Make Up Your Mind: Towards a Comprehensive Definition of Customer Value in Large Scale Software Development

Helena Holmström Olsson¹ and Jan Bosch²

¹ Malmö University, Department of Computer Science and Media Technology,
205 06 Malmö, Sweden
e-mail: helena.holmstrom.olsson@mah.se

² Chalmers University of Technology, Department of Computer Science and Engineering,
412 96 Göteborg, Sweden
e-mail: jan.bosch@chalmers.se

Abstract. Today, connected software-intensive products permeate virtually every aspect of our lives and the amount of customer and product data that is collected by companies across domains is exploding. In revealing what products we use, when and how we use them and how the product performs, this data has the potential to help companies optimize existing products, prioritize among features and evaluate new innovations. However, despite advanced data collection and analysis techniques, companies struggle with how to effectively extract value from the data they collect and they experience difficulties in defining what values to optimize for. As a result, the impact of data is low and companies run the risk of sub-optimization due to misalignment of the values they optimize for. In this paper, and based on multi-case study research in embedded systems and online companies, we explore data collection and analysis practices in companies in the embedded systems and in the online domain. In particular, we look into how the value that is delivered to customers can be expressed as a value function that combines different factors that are of importance to customers. By expressing customer value as a value function, companies have the opportunity to increase their awareness of key value factors and they can establish an agreement on what to optimize for. Based on our findings, we see that companies in the embedded systems domain suffer from vague and confusing value functions while companies in the online domain use simple and straightforward value functions to inform development. Ideally, and as proposed in this paper, companies should strive for a comprehensive value function that includes all relevant factors without being vague or too simple as is the case in the companies we studied. To achieve this, and to address the difficulties many companies experience, we present a systematic approach to value modelling in which we provide detailed guidance for how to quantify feature value in such a way that it can be systematically validated over time to help avoid sub-optimization that will harm the company in the long run.

Keywords: Data-driven development, customer value, value function, comprehensive value function, systematic value modeling.

1 Introduction

We live in a data-centric world where companies across domains have been collecting data from their customers and from products in the field for decades. With software-intensive products that are increasingly being connected to the Internet, everything we do and the way the products behave can be monitored and recorded [1], [2], [3], [4], [5], [6]. While it is easy to lament about the potential negative implications, there are great benefits provided by the data in terms of unprecedented quality of user experience and product performance as companies use the data for significantly improved decision-making during development. For example, the automotive companies started collecting diagnostics data from vehicles already in the early 90's to use as the basis for maintenance whenever a truck or a car was taken to a garage for service. More recently, and as a result of vehicles becoming connected to the Internet and with practices such as continuous deployment in place [3], [7] car manufacturers can push software updates to the vehicle on a continuous basis without taking the vehicle out of traffic. This allows for preventive maintenance and has become key to prolong the lifetime of a vehicle and avoid costly repairs. Also, this proactive use of data allows car manufacturers to detect errors while the vehicle is running and before the customer is even aware of them. Similarly,

online companies have been collecting data from customers and from products in the field for years [8], [9]. With pure software products and with the opportunity to run frequent A/B experiments with customers [9], these companies collect data revealing not only system operation and performance but also customer behaviors and preferences. For instance, game development companies collect data that help them understand when players experience problems, what situations that cause players to leave a game and how likely players are to continue to the next level in the game. Based on these insights, decisions are taken on what actions to take to either help the player reach the next level or have the player struggle depending on whether the analysis of the data shows that the player is likely to quit the game or if the player will stay and keep trying. As the company gains more and more insights from the data they advance their overall understanding of how the product is used by its customers. Over time, these insights are hardcoded into the product in order to have the system itself perform actions that were previously a manual effort.

However, despite the promise of big data, the fact of the matter is that companies collect extensive amounts of data but fail to make use of the data sitting in the data warehouses [10]. Most often, their data-driven practices inform only smaller improvements at a team level [4], [5], while having little, or no, impact on high-level business decisions and goals. Previous research shows that more than half of the features developed do not deliver on the value expected at the time of feature prioritization [6], [10], [11], [12]. In addition, we see that companies spend vast amounts of development resources in commodity functions of their products, rather than on innovative and differentiating features [10], [13]. Adopting data-driven and evidence-based decision making can provide a powerful antidote for these ways of investing in and performing software development [6], [10].

In this paper, we explore the use of data-driven development practices in companies in the embedded systems as well as in the online domain. Based on our findings, we see that embedded systems companies suffer from *vague* value functions, i.e. multi-dimensional value functions that are not agreed upon and that causes teams to optimize for conflicting factors as they include both relevant and irrelevant aspects. An example of a vague value function is ‘user experience’ which is too broad and includes too many aspects for development teams to use as an effective factor to optimize for. The online companies on the other hand, typically use simple value functions, i.e. a one-dimensional value function, including only one aspect, that are straightforward and very narrow. An example of a simple value function is ‘conversion’ which includes only one relevant aspect for development teams to use as a factor to optimize for. Ideally, and as proposed in this paper, companies should strive for a *comprehensive value function* that includes all relevant aspects without being vague or too simple as is the case in the companies we studied. A comprehensive value function might still be multi-dimensional but it is a value function that is agreed upon by all stakeholders involved and that aligns team level optimizations with high-level business goals. To achieve this, and to address the difficulties many companies experience, we present a systematic approach to value modelling that helps companies (1) *model feature value*, (2) *execute an experiment to evaluate this value proposition* and (3) *evaluate the outcome of the experiment in order to determine if the expected value was achieved*. Our approach involves ten steps and provides detailed guidance for how to quantify feature value in such a way that it can be systematically validated over time to help avoid misalignment of factors at different levels and sub-optimization that will harm the company in the long run. In our previous research [5], we present an initial version of the approach. In this paper, we extend the approach and we provide *industry-specific guidance* for each of the ten steps as the embedded systems domain and the online domain are inherently different in character which influence the way in which data-driven practices are adopted and applied.

The contribution of this paper is three-fold. First, we show empirical evidence of data-driven development practices that are currently used in companies in the embedded systems and in the online domain. Second, we identify the key challenges that these companies experience in relation to feature experimentation and we define three types of value functions of which the comprehensive value function is proposed as the most beneficial one for companies across domains. Third, we present a systematic approach to value modelling in which we provide detailed guidance for how to quantify value in such a way that it can be systematically validated over time.

The paper is organized as follows. In section 2, we outline the background and the limitations of current practices in relation to identifying and validating customer value. In section 3, we outline the research method and the case companies involved in our study. In section 4, we present our empirical findings and we identify key challenges and three types of value functions. In section 5, we present an approach to systematic value modeling and in section 6 we conclude the paper.

2 Background

2.1 Requirements Engineering: What customers say they want

There exist a number of well-established practices and techniques for defining and determining customer value. For decades, requirements engineering has been the primary practice for identifying, eliciting, and documenting customer requirements for a software system. It involves the identification of requirements and the modeling of these in order to develop an agreed upon understanding of what a future software system will look like in order to provide value to the customer [14]. As the basis for the requirements engineering tradition, there exist a wide range of techniques to help the development team ensure that the requirements are complete, consistent and relevant [15]. As recognized in previous research, the goal of the requirements engineering process is to identify what functionality to build before development starts in order to avoid, or at least reduce the risk, of costly rework [14], [15]. The reasoning is that mistakes that are revealed in the later stages of the development process are more expensive to correct, and that this can be avoided by identifying a stable set of requirements before development resources are allocated and system design and implementation activities start.

