2009

Digital differentiation, software product lines, and the challenge of isomorphism in innovation: A case study

Lena Andreasson
Viktoria Institute, lena.andreasson@viktoria.se

Ola Henfridsson
Viktoria Institute, Sweden & Oslo University, Norway, ola.henfridsson@wbs.ac.uk

DIGITAL DIFFERENTIATION, SOFTWARE PRODUCT LINES, AND THE CHALLENGE OF ISOMORPHISM IN INNOVATION: A CASE STUDY

Lena Andreasson, Viktoria Institute, Hörselgången 4, 417 56 Gothenburg, Sweden, lena.andreasson@viktoria.se

Ola Henfridsson, Viktoria Institute, Hörselgången 4, 417 56 Gothenburg, Sweden, ola.henfridsson@viktoria.se

Abstract

This paper examines the adoption of software product line engineering to implement digital differentiation of physical products. The introduction of such software-based variety can typically be challenging for firms innovating within the realm of a manufacturing paradigm. In particular, the mutual dependency between the organization design and product design of new product developing firms may counteract attempts to induce change through software product line engineering. On the basis of innovation theory and the notion of isomorphism, the paper presents a case study of digital differentiation at one of the world's largest automakers, GlobalCarCorp. Relating to the literatures of software product lines and product families, the contribution of the paper is a lens through which to understand the role of isomorphism in implementing digital differentiation in new product development. In addition, practical implications are derived from this in-depth study.

Keywords: digital differentiation, IS implementation, software product lines, innovation, innovation management, isomorphism.

1 INTRODUCTION

The digitization of physical products is radically challenging the innovation processes of established firms. On one hand, it multiplies the space of digital options for not only incrementally improving existing offers but also to launching radically new services on the market (Jonsson et al. 2007). On the other hand, seizing emergent digital options is difficult because current product innovation practices may not involve necessary IT capabilities and organizational agility (Sambamurthy et al. 2003).

In the innovation literature, the tension between options provided by a new technology and institutionalized practices established over longer periods of incremental innovation has received significant attention (Abernathy 1978; Abernathy and Utterback 1978; Anderson and Tushman 1990; Hargadon and Douglas 2001; Tushman and Anderson 1986). In particular, the notion of dominant design captures how institutionalized innovation practices typically congeal over time as a template for product innovation within an industry (Teece 1986). Dominant designs help firms organize their innovation processes so that they capitalize on their intellectual, relational, and technical resources. In particular, successful firms orchestrate a reciprocal relationship between organization design and product design. Baldwin and Clark (2000) refer to this relationship as the fundamental isomorphism between task structure and design structure.

While the fundamental isomorphism between organization design and product design is important to exploit a dominant design, it lowers a firm's capability to respond to technological discontinuities (Anderson and Tushman 1990). The introduction of digital technologies in physical products represents such a discontinuity for manufacturing firms. While existing processes embed an innovation logic fine-tuned for a tangible, hardware-based business, embedding software into existing product architectures introduces an alien innovation logic requiring new architectural knowledge (Andersson et al. 2008). A pressing issue is therefore how to handle these parallel logics when organizing innovation processes.

This paper focuses on the adoption of software product line engineering for implementing digital differentiation of physical products. This focus directs the attention to two literatures on modularity, currently divided along disciplinary lines. First, the body of literature on software product lines is concerned with strategies for managing software families in a way that facilitate flexibility and modularity (Clements and Northrop 2001; Pohl et al. 2005). This software engineering-literature has its origins in Parnas' (1976) work on software families, and has evolved as a result of the growing complexity of handling and exploiting software innovations. Second, the product family literature is concerned with platforms as a means for achieving product differentiation without compromising mass production advantages. This body of literature has emerged within the context of the product innovation literature, addressing the concerns of manufacturing companies in areas such as the automobile industry (Karlsson and Sköld 2007), camera industry (Robertson and Ulrich 1998), and office material industry (Cooper et al. 2001) .

The contribution of this paper is a lens through which to understand the mutual dependency between organization design and product design in implementing digital differentiation in new product development. Drawing on innovation theory and the notion of isomorphism (Baldwin and Clark 2000), we present a case study of such differentiation through software product line engineering at one of the world's largest car manufacturers: GlobalCarCorp. The following research question is addressed: How can software product lines be implemented to leverage digital differentiation in firms innovating in a manufacturing paradigm?

