Towards Data-Driven Product Development: A Multiple Case Study on Post-deployment Data Usage in Software-Intensive Embedded Systems

Helena Holmström Olsson[1] and Jan Bosch[2]

[1] Department of Computer Science, Malmö University, Malmö, Sweden
helena.holmstrom.olsson@mah.se
[2] Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden
jan.bosch@chalmers.se

Abstract. Today, products within telecommunication, transportation, consumer electronics, home automation, security etc. involve an increasing amount of software. As a result, organizations that have a tradition within hardware development are transforming to become software-intensive organizations. This implies products where software constitutes the majority of functionality, costs, future investments, and potential. While this shift poses a number of challenges, it brings with it opportunities as well. One of these opportunities is to collect product data in order to learn about product use, to inform product management decisions, and for improving already deployed products. In this paper, we focus on the opportunity to use post-deployment data, i.e. data that is generated while products are used, as a basis for product improvement and new product development. We do so by studying three software development companies involved in large-scale development of embedded software. In our study, we highlight limitations in post-deployment data usage and we conclude that post-deployment data remains an untapped resource for most companies. The contribution of the paper is two-fold. First, we present key opportunities for more effective product development based on post-deployment data usage. Second, we propose a framework for organizations interested in advancing their use of post-deployment product data.

Keywords: Post-deployment data collection, product data, software-intensive systems, embedded systems.

1 Introduction

The evolution of software, and the variety of domains in which it is used, is impressive. From being considered a configuration mechanism for electronic systems, software has become the core of most modern systems supporting individuals, companies and societies [1]. As a result, more and more organizations are realizing that while their main business may be in areas such as hardware development, telecommunication, transportation, home automation or finance, the software part of their products is responsible for a majority of the functionality, as well as for a majority of the development costs and investments. In this transition, the ability to learn about customers, and especially the way in which customers use software functionality, becomes increasingly important. Hence, agile software development practices that are flexible, responsive and adaptive to customers [2, 3] are gaining momentum. In advocating customer collaboration and the importance of test-driven development practices [4], agile practices have attracted not only small software development companies, but also companies involved in large-scale development of software-intensive embedded systems.

However, while many companies have succeeded in applying agile practices and, as a result, leveraged the benefits of close customer collaboration and continuous validation of functionality in pre-deployment phases, there are few examples of companies that have succeeded in maintaining this close relationship to customers also after product deployment. One technique that has emerged due to the online nature of most software-intense systems today is the opportunity to continuously collect post-deployment data, i.e. data generated by the product after commercial deployment. This data can be operational data reflecting product performance, it can be diagnostic data recording product behavior, and it can be data indicating feature usage. For online technologies such as Web 2.0 software, software-as-a-service (SaaS) systems, and cloud computing services, the collection of post-deployment data is a well-established technique used for continuous collection of information about product usage. In this domain, companies like Microsoft [5] and Intuit [6] successfully collect post-deployment product data and use this as a basis for continuous improvement of existing products, as well as for input to innovation and new product development.

Interestingly, the opportunity to collect post-deployment data extends also to software-intensive embedded systems. Today, these systems are increasingly connected, bringing with it the opportunity to collect data from real-time usage of these systems. For example, companies developing systems within the telecom and automotive industry, i.e. mobile phones and cars, are starting to explore the advantages of collecting product data from their systems in the field. However, while this trend is discernible, research in this domain is still scarce, resulting in companies investing significantly in development efforts without having an accurate way of continuously validating whether the functionality they develop is of value to their customers.

In this paper, we present a multiple case study on three companies developing software-intensive embedded systems. While in different domains, all companies develop products consisting of an increasing amount of software, and they all collect large amounts of data from the products they release. In our study, we explore *what* data they collect, and especially, *how* this data is *used* for increasing their understanding of how their products are used by customers. The contribution of the paper is two-fold. First, we present key opportunities for more effective product development based on post-deployment data usage. These key opportunities were identified by key stakeholders within the companies, and work as drivers for increasing the use of post-deployment data. In our discussion, we refer to 'effective' as the ability to confirm that the functionality that is developed is used and appreciated by customers, i.e. the ability to continuously evaluate development investments and efforts. Second, we propose a framework for organizations interested in advancing their usage of post-deployment product data. Our framework reflects the different levels of post-deployment data usage, as well as the mechanisms needed for improving post-deployment data usage.

