Digital Product Innovation:

Building Generative Capability through Architectural Frames

Fredrik Svahn

Department of Informatics
Doctoral Dissertation
Umeå University
Umeå 2012

Department of Informatics
Umeå University
SE-90187 Umeå
fredrik.svahn@viktoria.se

ISBN: 978-91-7459-458-4
ISSN: 1401-4572, RR-12.02
Printed by: Print & Media, Umeå University
Umeå, Sweden 2012

Abstract

Over the last decades we have witnessed a profound digitalization of tangible products. While this shift offers great opportunities, it also exposes product developing industries to significant challenges. In these industries organizations, markets, and technologies are tuned for mass production, providing competitive advantage through scale economics. Typically, firms exercise modular strategies to deliver such scale benefits. Rooted in Herbert Simon's notion of near decomposability, modular product architectures allow for production assets, such as tools, processes, and plants, to be effectively reused across product variants and over generations of designs. However, they come at a price; modularity requires overall design specifications to be frozen well before production. In practice, this tends to inscribe functional purpose in the structures of the system, effectively preventing firms from taking advantage of the speed by which digitized products can be developed and modified.

The main objective of this thesis is to investigate and explain how product developing organizations adapt architectural thinking to balance the proven benefits of modularity and the emerging opportunities provided by digital technology. In doing so, it introduces a complementary architectural frame, grounded in Christopher Alexander's seminal work on patterns. This frame associates the concept of architecture with generativity and reuse of ideas, rather than scale economics and reuse of physical assets.

Sensitizing the theoretical framework through a longitudinal case study of digital product innovation this thesis derives several implications for theory and practice. Across four embedded cases in the automotive industry it demonstrates that generative capability follows from a shared organizational view on products as enablers and catalyzers of new, yet unknown functionality. Such an emergence-centric view requires product developing firms to rethink existing governance models. Rather than exercising control through specific functionality, inscribed in modular product structures, it offers the benefit of influencing innovation through general functional patterns, serving as raw material in distributed and largely uncoordinated innovation processes. This shift in focus, from specific functionality to general functional patterns, enables a new strategic asset for product developing firms. It opens up for proactive rather than reactive strategies, where the architecture makes an instrument to cultivate new ideas and business opportunities, rather than a tool for cost savings.

Keywords: digital innovation, product innovation, generativity, modularity, patterns, architecture, architectural frames.

Acknowledgments

Writing a dissertation is an individual journey full of unexpected surprises and deep challenges, but also of joy and great satisfaction. Perhaps my journey has been particularly blurred in its contours for the simple reasons that I have never had a PhD position. While some parts of a PhD process fit well with an institute research practice, other parts tend to become spare-time activities. Finding a working combination between applied research and thesis writing takes some unconventional thinking, pragmatism, and a lot of inspiration. I am deeply indebted to a whole range of people for supporting me in this process. Without your constructive feedback, generous support, and frequent encouragements my academic career would have been short.

First, I would like to express sincere gratitude to my colleague, research manager, and supervisor Ola Henfridsson. You have offered me a tailored path for uncovering some of the intellectual craftsmanship of social science research. Acting as a mentor, rather than a formal supervisor you have encouraged me to draw on my own unique experiences and interests instead of following some predefined trail. In retrospect I can see that this attitude is far from typical in an academic context. By listening you have always shown interest in my ideas. By challenging them you have shown me respect.

I would also like to thank my co-supervisors, Jonny Holmström, Lars Mathiassen, and Youngjin Yoo. You have contributed in different, yet equally important ways. In particular, you have offered me different perspectives on my own work, but also on the scientific discipline we all belong to. Further, I would like to express my appreciation to Rikard Lindgren. Although we have not explicitly worked together you have often taken your time to read and listen. Thank you for constructive feedback and – not least – encouragement and spirit. I am also forever indebted to my great colleagues at Viktoria Institute, from researchers to administration and management. However, I would like to express particular gratitude to Lisen, Lena, Magnus, Taline, and Mats for being so patient with my ideas. It is a pleasure going to work with such colleagues. I would also like to give attention to my new colleagues in the IT Management group at Umeå University. Thank you for your open, welcoming attitude and the time you have invested in my work. Finally, I would like to thank my old friend and brother-in-law Per Wikman Svahn for intense and rewarding late night discussions, always providing new perspectives.

Taking a wider perspective on this thesis I am forever indebted to a few persons. Hans Bjersing, without your encouragement and funding from Guide Konsult I would not have approached Viktoria Institute in the first

place. Per-Åke Olsson, your support allowed me to back away from some duties to focus on thesis writing. Kicki and Jan, your generosity in supporting with child care made everyday family life a lot easier. However, I would also like to express deep respect and great appreciation to my parents, Lasse and Hjördis. You gave me strength to follow uncomfortable paths.

Finally, this would not have been possible without support and encouragement from my beloved wife Annika. You have been by my side from the very beginning of this process. You have listened, you have celebrated with me, and you have encouraged me in times of doubt. I could not have done this without you. Last of all, I would like to thank my wonderful children Elsa and Oscar. Your curiosity, unlimited energy, and unconditional love cannot be underestimated.

Frillesås, June 2012

Fredrik

Table of Contents

1 Introduction

Thursday January 21, 2010 Nokia launched free walk and drive navigation as a part of their new version of Ovi Maps[1]. This aggressive move towards a mobile service market was enabled by Nokia's acquisition of the leading digital map provider Navteq in July 2008. Consulting traditional business logic, it simply does not make sense for a product developing firm to give away an $8.1 billion investment, without any explicit revenue generation. However, a press release from February 3 gives an indication of the rationale behind Nokia's strategy; in about one week the new Ovi Maps service was downloaded by 1.4 million users. Anssi Vanjoki, executive vice president at Nokia explains why these new users are so important to Nokia[2]:

> *This is great news for our 3rd party application developers. Within a matter of days there is an installed base of more than 1 million active users all potentially hungry for new and innovative location-aware apps. […] For the operators too there is a growing opportunity to sell more data-plans and a complete navigation package to existing and new customers.*

As illustrated in Businessweek Online, August 2006, Nokia's previous success in the mobile handset business was tightly connected to its

[1] Ovi Maps (now Nokia Maps) is a free mapping service provided by Nokia for its mobile phones and smartphone multimedia devices.

[2] http://press.nokia.com

manufacturing machinery (Reinhardt 2006). A key element of this machinery was the strong commitment to modular product architectures. This architectural strategy allowed Nokia to offer product variation, while transferring marginal cost to commoditized components. It served Nokia well over several years, providing significant scale advantages and the highest operating margins in the business. Against this backdrop Vanjoki's statement uncovers a new Nokian perspective on innovation and technological progression. First, it suggests that Nokia is adopting a new architectural philosophy with a clear distinction between platform and services. Although the release of Ovi Maps attracted enormous attention at the level of users thanks to the new functionality, the quote indicates that Nokia looks upon navigation primarily as an important part of a platform. Second, the statement denotes a new organizing logic, where functionality is expected to emerge from the more or less independent work of 3rd party application developers. This opens up for unconstrained creativity and alternative modes of value generation. Third, it recognizes a new market dynamic, breaking with traditional ways to do business. With end-user value primarily supplied by 3rd party developers, Nokia customers will seek a prospering service ecosystem, feeding multiplicity and heterogeneity. Therefore, the decision to invest in a handset will correlate to the number of developers engaged in the ecosystem. At the same time, developers seek an outlet for their applications, making the number of potential customers a key aspect in the decision to engage in the ecosystem. Obviously, the installed base of mobile handset is an entry key for Nokia in their struggle to set up and capitalize on this type of two-sided market.

Over the last decades we have witnessed a profound digitalization of tangible products (Yoo et al. 2010b). Nokia's Ovi Maps initiative is nothing but a specific example of a general trend in product developing industries. Turning to the automotive industry, as an example, a modern car embeds numerous onboard computers, more than 10 million lines of code, and is increasingly connected to mobile devices and telematics services (Barabba et al. 2002; Broy et al. 2007; Henfridsson and Lindgren 2005). It is argued that as much as 80% of all car innovations can be traced to digital technology (Leen and Heffernan 2002). Given this wide adoption of software and digital technology, product developing organizations are triggered to rethink established models of innovation. Rather than centering on the corporate R&D department, they acknowledge that innovation is an increasingly distributed activity (Yoo et al. 2008), taking place in networks (Boland et al. 2007; Powell 1990; Tuomi 2002) or ecosystems (Basole 2009; Rosemann et al. 2011; Selander et al. 2010; Selander et al. in review) rather than within hierarchies. In turn, this calls for new forms of governance (Demil and Lecocq 2006; Ghazawneh and Henfridsson 2011; Ghazawneh and

Henfridsson forthcoming; Markus 2007), alternative business models (Economides and Katsamakas 2006; Eisenmann et al. 2006), and generative technologies (Remneland et al. 2011; Zittrain 2006), encouraging spontaneous and uncoordinated innovation.

This thesis is rooted in the observation that product developing industries need to *combine* different innovation regimes to tackle digitalization (Godoe 2000; Svahn and Henfridsson 2012; Svahn et al. 2009). On the one hand, cars, heat pumps, and washing machines will remain physical products, delivering tangible value – transportation, heating, and cleaning. On the other hand, this value is increasingly enabled by software and digital technology, calling for a new perspective on innovation.