The remainder of the paper is structured as follows. Section two reviews the extant literatures on product families and software product lines. Section three outlines the theoretical framework including the notion of isomorphism. While section four presents the research methodology, section five outlines the case of software product line engineering at GlobalCarCorp. Analyzing the case, section six outlines implications for theory and practice, while the last section concludes the paper.

2 RELATED LITTERATURE

Throughout this paper, we refer to digital differentiation as a software-enabled process that identifies, implements, evaluates, and maintains distinctive characteristics of a physical artifact relative to other artifacts of the same class. For the purposes of this paper, we review two streams of literature that relate to digital differentiation but yet are divided along disciplinary lines. First, reviewing the innovation literature, product differentiation has been an established topic for long. The research on mass customization (Pine II 1993) as well as product platforms and families (Halman et al. 2003) are of specific relevance in the context of this paper. Second, research on software-based differentiation has emerged within the software engineering community, typically oriented towards methodologies for managing variety in specific software families (Clements and Northrop 2001; Clements and Northrop 2003; Pohl et al. 2005).

2.1 Product Family Research

Product differentiation has been addressed quite extensively in the innovation literature (Aaker 2003; MacMillan and McGrath 1997; Pine II 1993; Robertson and Ulrich 1998). Conceptualized as product family, platform, or line research (Halman et al. 2003; Robertson and Ulrich 1998), one of the main themes of this literature has been the challenge of achieving a differentiated offer to customers without increasing cost and development time proportionally. The implementation of differentiation attributes on a common architecture promises to maintain or increase the market share and keep customer attention (Aaker 2003). The typical approach to this challenge is the development of a common architecture, which allows for the delivery of well-adapted products while maintaining a high degree of commonality of components (Lundbäck and Karlsson 2005). In this regard, mass-scale advantages can be combined with customization (Pine II 1993).

Managing product families can be challenging. As an example, Karlsson and Sköld (2007) note how multi-branded strategies typically involve counteracting forces linked to commonality and differentiation. On one hand, the commonality force directs attention to identifying a common solution for a set of brands identified as a product family. On the other hand, differentiation within this family is important as to avoid brand "cannibalization". It has been observed that careless search for commonality opportunities through common architectures may cause negative revenue effects (Kim and Chhajed 2000).

2.2 Software Product Lines

In the software engineering community, software product lines have emerged as an increasingly important research theme (Clements and Northrop 2001; Pohl et al. 2005). The introduction of digital technology in physical products enables new forms of differentiation that do not rely on hardware changes. In this regard, costly development of new tools and models for each product change is virtually avoided. Firms introducing software product line engineering are promised to achieve product differentiation in a more cost effective way (Kreuger 2002). Indeed, it is argued that the use of software product line engineering not only results in lower costs, shorter lead times, and increased variety, but also in higher quality of software products belonging to the same family (Ereño et al. 2006).

Reflecting its software engineering origin, the literature on software product lines is design-oriented in its emphasis on better methodologies for managing software families. As outlined by Clements and Northrop (2003, p23) "a software product line is a set of software intensive systems sharing a common, managed set of features that satisfy the specific needs or mission and that are developed from a common set of core assets in a prescribed way". To this end, domain engineering (Pohl et al. 2005), or core asset development (Clements and Northrop 2001), is the process where identification of common and variable features of a product family is accomplished.

Pohl et al. (2005) distinguish five iterative steps of domain engineering to ensure variability of the common assets: product management, domain requirements engineering, domain design, domain realization, and domain testing. First, product management sets the direction of the project and manages consumer goals throughout the software development process. The objective is to develop the product roadmap to the extent it is possible in a given point in time. The second step is to determine common requirements of the domain. The following two steps involve designing a solution, which is later implemented. The last step of this iterative process is to test the domain solution. While these steps resemble any single system development process, the emphasis on identifying commonalities within the software family and the options of variability makes software product line engineering promising for accomplishing software-based differentiation.

3 THEORETICAL FRAMEWORK

The literatures on product families and software product line engineering are important backdrops in our quest of understanding digital differentiation of physical products. However, in these streams of literature, little attention is paid to the relation between organization design and product design. Given the received innovation literature on dominant design (Anderson and Tushman 1990; Teece 1986; Tushman and Anderson 1986) it would be useful to further investigate the institutional challenges evoked in introducing digital differentiation in the new product developing process in firms designing and manufacturing physical products.