However, over the years, a number of limitations have been recognized in relation to the traditional requirements engineering tradition. Primarily, the assumption that customer requirements can be identified before development starts is questioned as more agile ways of working have emerged and started being adopted across industry domains [16], [17]. Moreover, although there exist various techniques to help elicit customer expectations, these tend to focus on what customers say they want rather than what they do in practice [10], [11]. As a result, requirements that can be made explicit are the ones that can be fully captured in the traditional elicitation process while the more implicit ones are difficult to capture. Third, with the most common techniques for requirements elicitation being brainstorming, interviews, focus groups, observations, prototyping and analysis of documents and interfaces [18], the amount of data that is collected is relatively small and primarily qualitative in its nature. More recently, the requirements engineering tradition has adopted more agile and collaborative ways of working in relation to planning, executing and reasoning about requirements [19], [20], [21], [22]. As a result, traditional challenges such as communication gaps between development teams and customers, as well as the problem of a too large scope for development, can be overcome. However, the transition towards more agile requirement engineering practices is not without challenges and it has proven difficult to striking a good balance between agility and stability, and ensuring sufficient competence in cross-functional development teams which is one of the core practices in agile development but not yet well-established in more traditional development contexts.

2.2 Data-driven Development: What customers do in practice

As a way to address many of the limitations in requirements-driven engineering, data-driven development practices are gaining momentum as the new practice to learn how a software system performs in the field, how it is used by its customers and what usage behaviors that evolve over time [6], [13], [25]. With systems being connected to the Internet and technologies that facilitate data collection and analysis, companies are increasingly adopting continuous deployment practices. Continuous deployment is a software engineering practice in which incremental software updates and improvements are developed, tested and deployed to the production environment on a continuous basis and in an automated fashion [7]. As a result, queries can be processed more frequently to provide software developers and managers with rapid feedback on system performance as well as on user behaviors. This reflects an interesting shift from a requirements-driven approach to software engineering in which the understanding of customer value is often based on opinions and internal assumptions, or by asking the customer, towards a data-driven approach in which companies use customer data and data from products in the field as the basis for a continuously evolving understanding of customer value [5], [8], [11]. Data-driven companies acquire, process, and leverage data in order to create efficiencies, develop new product concepts and navigate the competitive landscape [4], [5], [6], [11], [12], [25], [26], [27], [28]. In the transition towards data-driven development practices, there is a predictable set of steps that companies evolve through [6]. Starting off as ad-hoc in their data collection practices with manual and time-consuming processing of data, companies advance by implementing automated processes for data collection and analysis mechanisms. At the final step of this transition, companies use customer and product data as the basis for decision-making at all levels in the organization. As a result, previous opinions and assumptions about customer needs are challenged by continuous experimentation and validation with customers. In online software development companies, customer data from A/B tests are the norm for evaluating ideas and understanding customer value [9], [11], [23]. Already, there exists a number of models that provide companies with guidance on how to run A/B tests

and experiments with customers. These models describe the experiment cycle [11], [29], the data collection techniques that companies need to implement, the data types they should collect from customers and from products in the field [30] and the underlying infrastructure that is required for running successful A/B tests and experiments with customers [23], [24].

However, although data-driven development practices have already proven useful for continuous definition of customer value [4], [5], [6], [12] and with existing models providing advice on roles and infrastructures for experimentation, there exists little guidance for how to express and model value in order to ensure that data collection and experiments conducted by development teams support and align with the overall business strategy and goals. Based on our experiences from working with a number of software-intensive companies across domains, we see that development teams run the risk of using a set of metrics identified at the team-level in order to optimize certain features, but without the opportunity to continuously ensure that these metrics align with high-level metrics associated with the overall product portfolio and long-term business goals [5], [31]. Moreover, most organizations adopting data-driven practices report on difficulties in having the insights generated from experiments accumulate [4], [5], [12], [26]. Despite an infrastructure and an organization that enables continuous experimentation the knowledge that is generated is limited and tend to stay within a team, a unit or a department [6], [26]. As a result, data-driven development practices support only smaller improvements of specific features rather than having effective impact on business value.

2.3 Product Management: Changing practices

As one of the most important roles during the process of identifying customer value, product managers seek to understand customer needs and translate these into product requirements [32]. During the requirements engineering process, the translation of customer needs into product requirements is key for being able to prioritize among requests and for making decisions on how to scope an upcoming product release. In this process, product managers use a number of techniques to prioritize among different feature requests and, when possible, they work closely to customers to better understand what constitutes customer value and how this can be realized in a new software system. Often, the identification as well as the validation of a customer request is facilitated by early mock-ups and prototypes that help customers communicate their needs. However, although traditional product management practices strive for close collaboration with customers, and recognize this as key for understanding what will deliver value to a certain customer segment, this value proposition is seldom validated in practice. Often, product management is unaware of the intent behind the features requested by customers. As a result, they implement their own interpretation of that intent with the risk of developing an internal model of what constitutes customer value and use this as the basis for implementation [26]. In our previous research [11], we identified the ‘open loop’ problem referring to a situation that most large software development companies experience in that product managers lack the opportunity to validate whether the features they prioritize are actually used by customers and generating the revenue they expected. In many companies, we see that the understanding of what constitutes customer value is more often based on existing and internal assumptions rather than continuous and external validation with customers [26].

Recently, and as a result of data-driven development practices gaining momentum, the traditional role of product management is changing. With a continuous flow of customer and product data, management and development teams learn about their deployed products and what features that add value to customers. To extract value from this data, product managers work in close collaboration with development and data analytics teams with dashboards as dynamic reporting mechanisms. During feature development, development teams, product management teams and data analytics teams use insights from the dashboard to prioritize features, to scope a release and to continuously learn what adds value to customers. In previous work [6], we report on the transition towards data-driven development and we detail how traditional roles such as e.g. product management change as new roles such as e.g. data scientists emerge. In this research, we see that as companies move closer to data-driven development practices, customer and product data becomes the primary tool for decision-making, as well as the basis for product improvements and innovations.

2.4 Value Modeling: Qualitative and quantitative validation of value

Value modeling refers to the process of understanding customer value in products, systems and services [33], [34], and helps structuring information associated with value creation. Typically, value modeling frameworks support the process of defining and estimating customer value and they provide guidelines for market segmentation, discover of new business opportunities and product launch. The dialog and customer interaction that is part of the value modeling

process is used to discover and determine which potential product features and functionality would create the most value for customers. As the most common technique, on-site interaction is used to frame and define new features with the main purpose of focusing on benefits and value [33]. There are several methods and approaches used to create customer value models. All of these approaches appear to depend on substantial customer interaction and on-site interviews and observations of customers' challenges related to the product or service being valued.

However, although value modeling is not a new phenomenon, there are few attempts to quantify feature value in such a way that it can be systematically validated over time. As recognized in previous research, most existing approaches appear to depend on substantial customer interaction and on-site interviews and observations of customers' challenges related to the product or service being valued [33], [34]. Also, existing approaches tend to focus on evaluation of existing products and what value they provide to customers when deployed, rather than being used already before development starts and in order to better understand the interplay between team-, system- and business factors that are all important for continuous improvement and optimization of customer value.

3 Research method

3.1 Case study research

The research reported in this paper builds on longitudinal multi-case study research conducted in close collaboration with software-intensive companies in the embedded systems and in the online domain. As recognized in literature, case study research is becoming increasingly attractive in the software engineering field as an approach for exploring real-life challenges where control over the context is not possible, and where the focus is to create an in-depth understanding of a particular phenomenon [35], [36]. For the purpose of this research, we engaged in close collaboration with a total of twelve companies to investigate (1) their current use of data driven development practices, (2) what challenges they experience and (3) how to address these challenges in order to improve modelling of value. As the motivation for this research, and as recognized in previous literature, we see that although data-driven development practices have proven useful for continuous validation of customer value [4], [5], [6], [12] and with existing models providing advice on roles and infrastructures for experimentation, there exists little if any, guidance for how to express and model value to ensure that feature experiments conducted by development teams support and align with the overall business strategy and goals.