In addressing the challenge of combining innovation regimes, this thesis draws on an assessment of innovation literature (2.1). This is a large and rather fuzzy body of literature, ranging from economics to entrepreneurship, via technology management and organizational science. However, for this particular purpose I have concentrated my efforts on two, relatively homogeneous streams; product innovation and digital innovation. Product innovation is an established branch that we can trace at least back to the work of Schumpeter (cf. Schumpeter and Opie 1934). Although being a wide body of literature, it is relatively sharp in its contours. Researchers essentially refer to the same knowledge base when using the notion of product innovation. Turning to software there is not yet such a clear body of innovation literature. Software-enabled ERP[3] systems have transformed corporate governance, embedded software has revolutionized electronics, and open source software has forever changed our view on the incentives involved in innovation. Not surprisingly, the wide application of software has translated into several fields of research. At the same time, researchers generally recognize that software separates the meaning and functional behavior of a product from the physical product in itself. As a consequence, there is reasonable consensus that innovation processes, centered on software, follow a different logic. I present this logic in under the notion of digital innovation. The review suggests that these two streams of research approach innovation with inherently different perspectives (Table 1):

Product innovation is essentially firm-centric. Innovation is shaped in vertical industries where these firms develop new physical artifacts in a teleological and reductionist manner. Applying waterfall models, value is created in linear processes, governed by behavioral control mechanisms. Markets are characterized by competition over price since significant

[3] Enterprise Resource Planning

marginal cost pushes for commoditization and economies of scale, eventually making an influential force for dominant designs and homogeneity on markets. When architecting tangible products, product developing firms center on the physical structure. Modular designs allow for efficient reuse of assets, such as production tools and machineries. A strictly hierarchical decomposition of products preserves the overall functional setup, while allowing for change at the level of details.

Digital innovation is largely network-centric. Innovation is shaped by horizontal industrial structures, where essentially independent actors together shape value in a non-linear and emergence-oriented manner. Without centralized authority these processes are governed by output control, rather than direct influence over processes. In practice, that makes an evolutionary approach to innovation, providing variation and multiplicity to markets that constitute an ultimate selection mechanism. In a digital innovation regime, such markets normally take the form of two-sided markets, centered on a shared platform. Rather than competition over price, these markets are characterized by competition over attention. The architectures of digital products are normally centered on the functional structure of solutions and problems. An overall objective is to create generative designs, encouraging reuse of general functional patterns in ever new specific applications, not know at the time of platform design.

Product innovation and digital innovation make distinct regimes in the sense that organizing logic, market dynamics, and architectural design principles make sense together, as a whole. They manifest two consistent logics for innovation, where a range of different forces are intertwined and largely inseparable. To some extent it appears as if product innovation and digital innovation seem incompatible; digital innovation is powered by multiplicity and choice while product innovation is associated with commoditization and dominant designs. Product innovation is teleological in the sense that value is created linearly from an idea to a diffused product. Further, a product innovation regime rewards the stability coming out of efficient reuse of components, tools, and machineries, while digital innovation is centered on the generative reuse of functional patterns to encourage novelty and change. Contrasting product innovation literature and digital innovation literature certainly reveals a range of more or less fundamental tensions between the two regimes, but the literature is largely silent on theoretical as well as empirical evidence on how to combine them.

Organizing logic, market structure, and architecture of products are certainly deeply intertwined and cannot be studied in isolation. At the same time, it is not a bold statement to say that the radical changes of today's society are pushed by technological change. Few people would argue that publishing is

reforming in response to new organizational needs or customers desire for new business models. There is overwhelming evidence suggesting that publishing, photo, consumer electronics, and many other industries primarily change as a consequence of internet, tablets, electronic payment, etc.

This thesis addresses the gap in the literature by investigating how different ways to conceptualize products resonate with innovation processes. The introduction of digital technology in product developing organizations tends to be dialectical (Henfridsson et al. 2009b). It is dialectical in the sense that new paths are created in a tension between a familiar past and an uncertain future. Therefore, this research is centered on how architecture and architectural thinking defines change across generations of products. Being a link between historical achievements and future potentialities, the architecture is an instrument for path creation as well as a shackle of path dependency. Whether product developing firms will be able to transform innovation practices and leverage the opportunities of IT relies on their capability to internalize the architectural thinking of a digital innovation regime and combine it with the architectural perspective of product innovation. Therefore, the research question of this thesis is: *how do product developing firms architect digitized products to leverage the generative capability of IT*?

As a lens to be applied in my study of digital innovation in product developing organizations I have composed a theoretical framework centered on the concept of *architectural frames* (see also Henfridsson et al. in review). On a general level, architectural design is viewed as a way to manage complexity. Management of complexity is a critical activity in that it has implications on product change, product variety, standardization, and product performance (Alexander 1999; Simon 1962), but it also allows for division of labor, concurrent design, and accommodation of uncertainty (Baldwin and Clark 2000; Kirsch 1996; Ouchi 1979). As emphasized by Herbert Simon (1996, p. 215), complexity is not an invariant aspect of technology. Rather, "how complex or simple a structure is depends critically upon the way in which we describe it". Anchored in this perspective, I see architectural frames as schemas for thinking about and representing a complex product's architecture. Hence, I conceive of architectural frames as cognitive processes crystallized as a particular way of managing complexity in designing complex products. An architectural frame guides how complex forms can arise from simpler ones. Using the language of Brian Arthur (2009), it makes a distinct template for how technology is bootstrapped upwards, from the few to the many and from the simple to the complex.

Product innovation exercises modularity to handle complexity. This architectural frame relies on what Simon referred to as near decomposability (Simon 1962; Simon 1996; Simon 2002). A nearly decomposable system is a hierarchy of parts, where any level of analysis reveal a system of parts where each of those components is, in turn, a system of finer parts. Limited coupling between parts result in stable subassemblies, essentially defined by their respective interfaces. Such a stable subassembly makes a distinct building block, which can be used without paying much attention to its inner structure or legacy. As long as the interfaces are preserved an automaker can reuse e.g. a specific navigation component in a range of different car models. However, changing the interface of the navigation system is a major decision, with system level implications on every product using the component. Therefore, the original decomposition of a modular system is critical. In fact, this decomposition is "the scheme by which the function of a product is allocated to physical components" (Ulrich 1995, p. 419).

In practice, the hierarchical structure of a modular system emerges as designers recurrently practice decomposition and aggregation in the design of products. With the decomposition of products into parts designers seek to establish and preserve stable, loosely coupled subassemblies. Such stable subassemblies hide complexity and delivers functionality through well defined interfaces. In the aggregation of parts into products designers take the opposite perspective and seek different configurations of parts to create product variability. From now I will label this architectural frame *hierarchy-of-parts*.

We have seen that the increasing digitization of products call for innovation practices where functionality is the output from a generative platform, rather than the input to the decomposition of a modular system. This development put into question the dominant position of modularity and its application of Herbert Simon's notion of near decomposability. In a modular system coupling is low since every element has a well defined functional purpose, complying with the overall system design. In contrast, a generative system encourages coupling in that general functional building blocks are designed to be easily reused in large amounts of specific, yet unknown applications. Drawing on Christopher Alexander's seminal work on patterns (Alexander 1964; Alexander 1979; Alexander 1999; Alexander 2002; Alexander et al. 1977) I propose a complementary architectural frame that is resilient to the challenges of increasingly digitized products.

A cornerstone in Alexander's writing as an architect is that sound and harmonic environments show significant multiplicity and variation, not the uniformity associated with standardized components and dominant designs. According to Alexander, "the 'elements', which seem like elementary

building blocks, keep varying, and are different every time that they occur" (Alexander 1979, p. 84). On a general level, he offers a rather simple explanation; good designs – that is designs which we perceive as "whole" and "living" – are adapted to their contexts. People simply have to "shape their surroundings for themselves" (Alexander 1979, p. 74). Therefore, we will not find a deeper structure if we focus on the decomposition of physical artifacts. Instead, we have to direct our attention to the processes *generating* these artifacts. That is the common sense, shared legacy, and formal knowledge that directs us in designing and producing things. We need to look for standardized processes, not standardized components.

Patterns are the basic elements in Alexander's perspective on architecture. A pattern expresses "a relation between a certain context, a problem, and a solution" (Alexander 1979, p. 247). Thereby, it can be viewed as an instruction, which shows how a particular solution to a problem can be reused, over and over again, wherever the particular context is relevant. We find structure in the fact that individual patterns are not isolated. Every pattern "depends both on the smaller patterns it contains, and on the larger patterns within which it is contained" (Alexander 1979, p. 312). Together, this network of interconnected patterns forms what Alexander refers to as a "pattern language".

In practice, these networks of patterns emerge as designers recurrently practices generalization and specialization in the design of products. Generalization is a way to manage complexity, where designers seek increasingly generic representations of the functionality associated with an artifact. These representations, or patterns, are distinct solutions for particular problems, defined by a given context. In the process of generalization patterns are repeatedly disassembled into increasingly generic elements, relating to each other through inheritance. Exercising specialization designers take a bottoms-up perspective, seeking to extend the application of generic patterns by reusing them for new purposes. From now I will label this architectural frame *network-of-patterns*.

Seeking a better understanding of how product developing firms architect digitized products to leverage the generative capability of IT I have applied the theoretical framework to digital product innovation practices at the automaker CarCorp. The study is longitudinal in its character and spans a period of approximately one decade. It contains four different, yet temporally interlinked, embedded cases. Briefly, the objective of the study is (1) to understand how architectural thinking in product innovation and digital innovation can be conceptualized as architectural frames, how these frames (2) relate to each other, how they (3) can be combined, and, finally, how they (4) together influence innovation practices.