Isomorphism is an established concept within institutional theory (DiMaggio and Powell 1983; Meyer and Rowan 1977; Scott 1995; Zucker 1977). In this paper however, we draw on Baldwin and Clark's research (2000) on design and industry evolution. In particular, we apply their layers of structure (LoS) model and its relation to fundamental isomorphism as a lens with which to understand the role of institutional arrangements when transforming innovation processes assembled for hardware-based differentiation into processes suited for digital differentiation. The LoS model distinguishes three different layers of structure important in product innovation: artefact, design, and task structures. On the most basic level, the artifact structure refers to the tangible instantiation of a particular design. In other words, the artifact structure is what can be seen, heard, touched, and used. It simply performs functions that create value for its user. Encompassing its architecture and functions, the design structure is a description of the artefact. In this regard, it is not only the technical blueprint of the artefact but also defines the adaptability of the design to changing circumstances and markets. Typically, physical products embed a modular architecture, enabling flexibility because of its implementation of independence between structural elements of the design (Baldwin and Clark 1997; Sanchez and Mahoney 1997; Ulrich 1995). Lastly, task structure refers to the set of activities needed to be performed to realize a design. The task structure implies organizational design elements including communication channels, information filters, and standard operating procedures (Henderson and Clark 1990).

In this paper, we are particularily interested in the relation between task structure and design structure. The received innovation literature describes alignment of these structures as imperative to enable successful new product development (Henderson and Clark 1990; Sosa et al. 2004). Referring to this relationship as the fundamental isomorphism, Baldwin and Clark (2000) not only underline the tight coupling between the two structures but also suggest the challenges involved when initiating change in either structure. In what follows, we apply the LoS model in general and the concept of isomorphism between design structure and task structure in particular for understanding digital differentiation through software product lines at a global automobile manufacturer. The next section describes our methodology.

4 RESEARCH METHODOLOGY

4.1 Research Setting and Design

Our case study research was conducted at one of the world's largest automakers, referred to as GlobalCarCorp throughout this paper. At GlobalCarCorp, R&D operations serve many brands. The multiplicity of brands creates incentives to identify and develop product platforms for making differentiation economically feasible.

In 2005, one subdivision of GlobalCarCorp in Sweden received global responsibility for R&D on instrument clusters. This responsibility came in a transition period of instrument cluster design. Digital technologies were introduced into the design of instrument clusters including speedometer, tachometer, and other types of driver information during this time. This change in technological basis introduced a range of digital options for new types of differentiation within product families. Given its traditional hardware basis, however, digital differentiation also breaks with extant task and design structures.

Designed as case study research (Eisenhardt 1989), our 12-month study was initiated in November 2007 and was specifically focusing the 3-year process by which digital differentiation was implemented through software product lines at GlobalCarCorp. Exemplifying process research rather than variance research, the intention has been to identify and predict "patterned regularities over time" (Markus and Robey 1988).

4.2 Data Collection and Analysis

The data collection can be described as an iterative process. It consisted of three phases, as described below. Concurring with the typical case study, data collection also included multiple data sources including semi-structured interviews, meeting notes, workshop documentation, and email correspondence.

The first data collection phase included meeting attendance as to frame the area of concern. Analysis of meeting notes generated five areas of significant interest (development methods, organization, digitization, differentiation, and architecture), which, together with relevant literature, guided the remainder of the data collection process.

The data collection of the second phase mainly included recorded and transcribed semi-structured interviews. All in all, 25 interviews, ranging between 40 minutes and two hours, were conducted. The interviews were based on an interview template developed on the basis of the themes identified in the first phase. Respondents ranged from managers to developers, and they covered expertise such as software, architecture, graphical interfaces, ergonomics, design, and market. The semi-structured interviews improved the understanding of the organization's task and design structures and how these changed with the introduction of the SoftCluster tool. Statements from respondents concerning one or many of the five significant areas of interest were especially focused on.

The third phase was confirmatory in character. After completing the first two phases, process charts of the software development and differentiation process were developed. These process charts were discussed with GlobalCarCorp employees as to verify and extend our preliminary understanding of the case. The findings were assessed in view of relevant literature. Then, the theoretical framework was applied to extend our ideographic understanding beyond the specific case setting.

Exemplifying engaged scholarship (Van de Ven 2007), workshops were organized with the project team over the last two phases to feedback new perspectives on their work practice based on prior research and empirical findings.

