Commodity Eats Innovation for Breakfast: A Model for Differentiating Feature Realization

Aleksander Fabijan[1]✉, Helena Holmström Olsson[1], Jan Bosch[2]

[1] Malmö University, Faculty of Technology and Society, Nordenskiöldsgatan 1, 211 19 Malmö, Sweden
{Aleksander.Fabijan, Helena.Holmstrom.Olsson}@mah.se
[2] Chalmers University of Technology, Department of Computer Science & Engineering, Hörselgången 11, 412 96 Göteborg, Sweden
Jan.Bosch@chalmers.se

Abstract. Once supporting the electrical and mechanical functionality, software today became the main competitive advantage in products. However, in the companies that we study, the way in which software features are developed still reflects the traditional 'requirements over the wall' approach. As a consequence, individual departments prioritize what they believe is the most important and are unable to identify which features are regularly used – 'flow', there to be bought – 'wow', differentiating and that add value to customers, or which are regarded commodity. In this paper, and based on case study research in three large software-intensive companies, we (1) provide empirical evidence that companies do not distinguish between different types of features, which causes poor allocation of R&D efforts and suppresses innovation, and (2) develop a model in which we depict the activities for differentiating and working with different types of features and stakeholders.

Keywords: Customer feedback, innovation, commodity, wow feature, flow feature, duty feature, checkbox feature

1 Introduction

The amount of software in products is rapidly increasing. At first, software functionality was predominately required in order to support tangible electrical, hardware and mechanical solutions without delivering any other perceptible value for the customers [1]. Today, software functionality is rapidly becoming the main competitive advantage of the product, and what delivers value to the customers [2]. However, the way in which software features are being developed, and how they are prioritized is still a challenge for most organizations. Often, and due to immaturity and lack of experience in software development, companies treat software features similarly to electronics or mechanics components, with the risk of being unable to identify what features are differentiating and that add value to customers, and what features are regarded commodity by customers. As a consequence individual

departments continue to prioritize what they find the most important and miss the opportunities to minimize and share the investments into commodity features [3], [4].

In this paper, we identify that the lack of distinguishing between different types of features is the primary reason for inefficient resource allocation that, in the end, make innovation initiatives suffer.

The contribution of the paper is twofold. First, we give guidelines on how to distinguish between different types of features that are being developed and we provide empirical evidence on the challenges and implications involved in this. Second, we present a conceptual model to guide practitioners in prioritizing the development activities for each of the feature types. With this model, companies can develop only the amount of feature that is required for commoditized functionality and, on the other hand, maximize their investments in innovative features.

2 Background

In most companies, customer feedback is collected on a frequent basis in order to learn about how customers use products, what features they appreciate and what functionality they would like to see in new products [5],[6], [7], [8]. The number of requests and ideas that originate from this feedback often outnumbers available engineering resources and prevents companies from realizing all of them [9]. To help practitioners control the information overload originating from customer feedback, Knauss et al. [10] propose a feedback-centric requirements approach, together with a tool that elicits the most important information. Recently, Johansson et al. [11] stressed the importance of complementing the qualitative customer feedback with quantitative input by showing its implications on the product managers prioritization decisions. Moreover, and in order to further develop only those requirements that will deliver the most business value, various prioritization techniques have been introduced in requirement engineering and product development literature [12] [13], [13], [14], [15], [16]. However, these do not consider market factors such as the availability of the features being assessed in competitors' products [17]. Also, very little is known on how to prioritize the development activities for different types of features.

To recognize the importance of distinguishing between different types of functionality from a complexity point of view, Bosch [18] developed the 'Three Layer Product Model'. The model provides a high-level understanding of the three different layers of features, i.e. commodity, differentiating and innovative, however, does not give guidance on how to distinguish between the different types, neither which activities to invest into for each of them. The model distinguishes between three types of functionality layers, i.e. *commoditized functionality* (functionality necessary for system operation that customers take for granted), *differentiating functionality* (the functionality that differentiates the product from its competitors) and *Innovation functionality* (functionality providing significant value).

3 Research Method

This case study [19] research builds on an ongoing work with three case companies (see Table 1 below) involved in large-scale development of software products. It was conducted between August- December 2015.

Table 1. Description of the companies and the representatives.

