The Effect of Stakeholder Inertia on Product Management

Krzysztof Wnuk, Richard Berntsson-Svensson
Department of Computer Science,
Lund University, Sweden
{wnuk, rbsv} @cs.lth.se

David Callele
TRLabs
Saskatoon, Saskatchewan, Canada
dcallele@trlabs.ca

***Abstract* — One of the goals of requirements engineering is to capture and document innovation in the form of new product requirements. These product requirements need to express new system functions or new qualities that are most desired by customers while maintaining customer familiarity with existing products. This paper explores the contradiction between the customer desire for revolutionary advancement and their desire to maintain familiarity with existing systems. This customer inertia creates a bias toward incremental (evolutionary) advancement, potentially multiplying the risks associated with revolutionary innovations. We present a review of scenarios illustrating this stakeholder bias and propose a research agenda for further work in the area.**

Keywords: Stakeholder bias, inertia, evolution, revolution, innovation.

I. INTRODUCTION

Requirements Engineering (RE) is a key component of a successful software engineering process [1] and meeting customer requirements is critical to the process of creating value in software products [2]. The creation of software product value is inevitably associated with innovation identification and capitalization, typically in the form of executable software features or components. As a result, RE activities are primarily focused on elicitation, specification and verification of new functionality.

Software Product Lines and the concept of mass customization help software companies to capitalize on the potential of their products [3]. Product Managers and business managers often work with products that are in the market and generating revenue for years. For many of these products, customers receive new functionalities on an incremental basis – adding new capabilities in small increments that slowly evolve the user experience. Customers learn to use each increment, adapting as necessary. In contrast, large-scale changes to the user experience are not only costly to develop, they can meet with significant user resistance as customers may not readily accept revolutionary changes to systems with which they are already familiar.

This user behavior pattern indicates that there is a tension between the need to introduce revolutionary innovations [4] and customer resistance to change. Customer familiarity with the current version of a product or service creates significant inertia that strongly influences future purchasing decisions. Products that are evolutionary are less "frightening" than products that are revolutionary and may experience greater market success. In this paper we explore this tension between revolutionary innovations and customer resistance to change from the requirements engineering, innovation and decision making perspectives – focusing on the role of quality perception and quality requirements in the customer satisfaction [12] and excitement creation process.

This paper is structured as follows: Section II presents examples of customer inertia affecting market success in both the software and systems domain. Section III reviews related work in innovation quality requirements and requirements engineering release planning and decision making while Section IV analyses and comments on the current state of the research. Section V outlines research challenges and Section VI presents conclusions and a research agenda for improving our understanding of customer inertia in requirements engineering.

II. CUSTOMER INERTIA AND MARKET ACCEPTANCE

Customer inertia can be highly individual and identifying an optimal blend of evolutionary and revolutionary innovation is necessary for maximizing customer satisfaction as expressed by product requirements. We present here a simple example of customer inertia affecting market acceptance.

We analyze the instrument clusters for two car manufactures, BMW and Citroen. In the Citroen case, the designers decided to radically change the appearance of the dials when introducing a new Citroen C4 MK1 model in 2004 [5] (Figure 1a). Despite winning the 2005 European Car of the Year award [6], this radical change was not appreciated by many customers. Citroen has now returned to more 'classic' dials as shown, for example, in this image of the 2011 Citroen C4 instrument cluster (Figure 1b).

Figure 1a. Citroen C4 instrument cluster, 2004.

Figure 1b. Citroen C4 instrument cluster, 2011.

In comparison, BMW maintains a consistent instrument cluster design, apparently very careful when introducing changes to what information is displayed on the dashboard and where it is placed. The instrument cluster of the 2004 BMW model (Figure 2a) and the 2011 BMW model (Figure

2b) are very similar – any design changes are evolutionary and not revolutionary.

Customer opinions of products have more facets than customer inertia. Requirements for individual (sub)systems may be accurate but the interactions of these (sub)systems when integrated into systems of systems may produce an unexpected customer opinion.

Figure 2a. BMW instrument cluster, 2004.

Figure 2b. BMW instrument cluster, 2011.

In the automotive example of this section, the shape of the car, the design of the front grill and the quality of the brand are other factors that help form the customer's opinion. For example, a car may be deemed "exciting" as a result of substantially improved acceleration or even a 'cool' user experience innovation such as changing the color of the instrument cluster backlighting depending on operating mode.

III. Related Work

Requirements engineering and software engineering tend to focus on techniques for defining and providing product functionality [1][13]. To create a successful system and ensure its quality, it is not enough to fulfill the functional requirements. In relative terms, Quality Requirements (QR) receive less attention, perhaps because they are difficult to manage [1] or, for example, the quality requirements are relatively unknown at the requirements phase.