5 SOFTWARE PRODUCT LINES AT GLOBALCARCORP

In 2005, GlobalCarCorp designers working with instrument clusters introduced the idea of implementing software product line engineering. There were three main reasons for doing this:

- Controlling and managing software variants became significant as GlobalCarCorp decided to leverage from its portfolio of multiple brands.
- GlobalCarCorp initiated an economics of scale strategy with the intention to use the same hardware for many products, while accomplishing differentiation through software.
- The digitization of the car necessitated increased software development professionalism.

While the instrument cluster traditionally consists of hardware with little, or no, modification possibilities, the digitization of interfaces in the instrument cluster opened up new possibilities to present information in flexible and differentiated ways (see Figure 1). For instance, context-aware information presentation can be used to reflect the current condition of the car or immediate traffic situations. It also allows the introduction of new functionality and differentiation features in the instrument cluster.

Figure 1. Instrument cluster with fully reconfigurable display

GlobalCarCorp identified five product families to manage instrument cluster variants for over 80 car models, amounting to approximately 3.8 million cars each year until 2012. In addition, the new strategy involved separating hardware and software development. GlobalCarCorp intended to increase flexibility in the selection of hardware and software suppliers.

Different hardware requirements and cost requirements guided the process. Each cluster family was divided in different levels: base, mid, and high, representing levels of appearance and features. In other words, the complexity of handling software variants was considerable. Because of this complexity, GlobalCarCorp decided to adopt a tool that would facilitate the process.

5.1 The SoftClusterTool

The SoftClusterTool is a comprehensive XML-based tool for differentiation of the software architecture of instrument clusters. The tool enables the development of one basic software module for all types of displays in a specific product family. By using an editor, a HMI (Human Machine Interaction) engineer can differentiate the common cluster through changes to the XML-files controlling static data such as font, language, colors, layout, and graphics, or the dynamic data produced by functional units of the car. Examples of functional units are the FM tuner, phonebook, and thermometer, as well as units related to the condition of the car.

5.2 Implementing the New Process

Before adopting the SoftClusterTool, HMI engineers were responsible for the HMI portion of the instrument cluster on the basis of input from the design department and managers of functional units. HMI requirements and the overall cluster design would later be implemented by the selected instrument cluster supplier (see Figure 2). In this regard, the supplier was responsible for implementing and delivering the integrated cluster, including both hardware and software. Once implemented, GlobalCarCorp tested and evaluated the cluster. Changes and updates were then

collected and submitted to the supplier as change requests. This process was unique, although similar, for all brands within GlobalCarCorp

A common problem of the traditional design process was differences in requirements interpretation between GlobalCarCorp designers and the supplier. In other words, there were often a gap between the final implementation of the supplier and GlobalCarCorp's original intention. In addition, new requirements emerged over time, including seemingly minor issues such as change of a word, bitmap or a color. Since every update required contacts with the supplier, the process was both time consuming and costly. In fact, as a designers noted, late change requests were anticipated by the supplier and essentially an important element in their business model.