Also, previous literature identifies sub-optimization as one of the main challenges with experimentation, i.e. having teams define metrics but without the opportunity to continuously ensure that these metrics align with high-level business metrics [5], [31]. In our research, we seek to address this situation by proving a systematic approach for value modeling that takes into account both team and business level metrics. Below, we provide a short description of each of the case companies that were involved in our study:

Embedded systems companies:

- *Company A* is a provider of communication systems and equipment for mobile and fixed network operators. The company is among the world leading companies in its domain and with globally distributed development spanning multiple sites across the world.
- *Company B* is a manufacturer and supplier of transport solutions for commercial use. The company is highly distributed with hardware and software units and sites across the world and a workforce covering hardware, software, mechanical and electrical engineering.
- *Company C* is a software company specializing in navigational information, operations management and optimization solutions. The company is part of a multinational aerospace corporation manufacturing products and services for commercial use as well as for the defense.
- *Company D* is a developer of network video surveillance solutions with a large number of partners in close to 200 countries around the world.
- *Company E* is a pump manufacturer producing circular pumps for e.g. heating and air conditioning and with a large and globally distributed workforce.
- *Company F* is a developer of connected monitoring and alarm solutions with more than ten thousand employees around the world.

Online companies:

- *Company G* is a provider of business intelligence and visualization software to close to 50 000 customers worldwide.
- *Company H* is a fast-growing e-commerce company that provides payment services for individuals.
- *Company I* is a media streaming company with offices around the world and with a fast growing work force.
- *Company J* is an entertainment media company developing online games for individuals all around the world.
- *Company K* is an entertainment media company developing online games for individuals all around the world.
- *Company L* is a large multinational developer of IT solutions for businesses, developers, individuals and children.

In addition to the case companies, we organized a *knowledge exchange workshop* to which we invited a selected sub-set of the case companies, and also the following two external companies that we know from other engagements and that we regard especially valuable for exploring data driven development practices:

- *External company 1*: A growing start-up company focusing on data integration and analytics.
- *External company 2*: A company that provides online travel and accommodation services for individuals. The company is among the world leading companies in its domain.

3.2 Data collection and analysis

3.2.1 Case company interviews and workshops

During the study, we conducted a series of activities in the case companies (the research process is summarized in Figure 1). In the embedded systems companies, we conducted qualitative interviews, workshops, demonstration- and validation sessions during a period of five years. The collaboration with these companies is an on-going relationship and part of a research initiative involving eleven software-intensive companies and five universities. Our research on data-driven development was initiated already in 2012 and over the years, and for the purpose of this research topic, we have met with project managers, product managers, product owners, software developers, software and system architects, technical leaders as well as sales and marketing people in all the companies. We have engaged with a large number of different units within the companies and we have met with people at top-level management as well as people at the team level and we have interviewed a sub-set of these. In addition to frequent site visits to all case companies where we explored their practices in detail, we arranged a series of cross-company workshops where representatives from all companies met to discuss topics related to data collection and analysis, feature experimentation, value modeling of features and how to more effectively learn from the practices that constitute data-driven development. These workshops were valuable not only from a research perspective but also from a knowledge sharing perspective as they allowed all participating company representatives to continuously learn from each other and share experiences. To validate our findings, we organized cross-company validation sessions to which all companies were invited to evaluate our results and to provide additional feedback to the findings. Our empirical data consists of interview transcripts, meeting and workshop notes, presentation material from the companies, notes from informal meetings, e-mails and telephone conversations.

The collaboration with the online companies was initiated in early 2015 when we were seeking to complement our insights from the embedded systems domain with insights from additional domains. For the purpose of our research, the online companies represent a more advanced approach to data collection and analysis and although they experience a number of challenges they are not restricted to hardware dependencies and safety regulations to the same extent as the embedded systems companies we work with. Due to this, they are able to accelerate their data-driven development practices and feature experimentation such as e.g. A/B testing has become the norm. So far, our collaboration has resulted in a series of meetings with project and product managers in all the companies, three presentations for larger groups (20-25 people) within three of the companies, workshop sessions and qualitative interviews with roles such as developers, project managers and product managers.

3.2.2 Knowledge exchange workshop

In addition to the interactions with the case companies as mentioned above, we organized a knowledge exchange workshop in April 2017 to which we invited a selected sub-set of the case companies (two online companies and one embedded systems company), and also two external companies that we know from other engagements and that we regard especially valuable from a data driven development perspective. One of these companies is a start-up focusing on data integration and analytics. The other company provides online travel and accommodation services. The purpose

of this workshop was to allow for face-to-face knowledge exchange between some of the most advanced companies in terms of data collection and analysis and we had invited all companies to present their current practices, the challenges they experience and what future practices and/or solutions they consider key in going forward.

3.2.3 Data analysis

During analysis of our empirical data, the transcribed interviews, the workshop notes and all other written documentation were read carefully by both researchers with the intention to identify recurring elements and concepts [37]. An interpretive approach was adopted for analysis [35] as it has similarities with the qualitative grounded theory approach but with a less strict coding process [35], [37]. Following this approach, we documented our impressions during the research to then carefully reflect on what could be learnt and what implications could be drawn from the field data [35]. In addition, white board illustrations from the workshops were documented using a camera and these were revisited during data analysis.

Figure 1. Summary of research process

4 Case study findings

In this section, we present and discuss our case study findings. These findings are based on the interviews and the workshop sessions with the case companies, as well as the knowledge exchange workshop session with two external companies and a selected set of the case companies. In section 4.1, we present four practices for data-driven development and we explore to what extent the case companies use them. In section 4.2, we report on the key challenges that the case companies experience in relation to feature experimentation as one of the key practices of data driven development. Finally, in section 4.3, we identify three types of value functions and we propose one of these as the desired one for effective value modeling.

4.1 Data driven development practices

We present our findings by categorizing them into the four main practices that we see companies use when adopting data-driven development. The four practices are the following:

1. **Modeling of expected outcome (pre-development) practices** in the companies. These reflect the companies' ability to define the expected value of a new feature in a way that allows for continuous measuring and evaluation.
2. **Feature experimentation practices** in the companies. These reflect the companies' ability to expose customers to different versions of a software feature and use data from customers and from products in the field to inform further development.
3. **Post-experiment reflection practices** in the companies. These reflect the companies' ability to adjust and adapt development based on the insights gained from experimentation, as well as the ability to revise and redefine existing metrics.
4. **Generalized use of experiment results practices** in the companies. These reflect the companies' ability to accumulate knowledge from multiple experiments, as well as the ability to have data influence decision-making at all levels in the organization.

In Table I, we summarize our case study findings as presented in the below sections (section 4.1.1 – 4.1.4). In the table, Company A – F represent the embedded systems companies and Company G – L represent the online companies. We use the following notation to indicate to which extent the different data driven practices are used in each of the case companies:

- **N:** Not done at all
- **A:** Ad-hoc, non-systematic
- **T:** Team-level, aligned with team goals
- **B:** Business-level, aligned with business goals

Data-driven practices	Case Companies											
	A	B	C	D	E	F	G	H	I	J	K	L
Generalized use of experiment results	A	N	N	T	N	T	N	T	T	B	A	T
Post-experiment reflection	T	N	A	T	N	T	N	T	T	T	A	T
Feature experimentation	T	N	A	A	N	T	N	T	T	T	A	T
Modeling of expected outcome (pre-development)	B	N	A	A	A	T	A	B	A	A	N	B

Table 1. Current use of data-driven practices in the case companies.