2 Background

2.1 Pre-deployment Data Collection

Typically, feedback on a software system is collected during pre-deployment phases, i.e. before and during development. Most often, this is done by applying techniques that allow customers to engage in problem definition, requirements engineering and system evaluation and validation. To involve customers in the development process is not a new phenomenon and it is well elaborated upon in user-centered development approaches such as participatory design [7], cooperative design [8], and joint-application design [9]. In these approaches, techniques such as use cases, scenarios, prototyping, stakeholder interviews, joint requirements sessions, joint application design sessions etc. are common. Likewise, alpha- and beta testing, observations, expert reviews, and prototyping are efficiently used before and during development in order to continuously validate that the functionality that is developed is of value to customers. Also, and as can be seen in research on agile methods [4, 10, 11, 12], and requirements engineering [13, 14], these techniques are efficient for capturing generic needs for mass-market products [15]. Likewise, large-scale software development organizations often use product management as a proxy for communicating customer feedback before and during development of the product [16].

2.2 Post-deployment Data Collection

System use evolve over time and hence, system characteristics need to be adjusted, adapted and updated according to emerging customer requirements and needs. This implies that mechanisms for post-deployment customer collaboration are as important as those used during pre-development and development phases of a system. With regard to post-deployment data collection, the concept of 'lead users' is often used to reflect close collaboration with innovative customers in order to use their feedback for improvement and innovation of products [17]. In similar, the 'software ecosystem' approach is referred to as a way for companies to involve customers in improvement activities that start after product delivery. As can be seen in research within this field, many software development companies have realized the economic and strategic potential of establishing collaborative communities of third-party organizations, customer organizations and individual developers who contribute to the product development process [18]. According to Jansen et al [19], a software ecosystem is described as a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. While the definitions of a software ecosystem are numerous, they all involve the notion of interactions, relationships and co-evolvement among stakeholders such as development organizations, suppliers and customers.

However, with the more recent introduction of Web 2.0 systems, cloud computing, and online Software-as-a-Service (SaaS) technologies, the opportunity to collect post-deployment data from the product itself has significantly increased. Due to the online nature of these systems, data is generated and hence, can be collected, as soon as customers use the systems, and the advantage is that the cost of collecting data from, and about, the customer is low [20]. Examples include the amount of time a user spends using a feature, the frequency of feature selection, the path that a customer takes through product functionality, etc. If continuously collected and analyzed, product data can be used as efficient input for improvement of the existing product, and as a basis for new product development and innovation. As a result, these online systems allow for an approach where instead of freezing the requirements before development starts, requirements evolve in real-time based on data collected from the products.

Interestingly, and as the focus of this paper, these benefits extend also to software-intensive embedded systems. Today, companies developing connected embedded systems, from mobile phones to cars, are starting to exploit the advantages of continuous collection of product data. For example, connected cars can collect diagnostic data such as fuel efficiency and energy consumption data, whereas telecom equipment can collect performance data such as real-time bandwidth, restarts, outages and upgrade success rate etc. Therefore, although the first area of post-deployment data collection can be found in online services such as 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, and that can be connected to the Internet for data access and retrieval. This includes software-intensive embedded systems intended for a mass-market from which evolving needs might be difficult to capture during pre-deployment phases, and where post-deployment customer involvement might be difficult to maintain.

3 Research Site and Method

3.1 Research Site

This paper presents on-going research based on a multiple case study conducted at three software development companies. Today, all the companies are collecting large amounts of data from the products they release to customers.

Company A is a provider of telecommunication systems and equipment, communications networks and multimedia solutions for mobile and fixed network operators. The company has a number of post-deployment data collection mechanisms in place, and is currently collecting data related to system operation and performance. For the purpose of this study, we met with key stakeholders at two different company sites in two different countries:

- *Site 1:* The first site is involved in the development and maintenance of nodes within the 3G networks. At this site, we met with a group of four people involving the head of system and architecture, two system managers and a deputy manager. During this meeting, a high level discussion on the topic was held, and we decided on what people that would be of relevance for a more focused group interview related to our research questions. As a second step, we conducted a group interview with people identified as key stakeholders within the organization. In total, we met with 15 people during the group interview, including product managers, project managers, support managers, product specialists and integration leaders. Finally, a workshop was held in which we met with all involved in the project, i.e. people from the initial meeting as well as people from the group interview, as well as a few additionally invited managers, to discuss and confirm our findings.