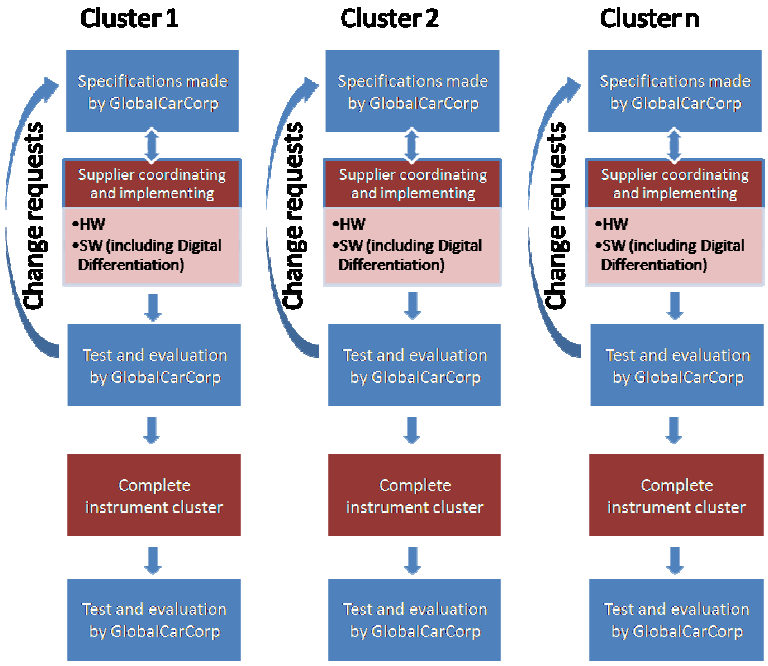

Figure 2. GlobalCarCorp's old task structure for instrument cluster design

In view of the problem with change requests, the tool was adopted in 2005. The intention was to in-house the software for realizing digital differentiation, also called HMI software, side of realizing instrument clusters. Rather than having the supplier implementing change requests related to HMI software updates, the new process, enabled by the tool, attributed the instrument cluster suppliers the role of a component supplier only. Without prior experience of software implementation, new roles and tasks had to be defined at CarCorp.

Given that the software implementation previously was done by the supplier, cognitive ergonomics and interaction design competences had been prioritized within CarCorp's HMI design group. In view of the new tool, people with programming skills in general and XML competence in particular were necessary. In addition, a new role was defined to cater for the idea of using the tool and the new process for accomplishing digital differentiation between brands in GlobalCarCorp's product families. Before separating software and hardware, as was done in the new process (see Figure 3), each brand-specific cluster had a responsible manager.

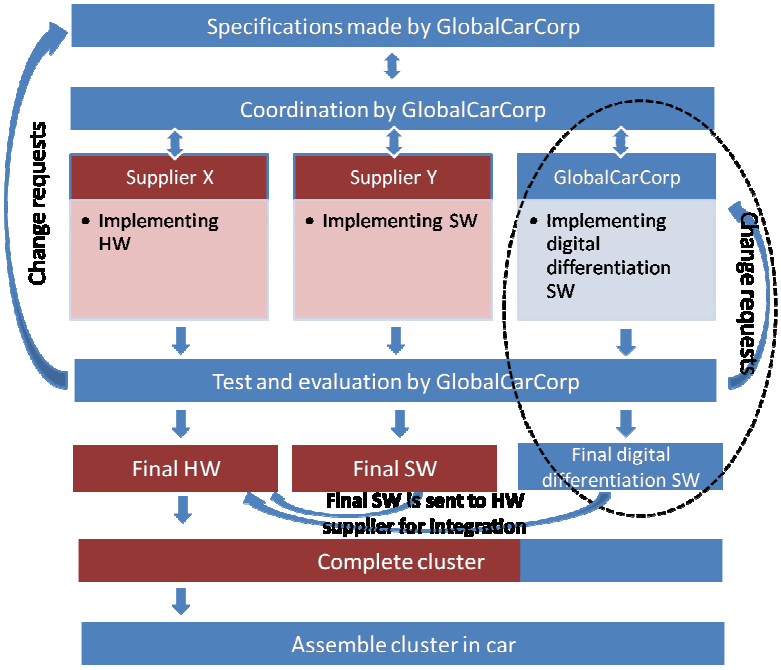

Figure 3. Current task structure for cluster design. The circled area is enabled by the tool

Due to the significant effort put into developing the SoftClusterTool, however, little focus was nevertheless on adjusting the organization and process to harvest the anticipated benefits of the tool. The traditional way of developing information cluster with significant focus on hardware components and their design structure and coupled task structure remained at GlobalCarCorp.

One example of this prevalence of institutionalized practices was the lack of standard operating procedures for gathering and communicating software differentiation requirements from GlobalCarCorp stakeholders. In this regard, programmers were required to work proactively to acquire relevant information. This was both time-consuming and challengeing because the programmers had to know exactly who to contact and what information to ask for. As a result, programmers often had to make decisions outside their obligations to meet deadlines and the overall productivity pressure. Essentially, they had to develop informal ways of gathering requirements, drawing on requests from people they already knew or were referred to by other employees. Ideally, such requests would involve perspectives from employees working on branding, interaction design, functional units (e.g., radio, antenna, or navigation) or cluster hardware. Without any general process in place, however, programmers of the HMI group often faced conflicting requirements and had to rely on individual judgment.

Before adopting the SoftClusterTool, the task of a HMI engineer at GlobalCarCorp was separated from other modules of the instrument cluster. Many decisions were made by the supplier, including the important task of pooling and integrating different requirements. While programmers face the consequences of the new tasks that the SoftClusterTool implies, other stakeholders at GlobalCarCorp were not fully aware of the new way of gathering requirements. Consequently, they drew on work processes already established before introducing software product lines. The anticipated benefits of the SoftClusterTool therefore remained to be seen.

