4-24-2010

Design-task Linkages in Digital Innovation: Software Platforms at Globalcarcorp

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

Ola Henfridsson
Viktoria Institute, ola.henfridsson@wbs.ac.uk

Lisen Selander
Viktoria Institute, Lisen.Selander@viktoria.se

芽|Sprouts Working Papers on Information Systems ISSN 1535-6078

DESIGN-TASK LINKAGES IN DIGITAL INNOVATION: SOFTWARE PLATFORMS AT GLOBALCARCORP

Lena Andreasson
Viktoria Institute, Sweden

Ola Henfridsson
Viktoria Institute, Sweden

Lisen Selander
Viktoria Institute, Sweden

Abstract
The adoption of software platforms in product design can be challenging for manufacturing firms. In particular, embedded linkages between the organization design (task) and product design (design) may counteract attempts to induce more agile and flexible innovation processes. Yet, little research has investigated the influence of software platforms on design-task linkages in digital innovation. This paper addresses this research problem by examining the use of software platforms for instrument cluster design at a global automaker. Drawing on innovation theory, we identify and explicate two types of tensions emerging when digitizing physical products. Related to temporality and design hierarchy, these tensions form the basis for a set of implications for the literatures on platforms and digital innovation.

Keywords: digital innovation, software platform, organization design, product design, case study, automotive industry

Permanent URL: http://sprouts.aisnet.org/10-25

Reference: Andreasson, L., Henfridsson, O., Selander, L. (2010). "DESIGN-TASK LINKAGES IN DIGITAL INNOVATION: SOFTWARE PLATFORMS AT GLOBALCARCORP ," Viktoria Institute, Sweden . *Sprouts: Working Papers on Information Systems*, 10(25). http://sprouts.aisnet.org/10-25

Introduction

The increasing digital content of physical products is radically challenging the innovation processes of established firms (Lenfle & Midler, 2009). On one hand, it multiplies the space of digital options available for augmenting existing offers and launching radically new ones (Jonsson et al., 2008; Yoo et al., 2008). On the other hand, seizing emergent digital options is difficult because established product innovation practices may not involve the necessary IT capabilities or organizational agility (Henfridsson et al., 2009; Sambamurthy et al., 2003).

In the innovation literature, significant attention has been paid to the tension between options provided by new technology and institutionalized practices established over long periods of incremental innovation (Anderson & Tushman, 1990; Hargadon & Douglas, 2001). In particular, the notion of dominant design has been coined to capture how innovation practices typically congeal over time as a template for product innovation within an industry (Teece, 1986; Murmann & Frenken, 2006; Suarez, 2004). Dominant designs help firms organizing their innovation processes so that they capitalize on their intellectual, relational, and technical resources. In particular, established 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 tight linkages between design and task is important to exploit a dominant design, such coupling lowers a firm's capability to respond to technological discontinuities (Anderson & Tushman, 1990; Baldwin & Clark, 2000). The integration of digital technologies in physical products represents such a discontinuity for manufacturing firms (Yoo, Forthcoming). 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 (Svahn et al., 2009) that requires new architectural knowledge (Andersson et al., 2008; Henderson & Clark, 1990). A pressing issue is therefore how to handle these parallel logics by building appropriate linkages between the design structure and task structure of innovation processes.

We investigated this pressing issue by conducting case study research of a global automaker's, GlobalCarCorp, adoption of a software platform for differentiating in-car user interfaces across brands and regions. The adoption of this software platform simultaneously provided new digital options and challenged the established design-task linkages. While platforms for managing differentiation through product families are commonplace at manufacturing firms such as GlobalCarCorp (Cooper et al., 2001; Karlsson & Sköld, 2007; Robertson & Ulrich, 1998) the adoption of platforms for managing software families (Clements & Northrop, 2001; Pohl et al., 2005) is a recent phenomenon. In our case study research at the global automaker, we found a disconnect between traditional platform practices and those evoked by using the new software platform. The research question addressed in this paper is: *how does the adoption of software platforms affect established design-task linkages in the innovation of physical products with increasing digital content?*

The remainder of the paper is structured as follows. Section two reviews the existing platform literatures on product families and software product lines. Section three outlines a theoretical framework with which to understand task-design linkages in digital innovation. While section four presents the research methodology, section five outlines the findings of our case study. Analyzing the case, section six outlines implications for theory and practice, and concludes the paper.

Related Literature

Halman et al. (2003, p.150) depict a platform as "the common basis of all individual products within a product family". It provides a set of generic resources with which to generate derivative products (Pohl et al., 2005). Platforms increase the speed and agility of the development process (Yang & Jiang, 2006), promote diversification (Kim & Kogut, 1996), and facilitate brand differentiation (Sköld & Karlsson, 2007). The adoption of platforms in product innovation is both a question of devising new organizing principles (Cusumano & Gawer, 2002) and adopting new principles for systems development and design (Lyytinen et al., 1998). In this regard, platforms concern both the task structure and the design structure of innovation design (cf. Baldwin & Clark, 2000; Smolander et al., 2008).

For the purposes of this paper, we review two streams of platform literature. We first review the extant body of knowledge on product families in the innovation and technology management literature (see e.g., Halman et al., 2003; Krishnan & Gupta, 2001). We then examine platform thinking for developing software families (Clements & Northrop, 2001; Pohl et al., 2005).

Product Platform Research

Product platforms are important to accomplish differentiation (Aaker, 2003; MacMillan & McGrath, 1997; Pine II, 1993; Robertson & Ulrich, 1998). Conceptualized as product family, product platform, or product line research (Halman et al., 2003; Robertson & Ulrich, 1998), this literature addresses the problem of offering a differentiated product without increasing cost and development time proportionally. For instance, new approaches for designing product platforms (see e.g., Chen & Wang, 2008; Muffatto & Roveda, 2000); and understanding multi-branded product families (Sköld & Karlsson, 2007) have been developed. Much of the product platform research has been conducted in the automotive industry, where the combination of scale advantages and variety in the product portfolio is important.

The role of platforms in developing product families is to provide a common architecture. The common architecture can be used to implement differentiation attributes to maintain or increase the market share and keep customer attention (Aaker, 2003). It allows for the delivery of well-adapted products while maintaining a high degree of commonality of components (Lundbäck & Karlsson, 2005). In this regard, mass-scale advantages can be combined with customization (Pine II, 1993).