- *Site 2:* The second site is involved in the development, supply and support of media gateways for mobile networks. At this site, we conducted a group interview in which we met with a group of six people involving two department managers, a support manager, a senior specialist, a product manager and an integration leader.

Company B is a manufacturer and supplier of transport solutions for commercial use. The company has a number of sophisticated data collection mechanisms implemented in their products, and the majority of the data that is collected is related to diagnostics of the vehicles. The products consist of a huge number of microprocessors and sensors with the potential to collect data for more advanced purposes. For the purpose of this study we met with two attribute leaders, two developers, and one software expert focusing on software process improvement. In addition, we met with a group of managers and developers focusing on the human machine interface of the vehicles, and with significant experience on user interface design and different collaboration techniques for this.

Company C is world leading in network video and offers products such as network cameras, video encoders, video management software and camera applications for professional IP video surveillance. The company has a number of post-deployment data collection mechanisms in place in their products. The data they collect is primarily performance data related to the operational use of the products. For the purpose of this study we met with a group of seven people involving the company CTO, two team leaders, a test manager and two software architects. During this meeting, a high level discussion on the topic was held, and we decided on what people that would be of relevance for more focused group interviews. As a second step, we conducted five group interviews in which we met with developers, testers, system architects, product owners, project managers and product specialists. In total, we met with 5 groups and a total of 44 people.

3.2 Research Method

Our paper reports on a multiple case study [21] involving three companies involved in large-scale development of embedded software products. The main data collection method used was semi-structured group interviews with open-ended questions [22]. In total, eight group interviews were conducted. All group interviews were conducted in

English and lasted for two hours. During the interviews in company A (site 1) and B, we were two researchers sharing the responsibility of asking questions and facilitating the group discussion. Notes were taken and after each interview these notes were shared among the researchers. The interviews in company A (site 2) and company C were conducted by one of the researchers. These interviews were also recorded, and notes were taken to summarize the discussions. The recordings, as well as the summarizing notes, were shared between the two researchers to allow for a discussion and interpretation of the interview findings. In total, we had 18 hours of recorded interviews and 58 pages of summarizing notes. During analysis, the summary notes were used when coding the data, and as soon as any questions or potential misunderstandings occurred, the recordings were used to replay the discussion and capture all interview details.

In terms of data analysis, a qualitative grounded theory approach was adopted [23]. In this process, the empirical data was coded using open coding principles, and clusters and categories emerged as a result of reading the transcribed data with the intention to identify similarities in the respondents' experiences. A problem that has been identified in relation to qualitative research is that different individuals may interpret the same data in different ways [24]. This problem was addressed in two ways. First, the coding processes prescribed by grounded theory provide an audit trail of the process by which conclusions are reached. Second, we used a 'venting' method, i.e. a process whereby interpretations are discussed with professional colleagues [25]. By sharing notes, and by discussing the results of each group interview, we could develop an accurate understanding of the different contexts and hence, explore the research questions guiding this study: (1) *What* post-deployment product data do the software development companies involved in our study collect? (2) *How, and for what purposes,* is this data *used* for increasing their understanding of how their products are used?

4 Findings

In this section, we present our interview findings. Also, we present key opportunities for more effective product development. The key opportunities were identified during our study and expressed as important by our interviewees when reflecting on ways in which post-deployment data collection can help advance their development practices.

4.1 Post-deployment Data Collection

Our interviews reveal a wide range of data that is collected after product release. All companies have mechanisms in place for collecting data from their products, and all agree that post-deployment product data constitutes an important asset for product improvement and innovation. In company A, huge amounts of post-deployment data are collected in relation to system operation and performance. Information on re-starts, system outage, faults, card re-booting and upgrade success rate is continuously collected and used for assessing system performance and behavior. In addition, dimensioning data such as CPU load, licenses sold etc. serve as important input for system configuration and capacity, as well as for producing sale statistics and market assessments. Most often, post-deployment data becomes useful when a trouble report or a customer request is reported. As a way of answering to a query, company A uses this data to track system behavior, identify a problem, and compare system performance before and after an intervention with the system. As mechanisms for collecting this data, company A reports on a number of support logs and counters, monitoring systems such as the 'Event Based Monitoring System', customer satisfaction indexes based on Key Performance Indicators (KPI's), and tools for collecting and storing trouble reports, trouble tickets and customer requests. While all respondents agree that post-deployment data is important, they experience difficulties when it comes to getting an overview on what is collected and for what purpose. One of the respondents touch upon this when saying that: *"We have all the mechanisms we need for collecting data from our products…we only have to agree on what to collect and why…"*