Customer satisfaction is strongly influenced by QR and end-users are often dissatisfied with software quality [17]. Perceptions of software quality play a central role in customer satisfaction and sound QR management practices can be seen as a key competitive advantage, playing a critical role in software product development [12].

There are several methods and techniques in the literature that address the handling and managing of quality requirements. Elicitation methods tend to rely upon brainstorming or the use of checklists. A method for eliciting, analyzing, and tracing QR using a language extended lexicon is proposed in [12]. An elicitation method where functional use-cases are created, and then associated QRs are identified by the use of a checklist is proposed by Doerr *et al.* in [14]. A similar approach is proposed by Kaiya *et al.* [15] but they use the goal-question-metric (GQM) model to explore quality requirements and their interdependencies.

In addition to elicitation methods, several QR modeling and analysis techniques are proposed in the literature. Goal-oriented methods [16][1] focus on the actual software development process where the software product's goals are the focus. The NFR Framework [13][17][1] is one of the most comprehensive methods for QR. The method defines quality goals, potential implementation solutions, and interdependencies between QR. The important quality goals are decomposed by the use of the soft-goal interdependency graph (SIG) using AND / OR refinement.

Research in innovation management recognizes that companies need to encourage innovation [8] to remain competitive. In the development of new systems, companies are facing the dichotomy of long-term *vs.* short-term strategies. Long-term strategies are often associated with revolutionary innovation [9]. Adaptability to market pressures may require risk-taking and seeking cutting-edge innovation to ensure the company's long-term viability and sustaining their competitive advantage.

A company also needs to generate short-term revenue to ensure a sound financial base; incremental innovations [10] may provide this short-term success. Gorschek *et al.* [11] recognize that the key for software companies' survival is selecting new product ideas from a range of potential innovation candidates; candidates that support the business strategy and have the highest financial impact.

Companies must also ensure that they focus on more than enabling innovation. The example of Section II illustrates customer reluctance to change and how this inertia may lead to market failure. Too much innovation may even cause the loss of customers due to a perception of quality impairment. Known as *reverse quality* in Kano's model [19], reverse quality occurs when a developer delivers large quantities of innovation. The number of innovations are deemed excessive by at least a portion of the customer base and this behavior may cause the developer to lose those customers.

To the best of our knowledge, existing QR techniques do not attempt to address the issues of customer inertia. Companies need incremental innovation to maintain or, perhaps, expand their market share. But, how much innovation is needed to maintain market share? How much innovation is needed to expand market share? And, how much innovation is too much innovation?

IV. Analysis and Commentary

Let us now look more closely at the examples of Section II. If we look at the paths taken by the automobile manufacturers we can see that Citroen chose to take a revolutionary approach between product generations. The transition from an analog instrument cluster to a digital instrument cluster was not received by the target market as Citroen expected. In contrast, BMW took an evolutionary approach, maintaining the same general look and feel while modifying only the number and relative size of the instruments. This relatively gentle evolutionary approach was well received by the marketplace. We postulate that evolutionary changes modify only a small number of the visual variables identified by Bertin [18] (horizontal position, vertical position, shape, size, color, brightness, orientation, texture) per iteration whereas revolutionary changes introduce complete, or near-complete replacement. However, if customers perceive a product to be exciting (in the sense of Kano's model [19]) then they are more willing to accept a revolutionary product and take the burden of learning how to use it than if a new product just offers "more of the same."

In the realm of COTS software we see a similar response: when Microsoft added ribbons to the user interface of their office applications product suite approximately 80% of the surveyed users had a negative reaction to this revolutionary change. Moreover, some survey respondents said that the new ribbon interface actually reduced their productivity by about 20% [7].

Alternatively, we can compare the reaction of users when they are asked to change between rival operating system user interfaces. The transition from Microsoft Windows to the Apple desktop or to any of the variants of the Linux desktops can be traumatic, the learning curve can be very steep, and productivity (at least in the short term) can suffer significantly.

We can also look to the arts and the common practice of sequels for books and movies. It appears that audiences (customers) have a strong desire for new experiences set in familiar environments populated by familiar characters, further evidence of customer inertia promoting evolutionary innovation.

We note that, in all cases that we have cited here, there is an element of product line positioning. In each case it is the user experience that is either evolutionary or revolutionary and in each case the revolutionary change in the user experience had a negative market response. This implies that evolutionary advancement may have a significantly higher probability of market success, when considered across all products and services, compared to revolutionary advancements.

The classic counter-example of revolutionary success is the introduction of the Apple iPhone. A revolutionary change with significant market success, the iPhone's success appears to be based on a deep knowledge of the consumer, knowledge that is expensive to acquire and which is more likely to be obtained by large companies than small companies [10].