Software Platform Research

Even though Parnas (1976; 1978) introduced the idea of software families as early as in the seventies, the bulk of literature on this topic has appeared during the last ten years (Clements & Northrop, 2001; Meyer & Seliger, 1998; Pohl et al., 2005). Reflecting its software engineering origin, the literature on software platforms and software product lines is design-oriented in its emphasis on platform development for creating and managing software families. As outlined by Clements and Northrop (2001, p. 5) "a software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way".

The emphasis on identifying commonalities within the software family and the options of variability makes software product line engineering geared towards software-based differentiation. The creation of variation through software product lines is enabled by tailoring components of a common platform through preplanned variation mechanisms. Variation mechanisms include assembling and adding new components (Northrop, 2002). Using a

common platform, costly development of new tools and models for each product change is virtually avoided. Thus, firms adopting software platforms and software product line engineering are promised to achieve product differentiation in a more cost-effective way. Indeed, it is argued that the use of software platforms for creating software families 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).

Transcending the Disciplinary Boundaries

The increasing digital content of physical products makes software platforms and families increasingly relevant for manufacturers of physical products such as airplanes, automobiles, and consumer devices. As Lenfle and Midler (2009, p. 156) note, however, "this does not mean that physical goods disappear from our universe but that they are more and more associated with complex services". In other words, the adoption of software platforms to increase variation and differentiation cannot be done at the expense of traditional product platform thinking.

A software platform initiative typically constitutes a major design effort that might transform established development practices while providing opportunities for new ways of organizing (Jones, 2003). In other words, the problem facing firms engaged in digital innovation is that product platforms and software platforms must be handled simultaneously. Given the existence of a dominant design (Murmann & Frenken, 2006; Teece, 1986) manifested in product platforms, this problem will involve leveraging software platforms within and across established design-task linkages. Next section introduces our theoretical framework and the notion of design-task linkages.

Theoretical Framework

Our theoretical framework draws on literature that examines the relationship between organization design and product design in innovation (Baldwin & Clark, 2000; Henderson & Clark, 1990; Sosa et al., 2004). The alignment of these entities in the innovation process is important to enable effective product development (Henderson & Clark, 1990). In fact, Baldwin and Clark (2000) even refer to this relationship as the "fundamental isomorphism". In doing so, they not only underline the reciprocal relationship between the organization design and product design but also the inherent difficulty to change either of them.

The organization of an innovation process can be described as a set of tasks needed to be performed to realize a design (Baldwin & Clark, 2000). The task structure implies organizational elements that link the organization design to the product design. Examples of such elements are communication channels, information filters, and problem-solving strategies (Henderson & Clark, 1990). *Communication channels* represent the formal and informal ways by which different groups of people interact and transfer knowledge within the organization (Baldwin & Clark, 2000; Sosa et al. 2004). In such communication, organizations develop shared *information filters* for distinguishing relevant information from irrelevant information, working as a type of attention-structures (Henfridsson, 2000; March & Olsen, 1976). Finally, organizations develop *problem-solving strategies* for handling routine problems in a standardized way.

Henderson and Clark (1990) demonstrate that the task structure used by a firm typically embed the architectural knowledge of the product. In this regard, communication channels, information filters, and problem-solving strategies work as the glue, or *linkages*, between tasks and the product design. Encompassing its architecture and functions, the product *design* structure is a description of the innovation (Baldwin & Clark, 2000). In this regard, it represents not only the technical blueprint of the innovation but also defines the

adaptability of the design to changing circumstances and markets. Typically, physical products embed a hierarchical structure (Alexander, 1964; Clark, 1985; Murmann & Frenken, 2006), enabling flexibility because of its implementation of independence between decomposed elements of the innovation (Ulrich, 1995; Baldwin & Clark, 1997; Sanchez & Mahoney, 1997).

Given the theoretical background above, we refer to design-task linkages as *the institutional arrangements enacted by designers that align the set of tasks performed to realize an innovation and its design architecture.* Following Henderson and Clark (1990), we refer to institutional arrangements as *communication channels, information filters*, and *problem-solving strategies*. As Sosa et al. (2004) argue, however, it would be naïve to expect a perfect match between task and product in complex product development. This paper is based on the assumption that the introduction of digital content in a physical artifact, such as a car subsystem, will disturb the linkages between task and design in the innovation process. The established linkages build on a dominant design conveying a hardware and product platform focus, while the new and emerging linkages need to handle the parallel logics of hardware and software (Svahn et al., 2009).

Research methodology

Research Setting and Design

In order to address the research question of how the adoption of software platforms affects established design-task linkages in the innovation of products with increasing digital content, we conducted case study research at one of the world's largest automakers, GlobalCarCorp. We selected GlobalCarCorp because they had recently adopted a new software platform for designing a family of user interfaces for car instrument clusters. Because GlobalCarCorp hosts many car brands that are sold globally, the automaker had strong incentives to identify and develop product platforms for making differentiation across brands and regions economically feasible.

Designed as case study research (Eisenhardt, 1989), our 20-month study was initiated in 2007 and focused on the process by which differentiation of instrument clusters was accomplished at GlobalCarCorp. Exemplifying process research rather than variance research, the intention was to identify and predict "patterned regularities over time" (Markus & Robey, 1988). This emphasis guided our data collection and analysis towards the decision, events, and activities (Langley, 1999) related to the adoption of the software platform (SoftClusterPlatform) at GlobalCarCorp.

While there existed a number of different roles related to HMI (Human Machine Interface) development at GlobalCarCorp, the roles can be divided into two main categories; HMI engineers (such as ergonomists and interaction designers) and design engineers (such as product design engineers, marketers, and functional unit engineers). With the implementation of the SoftClusterPlatform however, the HMI development group expanded to include yet another category; software engineers. Prior to the SoftClusterPlatform, the HMI engineers' main responsibility were to develop specifications for suppliers. Meanwhile, the design engineers focused on graphical design, marketing strategies, and development of functional units (e.g., FM tuners and navigation). Design engineers also provided the HMI engineers with necessary information for specification development. However, as indicated, the relationship between these groups changed considerably with the implementation of the SoftClusterPlatform and the decision of in-housing software development. This will be further elaborated in the following sections.

Data Collection

Concurring with the typical case study, our data collection included multiple data sources such as participant observation, semi-structured interviews, workshops, and documents review. Figure 1 depicts the time line of the data collection.

First, the first author of this paper spent considerable time at GlobalCarCorp with the intention of developing a thorough understanding of design-task linkages. During a six-month period (October 2007- May 2008), she even had a desk at the research site, attending meetings and interacting with employees at GlobalCarCorp on a weekly basis. The first author's presence at GlobalCarCorp enabled informal conversations with respondents during coffee breaks, lunches, and meetings. Such conversation added depth to the data material and enabled a fine-grained understanding of the software platform initiative at GlobalCarCorp. A total of about 80 contacts were documented during this *participant observation*.