4.1.1 Modeling of expected outcome (pre-development)

The first practice we studied reflects the companies' ability to define and model the expected value of a new feature before development. In general, we saw few attempts to this. Rather than defining the expected behavior before developing a new feature most companies react only afterwards and use data from already deployed features to evaluate and determine the impact of the new feature. For example, Company C reports on a situation when they experienced severe problems in customer uptake with one of their new features. To address this, they instrumented the already deployed feature with metrics such as e.g. number of clicks, time spent, menu scrolling and selection etc., that would help them understand how customers used the feature. However, the company doesn't have practices that help them define such metrics before development which is reflected in a quote from a developer when saying: *"We are good at reactive data-driven development but not so good at proactive development where we first define hypotheses and then test and measure if these are true."*

Similarly, Company D and E track system behavior using metrics such as e.g. configuration data and performance data that help define the expected outcome of new features but this is an ad-hoc and a non-systematic practice. There are, however, a few examples of pre-development modeling of features in two of the embedded systems companies we studied. In company F, the team developing the mobile app for their system has defined metrics they optimize for when modeling new features. These metrics are agreed upon within the team and they are used to inform development. As the most advanced company among the embedded systems companies, Company A collect and analyze data in order to prove value of existing features and to discover opportunities for new features. Teams have a large set of counters that help them track feature behaviors and these work as the basis for understanding the outcome of new features. For this company, to "prove" value beforehand is also something they monetize as a service to customers.

In the online companies, modeling of feature value is a mature practice in Company H, I and L. Here, systematic and continuous measuring of key metrics help the organization track and evaluate the outcome of new features and what they add in terms of value to customers. Company H has a clear separation between teams and responsibilities and each team does pre-development modeling of the features they are responsible for. In Company I, hypotheses are modeled and evaluated in a lab environment based on metrics such as e.g. daily/monthly users, click through rate, conversion and customer satisfaction that reflect customer value. As the most advanced company in our study, Company L predict the value of all new features in an 'opportunity analysis' where they use qualitative and quantitative techniques to model expected outcome. They have a large number of metrics such as e.g. click through rates, number of sites visited, conversion rate, time spent on a certain site etc. that they use for tracking their features and during early feature modeling they typically use a sub-set of these. The typical process is described by one of the

project managers: *"We use the data and insights from previous experiments to help us predict early if a new feature will add value or not. These insights might also lead to new metrics. We do hypotheses early on and then we validate with data that we collect over time."*

4.1.2 Feature experimentation

The second practice we studied reflects the companies' ability to expose customers to different versions of a software feature and use data from customers and from products in the field to understand what adds value to customers. Contrary to pre-development modeling, the majority of the companies we studied run some type of feature experiments with customers and in their deployed products. While the embedded systems companies typically run sequential experiments focusing on product performance (e.g. different optimizations algorithms, different configuration alternatives etc.), the online companies run multiple experiments in parallel and with a focus on customer behavior and product use (e.g. different versions of menus and icons, different colors on links, variation in page design etc.). The experiments that are run in the embedded systems companies are more ad-hoc in nature and they tend to have less impact on the overall business goals. Instead, they focus on helping teams advance their understanding of specific features and how these perform.

The majority of the online companies run feature experiments at a team level with key metrics that help the different R&D teams optimize certain aspects of the features they develop. Company J has around ten experiments running in parallel in their products and Company I run dozens of experiments per month. In Company L, all teams use a set of key metrics that reflect high level business goals such as e.g. 'revenue' and 'customer satisfaction'. From these, subsets of metrics are derived for each individual team in order to align team optimization of certain features with the overall business goals. The profound impact of feature experimentation is well expressed in the following quote: *"Everything is instrumented and we base what to ship on the results we get from experimentation and the actions we see users take"*. Another project manager adds to this when saying: *"We know how much you click, how fast you click, how long you stay, when you come back... This translates into metrics that span across page level metrics, script errors, performance, frequency of pop-up's, queries per user, click through rate, sessions per user, re-visits of users..."*.

4.1.3 Post-experiment reflection

The third practice, i.e. post-experiment reflection, is fairly established in both domains for those companies that run feature experiments. Of the twelve companies we studied, only two (Company E and G) don't do post-deployment reflection with the reason being that they have not yet introduced feature experiments as part of their development process. In all other companies, some form of adjustment is done based on insights from experiments although these might still be ad-hoc or at a team level rather than adjustments that influence overall business metrics. Also, most companies report on at least yearly revision and redefinition of existing metrics as the experiments help them gain further insights into these. This is illustrated in a quote from one of the product managers at Company L: *"The metrics reflect the accumulated knowledge of the experiments. We stick to a collection of metrics and these develop and improve over time due to the insights we get in the experiments."*

Company L is far ahead when it comes to post-experiment reflection. They update their metrics on a regular basis, they use these to gain new insights about new features and they continuously adjust and adapt development based on the data these metrics yield. In similar, Company I report on a continuous evaluation of metrics to ensure that team-level metrics reflect business-level goals. Here, the interplay between metrics is becoming increasingly important as reflected on by one of the developers:

"We used to understand and optimize individual metrics. Now we realize that there are relations between metrics that might make them look negative from one perspective while they have a positive effect from a different perspective."

4.1.4 Generalized use of experiments results

The fourth and last practice we studied reflects the companies' ability to accumulate knowledge from multiple experiments, as well as the ability to have data influence decision-making at all levels in the organization. In relation to this practice, we see some differences between the embedded systems companies and the online companies. With defined metrics that align team-level efforts with business level strategy, one of the online companies manages to have results from team-level experiments directly influence the product. One example is their ability to continuously track and analyze the way in which customers use their product in order to discern patterns that can then be implemented in the software. In this way, the product is instrumented to take action to e.g. help a user to solve a problem, without

having any manual effort by the development team involved in this. In most other online companies, the generalization of experiment results is done at a team level with multiple teams learning from each other's experiments to determine what metrics to use, how these depend on each other and when impact on high-level metrics is so low that team-level metrics need to be redefined. Also, and as noted in Company I, knowledge sharing between teams is important to avoid the risk of sub-optimizing: *"Teams need to understand that they cannot affect key metrics negatively by doing team-level tweaks but without being aware of the high-level metrics."*

It should be noted however, that although there exist examples of advanced use of data as presented above, also the online companies struggle with generalizing from their experiment results. Even in the companies we consider the most advanced, people struggle with accumulating insights and have experiments influence decision-making. This is evident in the following quotes from a project manager in one of these companies: *"To accumulate knowledge from experiments is very difficult as there aren't a lot of insights that you can generalize. Typically, we see some general behaviors but an overall 'framework' that is applicable over time is difficult to achieve."*

In the embedded systems companies, the generalization of experiment results happens mostly at the team level or they happen ad-hoc, but there are also companies in which this does not happen at all. With poor definition of metrics that align teams and business and with non-existing experiment practices, two of the companies fail in having the data they collect effectively influence decision-making practices in the company.