Together, the four embedded cases demonstrate how architectural thinking shifted in the automotive industry as a response to digital technology. The first case describes the adoption and appropriation of media-oriented systems transport (MOST), a radically new service-oriented and event-driven architectural solution to infotainment[4] system design. With MOST the network-of-patterns frame was officially recognized, for the first time. With the new frame automakers were offered the opportunity to exercise generalization, resulting in a harmonized infotainment solution, sharing many basic resources and functions. The second case is centered on the emergence of a new architectural concept for commoditization of instrument panel clusters - SoftCluster. The story explicates how CarCorp combined architectural frames to leverage differentiation in software. The SoftCluster platform was generative in the sense that it was not designed up-front for a particular functional setup. Instead, it allowed for recurring specialization of instrument cluster functionality across the full range of brands and models in the portfolio of GlobalCarCorp, CarCorp's owner. Third, I follow a range of experimental setups for leveraging nomadic device integration in cars. This story is primarily colored by the architectural challenges of introducing leeway between automotive lifecycles and the faster rhythm of consumer electronics. Finally, I study the emergence and consolidation of new architectural solutions for open, in-car platforms in the automotive industry. In this phase, network-of-patterns thinking had taken hold beyond R&D. To reinforce the generativity of the new Android-based infotainment platform CarCorp reconsidered their relationship to suppliers, their business models, and the deep-rooted automotive perspective on market logic.

On the basis of this longitudinal case study of digital innovation practices in the automotive industry I demonstrate the theoretical framework and leverage differences between architectural frames. Largely, generative capability is about encouraging creativity outside established processes, organizations, and conventions. However, it is also about accessing and, eventually, profiting from such creativity. The longitudinal case study of CarCorp shows that as the network-of-patterns frame is assimilated, people reassess their perspective on products. Rather than viewing their products as carriers of pre-fabricated functionality, they increasingly see them as enablers and catalyzers of new, yet unknown functionality. Further, synthesizing the four embedded cases I also argue that unless such a view informs all the different actions and decisions across the organization a

[4] Infotainment refers to media providing a combination of information and entertainment. In the automotive industry it includes navigation, telematics, rear-seat entertainment, and similar systems.

product developing firm has little chance to build permanent generative capability. Finally, this thesis suggests that a product developing firm cannot build generative capability unless adopting a distinctly different governance model. Generative capability relies on unconstrained freedom to create new specific functions. Such freedom clashes hard into established modular governance models, where decomposition of products is guided by specific functionality. Unless product developing firms find ways to govern innovation through general patterns rather than specific they have little chance to build permanent generative capability.

The thesis also derives theoretical and practical implications for technology and innovation management. First, it is argued that the assimilation of network-of-patterns thinking redefines the role of architecture; rather than being a tool for cost savings, it turns into an instrument for cultivation of new ideas, eventually building new business opportunities. Therefore, the network-of-patterns frame opens up for proactive rather than reactive architectural strategies. Second, the network-of-patterns frame turns the spotlight from specific functionality to general functional patterns. Sensors, actuators, data, and other low-level elements of a system are no longer details that should be embedded in components and hidden to reduce complexity. Instead, they have turned into valuable resources that should be exposed to developers. Therefore, it can be argued that the network-of-patterns frame enables a new type of strategic asset. Finally, network-of-patterns thinking brings a new perspective on how products change. It furthers a view where the meaning of a particular product is not up-front defined, but can evolve over time. Specialization can occur independently of hardware design, generating a constant flow of new functions. This disconnects software-based functionality from hardware, not just in a technical sense, but from an innovation perspective. As an important consequence, the network-of-patterns frame allows for appropriation of value across the product life cycle.

This work is concluded with a few notes on challenges for future research. In particular, it identifies four distinct challenges for product developing organizations when architecting hardware. Installed base is critical to generative environments. To avoid breaking installed base apart, product developing firms have to be careful when designing variants. They have to ask; how can differentiation be achieved without exercising aggregation, potentially generating incompatible hardware configurations? Further, an urgent need to assist the reuse of functionality calls for a range of actions. Producibility entails decomposition into largely independent components. However, in modular practices low coupling is achieved by decomposing the system from the perspective of functionality. This effectively inscribes

functionality in the physical structure of the system, which prevents reuse for new purposes. This raises several questions for future research. How can physical products be decomposed for low coupling without inscribing functionality in the structure? In this vein, it is relevant to ask; is it possible to assist reuse by increasing hierarchic span? Such increased span would result in less deep hierarchies, reducing recursive, nested encapsulation and potentially expose more functionality to developers. Finally, product developing firms need to question how they design interfaces. Applying modularity interfaces are hard to reassess in retrospect. Therefore, they make an important mechanism for product developing firms to exercise control over innovation processes; the more specific interface, the more detailed control. This makes a stark contrast to the principles of generalization, exercising control from the perspective of general patterns, rather than specific. Therefore, an important question for future research is; what does best practice for general interface design look like?

The rest of this thesis is organized as follows. Section two explicates an assessment of the innovation literature, drawing on the concept of innovation regimes. From this assessment I identify a research gap which centers on the lack of contributions elaborating how digital technology influences innovation in product development. Next, I derive a theoretical framework to understanding digital product innovation. In particular, this framework is designed to explicate architectural thinking in traditional product developing settings increasingly exposed to digital technology. Following a description of my methods, I then provide an empirical analysis of an automaker's attempts to architect infotainment products in response to technological change. Finally, I discuss how this research extends current views on product architecture and contributes to the emerging literature on innovation.

2 Related Work and Conceptual Foundation

Innovation is an old concept with many facets. As revealed by its Latin roots – *nova* – innovation is centered on the notion of newness or novelty (see e.g. Luecke et al. 2003). Although a significant body of literature approaches innovation primarily as "new products and processes" (Tushman and Moore 1982), the distinction between invention and innovation (Schumpeter and Opie 1934) suggests that it is something more than an act of intellectual creativity generating new ideas and concepts. Innovation is also about the "the production or emergence of a new idea" (Gupta et al. 2007, p. 886). Essentially, this perspective recognizes the demanding journey towards practical application of an idea as a central part of innovation. At the heart of such reasoning is that a novel idea does not make an innovation until it is changing practice. Therefore, in addition to ideas and their tangible manifestations in products, processes, etc, innovation is about relevance and value. Innovation is "the embodiment, combination, or synthesis of knowledge in original, relevant, valued new products, processes, or services" (Luecke et al. 2003, p. 2).

Synthesizing this brief walkthrough of definitions a few aspects stand out as fundamental to innovation. First, innovation is obviously about ideas. Any innovation process holds an element of creativity. However, most people would argue that it is not primarily an artistic creativity, where the output emerges in an open-ended manner. Instead, innovation is about the capability to make abstractions of our everyday life and elaborate them for

specific purposes. Second, innovation is about practice. It is about the application of abstract concepts in real life. Therefore, innovation is a phenomenon highly intertwined with the concept of practice. Given a definition where practice is "a process by which we can experience the world and engage with it as meaningful" (Wenger 1999, p. 51), it is also about meanings. New meanings.

However, understanding the process of innovation, as we see it in contemporary industrial practice and research, is largely about understanding technology. Ideas are manifested as functions or services, mediated and enabled by technology. New practices emerge through the adoption and use of new technology. Since new technologies in one way or the other is birthed from previous ones, innovation is intertwined with the evolution of technology over time, in research often referred to as technological change.

Clearly, technological change emanates from the interplay between people and technology. New technology is often the source of inspiration in innovation, subjecting new opportunities (Svahn 2009; Svahn et al. 2009). At the same time technology may make a strong opposing force as innovators seek new solutions. With technology making the backbone of many solid structures in society, users and designers tend to inscribe meaning in it. Established practices simply make efficient barriers for humans to rethink technology and give it new meanings. As noted by Nelson and Winter (1982, p. 258), some directions of progression "seem much more compelling of attention than others. Particularly in industries where technological advance is very rapid, advance seems to follow advance in a way that appears almost inevitable". Such path dependency (Arthur 1989; David 1985) makes technology evolve according to an inherent logic that we cannot ignore.

At the same time existing research underlines that new technology may disrupt this logic. Ultimately, it may inaugurate the emergence of a new technological paradigm (Godoe 2000), seeding new paths of innovation. Following the literature, such new paths are variously termed as "technological regimes" (Nelson and Winter 1982), "technological trajectories" (Dosi 1982), "pattern of evolution" (Hughes et al. 1987), "technological guideposts and avenues" (Sahal 1985), and "basic designs" (Rosenberg 1982). A new technological paradigm inevitably brings a shift in "principles, norms and ideology, rules and decision-making procedures", recognized by Godoe (2000, p. 1034) as the transition to a new *innovation regime*. Such regimes make a new foundation for actors to form "expectations and actions in terms of the future development of a technology".

This thesis is rooted in the observation that traditional product developing industries need to rethink innovation to tackle digitalization. Cars, heat pumps, and washing machines are physical products, delivering tangible value – transportation, heating, and cleaning. Over the years practitioners and technology have developed a relatively stable interplay – an innovation regime – embracing established business models, organizational role models, architectural standard solutions, and best practices (Svahn et al. 2009). However, digital technology is inherently different from tangible products (cf. Yoo 2010). As transportation, heating, and cleaning is increasingly enabled by software and digital technology the established product innovation regime will be disrupted. New perspectives on materiality will subject new opportunities to designers. At the same time, designers implementing new technology will be hampered by the resistance of traditional technology.