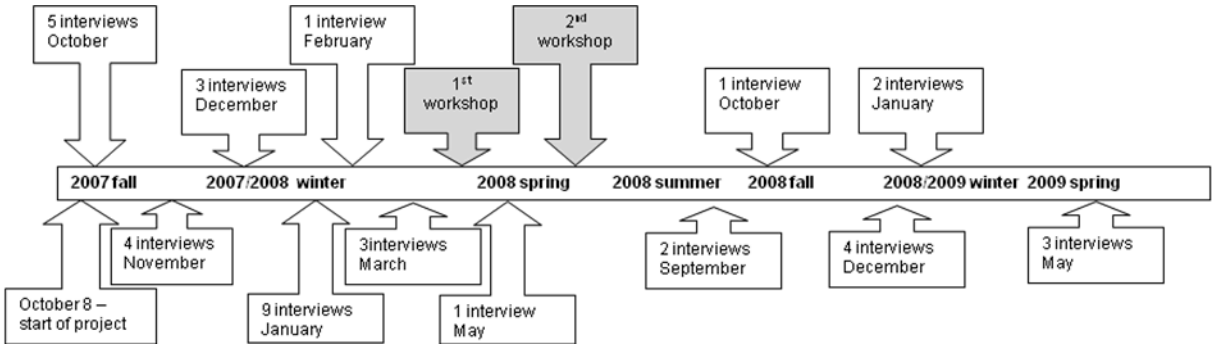

Figure 1: Time line of data collection

Second, we conducted, recorded, and transcribed 38 *semi-structured interviews*. The average length of the interviews was 64 minutes with a standard deviation of 24 minutes. Respondents ranged from managers to developers (see Table 1), covering expertise in areas such as ergonomics (HMI engineers), product design engineers (design engineers) and software development (SW engineers).

Category	*Role*	*Number of interviews*
HMI engineers	HMI managers	4
	Ergonomists	4
	Interaction designers	5
Design engineers	Product design engineers	2
	Marketing	1
	Functional unit responsible	1
SW engineers	SW managers	6
	SW engineers	15

Table 1: Overview of interviews; HMI development group

Third, two *workshops* were organized to validate the empirical findings over time. They allowed us to refine our understanding of the case and complement facts. In addition, consistent with the notion of engaged scholarship (Van de Ven, 2007), this was an

opportunity for GlobalCarCorp practitioners to reflect upon their practice in view of our findings and recommendations.

Finally, *documents review* included assessments of system specifications and concept descriptions. Such assessment was important for, e.g., tracking the specific details of the software platform investigated.

Data Analysis

The data analysis was supported by a qualitative data analysis software package (ATLAS.ti). The initial analysis involved open coding, that is, naming and taking segments of data apart (Charmaz, 2006; Strauss & Corbin, 1998). This open coding process generated hundreds of codes, which shaped the entire empirical frame from which we built our analysis.

The open coding procedure was followed by focused coding, searching for emergent core categories in the material (Charmaz, 2006; Miles & Huberman, 1994). This involved a linking and consolidating process in which we examined the relationship between codes, crystallizing the experiences and interpretations of the software platform initiative at GlobalCarCorp. All in all, the focused coding resulted in five coding families; *architecture, communication, software development process, differentiation* and *organization*. These coding families established a strong analytical ground from which to work with the data.

After discussing the coding scheme within the research team, we examined the empirical findings against the literature, understanding the coding families from a theoretical perspective. Drawing on innovation literature, linkages between design and task structures were deemed to be a suitable lens to use for exploring the disconnect between the traditional platform practices and the new software platform initiative at GlobalCarCorp.

The SoftClusterPlatform at GlobalCarCorp

In 2005, one subdivision of GlobalCarCorp received global responsibility for R&D on instrument clusters. This responsibility was awarded in a transition period of instrument cluster design. The use of digital displays in the instrument cluster enabled more flexible ways of presenting information to the driver. Digital displays enable presentation of a variety of driver information such as tire pressure, radio frequency, driving speed, and navigation information (see Figure 2).

Figure 2: Instrument cluster with reconfigurable digital display

This change in technological basis introduced a range of options for digital differentiation. GlobalCarCorp decided to exploit this potential by initiating a software platform initiative. Given its traditional hardware basis, however, the software platform initiative challenged the established linkages between task and design structures. In particular, it slowly changed the skills and capabilities required for excelling in innovation, since instrument clusters were increasingly dependent on embedded digital technology. In addition,

the global responsibility was transforming the scope of the instrument cluster group. A software manager described:

A decision was taken that we [GlobalCarCorp] should own all strategically important software. […] the first step was to take control over the HMI software, this was followed by a need to control the application software as well. For us, this meant that we were in a need of change.

Previously, the HMI development group designed clusters for a single car brand within GlobalCarCorp's multitude of brands. Now, they were confronted with the challenging task to handle a portfolio of different clusters across brands, car models, customer segments and regions. This was found to be a comprehensive task:

We are supposed to work as a central coordinator. In other words, we get information from different sources that we need to structure and determine which requirements to address. It's like 'I can fulfill your requirements but I can't fulfill his requirements at the same time' […] One cannot implement and support both requirements […] If you are lucky someone takes the decision of what to prioritize for you, but it's a process that needs some sort of structure.

One of the major barriers to handle the newly awarded responsibility was the traditionally slow pace of the product development process, and the difficulty of finding ways to communicate within the global organization. Serving multiple brands and models, the escalating number of change requests from departments across the globe was increasingly pressing. The HMI development group was supposed to coordinate all (world wide) incoming requirements.

It's all about costs. There used to be at least three different organizations within GlobalCarCorp that developed these systems and architectures based on their own perceptions on what they think is good or bad. This drives costs so now they want us to join as one.

Initiating this transformation, GlobalCarCorp pursued an economy of scale strategy with the intention to gain scale advantages on the hardware side, while accomplishing differentiation across brands, models, and regions through software. Initiating the work with in-housing the software development process, GlobalCarCorp identified five product families to organize and manage instrument cluster variants for over 80 car models, amounting to approximately 3.8 million cars each year until 2012. In particular, the new strategy involved novel software architecture for increasing flexibility and new ways of structuring current design processes, as noted by one of the software engineers:

The idea was to become more flexible and be able to implement and modify the HMI software very late in the development process. Management did put lot of weight on flexibility and on enabling new ways for differentiation between brands.