6 DISCUSSION

Operating in a manufacturing paradigm, product innovation in the automotive industry is typically geared towards integrating different hardware components, or modules, into a fully functional product. As an important element in such integration, requirements engineering with regard to the different

modules and their interfaces is a core competence (cf.Sanchez and Mahoney 1997). Following the digitization of the car, however, automotive design increasingly includes separating the software elements from hardware ones. Following this design change, the relationship between the supplier and the automotive firm is changing. Using our theoretical framework (Baldwin and Clark 2000), the design change occasioned a change in the layers of structure of designing instrument clusters at GlobalCarCorp. The digital options generated were challenging in view of the established isomorphism between the established design and task structures.

The case presented in this paper illustrates the adoption of software product line engineering in instrument cluster design at a global automaker. In particular, we examine the introduction of a tool that was designed not only to enable flexible catering for change requests related to software but also to allow for digital differentiation over product families. The SoftClusterTool was an attempt to harvest commonality benefits on the hardware side of instrument cluster design, while achieving differentiation between GlobalCarCorp brands with software. In this regard, the adoption of software product line engineering was an attempt to implement a vision of digital differentiation.

Referring to task structure as the set of activities needed to be performed to realize a design (Baldwin and Clark 2000), it is clear the adoption of the tool at GlobalCarCorp rendered some important implications associated with the established isomorphism between design structure and task structures in instrument cluster design: the redefined supplier role and the decoupled development of hardware and software. In what follows, we explore these implications in view of the LoS model in order to understand the innovation challenges faced by producers of physical products when attempting to seize digital options.

6.1 Isomorphism and Parallel Task Structures

As noted in our case, the old task structure of instrument cluster design was highly dependent on suppliers. While GlobalCarCorp specified, tested, and evaluated the implemented product before assembly into the car, some core activities in the task structure were performed at the supplier including hardware-software implementation and the coordination of hardware and software requirements. Given that this division of work tasks was not only institutionalized in GlobalCarCorp's supplier relationships but also reflected in the dominant industry structure, the new design structure occasioned by software product line engineering was challenging. Indeed, the debundling of hardware and software design instantiated through the SoftClusterTool created a new situation for both GlobalCarCorp and suppliers. At GlobalCarCorp, however, the task structure changes needed for mirroring the design structure changes have not yet been fully completed. At the end of the study, it could be noted that no real isomorphism existed between the design structure and task structure. As a tangible example of this, the final products, the instrument clusters, currently lack digital differentiation characteristics between brands (which was one of the main objectives at the outset). In other words, despite a change in the design structure and the implementation of the SoftClusterTool, prevailing task structures counteracted these initiatives.

There were three important aspects of digital differentiation and the established isomorphism between task structure and design structure at GlobalCarCorp. First, one tangible change amounted to the newly employed programmers, who were supposed to implement requirements submitted by stakeholders throughout GlobalCarCorp. Because stakeholders such as brand managers, interaction designers, and functional unit managers were unable to systemically supply such software-specific requirements to SoftClusterTool users, however, they had to take on tasks for which they were not responsible. This caused frustration among a key group of people involved in the digital differentiation initiative.

Second, as a multi-branded firm, GlobalCarCorp viewed digital differentiation and software product lines as elements in a strategy intended to combine commonality and differentiation. Given that such strategies traditionally applies to the hardware side of the car (cf. Sköld and Karlsson 2007), the corresponding strategy on the software side was completely new. It shifted major tasks of the domain

engineering process (Pohl et al. 2005) to GlobalCarCorp from suppliers. In this regard, complexity has increased and new patterns of requirements coordination and integration were needed. Even though GlobalCarCorp recruited programmers with XML knowledge, the new engineers could not fully accommodate the responsible task of handling the domain engineering process. Because GlobalCarCorp had no prior competence in the area, the new employees could not foster new requirement gathering processes with other stakeholders in the firm who still relied on the established hardware-related isomorphism between design structure and task structure As an illustration, the differentiation plan (Sköld and Karlsson 2007) for the five different cluster families only included strategies for hardware differentiation.