A new innovation regime will emerge as tangible products become increasingly digitalized. Such a regime unfolds from a different set of rules or fundamental mechanisms defining the elements and friction constituting the interplay between technology and people.

In what follows, I distinguish and review two distinct streams of innovation literature: product innovation and digital innovation. They represent innovation regimes in the way they manifest a particular view on innovation. At the heart of product innovation we find the tangible product, while digital innovation is centered on information technology, software and the IT artifact. As we shall see, these two streams deliver inherently different theoretical explanations to the concept of innovation, underline different challenges, and offer different architectural solutions to these challenges.

2.1 Assessment of Innovation Regimes

Product innovation literature is a well established research branch, with a track record of at least one century. We find seminal contributions in a variety of outlets, ranging from ASQ and Research Policy to AMR and Management Science. Product innovation is studied by strategy theorists (Porter 1985; Teece et al. 1997), economists (Nelson and Winter 1982; Schumpeter and Opie 1934), organization theorists (Cohen and Levinthal 1990; Tushman and Anderson 1986), and technology management researchers (Baldwin and Clark 2000; Van de Ven 1986).

Digital innovation is perhaps a less recognized label. Yet, this dynamic and slightly fragmented stream of research is gaining momentum across disciplines. While engineering-oriented outlets, such as IEEE journals, often have tried to translate product innovation to software settings (Boehm 1976; Parnas 1972; Royce 1970), we increasingly see recognition of the inherently

different properties of IT (Alexander 1999; Allen 2006; Jackson 2000). Similarly, outlets such as Organization Science show increasing interest in digital innovation, illustrated by an upcoming special issue labeled "Organizing for Innovation in the Digitized World" (Yoo et al. 2009). As digitalization is playing out at different levels of society we also see a translation of the information systems (IS) discipline. The traditional management perspective is gradually complemented with a growing interest in digital innovation. This is reflected in a range of special issues in premier journals. Fall 2010 Information Systems Research (ISR) published a special issue on digital systems and competition (Ferrier et al. 2007). The interest in questions relating to digital innovation was confirmed by ISR's 20th anniversary special issue, dedicated to "forward-looking commentaries on important topics and phenomena that are likely to frame a high-impact research agenda in the next few years" (Sambamurthy 2010). In this issue, a whole range out of totally 15 research commentaries are explicitly framed in this direction (Brynjolfsson et al. 2010; El Sawy et al. 2010; Tilson et al. 2010; Tiwana et al. 2010; Yoo et al. 2010b). Let us also note two similar calls for papers in MISQ on service innovation in the digital age (Barrett et al. 2010) and digital business strategy (Bharadwaj et al. 2010). We also see an increasing number of conference tracks and dedicated workshops (cf. Yoo et al. 2010c) centered on the rapid digitalization of society.

I will now turn to product innovation and digital innovation literature in an attempt to incarnate the two innovation regimes and illustrate essential differences between them. In particular, I will center my assessment on three key dimensions. First, I will discuss the basic organizational arrangements characterizing each regime. Next, I will elaborate the market dynamics empowering innovation in product innovation and digital innovation. Finally, I discuss similarities and differences in architectural perspectives, reflecting how the two regimes approach technological progression over time. While these three dimensions all together hopefully give life to the two innovation regimes portrayed in this thesis, they also make a tool for me rationalizing a focus on architecture. Together, they render a story that put organizing logic and new market dynamics in causal connection with architecture and architectural thinking.

2.1.1 Organizing Logic

IT and digital products largely seem to be incompatible with the firm-centric organizational structures developed over a hundred years of manufacturing. As one out of many examples, media industries are in the midst of a painful reorientation towards more network-centric structures. In this context, publishers and retailers desperately seek new organizational configurations as the distinction between producers and consumers is increasingly blurred

and fuzzy in the digital marketplace (Baudrillard 1998; Tapscott and Williams 2006). Similarly, the standardization and diffusion of the third generation (3G) mobile infrastructures have redefined innovation in the telecommunication area (Yoo et al. 2005). Mediating interests and motivations among a wide range of heterogeneous actors these standards have turned innovation of broadband mobile services into a collective achievement. Essentially, this put the once dominant operators in a new situation as the content and meaning enabled by their infrastructure is increasingly defined by networks out of their immediate control.

A product innovation regime seems to cultivate organizational configurations that are distinctly different from the structures and logic emerging in a digital innovation regime. Therefore, it is far from sure that established organizing logic will allow firms to "manage the imperatives of the business and technological environments in the digital economy" (Sambamurthy and Zmud 2000, p. 106). Development practices seem to be different as innovation essentially is an in-house activity, while IT progression is distributed. Governance is different as the capability to control innovation processes is inscribed in organization structure, while IT relies on mutual benefit and symbiotic relationships to prosper. Finally, industry structure is different since product innovation seems to develop vertical organizations, while firms in an IT context tend to focus their competences on particular layers in a value chain, eventually forming horizontal industries. In an attempt to identify and uncover the basic arguments behind these differences I now engage in a review of the organizing logic linked to a product innovation regime and a digital innovation regime respectively. Following Sambamurthy and Zmud (2000, p. 107) I refer to organizing logic as the "managerial rationale for designing and evolving specific organizational arrangements in response to an enterprise's environmental and strategic imperatives".

2.1.1.1 Firm-Centricity and the Exercise of Formal Control

Although the product innovation literature may be sliced in different directions and framed for different purposes two dimensions seem to be part of any perspective. These two dimensions are critical drivers that cannot be ignored when seeking explanations to technological progression in product developing settings. On the one hand, an organization has to master the transformation of captured knowledge into new products and diffuse these products to remote practices, eventually increase market shares and secure profit. Such capability calls for an incremental perspective on technological progression, where new solutions are combinations of existing. On the other hand, product innovation research recognizes that technological progression may be born out of radical change processes, breaking with an established

paradigm. To cope with such progression an organization has to be able to internalize foreign knowledge and technology for in-house innovation.

In a product innovation regime the organization is the epicenter of innovation activity and the natural container of innovation capability. A significant part of the research is focusing on challenges facing incumbent firms (Abernathy and Utterback 1978; Henderson and Clark 1990; Hill and Rothaermel 2003; Tushman and Anderson 1986; Utterback and O'Neill 1994), manufacturing physical products of significant complexity. An incumbent firm is already established in a market, occupying a central position. The goods produced by different actors are homogenous, leaving relatively low price differentiation. As markets are occupied with such dominant designs (Abernathy and Utterback 1978; Rosenberg 1982; Sahal 1985; Teece 1986; Utterback and Abernathy 1975) it makes perfect sense to view innovation as the fruit of incremental progression within organizations. In settings where material supplies are unreliable, production costly, and knowledge a scarcity, such incremental change is most efficiently managed through hierarchical organization structures (Clark 1985; Williamson 1973), hosting the development of modular products (Baldwin and Clark 2000; Sanchez and Mahoney 1996; Simon 1962) under strictly linear development processes (Godin 2006; Porter 1985; Takeuchi and Nonaka 1986). This translates to vertically oriented industries (Chandler 1977) where competitive advantage derives from an organization's capability to enforce absolute control over its entire value chain. Following transaction cost theory (Coase 1937; Williamson 1971) the vertical integration we see in product developing organizations illustrates that it is cheaper to administer incremental innovation processes internally than outsource it to a market (Carlton 1979; Klein et al. 1978). As a final remark, it is worth noting that incremental innovation does not condition new ideas to derive from internal processes. However, it prescribes a distinct way for organizations to absorb new ideas; they are mangled through existing practices to align with established knowledge, organizational structures, and products.

While incremental progression seems to be the dominant mode of innovation in product developing settings, the literature pays substantial attention to change processes where firms are forced to rethink established knowledge, routines, and organizational structures. It acknowledges that relatively stable periods of incremental innovation are recurrently interrupted by technological breakthroughs or radical innovations, overturning existing paradigms and, eventually, seeding new paths of incremental change. Such disruptions are potentially lethal to organizations tuned for incremental innovation. To maneuver in the uncertainty introduced by radically new technology organizations have to build

capability for creative destruction (Abernathy and Clark 1985; Schumpeter 1942). Essentially, such capability allows for the development of new knowledge, coming at the expense of existing explanations to everyday problems. Such destruction is particularly problematic when the innovation changes the architecture of a product, since architectural knowledge tend to be deeply embedded in organizational structures and information-processing procedures (Andersson et al. 2008; Henderson and Clark 1990). Cohen and Levinthal (1990) introduced the notion of absorptive capacity as a measure of an organization's ability to implement this transformation, where new external knowledge is used to restructure established, internal innovation processes.

Although research pays significant attention to various phenomena related to radical progression of technology, established firms tend to be organized for incremental innovation (e.g. Hill and Rothaermel 2003). Clearly, technological discontinuities (Tushman and Anderson 1986) are hard to predict and almost impossible to plan for. Although the literature is rich in explaining phenomena such as creative destruction and absorptive capacity, it is relatively weak in elaborating organizational implications. The theory on absorptive capacity (Cohen and Levinthal 1990) comes with a model on how to direct R&D expenditures, but without opposing traditional structures. Similarly, Henderson and Clark (1990) suggest that organizations have to set up communication channels, information filters, and problem-solving strategies to build architectural knowledge, but without questioning the basic organizational arrangements.