This need of increased flexibility was guided by organizing clusters into different families. Each cluster family was divided into different levels: base, mid, and high, representing levels of appearance and features. Given the global responsibility and the multitude of clusters, the complexity of handling software variants became considerable.

Previous Design and Task Linkages at GlobalCarCorp

Prior to the software platform initiative, different groups of HMI engineers, located worldwide, were responsible for the instrument cluster. Based on input from design departments and functional managers, they developed HMI requirements and specifications,

corresponding to the overall cluster design, that later would be implemented by the selected instrument cluster supplier (see Figure 3).

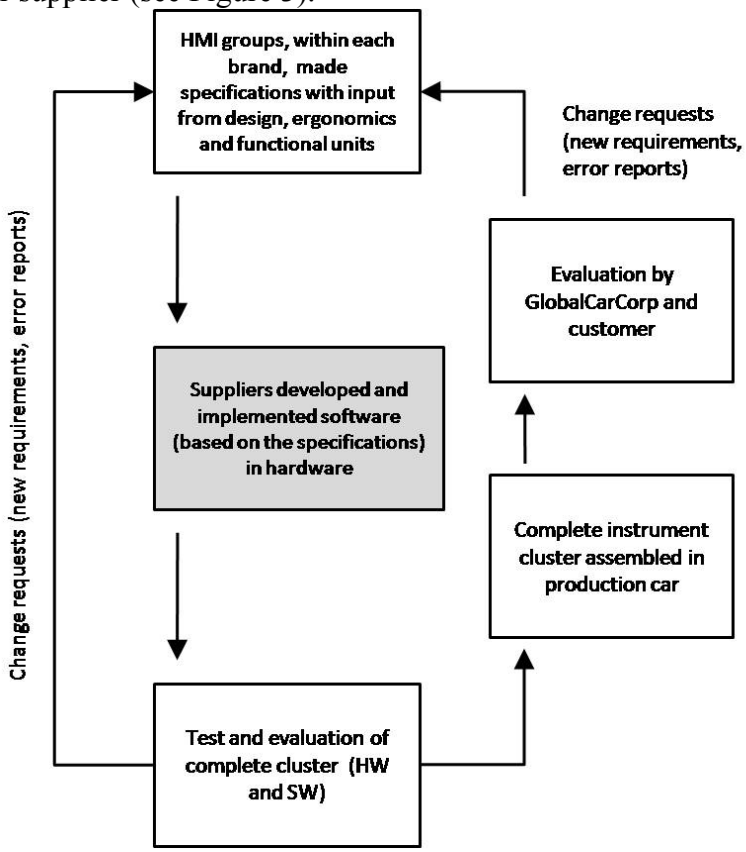

Figure 3: GlobalCarCorp's old task structure for instrument cluster design

This task structure had been institutionalized over years of HMI development at GlobalCarCorp. 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, updates, and new requirements were collected and submitted to the supplier as change requests. In effect, the iterative element between the supplier and GlobalCarCorp was comprehensive, sometimes even tedious. As illustrated by the words of a HMI engineer:

The easiest way to describe our former development process was that we verbally described what we wanted to accomplish. […] We did a lot of flowcharts and power points, that type of visual documentation.[…] and then we went to the supplier and interacted with them, asking each other questions like "how should we solve this" and "this situation is unique, we have to make an adjustment". After that we updated the specification with the information that the supplier needed. It was an interactive process between us and the supplier.

Moreover, each iteration included tasks, such as, specifying, testing, and improving the final cluster design. This process was not only time-consuming and expensive due to much iteration between GlobalCarCorp and the supplier; it was unique, although similar, for each brand within GlobalCarCorp.

Differences in requirements interpretation between GlobalCarCorp engineers and the supplier often resulted in a gap between the final implementation and GlobalCarCorp's original intention. These misunderstandings between suppliers and GlobalCarCorp were frustrating, not least because new requirements emerged over time as well. Such requirements

typically included seemingly minor issues such as the change of a word, bitmap, or a color. An HMI engineer described the requirement process:

We felt as if we were in the hands of the suppliers. Even a minor change such as a spelling mistake, a change of color or appearance demanded several iterations [with the supplier] and cost a lot of money.

In fact, as another engineer noted, late change requests were anticipated by the supplier, essentially working as an important component in their business model. So, in 2005, with the idea of the new software platform, GlobalCarCorp attempted to change the old cluster task structure by in-housing the HMI software development process. As a response to the problems documented above, GlobalCarCorp hired two software firms to help them develop a software platform that would facilitate flexible generation of variants within a cluster family. We refer to this platform as the SoftClusterPlatform, referring to its intention to speed up and facilitate the handling of change requests through software.

The SoftClusterPlatform

The SoftClusterPlatform consisted of four main components; a database, a communication protocol, a HMI engine and an editor that facilitated design, development and deployment of digital user interfaces for instrument clusters (see Figure 3).

First, the XML-based database allowed for differentiation of software embedded in instrument clusters. It was developed to help mapping existing and forthcoming product families by pre-defined software components and templates. The database contained all the logic and appearance needed for a graphical interface, for example priority handling, animations, and visual effects, as well as text strings for different languages.

Second, the communication protocol allowed for a more flexible way of managing functional units such as the tuner, phonebook, and thermometer, as well as units related to the condition of the car. Moreover, dynamic data such as the set radio frequency from the tuner or current incoming call from phonebook could be handled. Essentially, the protocol was designed to decouple the functional units from the HMI software.

Third, using the editor, a software engineer could differentiate the common cluster display through changes to the XML-files controlling static data such as font, language, colors, layout, and graphics. The editor facilitated specification of an entire graphical interface with all its elements along with converting the database to a binary format for the HMI engine to use.

Finally, the HMI engine interpreted the database and received dynamic information from the functional units. As illustrated in Figure 4, static data from the database, dynamic data from functional units, and user inputs employed the HMI engine to generate the graphical user interface on a display.