In company B, post-deployment data is continuously collected in order to assess system behavior of the vehicle. For evaluation purposes and development investments, performance data such as speed, fuel efficiency, energy consumption, acceleration, road conditions etc. is collected. In addition, diagnostic data such as trouble codes, failure reports etc., is continuously collected by the electronic nodes in the vehicle in order to help trouble shoot a problem whenever the vehicle is handed in for service at a garage. After having read the data from the different electronic nodes, the mechanics send this data to a central database in which all diagnostic data is stored. Finally, data is collected in order to fulfill legislation purposes since company B is involved in development of products where safety regulations are immense. Besides the electronic nodes that collect data, there is also ways in which 'flight recorders' collect important data. A 'flight recorder' is an instrument that is put in vehicles to track system performance while driving. The recorders are put in a limited number of test vehicles and only used within restricted areas, and serve as important input in assessing system performance when the product has been taken into use. In similar with company A, the respondents at company B agree that they have sophisticated mechanisms in place to collect post-deployment data, and that the challenge is to make it useful within the organization: *"For the diagnostic data we depend on the mechanics reporting it to us. Once we get it we store it in a database that is hosted centrally. What I don't know is how widely it is used for purposes other than those of the mechanics…"*

In company C, post-deployment data is collected on a continuous basis, and primarily for observing and assessing system performance. Here, data on frames per second, stability and usage hours is important as well as configuration data on camera models and number of sites. In similar with company A and B, the interviewees at company C find post-deployment data useful for answering to customer requests and for supporting the system.

4.2 Post-deployment Data Usage

Based on our interview findings, we see that collection of post-deployment product data is common practice at the companies involved in our study.

However, usage of this data is still scarce. In company A, both sites report on system operation and performance as types of product data being continuously collected. Also, bug report data is collected in order to learn about system behavior and use. Based on this data, statistical analysis and trend analysis is done and there is the opportunity to learn about current system operation and future dimensioning needs. However, while performance data, such as upgrade success and downtime reports, is collected, company A report on difficulties to use the data. As it seems, customer data is used for troubleshooting and for maintaining the current version of the product, but very seldom for improving functionality or as a base for developing new functionality. Managers at both sites describe a situation in which data is collected but not used, and they find it difficult to analyze the data to, for instance, learn about what features that are used and what features that are "waste". This is reflected upon by one of the developers when saying: *"We have an idea on what functionality is used…based on sales statistics… but we don't really know"*.

In company B, diagnostic data is collected when the vehicle attends service at an authorized garage. Based on this data, data mining techniques are used to learn about system performance. However, while this data is useful for the next iteration of development, i.e. for the next version of the product family, it is collected with very long intervals and is not used for improving the current version of the product. Also, to integrate and to visualize the data is regarded difficult. One of the software architects emphasizes this when saying: *"We use the data only for troubleshooting purposes – when something is already a problem for the customer. What we would like to do is to find ways in which we could make more efficient use of the data…"*

In company C, there are no established techniques for post-deployment data usage. While large amounts of data are generated in the systems, these are not used to systematically improve current versions of the systems. As a result, interviewees feel that they have only limited knowledge on what features of their product that are used, and they feel that whenever post-deployment data is used it is for troubleshooting and support of problems that have already occurred. One of the project managers reflects on this when saying: *"We get feedback only on things that don't work...things that are problematic. This is not necessarily an indication on what is used the most…"*

In summary, our study shows that product data constitutes an enormous asset for understanding product use, for informing product management and for improving deployed products. However, post-deployment data usage is scarce, and while all companies report on this data as useful for troubleshooting and support, they recognize that mechanisms for continuous improvement of existing products, as well as for innovation of new products, are not in place.