4.1.5 Summary and discussion of findings

Based on our empirical findings, we see that the first practice, i.e. modeling of expected outcome is an ad-hoc and non-systematic practice in the majority of the case companies, including both the embedded systems and the online companies. This situation reflects the lack of established value modeling practices and results in a feeling of being 're-active' rather than 'pro-active' as reported by some of the interviewees. Also, the lack of pre-development value modeling practices makes it difficult for the companies to improve effectiveness of their experiments as there is no established agreement on what values to optimize for before an experiment is initiated. As can be seen in Table 1, only four of the twelve case companies have established practices for modeling of expected outcome which means that they have agreed upon metrics for continuously validating the value of the features they develop. The lack of established practices for modeling of expected outcome is evident in both the embedded systems and the online companies and it seems a challenge regardless of domain or size of the company.

Second, and in relation to feature experimentation, our study shows that most of the online companies run experiments on a team level, meaning that the development teams in these companies run e.g. A/B tests with customers to evaluate what version of a feature adds most value to customers. In the embedded systems companies, this practice is less common and happens more on an ad-hoc basis and when there is a specific need. This is not surprising, as the embedded systems companies experience a complex dependency to hardware, as well as strict regulations, which typically cause experimentation practices to be more difficult to implement. However, some of the embedded systems companies run feature experiments with customers in certain parts of their systems. As an example, the automotive companies involved in our study run feature experiments in the infotainment system as this is not safety-critical nor hardware dependent. The majority of the online companies have established practices for feature experiments with platforms and tools that help teams run multiple experiments in parallel in their products. Here, the products are instrumented with metrics and customers are exposed to different versions of e.g. a website interface to determine what version is the optimal one. As a result, smaller improvements can be done on a continuous basis based on the data they collect. What is challenging in the online companies, as well as in the embedded systems companies, is the risk of having teams optimize for team level values that are not agreed upon and that might not align with business level goals. This is due to the lack of established value modeling practices and can be seen in all companies involved in our study as they report on difficulties with having team level experimentation align with business level goals.

Third, we see that post-experiment reflection, when done, happens at the team level in both company domains. This means that teams evaluate experiment results and as a result, they have the opportunity to adjust development and revise existing metrics. What is challenging in most companies however, is that the evaluation of experiments, and the impact of these, stays at the team level without reaching, aligning or influencing the system- and high-level metrics in the organization. As reflected in the interview findings, this might cause a situation in which teams optimize certain features to improve certain metrics, but without verified alignment with overall business goals. While this shortcoming can be found in all companies it is even more common in the large embedded systems companies with highly complex products involving hardware, software, mechanical and electrical parts.

Finally, the ability to accumulate knowledge from multiple experiments, as well as the ability to have data influence decision-making at all levels in the organization is challenging. Referred to as generalized use of experiment results, this practice is far from a well-established practice and something that companies in both domains struggle with regardless of size or product characteristics. In our study, only one company in the online domain has well-established practices for this and runs feature experiments that influence not only team level metrics and individual features, but also system level and high-level metrics that improve overall business goals.

4.2 Key challenges

During our case study research, as well as in the knowledge exchange workshop that was organized in collaboration with two external companies, we identified a number of challenges in relation to feature experimentation. In our previous research [5], [12], we report on an initial set of these challenges and below we add to these based on new findings in our research. As can be seen below, the challenges reflect the companies' difficulties to extract value from the data they collect and the lack of practices that help them agree on what to optimize for. Below, we list the challenges we identified in the companies involved in our study:

- **Lack of mechanisms to accumulate and scale the impact of experiments:** In the companies we studied, experiments support smaller improvements of features rather than having an impact on high-level business decisions such as larger re-designs, new product development or innovation initiatives. Often, experiments at a team level move the needle at a very small delta and the impact of an experiment is limited. This is demotivating for teams and it is also harmful for the company.
- **Insufficient guidance for identification of key value factors:** The identification of what values to optimize for is challenging as these are typically not agreed upon by all stakeholders. In many of the companies we studied, experiments are conducted without an agreement on the expected outcome. As a result, key value factors remain unclear and evaluation of the experiment result becomes difficult as there are no indicators to help judge success or failure.
- **Lack of practices for alignment of team level, system level and high-level business value factors:** To influence high-level business metrics is difficult in all companies we studied. Team level metrics that are used for experimentation focus on short-term goals, smaller improvements and factors that change fast. On the contrary, high-level business level metrics focus on long-term goals, bigger innovations and factors that change slowly. Despite an awareness of the different levels and the different metrics that serve these, most companies find it difficult to align value factors at the team level with value factors at the business level.
- **Poor evaluation criteria to assess experiment success or failure:** As a result of unclear key factors and difficulties to establish a common agreement on what to optimize for, the case companies experience a situation in which they find it difficult to assess whether an experiment is successful or not. In the embedded systems companies, this typically results in organizational resistance and hesitation towards further adoption of data driven development practices. In the online companies, and despite the fact that these practices are already well-established, this causes low confidence in data and experiment results.

In addition to the challenges mentioned above, the embedded systems companies experience difficulties in *translating and applying solutions* from the online domain to their own domain. While there exists guidance for how to optimally run experiments in the online domain, it is often unclear to what extent this guidance is applicable across domains and for companies outside of this domain.

4.3 Value function types

As can be seen in the challenges identified above, one of the main obstacles is the lack of agreed upon values to optimize for. While a couple of the online companies have an agreed upon value function in which they express the key factors that teams should optimize for, we learnt that the majority of the embedded systems companies experience a complex situation in which it is unclear what values teams should target. This problem became increasingly clear during the knowledge exchange workshop to which we invited a subset of the case companies and two external companies that are advanced in the way they collect and use data in their businesses. During the workshop, we could confirm our findings and advance our understanding of how important it is to have a value function that is agreed upon and that aligns team level experimentation with high-level business goals. If not, teams run the risk of sub-optimizing certain features without ensuring that the improvements they do align with overall business goals.

Based on the discussions during the knowledge exchange workshop, we identified three types of value functions that the companies use. In Table 2, we describe these in detail.

Type of value function	Definition:
<i>Vague</i> (confusing; including relevant and irrelevant elements or aspects)	Multi-stakeholder ecosystems and/or a multi-dimensional value function that is not agreed upon
<i>Comprehensive</i> (complete; including all relevant elements or aspects)	Multi-stakeholder ecosystem and/or multi-dimensional – and agreed upon – value function
<i>Simple</i> (straightforward; including one relevant element or aspect)	Single-stakeholder ecosystem and/or single-dimensional value function (e.g. ‘conversion’)

Table 2. Value function types.

The majority of the online companies in our study use a simple value function such as e.g. ‘conversion’. The value function is agreed upon by all teams and it is aligned with overall business goals. For the online companies, the use of a simple value function allows a straightforward approach to experimentation including a single, and relevant, factor to optimize for. In one of the companies that attended the knowledge exchange workshop, the only value function they focused on for a long period of time was ‘conversion’. This allowed for significant growth of the business in a short period of time. However, and as reported by this company, this approach might lead to a limited set of business opportunities in the long run due to the narrow scope of the value function. On the contrary, the embedded systems companies struggle with vague value functions such as e.g. ‘user experience’. A vague value function is broad in scope and difficult to define. As reported by one of the embedded systems companies, this might cause a situation in which experiments are run by teams without contributing to overall business strategy and goals, or in the worst case, even contradict these. During the study, we learnt that all case companies seek to achieve what we define as a *comprehensive value function*. This value function is an agreed upon and complete value function that allows teams to run experiments that align with business goals. In one of the online companies we studied, the transition from a simple value function towards a comprehensive value function had just started as a way to expand scope and allow for new business opportunities.