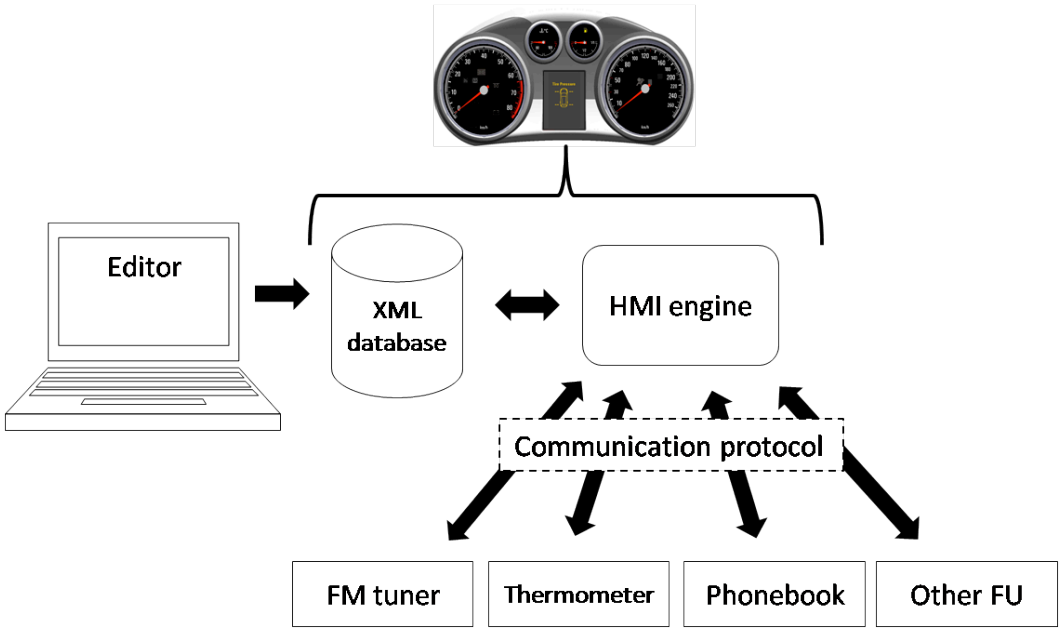

Figure 4. The SoftClusterPlatform.

So, unlike the previous cluster development process, the SoftClusterPlatform would enable the development of one basic software module for all types of displays in a specific cluster family. As an example, the SoftClusterPlatform handled text strings (notations and characters) in multiple languages. At this time, GlobalCarCorp implemented, updated, evaluated, and maintained 23 languages simply for the instrument cluster.

With the possibility to reuse and inherit patterns stored in databases along with a flexible way to handle new functional units, new graphical interfaces could be developed in an agile manner. One of the HMI engineers commented on the advantages of the platform:

The advantage of the editor [which is the component most visible to the people developing the HMI software] is that we can test, visualize and evaluate so many more possibilities. If I have an idea I can go to Peter or Mike and explain my idea and after a couple of minutes they show me the solution and ask "Is this what you were thinking of?" […] It can be visualized directly with the editor.

Traditionally, the process of implementing new interfaces was mediated through sketches using PowerPoint or Visio. With the SoftClusterPlatform, however, this process would be more structured and the cycle of implementation and evaluating new graphical interface proposals would be shortened.

Using the Platform

Before implementing the SoftClusterPlatform, each brand-specific cluster had a responsible manager. With the new initiative, however, the governance of the clusters was attributed to the HMI development group with a single HMI manager responsible for the SoftClusterPlatform. Developing this platform, however, little focus was put on transforming the task structure to harvest the anticipated benefits of the platform. This problem became increasingly clear to the software engineers:

It appears as if the organization cannot deal with this. It feels as if we have this "jet engine" assembled on a bicycle.

Given that the supplier previously did the software implementation, GlobalCarCorp's design group was dominated by cognitive ergonomics and interaction design competences, rather than software expertise. Using the SoftClusterPlatform, programming skills in general and XML competence in particular was required. However, the organization was reluctant to change and conform to the new situation. In particular, the traditional way of developing instrument clusters, with significant focus on hardware components and their design structure and coupled task structure, virtually remained at the automaker (see Figure 5).

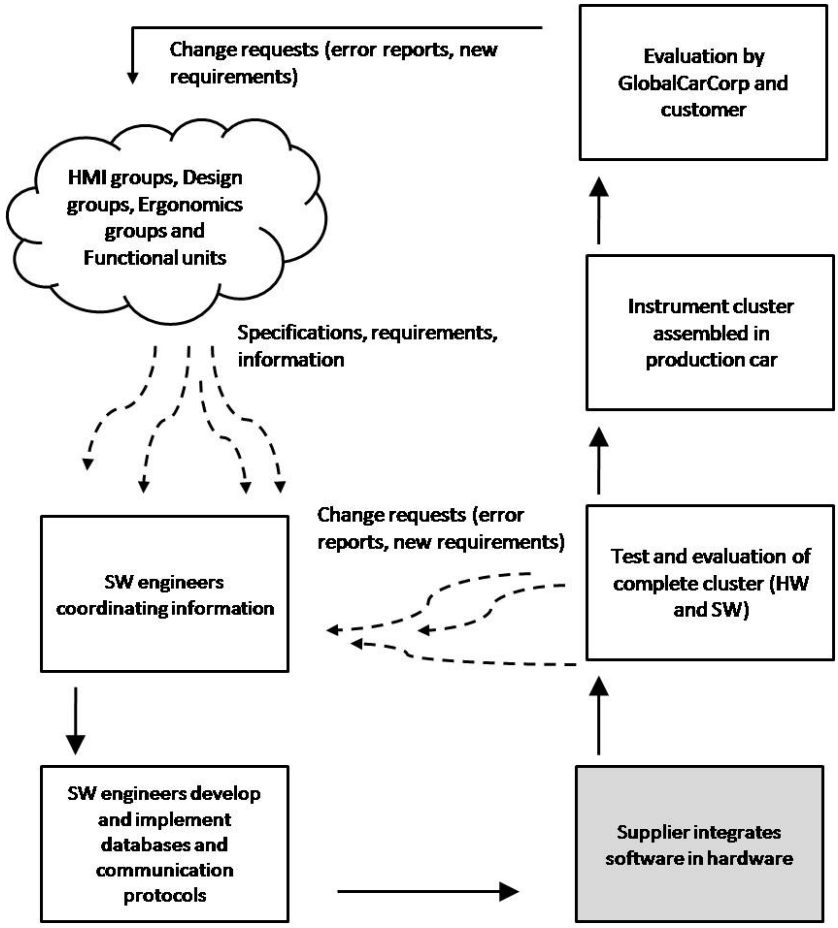

Figure 5 The emergent task structure for instrument cluster design

One example of this prevalence of institutionalized practices was the lack of standard operating procedures for gathering and communicating software requirements from GlobalCarCorp stakeholders. One of the group managers for software development observed:

We have never succeeded in uniting the people making the software with them doing prototypes and the people doing cliniques [tests and evaluations]. Such a process never existed; we haven't succeeded in getting that together at GlobalCarCorp. It is too much focus on the platform. We should have focused more on the process and organization than on the platform.