4.3 Key Opportunities

While still in the process of establishing techniques for post-deployment data usage, all companies view this activity as critical for continuous validation of their development efforts. In our interviews, we asked the interviewees to share with us why they consider post-deployment data collection important, and for what purposes this data could be used in their organization. From the interviews, we learnt that there are a number of key opportunities associated with post-deployment data collection and usage:

- To continuously validate what functionality customers value.
- To improve requirements prioritization.
- To optimize customers' use of the product.
- To increase the ability to anticipate future customer needs.
- To increase delivery frequency of functionality.

As can be seen, the interviewees consider post-deployment data critical for improving validation, prioritization, optimization, anticipation and delivery frequency of functionality. As a driver for these processes, post-deployment data collection and usage is viewed as an area of great potential and with a number of opportunities associated to it.

5 Discussion

Everyone involved in development of a software product has ideas on how to make it better. Typically, these ideas are collected and prioritized during the road mapping and requirements engineering process as part of the yearly release cycle and before any development starts. Often, the selection and prioritization of ideas are based on expert opinions originating from previous experience, and predictions by product management [6]. These opinions form the basis of hundreds or dozens of person years of development effort, and the confirmation of the correctness of these opinions usually takes place after the product has been released to its customers.

However, with the introduction of Web 2.0 systems, cloud computing, and online Software-as-a-Service (SaaS) technologies, the opportunity to collect post-deployment data from the product itself has significantly increased and brought with it interesting opportunities related to increasing the understanding for how customers use a product. Due to the online nature of these systems, data can be collected as soon as customers use the systems, and the cost of collecting data from, and about, the customer is low [20]. These benefits extend also to software-intensive embedded systems such as telecom equipment and vehicles. With the majority of functionality being software, and with the opportunity to be connected to the Internet, these systems are now increasingly interesting from a data collection perspective. Therefore, although the first area of post-deployment data collection can be found in online services, the techniques can be easily transferred to any product that is able to collect and provide data about its real-time usage and performance. From a product development perspective this is interesting as it opens up for continuous improvement of existing systems, i.e. companies do not need to wait until the next version of a system before they can improve it. Instead, they can deploy new functionality to the current version of the system and to customers already using the system.

5.1 Post-deployment Data as Pre-development Input

In our study, we explore post-deployment data usage in three companies involved in development of software-intensive embedded systems. On the basis of qualitative interviews we see that all companies collect huge amounts of product data. However, even though the companies have data collection mechanisms in place, they find it difficult to integrate, communicate and visualize the data so that it becomes accessible for people in their organization. As a result, post-deployment product data is used primarily as input to the next pre-development phase, i.e. as input for the next version of the system but not for improving the current version, or for innovation of new functionality. This shortcoming is recognized in previous research in which concepts such as 'online experiments' [5], 'test-and-learn' mind-set [6], and 'innovation experiment systems' [6] are used to denote techniques that allow for real-time use of post-deployment product data.

5.2 Post-deployment Data as Troubleshooting Input

While our study conveys a number of opportunities for efficient use of post-deployment product data, our interviews show that this data is currently used only for troubleshooting, for product support and maintenance, and for bug fixing purposes. Hence, our interviewees confirm previous research [1, 6] in that even though collection of post-deployment data is increasing, there is a range of opportunities still to explore for companies involved in software-intensive embedded systems development. In all companies involved in our study, people have a good understanding for system-level operations and system-level performance, but they lack insight in individual feature usage and user patterns related to specific system functionality. Hence, we see that while the data is used for troubleshooting purposes, it is not used for more advanced purposes such as improvement of existing functionality or innovation of new products.

5.3 Post-deployment Data Usage Framework

As a result of our study, and the insights provided by our interviewees with regard to how post-deployment data is used, we propose a framework that supports companies interested in advancing their usage of post-deployment product data (figure 1).

In the framework, we outline the different levels of post-deployment data usage, i.e. different purposes for which this data is used. At the first level, operational data represents data that helps companies understand how the system performs, i.e. data generated during real-time use and that is collected in order for companies to get a system understanding. However, at most companies, operational data is collected without a clear purpose of how to analyze and use it and therefore, primarily a high-level understanding of the system is obtained. At the second level, diagnostic data represents data that is collected with the specific purpose of troubleshooting activities. Here, data is collected in order to support bug fixing and error correction, and to provide input for system maintenance. At this level, a more systematic collection of data is required and companies make use of effective data storage in order to document and trace troubleshooting and maintenance processes. The third level, i.e. feature usage, represents a level at with companies collect data that helps them understand usage (or non-usage) of individual features. In comparison to the high-level system understanding that is provided by collection of operational data, this level requires mechanisms and tools that allow for a more sophisticated data analysis in which usage patterns of specific features can be discerned. At the two most advanced levels, i.e. feature improvement and new feature development, data is collected in order to support continuous improvement of current functionality, as well as for innovation and development of new features. To achieve these levels, advanced development practices that allow for new software functionality to be easily tested and integrated are required.