5 Discussion

Based on our empirical findings, we see that although the case companies collect large amounts of data they fail to extract value from this data. In particular, they fall short in pre-development modeling of the expected value of new features they introduce to their products. As a result, they are unable to validate if the value they had expected, and the revenue they were looking for, is actually delivered by introducing the new feature into the system. In earlier work, we refer to this situation as the ‘open loop’ problem [11]. The ‘open loop’ problem can, however, be addressed by creating a better understanding and definition of the expected value of a new feature.

5.1 Towards systematic value modeling

In what follows, and to address these shortcomings of existing approaches, we propose a *systematic approach to value modeling* that helps companies to identify the key values they optimize for. In Table 3, we detail our approach to value modeling. Our approach includes ten steps that help companies (1) *model feature value*, (2) *execute an experiment to evaluate this value proposition* and (3) *evaluate the outcome of the experiment in order to determine if the expected value was achieved*. The approach provides detailed guidance for how to quantify feature value in such a way that it can be systematically validated over time to help avoid misalignment of factors at different levels and sub-optimization that will harm the company in the long run. In our previous research [5], we present an initial version of the approach. In Table 3, we extend this and we provide *industry-specific guidance* for each of the ten steps as the embedded systems domain and the online domain are inherently different in character which influences the way in which data-driven practices are adopted and applied.

Step:	Definition:	Industry-specific guidance: Embedded systems companies	Industry-specific guidance: Online companies
Step 1: <i>Identify key value factors</i>	Key value factors are what the feature is intended to improve and for which metrics are implemented. Key value factors are defined for high-level, system level and team level goals.	As embedded systems companies suffer from vague value functions, this step should involve detailed break-down of high-level factors into system and team level factors that can be more easily measured.	As online companies optimize for simple value functions, this step should involve expansion of team level factors to system and high-level factors to help increase impact of experiments.
Step 2: <i>Agree on intended direction of key value factors</i>	Decide success metrics, i.e. if a factor should go up or down as a result of the experiment. In order to know whether an experiment is successful or not, there needs to be a common agreement on the direction of key value factors.	As embedded systems companies develop components that are typically interdependent, this step should involve careful analysis of what factors influence and depend on each other.	As online companies optimize for simple value functions, this step should involve careful analysis of an experiment to avoid sub-optimization by teams that optimize features that are fully decoupled.
Step 3: <i>Normalization of key value factors</i>	The key value factors need to be normalized so that they operate on a comparable scale. For example, while value factors such as ‘new users’ and ‘recurrent users’ have similar ranges, a factor such as ‘revenue’ will have a very different range and cannot be easily compared. To cater for this, each key value factor needs to have an upper and a lower boundary and a mapping function to map the result to a value between 0-1.	To improve preciseness, this step should involve the identification of factors that can be frequently monitored and measured.	To improve impact, this step should involve alignment of metrics so that team level experiments have the chance to influence high-level factors.
Step 4: <i>Prioritization of key value factors</i>	Teams need to prioritize the relative weight of value factors, i.e. the importance of these factors. The	Ensure that the prioritization of value factors is agreed upon by stakeholders such	Identify champions of factors besides the key value factor that drove the original

	prioritization is expressed in a value function by assigning the relative weight to each value factor, e.g. $0.2 \times \text{new users} + 0.3 \times \text{recurring users} + \dots$ etc. It is typically easiest to have the sum of the weights equal to 1.	as R&D teams, product management, marketing and sales, as these might have very different understandings of what constitutes a successful experiment.	success of the business. Providing priority to these additional factors requires dedicated attention.
Step 5: <i>Translation of hypothesis into an experiment</i>	Conversion of hypothesis into an executable experiment. The hypothesis represents an idea, i.e. an invalidated assumption about what adds value to customers, and is picked from the feature backlog in which potential new features are described.	Develop instrumentation for relevant data and execute the experiment in a selected set of systems and/or a part of a system.	Develop instrumentation for the new value factors and ensure quality is in line with core value factors.
Step 6: <i>Selection of a system or user base for experimentation</i>	Selection of appropriate base of deployed systems or active users. If the experiment, and the key value factors, relate solely to product operation and performance, a system or sub-system is selected. If the experiment is directed towards improving or optimizing customer-oriented functionality a suitable user segment needs to be identified. The selected base is subsequently divided into an experiment group and a control group.	Start the experiment small scale by selecting internal systems and/or users. Alternatively, identify a friendly customer with whom the experiment is initiated.	Reconsider customer segmentation in the context of the added value factors as it might require changes.
Step 7: <i>Establishing baseline</i>	The baseline is set before the experiment is initiated and represents the values of the key factors without any interference with the system. Typically, this is done by providing the control group and the experiment group with the same base solution ('A') in order to verify that there is no statistical difference between these groups (the 'B' version is turned off to ensure that both groups try the same solution before initiating the experiment).	Collection of baseline behavior needs to be highly prioritized as it is often missing for embedded systems. This data is critical for the next steps in the experiment process as it allows for ensuring the quality of the experimentation infrastructure.	Although online companies often have an established infrastructure for experimentation, the new value factors require renewed verification of the base line and guardrail metrics to ensure alignment with business goals.
Step 8: <i>Establishing length of experiment</i>	To capture both weekdays and weekends, experiments run at least 7 days. Some experiments run for 2-3 weeks to make sure that a change is not only a novelty	Due to system complexity and the relationship to hardware, the experiment should run for a longer time	The new value factors may have different longitudinal patterns than the existing value factors.

	effect. Too long experiments should be avoided as they cause users or systems to enter, leave and re-enter the experiment. This makes the results less trustworthy.	period than what is common practice in online companies.	Consequently, the organization needs to experiment with different durations to determine the optimal length for an experiment.
Step 9: Initiate experiment	The control group gets the base solution 'A' (as established as the baseline) and the experiment group gets the "treatment" solution 'B' of the software. During the experiment, user or system behaviors are measured to decide which version of the software that has the most positive impact on the key value factors. While the experiment is running there are two activities that need to be conducted. Activity 1 is to constantly verify the guardrail metrics (as described in step 3 above). Activity 2 is to constantly verify statistical validity of the data between the experiment and the control group.	Gradual rollout of the experiment and the ability to cancel the experiment without relying on network connections is critical to avoid unintended consequences in safety-critical systems.	Incorporate the fact that the new value function may cause existing factors to decrease and others to increase in value and still lead to a positive outcome. This is different from when a company optimize for a single (and simple) value factor.
Step 10: Completion of experiment	There are three possible results of an experiment. Either, there is statistical validity of the data and the experiment can be concluded. In this case, the base solution ('A') or new solution ('B') is implemented for all users or systems. Alternatively, no statistical validity is achieved. In this case, the experiment is concluded and the base solution ('A') is implemented for all users. Finally, if the experiment causes systems to get outside their guardrails, the experiment is aborted.	Ensure careful definition of guardrail metrics and real-time tracking of these to avoid potential economic damage from the system operating outside of specifications.	As the value function is more comprehensive, reaching statistical validity of experiments may require more users, a longer running time or both. Ensure proper tracking of guardrail metrics that may be affected by the new value factors.

Table 7. Value modeling framework.