In this regard, software engineers were required to work proactively to gather relevant information. In order to meet deadlines and overall productivity pressure, software engineers typically had to make decisions outside their obligations. Essentially, they had to develop informal ways of gathering requirements, typically drawing on requests from people they already knew or were referred to by other employees. As one software engineer explained:

I suppose that the lack of processes and general strategies for this [requirement process] is a result of the previous processes. Back then you collected all documented requirements and simply sent it to the supplier producing the cluster. All you said was 'do this and come back when you're done'. Everything was done outside of the organization by the supplier. […]Today the whole requirement process is vague, many of the requirements come to you as post-it notes or by e-mail. It causes confusion, you know 'too many cooks spoil the broth'.

Ideally, the requirement gathering process would involve input from people working on marketing, interaction design, and functional units (e.g., radio, antenna, or navigation). However, without any generic process in place, software engineers often faced conflicting requirements and had to rely on individual judgment. As illustrated by a software engineer:

We often need to run around and find people with whom to discuss what to implement. Then we get a pretty good picture of how the function, and its HMI [graphical interface], should look like, what kind of details and where they belong and things like that […] but there is no process, no formal way of doing it.

Before adopting the SoftClusterPlatform, the tasks of the HMI engineer were separated from other modules of the instrument cluster. In fact, many decisions were made by the supplier, including the important tasks of pooling and integrating requirements. With the SoftClusterPlatform, integrating requirements were in the hands of software engineers. This was a difficult transformation:

We have this platform, like a tool box, and we can do anything that the toolbox allows us to do […] but if the company does not know what's in the toolbox, then we have a huge problem. This is something that we need to improve on an organizational level, the design and implementation tasks need to be much more coordinated […] it's all about communication channels.

The new task and design structures required work groups to interact to build and renew knowledge that previously rested with the supplier. While the newly employed software engineers faced the results of the new platform, other stakeholders at GlobalCarCorp were largely unaware of the requirement process. As a result, they held on to the old task and design structures, established before introducing the software platform initiative.

Discussion and Implications

We set out to analyze how the adoption of software platforms affects design-task linkages in the innovation of physical products with increasing digital content. With this focus in mind, we conducted case study research of a software platform initiative at GlobalCarCorp. In what follows, we first use Henderson and Clark's (1990) innovation theory to zoom in on problem-solving strategies, communication channels, and information filters for tracing changes in design-task linkages when the SoftClusterPlatform was adopted at GlobalCarCorp. This analysis is then used for generating insights about tensions between the physical and the digital in product innovation. Lastly, we outline implications for research and practice.

Design-Task linkages at GlobalCarCorp

Problem-Solving Strategies. Organizations develop *problem-solving strategies* to tackle problems in its immediate environment. As linkages between product design and organization design, such strategies offer solutions to routine problems and is reflected in the organization's architectural knowledge (Henderson & Clark, 1990).

At GlobalCarCorp, years of experience of working with instrument cluster design had established a set of solutions to routine problems that were challenged by the adoption of the SoftClusterPlatform. In particular, problems-solving strategies associated with the traditional

set-up between GlobalCarCorp and suppliers did not resonate well with the virtues of the new platform. Considering that the suppliers used to deliver the cluster as a package consisting of both hardware and software, GlobalCarCorp's decision to take command over the software side of the cluster made some of the previous strategies obsolete. For instance, the requirements gathering process assumed the former set-up with suppliers. This was reflected in the difficulties for the newly appointed software engineers to gather change requests.

With the SoftClusterPlatform, GlobalCarCorp had the technical resources for accomplishing agile prototyping and update opportunities, problems frequently experienced when communicating with suppliers. However, the specific software knowledge rested with the suppliers, which made it difficult to draw on these resources in an efficient way. Essentially, GlobalCarCorp lacked routine solutions to everyday design problems related to the emergent task structure. This was particularly clear in the case of software updates (one of the promised benefits with the platform), which, in practice, failed to exploit the capability of the SoftClusterPlatform. Even with the recruitment of software engineers, the remainder of the organization continued to rely on established problem solving strategies. As a result, requirements gathering and change requests were handled ad hoc on a case-by-case basis, creating a stressful and dysfunctional work environment. The unmatched interactions (Sosa et al., 2004) between the software engineers and the rest of the organization indicated design-task misalignments at GlobalCarCorp.

Communication Channels. *Communication channels* represent the formal and informal ways by which different groups interact, and transfer knowledge, within the organization (Sosa et al., 2004; Baldwin & Clark, 2000; Henderson & Clark, 1990). As such, communication channels provide an indication of people's knowledge of how different product components are interconnected (Henderson & Clark, 1990; Sosa et al., 2004).

The communication channels at GlobalCarCorp reflected the HMI development groups, including design tasks and HMI engineering. This high degree of group specialization, associating people with a specific range of tasks, was greatly institutionalized at GlobalCarCorp. Although allowing concurrent design, it imposed barriers to cross-team interaction. Prior to the SoftClusterPlatform, the different HMI development groups communicated directly with suppliers. By-passing the supplier, GlobalCarCorp engineers perceived a capability gap at the interface between hardware and software.

The SoftClusterPlatform imposed cross team interactions, changing both internal and external communication channels at GlobalCarCorp. The network nature of the software platform triggered attempts to find new communication channels across teams. For instance, the software engineers, who implemented the HMI software, communicated interactively with ergonomics, HMI engineers, and different functional groups. Additionally, realizing the potential of the software platform, software engineers struggled with having other groups communicating with them on a regular basis about possible new designs.

Information Filters. Organizations develop *information filters* for identifying the most important information for certain tasks (Henderson & Clark, 1990). Over time, such filters become an important organizational element enabling efficient product development.

Traditionally, the differentiation of instrument clusters was in the hand of brand-specific groups at GlobalCarCorp. Over the years, these groups developed a number of practices that helped them distinguish relevant input from irrelevant input in the design process. For instance, they drew on a set of design principles that would distinguish their brand from other brands in the same product family. Such differentiation was typically accomplished through associating car brands with different visual expressions. Examples of such expressions in the cluster display would be particular uses of fonts and colors.