Hierarchical organizations, linear models of product development, and vertical industry structures simply seem to be the dominant organizing logic of a product innovation regime. One lens to understand this dominance is offered by control theory, in turn deriving from ideas in transaction cost economics. Incremental innovation practices are highly visible within firms and, thereby, provide significant "knowledge of the transformation process" (Ouchi 1979, p. 843). When organizations know in detail how behaviors and processes will transform inputs into outputs it is cheaper to apply formal behavioral control than informal outcome control. Therefore, incremental innovation tends to feed such formal control modes, centered on authority (Eisenhardt 1985; Kirsch 1996) and maintained by the various distinctive properties of a product innovation regime.

2.1.1.2 Network-Centricity and the Creation of Digital Options

A salient distinction between traditional product innovation literature and digital innovation is that the firm-centric view is largely shifted out. Technological progression is not seen as a phenomenon deriving from linear

development processes, hierarchical organizations, and vertical industry structures. Instead, digital innovation research underlines that digital technology destroys many barriers favoring incumbent innovation. Over time this cultivates boundary-spanning practices (Levina and Vaast 2005; Lindgren et al. 2008), involving an increasing variety of largely uncoordinated innovation sources (von Hippel 1988). As a result, innovation translates into a distributed activity (Yoo 2010; Yoo et al. 2008). Taking place in networks (Boland et al. 2007; Powell 1990; Tuomi 2002; Van de Ven 2005; von Hippel 2007) or ecosystems (Basole 2009; Selander et al. 2010; Selander et al. in review), rather than within hierarchies, such innovation feeds significant multiplicity in functions and services.

Distributed value creation, scattered across networks and ecosystems, "leads to the emergence of dynamic, non-linear patterns of digital innovation" (Yoo et al. 2010c, p. 3). Such non-linear innovation provides value, not by outperforming existing products, but through the establishment of genuinely new value networks (Christensen 1997; Åkesson 2009). Although Clayton Christensen's (1997) concept of disruptive innovation is not originally coined in response to digitalization, we can use it to understand how technology emerges in a digital innovation regime. When discussing disruptive technology he refers to the disruption of markets. Such disruptive technology takes root in simple applications at the bottom of a market and then relentlessly move 'up market', eventually displacing established competitors. At the heart of his concept we find the idea that remote sources of innovation, rooted at the bottom of a market, are able to break with established norms of how we appreciate value and create new meanings. As already discussed digital technology helps destroy many of the barriers that hold back disruptive technologies in a product innovation regime. It is simply significantly easier to reach an audience for a piece of new software than it is to create a market for a new generation of hard drives, which is a famous example from The Innovator's Dilemma (Christensen 1997). Obviously, this bottoms-up model of innovation makes a strong contrast to traditional, firm-centric innovation, where value is created linearly (Godin 2006; Porter 1985; Takeuchi and Nonaka 1986) as the product is refined from top to bottom using waterfall models (Boehm 1976; Royce 1970) in a strict design hierarchy.

Navigating in a distributed innovation environment, where ideas and knowledge derive from external sources at the bottom of a market, requires firms to organize for agility (Sambamurthy et al. 2003). Agility, referring to the ability to detect and seize market opportunities with speed and surprise, is by many considered to be an imperative for success in a digital innovation regime (Brown and Eisenhardt 1997; Christensen 1997). Referring to

absorptive capacity (Cohen and Levinthal 1990) or creative destruction (Schumpeter 1942) one can argue that the capability to identify and internalize new knowledge and technology is a key part of any innovation regime. However, in product innovation literature the argument for openness towards external environments is rooted in reinforcement of internal activity. Therefore, such openness is essentially inbound. Digital innovation literature increasingly distances itself from such unilateral action and emphasizes that it might be a better idea to share intellectual property than keeping it hidden from competitors. Recently, this thinking has been successfully framed through the concept of open innovation. The term was popularized and promoted by Henry Chesbrough, defining it as the "use of purposive inflows and outflows of knowledge to accelerate internal innovation, and expand the markets for external use of innovation respectively" (Chesbrough 2006, p. 1). Although relatively imprecise in its contours, open innovation is increasingly adopted by practitioners. At the same time, researchers try to carve out the distinctions of the concept to identify future research agendas (cf. Huizingh 2011; Lichtenthaler 2011; West and Gallagher 2006).

Taking an industry perspective rather than a firm perspective, a digital innovation regime seems to feed horizontally segmented industries. To survive in distributed innovation ecosystems, firms have to focus on building their distinctive competences, outsource the rest, and become nodes in value chain networks (Van de Ven 2005). As illustrated by the transformation of computer industry (Chandler 1997), accelerating pace of technological change and fierce competition forces product developing firms to focus on horizontal segments, rather than remaining vertical organizations.

A key consequence of a transition towards horizontally structured industries, networked collaboration forms, and largely non-linear, open innovation processes is that once effective governance mechanisms are increasingly useless. While product innovation cultivates detailed control over internal behaviors and processes (Ouchi 1979), a prospering digital innovation ecosystem seems to be characterized by extensive and unconstrained cross-fertilization, spanning firm boundaries. Essentially, such loosely coupled collaboration introduces uncertainty that prevents organizations from exercising formal control over the innovation process. As pointed out by Ouchi, it is pointless to enforce formal mechanisms to control the transformation of input to output in such environments:

> *Under conditions of ambiguity, of loose coupling, and of uncertainty, [behavior] measurement with reliability and with precision is not possible. A control system based on such measurements is likely to systematically reward a narrow*

> *range of maladaptive behavior, leading ultimately to organizational decline (Ouchi 1979, p. 845).*

With limited power to influence the details of innovation processes, firms operating in digital innovation regimes are directed to informal governance, controlling the output, rather than behavior. It is easy to argue that the inability to get involved increases the risk for a given firm. As expressed by Fichman (2004, p. 132) a digital innovation regime brings "uncertainty about expected payoffs [of engagement] and irreversibilities in the costs of implementation". Uncertainty derives e.g. from unpredictable evolution of a particular technology, potentially creating unwanted path dependencies. Irreversibility may arise from high learning and adaptation cost, as well as high switching cost, when phasing out a technology.

While recognizing that a given innovation process may hold significant risk for an organization, a digital innovation regime at the same time offers a powerful countermeasure. Open, distributed innovation "allows companies to scan a much wider range of the available technologies or new market developments" (Vanhaverbeke et al. 2008, p. 253), without mandatory commitments. Therefore, it makes a complementary mechanism that balances the risk of specific initiatives. A digital innovation regime gives a firm access to options that do not have to be exercised. Such rights, without obligations to take actions in the future are frequently discussed, across scientific disciplines, under the notion of real options. This concept extends from finance literature, where it is applied for decision-making processes under uncertainty (Dixit et al. 1994). However, in the 90th it emerged as a theoretical lens in strategic management (Amram et al. 1999), making a tool for firms to build managerial flexibility (Trigeorgis 1996).

As demonstrated by Vanhaverbeke et al. (2008), real option theory makes a excellent tool in understanding governance and organizing logic in digital innovation. Many researchers have stressed the need to rethink the trade-off between incentives and authority in governance (Demil and Lecocq 2006; Markus 2007; O'Mahony 2007; Shah 2006). This argument is rooted in an increasing awareness of coopetition, i.e. simultaneous competition and cooperation (Walley 2007). Progressive digital innovation is built around symbiotic relationships, formed to create mutual value for its members (Basole 2009). Instead of elaborating rather fuzzy tensions between informal, incentives-driven governance and formal, authority-based control, one can argue that governance in digital innovation is about the creation, maintenance, and, eventually, realization of options. Such options "create value by generating future decision rights and, in this way, providing strategic flexibility. This flexibility is more valuable the higher the level of uncertainty" (Vanhaverbeke et al. 2008, p.252). With this perspective on

governance in digital innovation, one can say that IT is a "digital options generator" (Sambamurthy et al. 2003).

Taking a step back, it makes sense to say that the multiplicity we see at digital markets is not primarily a measure of success, at least not in terms of profit, but rather an inherent property of digital innovation, needed for such distributed processes to work at all. While product innovation essentially seek formal control modes, centered on authority, hierarchical organizations and contractual agreements, the key to profitable digital innovation is found in the capability to domesticate the multiplicity of ecosystems and networks. To master such governance, firms have to organize for distributed rather than centralized knowledge bases, non-linear rather than linear value creation, and horizontal rather than vertical industries. Synthesizing the literature such organizing logic seem better tailored to informal control modes that ultimately aims for the creation of digital options.

2.1.2 Market Dynamics

The markets in contemporary western economies are flooded with products, not only in terms of volume but also by offering an almost indefinite range of options. However, zooming in on this reflection we can ask ourselves what constitutes these options. In what sense do markets offer consumers a range of alternatives? Tangible products are indeed offered in a wide range of forms, brands, and models. Still it can be argued that the technology is strikingly similar. The basic structure and functionality of a car, airplane, or heat pump is essentially the same, irrespective of brand and model. They all deliver transportation and heat, essentially using the same solutions.

In contrast, software-oriented markets seem to feed remarkable variety and multiplicity (Anderson 2006; Brynjolfsson et al. 2010; Brynjolfsson and Smith 2003), at least if we focus on functionality – the value delivered to customers. This is particularly salient for innovation ecosystems built around a shared platform, rather than an application area (Tiwana et al. 2010). The software offered at Apple's AppStore or Android Market share many fundamental properties, yet they serve an almost infinite range of different purposes. Product innovation seems to feed relatively few solutions, tailored and optimized for a particular purpose, while digital innovation seems to be an open-ended process, allowing for a wide range of alternative solutions to reach the market.