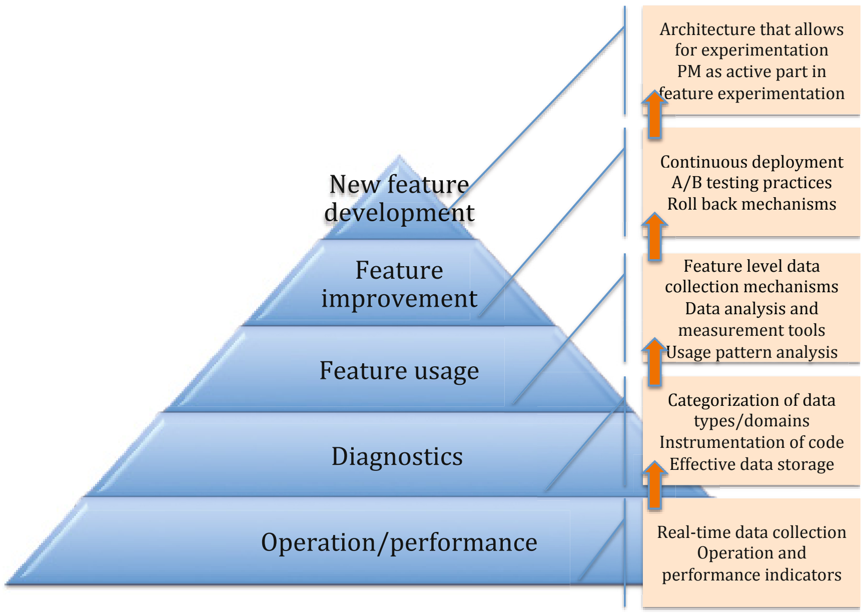
New feature development
Feature improvement
Feature usage
Diagnostics
Operation/performance
Architecture that allows for experimentation
PM as active part in feature experimentation
Continuous deployment
A/B testing practices
Roll back mechanisms
Feature level data collection mechanisms
Data analysis and measurement tools
Usage pattern analysis
Categorization of data types/domains
Instrumentation of code
Effective data storage
Real-time data collection
Operation and performance indicators

Fig. 1. Post-Deployment Data Usage Framework

Based on insights acquired during our interview study, as well as on our previous work on how to advance beyond agile development practices [26], our framework suggests mechanisms (see the boxes to the right in the figure) that are needed for climbing these levels and move towards more advanced use of post-deployment product data. These mechanisms are related to organizational processes and development practices that will allow a company to not only collect post-deployment data for operational and diagnostic purposes, but to use this data for more advanced purposes such as a detailed understanding of features, improvement of features and innovation of new features.

6 Conclusions

In this paper, we explore collection and usage of post-deployment product data. We highlight the existing limitations in post-deployment data usage, and the untapped resource that post-deployment product data remains. Based on a multiple case study at three software development companies, we present the following findings:

- Post-deployment product data is used primarily as input to the next pre-development phase, i.e. as input for the next version of the product, but *not* for improving the current version of the product, or for innovation of new functionality.
- Post-deployment product data is used for diagnostic purposes, i.e. troubleshooting and maintenance activities, and provide a good system-level understanding of operation and performance, but does *not* provide insight in individual feature usage and user patterns related to specific system functionality.
- There are a number of *key opportunities* that work as drivers for advancing the practices related to the collection and usage of post-deployment product data. These key opportunities are related to organizational processes and development practices that allow for more effective product development.

Finally, we propose a framework for post-deployment data usage in which we outline what development practices and organizational mechanisms that need to be in place for advancing the usage of post-deployment product data and hence, advance the development of software-intensive embedded systems.

Acknowledgements. This study was funded by Malmö University as part of a research collaboration between Malmö University and the Software Center at Chalmers University of Technology and University of Gothenburg, Sweden. We would like to thank the three companies involved in the study for the time and engagement allocated by all interviewees at these companies.