With the SoftClusterPlatform initiative, this differentiation was not only centralized to the investigated subdivision of GlobalCarCorp but also involved a new type of organizational

role: the software engineers. The actual deployment of visual expressions was now done by software engineers, rather than by HMI engineers in cooperation with the supplier. As a result, the traditional information filters, involving certain explicit and implicit design principles, were challenged.

This challenge was manifested in the difficulties that software engineers experienced when attempting to prioritize between different requirements. Without the long experience of the traditional HMI engineers, they were much reliant on other stakeholders when implementing cluster designs. They were unable to accomplish swift filtering of requirements, especially when requirements were in conflict with each other. In addition, they typically needed to be very active in collecting necessary information for implementing a requirement. Other stakeholders still operated with previous task structures in mind, meaning that the software engineers sometimes felt by-passed.

Despite serving as gatekeepers (Sosa et al., 2004; Tushman, 1977) software engineers' little experience from instrument cluster design and HMI engineering did not have a basis for filtering information. As a result, software engineers were frustrated about the seemingly ad-hoc basis of their work.

Digital-Physical tensions

So, how does the adoption of software platforms affect established design-task linkages in the innovation of physical products with increasing digital content? Our study at GlobalCarCorp suggests that increasing digital content in physical artifacts is a challenge that cuts across the organization. GlobalCarCorp's disintegration of the instrument cluster as a hardware component and the software used for driver information and visual expression offered new flexibility. The newly recruited software engineers could now potentially address change requests swiftly, traditionally handled through close supplier interaction with long lead times and high costs. The redefined supplier relationship created new roles (e.g., software engineer) and redefined roles (e.g., HMI engineer) at the automaker. Indeed, the software platform initiative rendered some important implications associated with the fundamental isomorphism between design structure and task structure (Baldwin and Clark 2000).

We propose two specific digital-physical tensions that influence linkages between organization design and product design. First, the digitization of physical artifacts is characterized by an inherent *temporal tension* between the processes linked to manufacturing and the processes linked to software design. As Fine (1998) succinctly points out, industries operate at different clockspeeds. The automotive industry is slower than the average industry, compensating massive investments in new car lines with product lifecycles of four years or longer. At the other end of the continuum, the software industry is characterized by a considerably faster rhythm, where the lack of physical constraints makes new product releases commonplace and frequent. In the digitized instrument cluster, these two industries met, creating dissonance with the traditional automotive way of working. The SoftClusterPlatform was an attempt to handle this dissonance but the anticipated benefits were not gained because of its disturbance of established problem-solving strategies, information filters, and communication channels.

Second, the increasing digital content of physical artifacts involves a *design hierarchy tension*. As documented in the product innovation literature (Clark 1985, Murmann and Frenken 2006), physical products imply a hierarchical structure where the product-lead firm (e.g., the automaker) controls the design and its different layers of suppliers. In practice, this hierarchical structure largely influences the everyday work of automotive engineers relying on clear-cut relationships with suppliers, expected to deliver according to requirements and change requests. In contrast, software is characterized by a networked structure, where software patterns can be reused as solutions in different contexts without any imposed

hierarchical order. Blending these two logics in the same design will inevitably create dissonance in established design-task linkages at industrial age companies such automakers.

Implications

Our analysis proposes two tensions – temporality tension and design hierarchy tension – that emerge when digitizing physical artifacts. Reflecting the inherent difference between the physical and the digital, these tensions are manifested in the linkages between the organization design and product design. Our examination of problem-solving strategies, communication channels, and information filters (Henderson and Clark 1990) at GlobalCarCorp shows that the established way of linking the task structure and the instrument cluster was severely challenged.

We argue that the management of the co-existing digital-physical logics requires careful attention to design-task linkages. Although our study was conducted in the automotive industry, this is also true for other industries where physical products are digitized (cf. Benner, 2010; Trispas, 2009). Looking at the two different streams of platform literature, little, if any, attention is paid to digital innovation and the tensions following the co-existence of product and software platforms. The perspective developed throughout this paper provides insights that are new to both platform literatures.

First, the literature on product families (Halman et al., 2003; Karlsson & Sköld, 2007; Robertson & Ulrich, 1998) pays scant attention to the increasing digitization of physical products. As a result, it does not recognize either temporality tensions or design hierarchy tensions. Originating in industries with long product life cycles, this stream of research cannot cater for variety produced through software, characterized by a higher clockspeed than physical products. It assumes that product families are long-term investments, where platform thinking primarily aims at making a product platform endure to achieve the mass-scale advantages required to harvest massive investments in technical development and production. Similarly, the software platform literature (Clements & Northrop, 2001; Pohl et al., 2005) focuses on the variety created through digital technology. However, it tends to disregard how software platform deployment is embedded in a surrounding task structure, dominated by a manufacturing paradigm.

Our research suggests that the increasing digital content of physical products paves the way for a new emerging stream of research. Accommodating findings in the product innovation and software engineering in a concerted effort may be useful in generating new insights on this topic. Given that these literatures are currently divided along disciplinary lines in their investigation of platforms, it may be argued that such accommodation is best done by researchers who appreciate both organizational and technical issues. IS researchers are therefore well-positioned to investigate the role of temporality and clockspeed, as well as design hierarchy tensions, as these issues emerge as fundamental problems for firms that orchestrate innovation in the digital age.

References

Aaker D (2003) The power of the branded differentiator. *Sloan Management Review*, 83-87.

Alexander C (1964) *Notes on the synthesis of form*. Harvard University Press, Cambridge, MA.

Anderson P and Tushman M (1990) Technological discontinuities and dominant designs: A cyclical model of technological change. *Administrative Science Quarterly* **35**(4), 604-633.

Andersson M, Lindgren R and Henfridsson O (2008) Architectural knowledge in inter-organizational IT innovation. *Journal of Strategic Information Systems* **17**, 19-38.

Baldwin C and Clark K (1997) Managing in the age of modularity. *Harvard Business Review* **75**(5), 84-93.

Baldwin C and Clark K (2000) *Design rules: The power of modularity* MIT Press, Cambridge, MA.

Benner M (2010) Securities analysts and incumbent response to radical technological change: Evidence from digital photography and internet telephony. *Organization Science* **21**(1), 42-62.

Charmaz K (2006) *Constructing grounded theory: A practical guide through qualitative analysis.* Sage Publications Ltd.

Chen C and Wang L (2008) Product platform design through clustering analysis and information theoretical approach. *International Journal of Production Research* **46**(15), 4259-4259.

Clark KB (1985) The interaction of design hierarchies and market concepts in technological evolution. *Research Policy* **14**(5), 235-251.

