customer feedback while the customer is in the process of performing a certain action. Although customers may experience this as an annoyance, when presented well, it can provide immensely valuable feedback that contains more semantic information than the purely passive data from product instrumentation.

New product: In-Product Marketing
A special case of advertising is in-product marketing of other products or of experimental features of the product currently in use. In this case, part of the screen real-estate is reserved for presenting marketing messages that preferably are as well aligned with the current context of use as possible. Similar to general advertising, this technique is often used for cross-sell and upsell purposes, but the technique can be used to determine the interest of customers during the pre-deployment stage or to draw attention to products or features in the non-commercial deployment phase.

New product: Labs website
Several companies have a separate "labs" website where prototypes and early versions of products can be made available free of charge with the sole purpose of collecting customer feedback. A labs website is especially valuable for companies that have an existing customer base that they can draw from for collecting feedback for new SaaS products that are under development. Existing, already deployed, products can then be used to encourage customers to try out the new product.

For connected, embedded systems this technique is less applicable. This type of experimentation requires the production of a set of prototype systems for the explicit purpose of collecting customer feedback during product development.

Commercial Deployment

Once the product is commercially deployed, the focus of the R&D investment shifts in two important aspects. First, as there now is a model of monetization, optimizing the actual use of the system, for instance through A/B testing of alternative feature implementations becomes an important part of development. Although the system will still be extended with new features, over time an increasing part of the R&D effort shifts to optimizing the existing features. Second, a new form of experimentation is often added to the equation: experimentation with alternative business models in order to drive revenue growth. The close, real-time alignment between business goals and operational metrics provides the basis for this.

Although most if not all of the aforementioned techniques can be used during this stage as well, there are some specific techniques exist that can be employed to drive the innovation experiment system specifically at this stage.

Similar to earlier sections, we discuss a few example techniques, based on the list shown in figure 2.

Optimization: Random Selection of Versions (A/B testing)
Possibly the most fundamental of mechanisms is the random selection of versions of use cases, features or other "chunks" of functionality and their subsequent presentation to customers. The versions are instrumented with data-collection functionality to capture the user behavior in a way that allows the team to select the preferred version. In online offerings, the notion of A/B testing is used extensively as a mechanism to determine the best web-page version. The metric for success differs based on the business goal. For instance, for websites that use advertising as a monetization mechanism, the user staying longer on the page is preferred, whereas a subscription-based web property offering productivity solutions would measure success in terms of the least amount of time spent on the page. Of course, a multitude of other metrics might constitute the definition of success.

Optimization: Ethnographic studies
Finally, ethnographic researchers, or for that matter, regular employees, can be sent out to meet with customers. Rather than interviewing the customer, the idea behind ethnography is to follow the customer in their daily life and study the use of the product or feature in context. This provides valuable information about the use of the product or feature in practice and is an important source for new work to be pursued. For example, at Intuit engineers collectively spend more than 10.000 hours in "Follow Me Home" sessions with customers.

4 Case Company: Intuit

Intuit is Fortune 1000 software product and services company that serves consumers, small businesses, accountants, financial institutions and healthcare providers with primarily finance and accounting related solutions. Intuit is organized in 10 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 Intuit's revenue has shifted to SaaS or SaaS related solutions and more than 70% of it's revenue now is pure SaaS or has a significant SaaS component.

As Intuit is a major SaaS company, there are numerous examples of SaaS offerings that are in one of the stages of the innovation experiment system model. However, for reasons of confidentiality, we can only share some aspects of each SaaS offering. During the time of the research, the author was an employee at Intuit and was indirectly involved with several SaaS initiatives. Consequently, the research underlying this case study is based on a participant-observer research method [1][3].

Pre-deployment

One of the products at Intuit with a significant SaaS component uses Solution Jams to prioritize the features to build even before investing any R&D budget. The solution

jam is organized at the product development organization level. It is a full-day event where 10-15 customers are invited and all PM and PD staff currently not full-time assigned to development projects attends. Also, senior PM and PD management is present. Typically, somewhere between 20 and 50 Intuit staff attend the event. The customers invited to the solution jam are selected based on their potential interest in the areas addressed by the customer pain statements.

Before the solution jam, the coordinator connects with individuals and already created teams to facilitate the formulation of "customer pain" statements, i.e. hypotheses about areas that customers would prioritize development to take place in. During the solution jam, the team fleshes out the customer pain statement and starts to design a solution in terms of "mock-ups". Based on customer feedback, the team continues to iterate the design.

The customer feedback at this early stage in the process is invaluable for two reasons. Receiving customer feedback at the very start weeds out the well-intended but ill-conceived hypotheses of customer "pains" before any significant investment is made. Secondly, even if the customer pain is correctly identified and significant, engineers may still focus on the wrong aspects initially rather than fleshing out the design of the most important aspects.