Company and their domain	Representatives
Company A is a provider of telecommunication systems and equipment, communications networks and multimedia solutions for mobile and fixed network operators. The company has several sites and for the purpose of this study, we collaborated with representatives from one company site. The participants marked with an asterisk (*) attended the workshop and were not available for a follow up-interview.	1 Product Owner 1 Product Manager 2 System Managers 2 Software Engineer 1 Release Manager 1 Area Prod. Mng.* 1 Lean Coach* 1 Section Mng.*
Company B is a software company specializing in navigational information, operations management and optimization solutions. All the participants attended the workshop and were interviewed.	1 Product Owner 1 System Architect 1 UX Designer 1 Service Manager
Company C is a manufacturer and supplier of transport solutions, construction technology and vehicles for commercial use. All the participants that attended the workshop were interviewed. In addition, one sales manager and one technology specialist wished to join the project at a later stage, and were interviewed.	1 Product Owner 2 Product Strategists 2 UX Managers 2 Function Owners 1 Feature Coord. 1 Sales Manager 2 Technology Spec.

3.1 Data Collection and Data Analysis

First, we conducted a workshop at each of the companies. Second, we conducted twenty-two interviews that lasted one hour. During analysis, the workshop notes, interview transcriptions and graphical illustrations were used when coding the data. The data collected were analyzed following the conventional qualitative content analysis approach [20] where we derived the codes directly from the text data.

To improve the study's construct validity, we conducted semi-structured interviews at the workshops with representatives working in several different roles and companies. Since these companies represent the current state of large-scale software development of embedded systems industry, we believe that the results can be generalized to other large-scale software development companies.

4 Findings

4.1 Feature Realization: Current State of Feature Differentiation

Features that are being developed are handed over from one development stage to another, together with their requirements and priorities. The differentiation strategy is unclear to the practitioners developing the features (see e.g. Table 2).

Table 2. The current State of Feature Differentiation.

Current State	Description	Quote
Vague differentiating strategy	Practitioners struggle to know if the feature is innovative and requires e.g. direct investment, or commodity and can be covered from e.g. running maintenance budget.	*"Should we go into Maintenance budget? Or should it go to investment budget and we prioritize there?"*, -Product Strategist from Company C.
Dev. investment level does not vary	Based on the interview data, we do not see a significant difference in defining the investment level allocated to the feature.	*"There is a lot of functionality that we probably would not need to focus on." - Technology Specialist from Company C.*
Feature prioritization processes is in favor of commodity	Prioritizing innovative features is suppressed with numerous commodity efforts that are needed to satisfy the standards and follow the competitors instead of accurately understanding what adds value to the stakeholders	"*Customer could be more involved in prioritization that we do in pre-development. Is this feature more important that the other one?*" –Product Owner from Company B.

4.2 Differentiating Features: Challenges and Implications

The current state advocates a situation where features are not differentiated in the strategy between being innovative, differentiating or commodity and, as a consequence, development activities do not differ between the features. Based on our interviews, we see that there are a number of challenges associated with this situation. We present the challenges in Table 3 below.

Table 3. The challenges with differentiating features.

Challenge	Description	Quote
Understanding the stakeholder and purpose of the feature	The way in how feedback is being collected is rather ad-hoc and it depends on the stage of the feature and not on the	"*We want to understand what the customer wants to have and also truly, what do they need*." -Product Manager from company A.
Incomprehen-sible high-level directives	Identifying whether a feature is commodity, differentiating or innovative, and investing into activities needed for each type is left to the practitioners developing the feature.	"*Functionality is software basically and the features are more subjective opinions and things that we can't really… it is hard to collect data*." – Function owner from Company C.
Commodity functionality is internally considered to be innovative.	Companies do not learn from customers fast enough and, occasionally, consider and invest in development activities for features they believe are innovative..	"*Those are the things that worries us the most. All of us, since it is so hard, you need to gamble a bit. If it turns out that you are wrong, then you are behind*." –Product Manager from Company A.

Due to an unclear differentiating strategy, our case companies experience a number of implications during the development process of a feature. We summarize them in Table 4 below.

Table 4. The implications of feature realization.

Implication	Description	Quote
Uniform stakeholder impression	Individuals in our case companies typically find it difficult to identify and truly understand the stakeholder and their needs that the feature is being developed for.	"*If you are sitting in a team…you see that this is the most important thing we need to do*." -Product Manager from company A.
Arbitrary investments in development activities	Companies risk to invest extensively in development activities that are not required for a certain type of feature.	"*We tend to focus on,…on the wrong things. We need to look at the benefit for their customers*." -Product Manager from company A.
Commodity suppresses innovation	As a consequence, features that are expected to be innovative do not get to be prioritized. Instead, and in a rush to keep the pace, our case companies study close competitors and analyze their most promising features.	"*In our organization is really hard to sew everything together, to make it work. That requires funding that is almost nonexistent*." –Software engineer from Company A.
Projecting competitors current state is the norm	Companies base themselves on competition right now instead of where it will be in the future.	"*We do also our own tests of competitors….We measure are we on track or not*." -Product Strategist from Company C .