Third, the institutionalized boundary relationships between suppliers and GlobalCarCorp were blurred. The old task structure had been instantiated and fine-tuned over many years. With the new initiative, a portion of that process remained, namely, the hardware and pre-installed software, while the differentiation software responsibility was taken on by GlobalCarCorp. This required new boundary-spanning behavior, where the old task structure needed to be split into two parallel ones (cf. Figure 3). In essence, the hardware set the rules and the software had to follow. As illustrated in Figure 3, the hardware was first developed and then software was added. This resulted in an inflexible way of handling software updates. While software is more flexible than hardware with regard to updates, adoptions, and adjustments, this sequencing downplayed the flexibility introduced by the SoftClusterTool. This advantage dissolved in the already introduced hardware constraints. It simply was not possible to make software updates as frequently and swiftly as the tool rendered possible due to lack of supporting task structures.

7 IMPLICATIONS

We report a case study of GlobalCarCorp's proactive attempts to introduce software product line engineering for combining commonality and differentiation in instrument cluster design. Using Baldwin and Clark's (2000) LoS model and the notion of isomorphism, we analyze this case for understanding the mutual dependency between organization design and product design in implementing digital differentiation in new product development. Our study has significant implications for the innovation literature on product families (e.g., Halman et al. 2003; Karlsson and Sköld 2007; Robertson and Ulrich 1998) and software product line engineering (Clements and Northrop 2001; Pohl et al. 2005).

In sum, in spite of attempts to introduce a new task structure around digital differentiation software in cluster design, our evidence pinpoints the difficulty for a manufacturing organization to establish software product line engineering processes that break with the fundamental isomorphism between task structure and design structure (cf. Baldwin and Clark 2000). Because the established isomorphism is based on a hardware paradigm that will remain in firms producing physical products, attempts to establish new design structures and task structures for the software portion of the sub-system are challenging. As illustrated in the requirement elicitation problems at GlobalCarCorp, the new task structure will be severely challenged by the dominant design and task structures related to hardware. This insight is a contribution to the software product line literature (Clements and Northrop 2001; Pohl et al. 2005), which tends to disregard surrounding processes, including hardware-related ones, when outlining their recommendations on how to accomplish software variety. In a similar vein, in its orientation on physical products only, the product family literature (Halman et al. 2003; Karlsson and Sköld 2007; Robertson and Ulrich 1998) cannot either cater for this observation. The innovation literature on product families tends to oversee the increasing digitization of physical products, thereby disregarding the new and relevant relationships emerging between differentiation across the digital and physical realms.

We believe that more research is needed in this domain in order to handle the innovation challenges and opportunities that manufacturing organizations face in a time when their manufactured products

are becoming digitized. After all, a new set of core competences seems to be needed when seizing emergent digital options.

8 REFERENCES

Aaker, D. 2003. "The Power of the Branded Differentiator," *Sloan Managment Review*), pp 83-87.

Abernathy, W.J. 1978. *The Productivity Dilemma*. Baltimore: Johns Hopkins University Press.

Abernathy, W.J., and Utterback, J.M. 1978. "Patterns of Innovation in Industry," *Technology review* (80:7), pp 40-47.

Anderson, P., and Tushman, M. 1990. "Technological Discontinuities and Dominant Designs: A Cyclical Model of Technological Change," *Administrative Science Quarterly* (35:4), pp 604-633.

Andersson, M., Lindgren, R., and Henfridsson, O. 2008. "Architectural Knowledge in Inter-Organizational It Innovation," *Strategic Information Systems* (17), pp 19-38.

Baldwin, C., and Clark, K. 1997. "Managing in the Age of Modularity," *Harvard business review* (75:5), pp 84-93.

Baldwin, C., and Clark, K. 2000. *Design Rules: The Power of Modularity* Cambridge, MA: MIT Press.

Clements, P., and Northrop, L. 2001. *Software Product Lines: Practices and Patterns*, (1 ed.). Addison-Wesley Professional.

Clements, P., and Northrop, L. 2003. "Software Product Lines." Retrieved 7th Nov, 2008, from http://www.sei.cmu.edu/programs/pls/sw-product-lines_05_03.pdf

Cooper, R., Edgett, S., and Kleinschmidt, E. 2001. "Portfolio Management for New Product Development: Results of an Industry Practices Study," *R&D Management* (31:4).

DiMaggio, P., and Powell, W. 1983. "The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality in Organizational Fields," *American Sociological Review* (48:2), pp 147-160.

Eisenhardt, K.M. 1989. "Building Theories from Case Study Research," *Academy of Management Review* (14:4), pp 532-550.