In order to shed some light on the differences in market offers and behavior I now engage in a review of the market dynamics linked to a product innovation regime and a digital innovation regime respectively. With the notion of market dynamics I generally refer to the mechanisms defining the forces of demand and supply at markets. While an economist is primarily

interested in resulting pricing signals, I focus my attention on how tensions are manifested and materialized. On a general level, the market dynamics explains why a producer is prevented from giving the consumer what it ultimately wants and vice versa. It defines an equilibrium at a market and the rules for mowing this balance point.

2.1.2.1 Competition over Price under Dominant Designs

Largely, product innovation research explains the relative uniformity at the level of markets through the concept of dominant design. At some point in the evolution of a technology the industry is moving from a system of "made-to-order" products to a standardized mass-market manufacturing system of a complex assembled product (Abernathy and Utterback 1978; Abernathy 1978). This turning point between flexible and specialized production marks the transition into a dominant design. The emergence of a dominant design is a subtle process which can be recognized in retrospect, but is almost impossible to appreciate in real time (Anderson and Tushman 1990). As reflected by Murmann and Frenken (2006), Abernathy (1978) identifies three distinct phases in the materialization of a dominant design. The first step is characterized by the introduction of a solution that has broader appeal in contrast to earlier product variants that focused on performance dimensions valued by only a small number of users. In the second phase attention is shifted away from performance and basic functionality towards the details of design as increasing market shares impose imitative design reactions among players competing at the same market. Finally, the dominant design is established as imitative behavior eventually enforces standardization throughout the industry and almost complete diffusion across the market.

There are several perspectives in the literature discussing the causal mechanisms behind dominant designs. One stream of research emphasizes that a dominant design becomes dominant simply because it delivers the best technological compromise among the different functional characteristics of the technology (Abernathy and Utterback 1978; Christensen et al. 1998; Suarez and Utterback 1995; Utterback and Suarez 1993). Other researchers focus on the self-reinforcing nature of dominant designs and argue that those designs initially gaining the lead in market share often will become dominant (Cusumano et al. 1992; Khazam and Mowery 1994; Liebowitz and Margolis 1995). Another recurring perspective on dominant design is based on the idea of network externalities (Baum et al. 1995; Frenken et al. 1999; Hagedoorn et al. 2001; Rosenkopf and Nerkar 1999; Wade 1995). The key point of this argument is that dominant designs are encouraged if the value of a particular technology depends on the number of other users who have adopted it (Arthur 1989; David 1985).

Finally, the most straight forward explanation to dominant design in product innovation literature derives from theories on economies of scale. On a general level, this concept refers to cost advantages an organization can achieve through expansion. Dominant designs simply are economies of scale that can be realized with standardized products (Hounshell 1984; Klepper 1997). Therefore, as dominant designs emerge market competition is shifted from functional performance to price (Teece 1986).

2.1.2.2 Competition over Attention through Shared Platforms

Since competition over price is inherently related to the concept of dominant design it makes a stable point of departure when trying to distinguish between our two innovation regimes. Producing tangible products entails significant fixed and marginal cost, while producing software does not. Tracing the cost of a car or airplane we will find that a majority is related to production tools, supply chains, factories, and distribution, but also to the marginal cost, such as materials making up the physical artifact. In contrast, as stressed by Microsoft's chairman Bill Gates, the cost of software derives almost exclusively from design. Elaborating the nature of software business in a Wall Street Journal article 2001 he underlines that the digital economy is different. "Say a piece of software costs $10 million to create and the marginal costs, because it's going to be distributed electronically, are basically zero." Once the costs of development have been covered, "every single additional unit is pure profit."

An innovation regime characterized by the absence of marginal cost and limited fixed cost induces new incentives and, therefore, gives rise to a new market logic. At the heart of this new logic we find a story about an abundance of critical resources. In digital innovation the bottlenecks standing behind a demand-side and a supply side are inherently different from the barriers in a product innovation regime. Software is realized, shipped, and consumed electronically, without consuming scarce, physical resources. Essentially, this eliminates price as a dominant force in innovation. Ultimately this promotes markets of infinite choice.

A central argument in the "The Long Tail" (Anderson 2006) is that scarcity of fundamental resources in product innovation enforces dominant designs, here discussed in terms of "hits". As one out of many examples, Anderson narrates that "if there are only a few slots on the shelves or [broadcast] airwaves, it's only sensible to fill them with the titles that will sell best. And if that's all that's available, that's all people will buy" (Anderson 2006, p. 8). As products are increasingly digitized such scarce resources are gradually marginalized and "the mass market is turning into a mass of niches". That way digital technology has "unleashed an extraordinary possibility for many

to participate in the process of building and cultivating a culture that reaches far beyond local boundaries" (Lessig 2004, p. 9). Such power changes markets and threatens established content industries.

Then, as multiplicity explodes in the wakes of digital technology, what drives competition? Obviously, an unlimited multiplicity cannot feed unlimited wealth. Not everyone can make money on digital products. What are the scarce resources of digital innovation?

As pointed out by Bill Gates in the Wall Street Journal article, that scarce resource is attention. On the one hand, profit may sky-rocket as soon as design cost is covered. On the other hand, "your demand can literally almost drop to zero" in the moment when someone comes up with a superior solution and user attention is shifted away. Although this phenomenon is frequently discussed in the literature (cf. Davenport and Beck 2001), it is particularly well articulated by Herbert Simon (e.g. 1971, p. 40-41). What information consumes, says Simon, is "the attention of its recipients. Hence a wealth of information creates a poverty of attention and a need to allocate that attention efficiently among the overabundance of information sources that might consume it." This translates well to digital markets. Multiplicity of software applications creates poverty of attention and a strategic need to allocate that attention efficiently across potential customers.

Trying to set up such strategies it is critical to take into account that digital markets increasingly are taking the form of two-sided markets (Economides and Katsamakas 2006; Eisenmann et al. 2006). Two groups – here end-users and application developers – are attracted to each other through at phenomenon identified by economists as the network effect (Katz and Shapiro 1994; Rosenkopf and Nerkar 1999; Wade 1995). The value of a particular network is largely depending on the number of users on the other side of the network. Game developers will direct their attention towards a community offering a critical mass of players. Similarly, players will favor communities with great variety of games.

Clearly, the product is an essential ingredient and critical glue in these two-sided markets. Sony's Playstation may be a console where developers can design high-quality games, but more important, the diffusion of it allows them to get the attention of users. Equally valid, it guarantees a rich variety of games at the level of end-users. As two-sided markets are increasingly important, competition is shifted towards platform-centric ecosystems (Katz and Shapiro 1994; Tiwana et al. 2010). In such ecosystems, platforms are "products and services that bring together groups of users in two-sided networks" (Eisenmann et al. 2006, p. 94). This perspective downplays the tangible dimensions of the product and put attention on the "infrastructure

and rules that facilitate the two groups' transactions". This marks a distinction towards other perspectives on the concept of platforms, discussed in marketing (Bagozzi 1986; Morein 1975), software engineering (Clements and Northrop 2001; Pohl et al. 2005; Thiel and Hein 2002), or product development (Karlsson and Sköld 2007; Robertson and Ulrich 1998).

Summing up this section, product innovation normally takes place in mature markets, characterized of fierce competition over price and dominant designs. In such environments it is essential to "meet the needs of diverse market segments while [at the same time] conserving development and production resources" (Robertson and Ulrich 1998, p. 20). With this view a platform is a "collection of assets that are shared between a set of [known] products". In contrast, digital innovation increasingly faces the abundance of two-sided digital markets, kept together through shared platforms. Rather than competition over price, deriving from a dependence on scarce material resources, such markets are characterized by a competition over attention. For innovation to prosper the shared platform has to be able to facilitate the two groups' transactions, which at the end of the day requires substantial diffusion across the market.

From the perspective of a product developing firm, experiencing increasing digitalization of products and processes, it is reasonably highly frustrating to face a new form of market, with an inherently new logic. However, causing impact on design and production, a new perspective on the product is significantly worse. Capitalizing on digital innovation simply requires them to develop new a new approach to technology and a fundamentally new perspective on design.

2.1.3 Architectural Design

The more expensive, complicated, and ephemeral a product or service is, the more important it is to build on earlier achievements. Designing from scratch is simply a bad idea in environments characterized by significant pace of change. Such accelerating clockspeed (Fine 1999) is a distinguishing feature of product developing industries as well as business environments centered on IT. However, they adopt different approaches to the design challenges coming with technological change. As we have seen, in a product innovation regime such change emanates from the center of organizations that exercise formal control to improve functionality and reduce cost of products subject to dominant designs. Not surprisingly, product developing organizations, such as an automotive manufacturer, aim for a core design structure – often referred to as architecture – which is relatively stable in time and allows for the depreciation of investments across a range of models and several generations of the product. Essentially, this approach to

architectural design is grounded in a need to identify the least common denominator of a range of known, or at least anticipated, product variants. Properly implemented such a strategy allows for extensive reuse of critical assets while, eventually securing alternatives and attractive pricing at the level of end-users.

We have also seen that digital innovations evolve in networks where a shared platform makes a tool to orchestrate a variety of heterogeneous knowledge in the harsh competition over attention. In such environments it is not surprising that platform designers direct their attention to application developers, rather than end-users. Google's Android platform, as an example, is largely designed to make life easy for developers by providing generic building blocks and proven solutions at the architectural level. The architecture cannot be viewed as the common parts of a range of known products. Offering a collection of best practice tools and inherent support for the reconfiguration and reuse of existing ideas it is rather a catalyzer for open-ended innovation in ecosystems of rich and heterogeneous knowledge.