Clements P and Northrop L (2001) *Software product lines: Practices and patterns*. Addison-Wesley Professional.

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), 361-380.

Cusumano MA and Gawer A (2002) The elements of platform leadership. *MIT Sloan Management Review Spring*, 51-58.

Eisenhardt KM (1989) Building theories from case study research. *Academy of Management Review* **14**(4), 532-550.

Ereño M, Landa U and Cortazar R (2006) Software product lines structuring based upon market demands. *SIGSOFT Software Engineering Notes* **31**(2), 13.

Halman JIM, Hofer AP and van Vuuren W (2003) Platform-driven development of product families: Linking theory with practice. *Journal of Product Innovation Management* **20**, 149-162.

Hargadon AB and Douglas Y (2001) When innovations meet institutions: Edison and the design of the electric light. *Administrative Science Quarterly* **46**, 476-501.

Henderson RM and Clark KB (1990) Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. *Administrative Science Quarterly* **35**, 9-30.

Henfridsson O (2000) Ambiguity in IT-adaptation: Making sense of first class in a social work setting. *Information Systems Journal* **10**, 85-104.

Henfridsson O, Mathiassen L and Svahn F (2009) Reconfiguring modularity: Closing capability gaps in digital innovation. *Sprouts: Working Papers on Information Systems* **9**(22),

Jones C (2003) Variations in software development practices. *IEEE Software* **20**(6), 22-27.

Jonsson K, Westergren U and Holmström J (2008) Technologies for value creation: An exploration of remote diagnostics systems in the manufacturing industry. *Information Systems Journal* **18**(3), 227-246.
Karlsson C and Sköld M (2007) Counteracting forces in multi-branded product platform development. *Creativity and Innovation Management* **16**(2), 133-141.
Kim D-J and Kogut B (1996) Technological platforms and diversification. *Organization Science* **7**(3), 283-301.
Krishnan V and Gupta S (2001) Appropriateness and impact of platform-based product development. *Management Science* **47**(1), 52-68.
Langley A (1999) Strategies for theorizing from process data. *The Academy of Management Review* **24**(4), 691-710.
Lenfle S and Midler C (2009) The launch of innovative product-related services: Lessons from automotive telematics. *Research Policy* **38**(1), 156-169.
Lundbäck M and Karlsson C (2005) Inter-firm product platform development in the automotive industry. *International Journal of Innovation Management* **9**(2), 155-181.
Lyytinen K, Rose G and Welke R (1998) The brave new world of development in the internetwork computing architecture (internca): Or how distributed computing platforms will change systems development. *Information Systems Journal* **8**(3), 241-253.
MacMillan I and McGrath R (1997) Discovering new points of differentiation. *Harvard Business Review* **75**(4), 133-145.
March JG and Olsen JP (1976) *Ambiguity and choice in organizations*. Universitetsforlaget, Bergen.
Markus ML and Robey D (1988) Information technology and organizational change: Causal structure in theory and research. *Management Science* **34**(5), 583-598.
Meyer MH and Seliger R (1998) Product platforms in software development. *Sloan Management Review* **40**(1), 61-74.
Miles MB and Huberman AM (1994) *Qualitative data analysis*. Sage, California.
Muffatto M and Roveda M (2000) Developing product platforms: Analysis of the development process. *Technovation* **20**(11), 617-630.
Murmann JP and Frenken K (2006) Toward a systematic framework for research on dominant designs, technological innovations, and industrial change. *Research Policy* **35**(7), 925-952.
Northrop LM (2002) Sei's software product line tenets. *IEEE Software* **19**(4), 32-40.
Parnas DL (1976) On the design and development of program families. *IEEE Transactions on Software Engineering* SE-**2**(March), 1-9.
Parnas DL (1978) Designing software for ease of extension and contraction. *Proceedings of the 3rd International Conference on Software Engineering*, IEEE Press, Atlanta, Georgia, United States, pp 264-277.
Pine II JB (1993) *Mass customization: The new frontier in business competition*. Harvard Business School Press, Boston, Massachusetts.
Pohl K, Böckle G and van der Linden F (2005) *Software product line engineering: Foundations, principles, and techniques.* Springer-Verlag Berlin.
Robertson D and Ulrich K (1998) Planning for product platforms. *Sloan Management Review* **39**(4), 19-31.
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), 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), 63-76.

Sköld M and Karlsson C (2007) Multibranded platform development: A corporate strategy with multimanagerial challanges. *The Journal of Product Innovation Managment* **24**, 554-566.
Smolander K, Rossi M and Purao S (2008) Software architectures: Blueprint, literature, language or decision? *European Journal of Information Systems* **17**(6), 575-588.
Sosa M, Eppinger S and Rowles C (2004) The misalignment of product architecture and organizational structure in comlpex product development. *Management Science* **50**(12), 1674-1689.
Strauss A and Corbin J (1998) *Basics of qualitative research: Techniques and procedures for developing grounded theory*. SAGE, Thousands Oaks, CA.
Suarez FF (2004) Battles for technological dominance: An integrative framework. *Research Policy* **33**, 271-286.
Svahn F, Henfridsson O and Yoo Y (2009) A threesome dance of agency: Mangling the sociomateriality of technological regimes in digital innovation. *ICIS: Thirtieth International Conference on Information Systems*, Phoenix, AZ.
Teece D (1986) Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy. *Research Policy* **15**, 285-305.
Trispas M (2009) Technology, identity, and inertia through the lens of "The digital photography company". *Organization Science* **20**(2), 441-460.
Tushman ML (1977) Special boundary roles in the innovation process. *Administrative Science Quarterly* **22**, 587-605.
Ulrich K (1995) The role of product architecture in the manufacturing firm. *Research Policy* **24**(3), 419-440.
Van de Ven A (2007) *Engaged scholarship: A guide for organizational and social research*. Oxford University Press.
Yang C and Jiang S (2006) Strategies for technology platforms. *Research Technology Management* **49**(3), 48-57.
Yoo Y (2010) Computing in everyday life: A call for research on experiential computing. *MIS Quarterly*, **34**(2), 213-231.
Yoo Y, Lyytinen K and Boland R (2008) Distributed innovation in classes of networks. *Proceedings of the 41st Hawaii International Conference on System Sciences,* Hawaii.

芽|Sprouts

Working Papers on Information Systems | ISSN 1535-6078

Office:
Sprouts
University of Amsterdam
Roetersstraat 11, Room E 2.74
1018 WB Amsterdam, Netherlands
Email: admin@sprouts.aisnet.org

芽|Sprouts