Ereño, M., Landa, U., and Cortazar, R. 2006. "Software Product Lines Structuring Based Upon Market Demands," *SIGSOFT Softw. Eng. Notes* (31:2), p 13.

Halman, J.I.M., Hofer, A.P., and van Vuuren, W. 2003. "Platform-Driven Development of Product Families: Linking Theory with Practice," *Journal of Product Innovation Management* (20), pp 149-162.

Hargadon, A.B., and Douglas, Y. 2001. "When Innovations Meet Institutions: Edison and the Design of the Electric Light," *Administrative Science Quarterly* (46), pp 476-501.

Henderson, R.M., and Clark, K.B. 1990. "Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms," *Administrative Science Quarterly* (35), pp 9-30.

Jonsson, K., Westergren, U., and Holmström, J. 2007. "Technologies for Value Creation: An Exploration of Remote Diagnostics Systems in the Manufacturing Industry," *Information System Journal* (18).

Karlsson, C., and Sköld, M. 2007. "Counteracting Forces in Multi-Branded Product Platform Development," *Creativity and Innovation Management* (16:2), pp 133-141.

Kim, K., and Chhajed, D. 2000. "Commonality in Product Design: Cost Saving, Valuation Change and Cannibalization," *European Juournal of Operational Research* (125:3), pp 602-621.

Kreuger, C. 2002. "Easing the Transition to Software Mass Customization," in: *Lncs 2290,* F. van der Linden (ed.). Berlin: Springer.

Lundbäck, M., and Karlsson, C. 2005. "Inter-Firm Product Platform Development in the Automotive Industry," *International Journal of Innovation Management* (9:2), pp 155-181.

MacMillan, I.C.I., and McGrath, G.R. 1997. "Discovering New Points of Differentiation," *Harvard business review* (75:4), p 133.

Markus, M.L., and Robey, D. 1988. "Information Technology and Organizational Change: Causal Structure in Theory and Research," *Mgmt Science* (34:5), pp 583-598.

Meyer, J.W., and Rowan, B. 1977. "Institutionalized Organizations: Formal Structure as Myth and Ceremony," *American journal of sociology* (83:2), p 340.

Parnas, D.L. 1976. "On the Design and Development of Program Families," *IEEE Trans. Software Engineering* (SE-2:March), pp 1-9.

Pine II, J.B. 1993. *Mass Customization: The New Frontier in Business Competition*. Boston, Massachusetts: Harvard Business School Press.

Pohl, K., Böckle, G., and van der Linden, F. 2005. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Berlin: Springer-Verlag

Robertson, D.D., and Ulrich, K. 1998. "Planning for Product Platforms," *Sloan management review* (39:4), p 19.

Sambamurthy, V., Bharadwaj, A., and Grover, V. 2003. "Shaping Agility through Digital Options: Reconceptualizing the Role of Information Technology in Contemporary Firms," *MIS Quarterly* (27:2), pp 237-263.

Sanchez, R., and Mahoney, J. 1997. "Modularity, Flexibility, and Knowledge Management in Product and Organization Design," *Strategic Management Journal* (17:Special issue: Knowledge and the Firm), pp 63-76.

Scott, W.R. 1995. *Institutions and Organizations*. Thousand Oaks: Sage.

Sköld, M., and Karlsson, C. 2007. "Multibranded Platform Development: A Corporate Strategy with Multimanagerial Challanges," *The Journal of Product Innovation Managment* (24), pp 554-566.

Sosa, M., Eppinger, S., and Rowles, C. 2004. "The Misalignment of Product Architecture and Organizational Structure in Comlpex Product Development," *Management Science* (50:12), pp 1674-1689.

Teece, D. 1986. "Profiting from Technological Innovation: Implications for Integration, Collaboration, Licensing and Public Policy," *Research Policy* (15), pp 285-305.

Tushman, M., and Anderson, P. 1986. "Technological Discontinuities and Organizational Environments," *Administrative Science Quarterly* (31:3), pp 439-465.

Ulrich, K. 1995. "The Role of Product Architecture in the Manufacturing Firm," *Research Policy* (24:3), pp 419-440.

Van de Ven, A. 2007. *Engaged Scholarship: A Guided for Organizational and Social Research.* Oxford University Press.

Zucker, L.G. 1977. "The Role of Institutionalization in Cultural Persistence," *American Sociological Review* (42), pp 726-743.