The practice of requirements engineers tends to focus upon the functional requirements for a given project. Once functional requirements have been captured the attention tends to turn toward the nonfunctional requirements – to identify them and to identify their constraints upon the functional requirements. A number of ontologies for nonfunctional requirements have been proposed [16]. However, these ontologies do not explicitly identify that customer inertia could be represented as a set of quality attributes or other non-functional requirements. Further, if we assume that non-functional requirements are elements that act as market differentiators, catering to customer inertia could be identified and positioned as a competitive advantage (*e.g.* preserve your training investments!).

Kano [19] proposed a customer satisfaction model for product development that classifies customer preferences into five categories: those that excite the customer, those that are conspicuous by their absence, those that must be present, those that need to be present but for which no credit is given, and those that are actually considered to engender a negative response. We shall focus our attentions upon those features that excite the customer and those features that repel the customer, particularly in the context of product line development.

There is a tendency in the market to believe that "more features are better". However, there is a significant risk that the user will perceive the addition of too many features in a single release as a revolutionary change rather than as an evolutionary change. Domain experts have a tendency to embrace a multitude of new features whenever they are released. However, the typical (or even novice) user can find the addition of all of these features at once an overwhelming proposition [20], lacking the ability to accept this many changes at once. They must also now invest significant time and effort in learning these new features. Or, if they do not believe that they need to these new features (or that these new features will help them in any way) then they may resent their addition to the product line. This resentment may be proportional to the investment necessary to adapt to the revolutionary changes or the cost to develop workarounds to maintain the *status quo*.

Further complicating customer acceptance is the decision complexity faced by the customer as the number of features increases in a given release. Figure 3 illustrates the complexity faced by the user when attempting to find and utilize a subset N features that they want to have out of a total of M new features in a release (*e.g.* ${}_{N}C_{M}$, for N=2,3,4 and M=2..10).

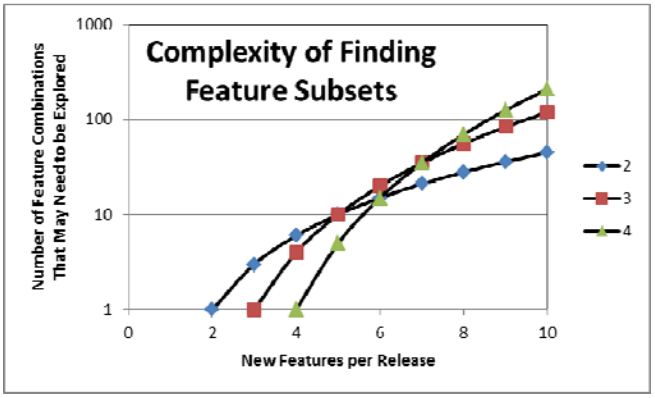

Figure 3 Complexity of Finding Feature Subsets

This visualization is a reminder that there is a significantly greater probability of customer satisfaction if the number of features offered by the release is close to the number of features that the customer can absorb for that release. A customer presented with an excess of features may not be even able to find the features of value to them, especially if the features are designed to work together. Of course, the challenge now becomes identifying those features that excite the customer and including them in the release rather than choosing features that repel the customer. It does, however, appear reasonable to conclude that an excessive number of features will repel the customer for the reasons discussed earlier.

We believe that for some scenarios it would be better to reduce the number of new capabilities added to iterations of the product. If the requirements process focused first upon identifying the range of possible features and then upon

prioritization the end result might be greater customer satisfaction. The delivery of exactly those features that are desired by the customer, in a timely manner, and at a cost that the customer is willing to pay appears to be a pattern for success. One could argue that the various Apple iProducts all follow this basic model. Apple products are often not as feature-rich as their competitors but the features that they do have are generally considered to be reliable and well crafted: for example, the first iPhone lacked MMS functionality.

V. Research Agenda

We do not yet have sufficient information on current industrial practice related to customer inertia but our industrial case experiences and the gaps in related work suggested in Section I provide indications for the following research directions:

- Integrate the customer inertia concept with the software product management, innovation and software product lines literature.
- Add a type of quality requirements that captures an assessment of customer inertia and investigates its relation with already defined quality attributes, *e.g.* usability.
- Discover methods for identification and specification of the customer inertia quality attribute as a part of the requirements engineering and software product management lifecycle, particularly prioritization.
- Research possible methods to identify which of the already known quality requirement types may be impacted by the customer inertia aspect of software solutions.
- Develop a method to assess and analyze the learning curve impact for customer inertia.

VI. Conclusions

Customer resistance to change, referred to herein as customer inertia, is a proposed new type of quality requirement identified in this work. This topic has received relatively little attention in the RE literature despite its demonstrable effect upon customer satisfaction.