It should be noted that there are different levels of key factors depending on what level of the organization they aim to measure, i.e. *high-level factors* referring to high-level business goals, *system level factors* referring to mid-level system goals and *team level factors* referring to improvement of specific features. While team level factors are 'leading indicators', i.e. easy to influence and fast changing, system level and especially high-level factors are 'laggard indicators', i.e. hard to influence and slow changing. For example, 'daily/monthly users' is an example of a team level factor in online companies ('leading indicator') that changes quickly and that can be easily influenced. 'Recurring

use' (system level) and 'customer satisfaction' (high-level) on the other hand, are examples of system and high-level factors ('laggard indicators') that change slowly and that are hard to influence. As a feature experiment is conducted to influence and change one or more factors each factor that is measured has to have the ability to change at the same rate as the development team can make changes to the software. If a team runs an experiment aiming at a too high level factor the risk is that the influence is so low, and the time it takes to see any impact is so slow, that any potential improvement cannot be distinguished from the "noise" in the data. Therefore, and in order to allow for continuous improvement of individual features, as well as alignment of high-, system-and team level metrics, we suggest that for a company that works with value modeling according to the approach we suggest, there are two types of experiments that need to be conducted. The first type is **horizontal experiments**. These are regular A/B tests with the purpose of exploring what variant of functionality that is the optimal one. They focus on changing leading indicators as defined by team level metrics and they move at the rate of the development cycle. For such a test, the impact of some variable on a set of factors are measured to learn e.g. what version of a website interface leads to better conversion rate (i.e. the proportion of people viewing an advertisement and going on to buy the product, click on a link, etc.), what menu alternative that attracts most users or what color of a link that increases the click rate etc. The second type is **vertical experiments**. These are A/B tests focusing on the relationship between leading and laggard indicators as defined by team level, system level and high-level metrics, and they are conducted to verify the relationship between these. In both types of experiments, and to ensure accuracy and relevance of the experiment, guardrail metrics need to be defined to set the ranges within which the system is allowed to operate. For any experiment, there are four types of outcomes. There are the *expected positive factors* and the *expected negative factors*. In addition, and as a potential surprise to most companies, there are the *unexpected positive factors* and the *unexpected negative factors*. Also, as the whole notion of experimentation and testing hypotheses is an iterative process, additional factors might have to be added to the value function during an experiment. For example, if a team realizes that what they optimize for, and the outcome of such an experiment, proves to harm overall sales this needs to be adjusted by adding or removing factors for subsequent experimentation.

5.2 Validity and limitations

With respect to validity, we made sure to continuously share and validate our understanding of key concepts such as 'data-driven development', 'customer value', 'value modeling' etc. with the company representatives. In this way, the researchers and the company representatives got a common view of the most important concepts. Also, all empirical data was independently assessed by two researchers to enhance reliability. With respect to external validity, the research we present is based on case study research in companies representing the embedded systems and the online domains. Hence, our results cannot be directly translated to other companies or other domains. However, with a large number of companies from different industries represented in our study, we believe that our findings, and the approach we propose for value modeling, is relevant to other companies operating in similar industries. Finally, it should be recognized that the approach we suggest has not yet been fully validated. However, it was derived based on insights from a large number of companies that also acknowledge the solution we propose.

5 Related work

There are a number of recent studies on data driven development and on feature experiments [9], [11], [24], [28] and the topic seems to only grow in terms of interest in the software engineering community [1]. Current studies tend to focus predominantly on the roles involved (e.g. data analysts, data scientists, product manager, software developer etc.), the task at hand (e.g. develop roadmap, design and analyze experiments, develop products, deploy product etc.), the technical infrastructure (e.g. APIs, experiment databases, analytic tools, instrumentation, integration and deployment systems etc.) but less on the business context of which the experiment is part. In similar, value modeling has been previously studied and there are relevant studies primarily in the management literature and in relation to new product development. However, the topic is less common in relation to software businesses and engineering of new software functionality. In previous studies, value modeling is defined as the process of understanding customer value in products, systems and services [33], [34], and as a way to help structuring information associated with value creation. Typically, value modeling frameworks support the process of defining and estimating customer value and they provide guidelines for market segmentation, discover of new business opportunities and product launch. The

dialog and customer interaction that is part of the value modeling process is used to discover and determine which potential product features and functionality would create the most value for customers. As the most common technique, on-site interaction is used to frame and define new features with the main purpose of focusing on benefits and value [33]. There are several methods and approaches used to create customer value models. All of these approaches appear to depend on substantial customer interaction and on-site interviews and observations of customers' challenges related to the product or service being valued.

6 Conclusions

In this paper, we propose an approach that help software-intensive companies (1) *model feature value*, (2) *execute an experiment to evaluate this value proposition* and (3) *evaluate the outcome of the experiment in order to determine if the expected value was achieved*. We do so by exploring the use of data-driven development practices in companies in the embedded systems as well as in the online domain. In particular, we explore how the value that is delivered to customers can be expressed as a value function that combines different factors that are of importance to customers.

Based on multi-case research in companies in the embedded systems and in the online domain, we see that embedded systems companies suffer from vague value functions while online companies use simple value functions to inform development. Ideally, and as proposed in this paper, companies should strive for a comprehensive value function that includes all relevant factors without being vague or too simple as is the case in the companies we studied. To achieve this, we present a systematic approach to value modelling in which we provide detailed guidance for how to quantify feature value in such a way that it can be systematically validated over time. Contrary to many existing techniques for value modeling, our approach allows for continuous experimentation with high-level, system level and team level factors and it provides guidance for how to verify alignment between these factors. Moreover, and for each step in the approach, we provide industry-specific guidance as companies in the embedded systems and in the online domain are inherently different in character which influence the way in which data-driven practices are adopted and applied.

By adopting the approach to systematic value modeling as proposed in this paper, companies have the opportunity to improve their effectiveness, increase customer value and avoid sub-optimization that might harm the business in the long run. To conclude, and as a result of our long-term engagements with case companies in the embedded systems and in the online domain, we share the following recommendations for how to advance data-driven development practices:

- Continuous collection and analysis of *customer and product data* is critical for understanding product use and for taking accurate decisions regarding what adds value to customers.
- *Systematic modeling of feature value* by defining team level, system level and high-level value factors helps to significantly improve the impact of experiments.
- Transition towards a *comprehensive value function* increases accuracy of experiments by focusing on an agreed upon value function in which irrelevant factors are excluded.
- Adoption of *systematic value modeling practices* to ensure alignment between key value factors at different levels reduces the risk of local sub-optimization that might harm the company in the long run.

Acknowledgements

We would like to thank all the companies involved in our research for their support and collaboration, their engagement and their valuable input during interview and workshop sessions.

References

1. Bosch, J. (2016) Future Trends in Software Engineering. *IEEE Software*, Jan/Feb. Vol. 33, Issue 1, pp. 82 – 88.
2. Olsson H. H., and Bosch J. (2013). Towards Data-Driven Product Development: A Multiple Case Study on Post-Deployment Data Usage in Software-Intensive Embedded Systems. *In Proceedings of the Lean Enterprise Software and Systems Conference (LESS), December 1-4, Galway, Ireland.*
https://doi.org/10.1007/978-3-642-44930-7_10
3. Eklund, U., and Bosch, J. (2012). Eternal Embedded Software: Towards Innovation Experiment Systems. *In Proceedings of the 5th International Symposium, (ISoLA), Heraklion, Crete, Greece, October 15-18, pp. 19-31.*
4. Fabijan, A., Dmitriev, P., Olsson, H. H., and Bosch J. (2017). The Evolution of Continuous Experimentation in Software Product Development: From Data to a Data-driven Organization at Scale. *In Proceedings of the 39th International Conference on Software Engineering (ICSE), May 20 – 28th, Buenos Aires, Argentina.*
<https://doi.org/10.1109/ICSE.2017.76>
5. Olsson, H. H., and Bosch, J. (2017). So Much Data; So Little Value A multi-case study on improving the impact of data-driven development practices. *In Proceedings of the Ibero American Conference on Software Engineering (CIBSE), May 22nd – 23rd, Buenos Aires, Argentina.*
6. Olsson, H.H., and Bosch, J. (2017). Towards Evidence-Based Development: Learnings from Embedded Systems, Online Games and Internet of Things. *IEEE Software*, October 2017, Issue 99.
7. Savor, T., Douglas, M., Gentili, M., Williams, L., Beck, K., and Stumm, M. (2016). Continuous deployment at Facebook and OANDA. *In Proceedings of the 38th International Conference on Software Engineering Companion, pp. 21-30.*
<https://doi.org/10.1145/2889160.2889223>
8. Bosch, J. 2012. Building Products as Innovations Experiment Systems. *In Proceedings of 3rd International Conference on Software Business, June 18-20, Cambridge, Massachusetts.* https://doi.org/10.1007/978-3-642-30746-1_3
9. Kohavi, R., and Longbotham, R. (2015). Online Controlled Experiments and A/B Tests. *Encyclopedia of Machine Learning and Data Mining*, pp. 1–11.
10. Bosch, J. (2017). *Using data to build better products: A hands-on guide to working with data in R&D – the basics.* Createspace Independent Publishing Platform.
11. Olsson, H.H., and Bosch, J. (2014). From Opinions to Data-Driven Software R&D: A Multi-Case Study on How To Close The 'Open Loop' Problem. *In Proceedings of EUROMICRO, Software Engineering and Advanced Applications (SEAA), August 27-29, Verona, Italy.* <https://doi.org/10.1109/SEAA.2014.75>