During the final hour, all teams get the opportunity to present for a panel consisting of engineering and product management as well as customers. The panel decides which of the presented concepts are considered to be the most viable. For more details, see [2].

Non-commercial Deployment

About a year ago, Intuit introduced a new online solution to a new type of customer. Rather than immediately driving for commercialization, the business unit decided that it was most important to maximize the customer base and to achieve a significant network effect. The team followed the development process described in this paper and worked hard to get solution in the hands of customers within a few months.

Once the product was deployed (non-commercially) the team focused on growing the customer base as rapidly as possible as a proxy for future business success. The best way to drive growth of the customer base of free products is through word of mouth, i.e. through viral means. The team tracked the usage of the product and the satisfaction of the customer through product instrumentation and in-product surveys and drove development based on the active and passive customer feedback.

Commercial Deployment

Some of the SaaS properties of Intuit have been commercially deployed for several years already and have large customer bases. Over the last years, the team responsible for one of these properties has adopted a vigorous experimentation approach using A/B testing.

The team has adopted a weekly cycle of experimentation where it will decide which experiments to run early in the week, take one or two days to develop alternative implementations of aspects of the system, deploy the solution 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 of the product and the customer satisfaction. Obviously, this affects the financial and business goals of the product quite positively as well.

5 Conclusion

Connected systems, be it SaaS solutions or software-intensive embedded systems, are affected by several important factors including the prevalent business models on the web and beyond, the deployment infrastructure, customer expectations, the network connectivity of increasingly many products and the real-time nature of online solutions. The aforementioned factors have lead to the evolution of a new software development model that is different from development approaches for traditional software. First, it is focused on continuously evolving the software by frequently deploying new versions. Second, customers and customer usage data play a central role throughout the development process. Third, development is focused on innovation and testing as many ideas as possible with customers to drive customer satisfaction and, consequently, revenue growth. Stepping back, we can recognize that R&D in this context is best described as an "innovation experiment system" approach where the development organization constantly develops new hypothesis (ideas) and tests these with groups of customers.

The innovation experiment system approach to software development that we present in this paper focuses on three phases, i.e. pre-deployment, non-commercial deployment and commercial deployment, as well as three scopes, optimization, new features and new products, and presents example techniques for collecting active and passive customer feedback in each phase.

In the management literature, apply experimentation to business is a well-established concept, e.g. Davenport [4], though not practiced consistently and broadly in industry. Also, in innovation, the role of experimentation is broadly recognized, e.g. [9]. Finally, the notion of experimentation in online software is not novel. For instance, in response to Ray Ozzie's call to arms [6] to the Microsoft organization, Kohavi et al. [5] have developed an experimentation platform.

In practice, however, experimentation in online software is often limited to optimizing narrow aspects of the front-end of the website through A/B testing and in connected, software-intensive systems experimentation, if applied at all, is ad-hoc and not systematically applied.

In this paper, we take a broader perspective and address the scope ranging from optimization to new features and products. Consequently, the contribution of this paper is as follows. First, we present a first systematization of this innovation experiment system approach to software development for connected systems. Second, we illustrate the model using an industrial case study, Intuit.

In future work, we intend to strengthen the validation of the development approach by studying other companies that employ similar principles to software development.

References

[1] Adler, P.A., Adler, P.: Observational Techniques. In: Denzin, N.K., Lincoln, Y. (eds.) Handbook of Qualitative Research, pp. 377–393. Sage, Thousand Oaks (1994)

[2] Bosch, J., Bosch-Sijtsema, P.M.: Introducing Agile Customer-Centered Development in a Legacy Software Product Line. Accepted for Software Practice and Experience (February 2011)

[3] Corbin, J., Strauss, A.: Basics of qualitative research, 3rd edn. Sage, Thousand Oaks (2008)

[4] Davenport, T.H.: How to Design Smart Business Experiments. Harvard Business Review (February 2009)

[5] Kohavi, R., Crook, T., Longbotham, R.: Online Experimentation at Microsoft. In: Third Workshop on Data Mining Case Studies and Practice Prize (2009)

[6] Ozzie, R.: Ozzie memo: Internet services disruption (October 28, 2005), http://news.zdnet.com/2100-3513_22-145534.html

[7] http://en-ca.nielsen.com/content/nielsen/en_ca/product_families/nielsen_bases.html

[8] Reichheld, F.F.: The One Number You Need to Grow. Harvard Business Review (December 2003)

[9] Thomke, S.H.: Experimentation Matters: Unlocking the Potential of New Technologies for Innovation. Harvard Business Review Press (2003)