5 The Feature Differentiation Model

In response to the empirical data from our case companies, combined with the findings and the implications that we presented above, we expand the 3LPM model to a new dimension and present our model for feature differentiation in the following section. The contribution of our model is twofold. First, we provide four different categories of features and their characteristics in order to give practitioners an ability to better differentiate features early in the development cycle. Second, and as a guidance for practitioners after classifying a feature, we provide a summary of development activities for every type of feature. Practitioners can use the guidance in prioritizing the development activities for the features that they are developing. With this model, companies can develop only the amount of feature that is required for commoditized functionality and, on the other hand, free the resources to maximize their investments in innovative features that will deliver the most value.

5.1 Differentiating Characteristics of New Feature Development

Our model advocates an approach in which four fundamentally different types of features are being developed. We name them "duty", "wow", "checkbox" and "flow" types of features. With "duty", we label the type of features that are needed in the products due to a policy or regulation requirement. "Checkbox" features are the features that companies need to provide in order to be on par with the competition that

provides similar functionality. With "wow", we label the differentiating features that are the deciding factor for buying a product. Finally, and with "flow", we label the features in the product that are regularly used. We depict the four types of features on Figure 2, where we place each of the types in relation to the 3LPM commodity-differentiating-innovative categorization.

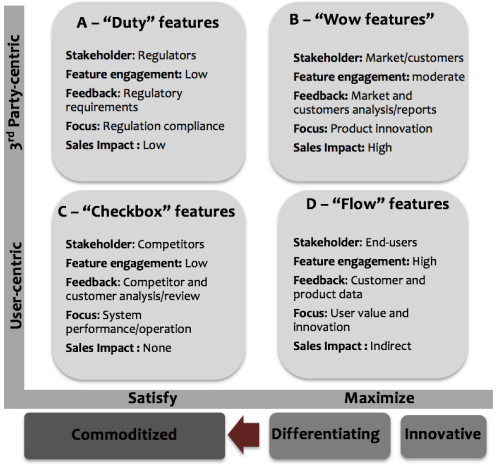

Fig. 1. The Feature Differentiation Model.

On the horizontal axis, we indicate the development extent for the features, ranging from "Satisfy" to "Maximize". On the vertical axis, however, we indicate the distinction between primarily "User-centric" and "3rd party-centric" features. By this, we divide between features that are primarily being developed for users of the product (hence "User-centric") and other, 3rd parties-centric features, which are not directly used by the users. And to help practitioners categorize a feature on these axis, we distinguish between five characteristic points; *the stakeholder* (e.g. the requestor of the feature), *feature engagement* (e.g. the expected level of feature usage) , the source of the feedback (e.g. the stakeholder generating the most feedback), *the focus of the feature* (e.g. is the feature indenting to minimally satisfy a known need, or to innovate in a new area?) and its *impact on driving sales* (is it a feature focusing on the customer paying for the product?).

5.2 The Development Process

For each of the four feature types, we suggest how to set the extent of the feature that should be developed. Here, the extent of the feature can be either defined once (constant) or dynamically adjusted during development and operation (*floating* - alternates, *following*- follows the competitors *or open* - no limitation*)*. Next, the sources that contain the information required to set the development extent need to be defined, together with the techniques that make it possible for the practitioners to collect relevant customer feedback. Next, we suggest the most important R&D activities. They are followed by the activities that do not deliver value for that type and should be avoided. Finally, we suggest how to set the deployment frequency. We summarize the most important differences between each approach in Figure 3.

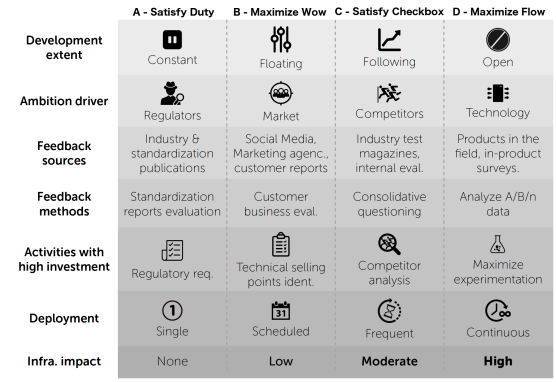

Fig. 2. The summary of the 4 different development approaches.

6 Discussion

Multi-disciplinary teams involved in the development of a software product are increasingly using customer feedback to develop and improve their products and features. Both qualitative techniques [5], [6], [8] and quantitative techniques [7], [8] are used to collect customer feedback and product data. Previous research shows that the number of requests and ideas that originate from this feedback often outnumbers available engineering resources [9]. And with increasing amount of ideas, prioritizing development resources and identifying which features to develop to what extent is a crucial step in new feature development process. Bosch [18] recognized the importance of dividing functionality between commodity, differentiating and innovative. However, and as shown in our empirical findings, companies still struggle with this and, consequently, invest into development activities that do not deliver value to the stakeholders. Innovative features are suppressed by commodity.