References

1. Bosch, J., Eklund, U.: Eternal embedded software: Towards innovation experiment systems. In: Margaria, T., Steffen, B. (eds.) ISoLA 2012, Part I. LNCS, vol. 7609, pp. 19–31. Springer, Heidelberg (2012)
2. Desouza, K., Awazu, Y., Jha, S., Dombrowski, C., Papagari, S., Baloh, P., Kim, J.Y.: Customer-Driven Innovation. Research Technology Management, 35–44 (May-June 2008)
3. Fogelström, N.D., Gorschek, T., Svahnberg, M., Olsson, P.: The Impact of Agile Principles on Market-Driven Software Product Development. Journal of Software Maintenance and Evolution: Research and Practice 22, 53–80 (2010)
4. Highsmith, J., Cockburn, A.: Agile Software Development: The business of innovation. Software Management, 120–122 (September 2001)
5. Kohavi, R., Longbotham, R., Sommerfield, D., Henne, R.M.: Controlled experiments on the web: survey and practice guide. Data Mining and Knowledge Discovery 18(1), 140–181 (2009)

6. Bosch, J.: Building Products as Innovation Experiment Systems. In: Cusumano, M.A., Iyer, B., Venkatraman, N. (eds.) ICSOB 2012. LNBIP, vol. 114, pp. 27–39. Springer, Heidelberg (2012)
7. Schuler, D., Namioka, A.: Participatory design: Principles and practices. Erlbaum, Hillsdale (1993)
8. Grønbæk, K., Kyng, M., Mogensen, P.: CSCW challenges: cooperative design in engineering projects. Communications of the ACM 36(6), 67–77 (1993)
9. Wood, J., Silver, D.: Joint application development. John Wiley & Sons, New York (1995)
10. Abrahamsson, P., Conboy, K., Wang, X.: 'Lots done, more to do': the current state of agile systems development research. European Journal of Information Systems 18(4), 281–284 (2009)
11. Beck, K.: Embracing Change with Extreme Programming. Computer 32(10), 70–77 (1999)
12. Larman, C.: Agile and Iterative Development: A Manager's Guide. Addison-Wesley (2004)
13. Nuseibeh, B., Easterbrook, S.: Requirements engineering: A roadmap. In: Proceedings of the 22nd International Conference on Software Engineering (ICSE), Limerick, Ireland, June 4-11 (2000)
14. Bennett, K.H., Rajlish, V.T.: Software maintenance and evolution. In: Proceedings of the 22nd International Conference on Software Engineering (ICSE), Limerick, Ireland, June 4-11 (2000)
15. Sommerville, I.: Software engineering, 9th edn. Addison-Wesley, Boston (2010)
16. Larman, C., Vodde, B.: Scaling lean & agile development: Thinking and organizational tools for large-scale scrum. Addison-Wesley (2008)
17. Von Hippel, E.: Democratizing Innovation: The evolving phenomenon of user innovation. Journal für Betriebswirschaft 55, 63–78 (2005)
18. Iansiti, M., Levien, R.: Strategy as ecology. Harvard Business Review 82, 68–78 (2004)
19. Jansen, S., Brinkkemper, S., Cusumano, M.: Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry. Edward Elgar
20. Bosch, J.: Building products as innovation experiment systems. In: Cusumano, M.A., Iyer, B., Venkatraman, N. (eds.) ICSOB 2012. LNBIP, vol. 114, pp. 27–39. Springer, Heidelberg (2012)
21. Walsham, G.: Interpretive case studies in IS research: Nature and method. European Journal of Information Systems 4, 74–81 (1995)
22. Runesson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14 (2009)
23. Corbin, J., Strauss, A.: Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Sage, California (1990)
24. Kaplan, B., Duchon, D.: Combining qualitative and quantitative methods in IS research: A case study. MIS Quarterly 12(4), 571–587 (1988)
25. Goetz, J., LeCompte, D.: Ethnography and Qualitative Design in Educational Research. Academic Press, Orlando (1984)
26. Olsson, H.H., Alahyari, H., Bosch, J.: Climbing the Stairway to Heaven. In: Proceedings of the 38th Euromicro Software Engineering Advanced Applications (SEAA) Conference, Cesme, Turkey, September 5-7 (2012)