Architecture is a subtle concept with many facets and angles. It is often described using notions such as abstraction, structure, and style (Garlan and Shaw 1994; IEEE Std 610.12 1990; Kruchten 1995; Perry and Wolf 1992; Ulrich 1995) and discussed in relation to concepts such as platforms (Karlsson and Sköld 2007; Robertson and Ulrich 1998), product families (Du et al. 2001; Jiao and Tseng 2000; Sanderson and Uzumeri 1995), and design rules (Baldwin and Clark 2000). When it comes to the rationale behind architectural investments and architectural thinking, it is argued that product architecture is profoundly linked to product change, variety, commoditization and standardization, performance, and product development management (Ulrich 1995).

Briefly synthesizing this range of perspectives we can tell that architecture is a more stable and broad concept than design. "Architecture *is* design, but not all design is architecture" (Clements et al. 2003). Clearly, an architecture is something that spans particular solutions, designers, and moments in time. Although it is remarkably hard to make a clear-cut illustration of the distinction, most people would agree that design is specific and concrete, while architecture is universal and abstract. Design is forward-looking, aiming for the solution of a particular problem or challenge, while architecture, in some sense, can be described as retrospective, representing some kind of best practice for how to solve a particular class of problems. An architect searches for invariability, or timelessness as expressed by Christopher Alexander in one of his seminal books, discussing the essence of architecture in the context of buildings.

> *There is one timeless way of building. It is a thousand years old, and the same today as it has ever been. The great traditional buildings of the past, the villages and tents and temples in which man feels at home, have always been made by people who were very close to the center of this way. It is not possible to make great buildings, or great towns, beautiful places, places where you feel yourself, places where you feel alive, except by following this way. And, as you will see, this way will lead anyone who looks for it to buildings which are themselves as ancient in their form, as the trees and hills, and as our faces are (Alexander 1979, p. 7).*

Although Alexander belongs to a different discipline, his reasoning captures a fundamental aspect of architecture that I will focus particular attention on in my attempt to distinguish between a product innovation regime and a digital innovation regime; *the magnitude of the concept architecture becomes visible across generations of designs*. While design work is directed towards a particular problem, defined by its time, architecture is a structure-preserving mechanism, passing sound solutions on from design to design and generation to generation. In connecting historical achievements with future potentialities, architecture is a key instrument for path creation, helping firms to create competitive advantage over time. However, what introduces inertia in change may, at the same time, transform into ballast, preventing an organization to improve. Ideally, architectures are structure-preserving and structure-enhancing (Alexander 2002) in the sense that they allow for innovation processes to take advantage of, yet not being obstructed by historical achievements. To some extent this is about the tricky balancing of defensive, retrospective forces and aggressive, forward-looking forces.

We have now established a perspective where architecture is a key concept in the continuous temporal transformation of technology. In order to distinguish between the architectural design in product innovation and digital innovation respectively, we now need to elaborate the basic mechanisms for how technology evolves. What drives change? Let us, as a first step, make use of the generally accepted idea that both natural and artificial systems tend to evolve in response to changes in their context or changes in their underlying components, seeking better “fitness” (cf. Holland 1992a; Holland 1996). Such fitness of a system is the degree to which the system and its context are "mutually acceptable" (Alexander 1964). According to Alexander this translates into an effort to achieve fitness between two entities: the system in itself and the context within which it exists. While the system is a solution to a problem, the context defines the problem.

In trying to achieve fitness with context, traditional engineering design would prescribe a methodology of constrained optimizations. On a general level, constrained optimization aims for "the highest level of product performance within some cost constraint or the lowest cost for a product meeting a minimum performance constraint" (Sanchez and Mahoney 1996, p. 65). A major problem following from this methodology is that it typically leads to highly integrated and tightly coupled designs. In turn, such monolithic designs require tight coordination of work forces, since changes in one component tend to trigger compensating changes on other interrelated components. Interdependencies in product designs simply entail isomorphic interdependencies in organizational structures (Andreasson and Henfridsson 2009; Baldwin and Clark 2000; Sosa et al. 2004). "Product designs composed of tightly coupled components will generally require development processes carried out in a tightly coupled organization structure coordinated by a managerial authority hierarchy" (Sanchez and Mahoney 1996, p. 65). Such designs incur high communication cost (Brooks 1975; Langlois 2002), making an opposing force to change. It is simply very hard to launch new ideas when a wide range of people have to be involved in its implementation. Therefore, the tightly coupled, integral structures produced by constrained optimization design may perform well when context is relatively stable and solutions last over time. Yet, as I will demonstrate in deeper detail, they are inappropriate for environments characterized by significant change.

However, before dealing with the intricate issue of change, let us elaborate a critical reflection on system design that can help us understand the nuances of coupling; sooner or later any design process will end up discussing the interplay between the whole and the parts. When changes in one component diffuses across a system and translates to unexpected effects in other components it is very difficult to understand the whole from its parts. This is at the heart of the concept of complexity, at least as interpreted by Herbert Simon:

> *Roughly, by a complex system I mean one made up of a large number of parts that interact in a nonsimple way. In such systems, the whole is more than the sum of the parts, at least in the important pragmatic sense that, given the properties of the parts and the laws of their interaction, it is not a trivial matter to infer the properties of the whole (Simon 1962, p.468)*

Simon emphasizes that complexity is not an invariant aspect of technology. Rather, "how complex or simple a structure is depends critically upon the way in which we describe it" (Simon 1996, p. 215). Extending such reasoning, complexity can be reduced by finding new structural interpretations of how systems as a whole relate to the parts of a system. There are indeed many

perspectives on complexity and how to reduce complexity. However, most researchers agree that complexity is most efficiently managed by reducing the coupling between parts of a system. This is illustrated by concepts such as information hiding (Parnas 1972) or Brooks' (1975, p. 78) commentary that programmers are "most effective if shielded from, rather than exposed to the details of construction of system parts other than his own".

Given a focus on architectural design and innovation, one aspect stands out as particularly salient; the autonomous innovation following from division of labor seems to outperform more cohesive approaches. Given that the overall properties of the product as a whole can be satisfied, decentralized processes "can have advantages in innovation to the extent that it involves the trying out of many alternate approaches simultaneously, leading to rapid trial-and-error learning" (Langlois and Richard 1992, p. 301). A decomposable approach to design seems to release the creativity in people as they can focus on distinct problems without continuously being obstructed by system level constraints.

Still, the main advantage of decomposable systems is not found in superior mechanisms for making abstractions in a given design process. Rather, we find its key strength in its ability to support technological change in the form of evolution. As demonstrated in his exemplary parable on the watchmakers Hora and Tempus, Simon (1962, p. 473) shows that "complex systems will evolve from simple systems much more rapidly if there are stable intermediate forms than if there are not". Langlois (2002) provides a condensed, yet intuitive outline of the basic reasoning:

> *In a nondecomposable system, the successful operation of any given part is likely to depend on the characteristics of many other parts throughout the system. So when such a system is missing parts (because it is not finished, for example, or because some of the parts are damaged), the whole ceases to function and the system becomes evolutionary shark bait. In a decomposable system, by contrast, the proper working of a given part will depend with high probability on the characteristics of other parts within its subassembly—but will depend with relatively lower probability on the characteristics of parts outside of that subassembly. As a result, a decomposable system may be able to limp along even if some subsystems are damaged or incomplete (Langlois 2002, p. 21).*

In its ultimate form, exercised by biological systems in nature, decomposability paves the way for a perspective on change which "assumes no teleological mechanism. The complex forms can arise from the simple ones by purely random processes" (Simon 1962, p. 471). In his ambitious work on the nature of technology Brian Arthur (2009) essentially adopts

such Simonian thinking when he argues that technology "bootstraps itself upwards from the few to the many and from the simple to the complex" (p. 21). Therefore, one can say that "technology creates itself out of itself". With this perspective systems are not designed, they emerge.

This far I have tried to portray a view of system design and complexity that essentially is shared between product innovation and digital innovation regimes. Competitive advantage over time is rooted in the ability to domesticate technological change in path creating processes. Managing technological change is largely about mastering complexity in design processes, which at the end of the day enforce structural decomposability. Finally, such decomposability opens up for evolutionary, rather than teleological motors of innovation, radically increasing pace of change.

Let us now put attention on a distinguishing aspect that translates into relatively different approaches to architectural design in product innovation and digital innovation. Although recognizing the intricate interplay between a system as a whole and the system in its parts, product developing firms tend to apply a reductionist perspective on complexity. Existing in a highly competitive environment where dominant designs make it largely impossible to question the role and meaning of a product, they turn their attention to the details. Making use of waterfall models (Boehm 1976; Royce 1970) these centralized organizations put things under a finer and finer microscope in order to make parts better or cheaper and then, eventually, put it together again, into a whole. As we shall see modularity is a standard strategy for such firms to handle complexity. It is an indispensable tool for making abstract designs but, above all, it allows for efficient reuse of critical assets in an evolutionary process of technological change. Properly applied modular architectures open up for continuous progression of product performance and cost, while at the same time reusing both components – "the part designs of a product, the fixtures and tools needed to make them, the circuit designs, and the programs burned into programmable chips or stored on disks" – and processes – "the equipment used to make components or to assemble components into products and the design of the associated production process and supply chains" (Robertson and Ulrich 1998, p. 20).

In contrast to the reductionist perspective adopted by product developing firms, designers operating in a digital innovation regime increasingly often approach complexity by looking in the other direction. The key question is not how the whole is to be described through its parts. Rather, they ask: how things assemble themselves? How do new patterns emerge from existing elements? Finding themselves in environments where attention is achieved and maintained by continuous supply of novel functionality, these distributed networks of innovators have to create architectural solutions that

facilitate the materialization of new meanings. In contrast to a reductionist view this makes an open-ended process, where the configuration of patterns may never be finished.