12. Olsson, H., and Bosch, J., and Fabijan, A. Experimentation that Matters: A multi-case study on the challenges with A/B testing. *In Proceedings of the International Conference on Software Business (ICSOB), June 12th – 13th, Essen, Germany, 2017.* https://doi.org/10.1007/978-3-319-69191-6_12
13. Bosch, J. (2016). *Speed, Data, and Ecosystems: Excelling in a Software-Driven World*. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series. CRC Press.
14. Pohl, K. (2010). *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated. <https://doi.org/10.1007/978-3-642-12578-2>
15. Paetch, F., Eberlein, A., and Maurer, F. (2003). Requirements Engineering and Agile Software Development. *In Proceedings of the 12th IEEE International Workshop on Enabling Technologies, p.* 307-313.
16. Zave, P., and Jackson, M. (1997). Four Dark Corners of Requirements Engineering. *ACM Transactions on Software Engineering and Methodology*, Vol. 6, No. 1, January, pp. 1-30. <https://doi.org/10.1145/237432.237434>
17. Beck, K. (1999). *Extreme Programming Explained*. Addison-Wesley.
18. Alexander, I., and Beus-Dukic, L. (2009). *Discovering Requirements: How to Specify Products and Services*. John Wiley.
19. Inayata, I., Salwah Salima, S., Marczakb, S., Danevac. M., and Shamshirbandd, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, Vol. 51, pp. 915 – 929. <https://doi.org/10.1016/j.chb.2014.10.046>
20. Bjarnason, E., Wnuk, K., and Regnell, B. (2011). A Case Study on Benefits and Side-Effects of Agile Practices in Large-Scale Requirements Engineering. *In Proceedings of the First workshop on Agile Requirements Engineering*, <http://lup.lub.lu.se/record/2173152>.
21. Maalej, W., Nayebi, M., Johann, T. and Ruhe, G. (2016). Toward Data-Driven Requirements Engineering. *IEEE Software*, Vol. 33, Issue 1, pp. 48 – 54. <https://doi.org/10.1109/MS.2015.153>
22. Becker, C., Betz, S., Chitchyan, R., Duboc, L., Easterbrook, S.M., Penzenstadier, B., Seyff, N., and Venters, C.C. (2016). Requirements: The Key to Sustainability. *IEEE Software*, Vol. 33, Issue 1, pp. 56 – 65. <https://doi.org/10.1109/MS.2015.158>
23. Fagerholm, F., Guinea, A.S., Mäenpää, H and Münch, J. (2014). Building Blocks for Continuous Experimentation. *In Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering.*, pp. 26–35. <https://doi.org/10.1145/2593812.2593816>
24. Fagerholm, F., Sanchez Guinea, A., Mäenpää, H., and Münch, J. (2016). The RIGHT model for continuous experimentation. *Journal of Systems and Software* 123, pp. 292-305. <https://doi.org/10.1016/j.jss.2016.03.034>

25. Fabijan, A., Olsson, H.H., and Bosch, J. (2015). Early Value Argumentation and Prediction: An Iterative Approach to Quantifying Feature Value. *In Proceedings of the 16th International Conference on Product-Focused Software Process Improvement (PROFES), December 2-4, Bolzano, Italy, December 2nd – 4th, 2015*, pp. 16–23. https://doi.org/10.1007/978-3-319-26844-6_2
26. Fabijan, A., Olsson, H.H., and Bosch, J. (2016). The Lack of Sharing of Customer Data in Large Software Organizations: Challenges and Implications. *In Agile Processes in Software Engineering and Extreme Programming (XP)*, pp. 39–52. https://doi.org/10.1007/978-3-319-33515-5_4
27. Patil, D.J. (2011). *Building Data Science Teams*, Oreilly Radar, pp. 1–25.
28. M. Kim, T. Zimmermann, R. DeLine, and A. Begel, (2015). The Emerging Role of Data Scientists on Software Development Teams, No. MSR-TR-2015–30, pp. 0–10
29. Olsson, H.H., and Bosch, J. (2015). Towards Continuous Customer Validation: A conceptual model for combining qualitative customer feedback with quantitative customer observation, LNBIP, vol. 210, pp. 154–166, 2015.
https://doi.org/10.1007/978-3-319-19593-3_13
30. Fabijan, A., Olsson, H.H., and Bosch, J. (2015). Customer Feedback and Data Collection Techniques in Software R&D: A literature review. *In Proceedings of the 6th International Conference on Software Business (ICSOB). June 10-12, Braga, Portugal*. https://doi.org/10.1007/978-3-319-19593-3_12
31. Fabijan, A., Dmitriev, P., Olsson, H.H., and Bosch, J. (2017). The Benefits of Controlled Experimentation at Scale. *In Proceedings of the 43th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Vienna, August 30th – September 1st, Austria, 2017*. <https://doi.org/10.1109/SEAA.2017.47>
32. Paasivaara, M., Heikkilä, V.T. and Lassenius, C., (2012). Experiences in scaling the product owner role in large-scale globally distributed scrum. *In 2012 IEEE Seventh International Conference on Global Software Engineering (pp. 174-178). IEEE*. <https://doi.org/10.1109/ICGSE.2012.41>
33. Lindstedt, P., and Burenius, J. (2003). *The Value Model: How to Master Product Development and Create Unrivalled Customer Value*. Nimba.
34. Ulaga, W. (2001). Customer Value in Business Markets: An Agenda for Inquiry. *Industrial Marketing Management*, 30, 315–319.
[https://doi.org/10.1016/S0019-8501\(01\)00151-1](https://doi.org/10.1016/S0019-8501(01)00151-1)
[https://doi.org/10.1016/S0019-8501\(99\)00122-4](https://doi.org/10.1016/S0019-8501(99)00122-4)
35. Walsham, G. (1995). Interpretive case studies in IS research: Nature and method, *European Journal of Information Systems*, Vol. 4, pp. 74–81.
<https://doi.org/10.1057/ejis.1995.9>
36. Runesson P., and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering, *Empirical Software Engineering*, Vol, 14. <https://doi.org/10.1007/s10664-008-9102-8>

37. Maxwell J. A. 2005. *Qualitative Research Design: An interactive approach*, 2nd Ed. Thousands Oaks, CA: SAGE Publications.