To address the concerns above, we develop the Feature Differentiation model, where we illustrate how development activities depend on the type of the feature being developed with respect to characteristics. The model helps companies to (1) differentiate between the four types of the features, and (2) prioritize the necessary development activities.

7 Conclusion

In this paper, and based on case study research in three large software-intensive companies, we (1) provide empirical evidence that companies do not distinguish between different types of features, i.e. they don't know what is innovation, differentiation or commodity, which is the main problem that causes poor allocation of R&D efforts and suppresses innovation. We (2) develop a model in which we depict the activities for differentiating and working with different types of features.

References

1. Boehm, B.: Value-based software engineering: reinventing. SIGSOFT Softw. Eng. Notes. 28, 3– (2003).
2. Khurum, M., Gorschek, T., Wilson, M.: The software value map - an exhaustive collection of value aspects for the development of software intensive products. J. Softw. Evol. Process. 25, 711–741 (2013).
3. Lindgren, E., Münch, J.: Software development as an experiment system: A qualitative survey on the state of the practice. In: Lassenius, C., Dingsøyr, T., and Paasivaara, M. (eds.) Lecture Notes in Business Information Processing. pp. 117–128. Springer International Publishing, Cham (2015).
4. Olsson, H.H., Bosch, J.: Towards Continuous Customer Validation : A conceptual model for combining qualitative customer feedback with quantitative customer observation. LNBIP. 210, 154–166 (2015).
5. Fabijan, A., Olsson, H.H., Bosch, J.: Customer Feedback and Data Collection Techniques in Software R&D: A Literature Review. In: Software Business, ICSOB 2015. pp. 139–153. , Braga, Portugal (2015).
6. Williams, L., Cockburn, A.: Introduction: Agile Software Development: Its About Feedback and Change, (2003).
7. Olsson, H.H., Bosch, J.: Towards Data-Driven Product Development: A Multiple Case Study on Post-deployment Data Usage in Software-Intensive Embedded Systems. In: Proceedings of the 4th International Conference on Lean Enterprise Software and Systems, LESS 2013, Galway, Ireland, December 1-4, 2013. pp. 152–164 (2014).
8. Bosch-Sijtsema, P., Bosch, J.: User Involvement throughout the Innovation Process in High-Tech Industries. J. Prod. Innov. Manag. 32, 1–36 (2014).
9. Bebensee, T., Van De Weerd, I., Brinkkemper, S.: Binary priority list for prioritizing software requirements. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 67–78 (2010).
10. Knauss, E., Lubke, D., Meyer, S.: Feedback-driven requirements engineering: The Heuristic Requirements Assistant. In: 2009 IEEE 31st International Conference on Software Engineering. pp. 587–590. IEEE (2009).
11. Johansson, E., Bergdahl, D., Bosch, J., Olsson, H.H.: Requirement Prioritization with Quantitative Data - a case study. In: International Conference on Product-Focused Software Process Improvement. pp. 89–104. Springer International Publishing, Bolzano (2015).
12. Kano, N., Seraku, N., Takahashi, F., Tsuji, S.: Attractive quality and must-be quality. J. Japanese Soc. Qual. Control. 14, 39–48 (1984).
13. Wiegers, K.E.: Automating requirements management. Softw. Dev. 7, 1–5 (1999).
14. Karlsson, L., Thelin, T., Regnell, B., Berander, P., Wohlin, C.: Pair-wise comparisons versus planning game partitioning-experiments on requirements prioritisation techniques. Empir. Softw. Eng. 12, 3–33 (2007).
15. Leffingwell, D., Widrig, D.: Managing software requirements: a unified approach. Addison-Wesley Longman Publ. Co., Inc. Boston, MA, USA. 10, 491 (1999).
16. Karlsson, J., Ryan, K.: A cost-value approach for prioritizing requirements. IEEE Softw. 14, 67–74 (1997).
17. Kakar, A.K.: OF THE USER , BY THE USER , FOR THE USER : ENGAGING USERS IN INFORMATION SYSTEMS PRODUCT. In: SAIS 2014 Proceedings (2014).
18. Bosch, J.: Achieving Simplicity with the Three-Layer Product Model. Computer (Long. Beach. Calif). 46, 34–39 (2013).
19. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14, 131–164 (2008).
20. Hsieh, H.-F., Shannon, S.E.: Three approaches to qualitative content analysis. Qual. Health Res. 15, 1277–88 (2005).