Assessments of customer inertia could be used as part of a requirements prioritization process: inertia, coupled with the intensity of innovation (evolutionary through to revolutionary) could be used to assess the risk of bundling certain groups of features together for a product release. Too much innovation could be met with customer inertia resistance while too little innovation may lead to loss of market status in the eye of the consumer.

An initial research agenda has been proposed and we intend to proceed toward integration of customer inertia with traditional requirements engineering beginning with a more thorough integration with the existing literature. Future work could also move toward those areas often the domain of the business analyst, incorporating greater knowledge of business goals and market intelligence in the requirements process.

References

[1] G. Kotonya and I. Sommerville, Requirements Engineering - Processes and Technique, John Wiley & Sons, 1998.

[2] A. Aurum and C. Wohlin, “A value-based approach in requirements engineering: explaining some of the fundamental concepts.” In Proc. of the 13th International working conference on Requirements Engineering: Foundation for Software Quality (REFSQ'07), Trondheim, Norway 11-12 June 2007, P. Sawyer, B. Paech, and P. Heymans (Eds.). Springer-Verlag, Heidelberg, pp. 109-115.

[3] C. Pohl, G. Böckle, and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*, Springer-Verlag, New York USA, 2005.

[4] C. Andriopoulos and M.W. Lewis, “Managing Innovation Paradoxes: Ambidexterity Lessons from Leading Product Design Companies”, Long Range Planning, vol 43, 2010, pp 104-122.

[5] The description of the Citroen C4 MK1 model can be found at http://en.wikipedia.org/wiki/Citroën_C4

[6] The European Car of The Year competition website http://en.wikipedia.org/wiki/European_Car_of_the_Year

[7] The discussion of the results of the survey about the Microsoft’s ribon introduction can be accessed at http://www.exceluser.com/explore/surveys/ribbon/ribbon-survey-results.htm

[8] F. Patterson, M. Kerrin, G. Gatto-Roissard, and P. Coan, “Everyday innovation: How to enhance innovative working in employees and organisations”, NESTA research reports, pp. 1-54, 2009.

[9] C. Andriopoulos and M.W. Lewis, “Managing Innovation Paradoxes: Ambidexterity Lessons from Leading Product Design Companies”, Long Range Planning, 43, 2010, pp 104-122.

[10] R. D. Deward and J.E. Dutton, "The Adoption of Radical and Incremental Innovations: An Empirical Analysis," *Management Science*, vol. 32, Nov. 1986, pp. 1422-1433.

[11] T. Gorschek, S. Fricker, K. Palm, and S.A. Kunsamn, “A Lightweight Innovation Process for Software-Intensive Product Development”, IEEE Software, Jan. 2010, pp. 37-45, doi: 10.1109/MS.2009.164.

[12] L.M. Cysneiros and J.C.S.P. Leite, “Nonfunctional Requirements: From Elicitation to Conceptual Models”, IEEE Transactions on Software Engineering, vol. 30, May 2004, pp. 328-349, doi: 10.1109/TSE.2004.10.

[13] L. Chung and J.C.S do Prado Leite, “On Non-Functional Requirements in Software Engineering”, Lecture Notes in Computer Science, vol. 5600, Mar. 2009, pp. 363-379, doi: 10.1007/978-3-642-02463-4_19. .

[14] J. Doerr and D. Kerkow and T. Koenig and T. Olsson and T. Suzuki, “Non-functional requirements in industry - three case studies adopting an experience-based NFR method”, Proc. of the 13th IEEE International Conference on Requirements Engineering 2005, pp. 373-382, doi: 10.1109/RE.2005.47.

[15] H. Kaiya and A. Osada and K. Kaijiri, “Identifying stakeholders and their preferences about NFR by comparing use case diagrams of several existing systems”, Proc. of the 12th IEEE International Conference on Requirements Engineering, April 2004, pp. 112-121, doi: 10.1093/ietisy/e91-d.4.897.

[16] L. Chung and B.A. Nixon and E. Yu and J. Mylopoulos, NFR in Software Engineering, Kluwer Academic Publishers, 2000.

[17] H-W. Jung, S-G. Kim and C-S. Chung, “Measuring software product quality: A survey of ISO/IEC 9126,” IEEE Software, vol. 21, Sep. 2004, pp. 88-92, doi: 10.1109/MS.2004.1331309.

[18] Bertin, J., Semiology of Graphics: Diagrams, Networks, Maps. 1983, Madison, Wisconsin, USA: University of Wisconsin Press

[19] The description of the Kano model can be accessed at http://en.wikipedia.org/wiki/Kano_model

[20] http://www.networkworld.com/community/blog/offering-too-many-products-or-features-bad-th