While modularity completely dominates product innovation literature on complex design, the significantly younger discipline of digital innovation is more fragmented. A wide range of researchers have tried to translate ideas from product innovation to fit a software context. Modular software design (Parnas 1972), component-based software engineering (Heineman and Councill 2001), and software product lines (Pohl et al. 2005) are prominent examples of such contributions. However, contemporary research increasingly downplays the usefulness of such direct translations. Digital technology seems to offer properties that do not allow for such morphing to work. As an example, pattern-oriented software design (Buschmann et al. 2008; Gamma et al. 1995) move focus from components, modules, and processes to problems, functions, and ideas. Turning to Christopher Alexander and his ideas on pattern languages, this stream of research argues that the key challenge in software design is to reuse and refine sound ideas. Representing key ingredients of "living structure" a pattern language will act as a sifter in a sandbox as evolution gradually reinforces sound ideas in a series of "structure-preserving and structure-enhancing transformations" (Alexander 1999, p.79).

As we shall see, the increasingly popular concept of generativity (Zittrain 2006) is gaining momentum as a theoretical guide in the design of complex digital systems. On a general level, the term generativity describes a technology's capacity to enable voluntaristic and spontaneous innovation driven by large, heterogeneous an essentially uncoordinated crowds (Remneland et al. 2011). Therefore, at the heart of generativity we find the ability to get quick turnaround on ideas. In other words, generativity allows for efficient reuse of ideas in an evolutionary process of technological change. Properly applied generativity increases the leverage, adaptability, ease of mastery, accessibility, and transferability of a product or service (Zittrain 2006).

Let us now leave the general discussion on architecture behind and engage in a specific review of literature relating to product architecture and software architecture respectively.

2.1.3.1 Product Architecture: Modularity and the Reuse of Assets

Ever since the birth of mass production organizations have been forced to spend significant effort on production processes. Although we see many stances of mass production, a common denominator is found in the need to set up concurrent and autonomous operations. An assembly line is built on a

sequential organizational model, where tools, machines, and knowledge are specialized for a given task. This requires loosely coupled organization structures (Orton and Weick 1990; Weick 1976).

Over the years many researchers have noticed that the structure of product developing organizations tend to match the structure of their products. This "fundamental isomorphism" of design structure and task structure (Baldwin and Clark 2000) suggests that decoupling between tasks at the process level is reflected as decoupling between components at the product level. This interplay between task structure and design structure certainly explains some of the strategic interest in product architecture. As underlined by Sanchez and Mahoney (1996, p. 64) a properly composed architecture may provide a form of "embedded coordination that greatly reduces the need for overt exercise of managerial authority to achieve coordination of development processes, thereby making possible the concurrent and autonomous development of components by loosely coupled organization structures".

It can be argued that product developing firms originally engaged in product architecture to improve production processes, eventually translating to competitive pricing. Yet, contemporary research emphasizes its substantial role also at the level of product design. Seeing product architecture as "the scheme by which the function of a product is allocated to physical components" (Ulrich 1995, p. 419), it largely defines how a particular product can be changed and varied, not only how it is assembled.

Today an overwhelming majority of product developing firms develop their products on the basis of modular architectures. There are voices reminding of integral solutions and their merits (Fixson and Park 2008; Schilling 2000), but an inherent and successfully demonstrated capability to cope with change has largely rendered this discourse obsolete in the context of product development. The power of the concept is illustrated by its impact on a wide range of disciplines. It has had significant influence in fields such as organization studies (Orton and Weick 1990; Sturgeon 2002), management (Baldwin and Clark 2003; Ethiraj and Levinthal 2004; Sanchez and Mahoney 1996), innovation (Robertson and Langlois 1995; Ulrich 1995; Von Hippel 1990), and various forms of design research (Baldwin and Clark 2000; Ulrich and Eppinger 2004). Although giving rise to such wide range of theoretical angles, it is grounded in two relatively simple observations, reflected in the concept of near decomposability (Simon 1962; Simon 2002).

First, Simon establishes that systems "produced by successive assemblies of small numbers of components will emerge much more rapidly than systems that are assembled in one step by uniting large numbers of components" (Simon 2002, p. 598). Such systems form hierarchies in the sense that any

level of analysis will reveal a system of components where each of those components is, in turn, a system of finer components. This recursive decomposition can continue until we reach some point at which the components are "elementary particles" or until science constrains our decomposition (Simon 1962). At the heart of this reasoning we find the idea that stable intermediate forms assist bootstrapping processes, where simple structures are recursively combined over time to form increasingly complex ones (Arthur 2009). An off-the-shelf GPS receiver holds a remarkably complex interior. Yet, it can be used as an elementary building block in a navigation system, without actually paying attention to its hidden complexity.

The second ingredient of near decomposability is that systems in which efficiency of design of each component is relatively independent of the designs of other components will increase their fitness much more rapidly than systems where components are interdependent. Therefore, nearly decomposable systems are manifested as “a hierarchy of components, such that, at any level of the hierarchy, the rates of interaction within components at that level are much higher than the rates of interaction between different components” (Simon 2002, p. 587). In its simplest form, the GPS receiver interface supplies position and time. That allows for satellite positioning technology to evolve relatively independent from contemporary applications. Similarly, applications development is unconstrained by GPS technology.

The idea of near decomposability is truly universal and can be applied in order to understand phenomena ranging from biological systems to human problem solving. Consequently, the notion of component may refer to many different things. However, in a product innovation regime, discussing near decomposability in terms of modularity, the notion of component generally refers to the physical, tangible building blocks that together aggregate into a product. Therefore, when talking about modular structures, the architect normally refers to a hierarchy of such physical components. As a consequence, this hierarchy of parts becomes the main lens to understand complexity of products.

Taking a step back, how does modular product architectures play out in the practice of a product innovation regime? In what sense does it allow organizations to develop competitive advantage in a product innovation regime? So far I have argued that product innovation unfolds from within organizations that exercise formal control modes to improve functionality and reduce cost of products subject to dominant designs. Reviewing organizing logic and market dynamics, we have seen that fit with context, making the primary force for change, is largely defined by the price/performance ratio. Product developing firms continuously improve

their products to keep up with technological progression and market expectations on improved functionality. At the same time the price of a product is a critical distinguishing factor when a dominant design makes the range of offers at a market relatively homogeneous.

Let us then ask how organizations can change their products in order to improve price/performance ratios when markets are characterized by dominant designs. Turning to the literature, it is relatively straight-forward to claim that dominant designs enforces change in the details, while preserving structures of the system as a whole. As emphasized by several researchers, a dominant design is characterized by a set of core design concepts, corresponding to the major functions of a product (Clark 1985; Henderson and Clark 1990; Marples 1961). It also comes with a general idea of how these core design concepts are embodied in physical components and eventually integrated into a product (Clark 1985; Henderson and Clark 1990; Sahal 1985). "Once any dominant design is established, the initial set of components is refined and elaborated, and progress takes the shape of improvements in the components within the framework of a stable architecture" (Henderson and Clark 1990, p. 14).

Hence, to stay competitive product developing organizations have to tune their architectural strategies towards the details. Structuring the parts of a system according to the principles of modularity allows them to focus their design attention on the internal properties of components. With this approach it is possible to feed markets with variety and change, while at the same time trying to preserve stable system solutions to conserve development and production resources for scale advantages (Robertson and Ulrich 1998). It makes little sense for an automaker to question the overall meaning, behavior, or structure of e.g. a navigation system. Instead, the fact that basic elements – map, routing, and guidance – as well as the interplay between these elements are defined by the dominant design allows for scale advantages in production. At the same time it is crucial to continuously improve fitness by a devotion to details. Response time in routing, level of details in maps, or precision in guidance instructions may be the distinguishing features directing the flow of customers from one brand to another. Such reasoning is at the heart of product platform literature (cf. Karlsson and Sköld 2007; Robertson and Ulrich 1998), arguing that a central challenge of product innovation is to, on the one hand, take advantage of the cost-saving potential in dominant designs and, on the other hand, differentiate the functional offer to end-users in order to escape the grip of price as the only discriminator between models and brands. Therefore, competitive advantage on markets characterized by dominant designs grows from the capability to continuously fine tune the fitness of a relatively stable

overall system solution by adapting its different parts (Abernathy and Utterback 1978; Clark 1985; Henderson and Clark 1990). Product innovation regimes feed reductionist perspectives on complexity, normally addressed by modular product architectures.

Then, let us try to uncover yet another layer of details in order to understand the role of modularity in a product innovation regime. How do modular architectures actually promote variety and change? Given that we see an architecture as a structure-preserving and structure-enhancing mechanism, how does it allow for the reuse of historical achievements in benefit of future potentialities? Let us walk through, at least in some detail, how modularity delivers variety and change in tangible products by reusing (1) production assets and (2) existing components. As we shall see, these are critical aspects when firms ask themselves "what product architecture should be used to deliver the different products while sharing parts and production steps across the products" (Robertson and Ulrich 1998, p. 21).

In order to achieve scale advantages in a product innovation regime production needs to be nearly algorithmic, with a well defined assembling process, enabling high-speed throughput (Chandler 1977). This push firms to deploy specialized capital, such as assembly lines, tooling, equipment, and various materials (Teece 1986). Such complementary assets make considerable investments for an organization, but with an absolute majority of fixed and marginal cost relating to production it pays off through lower unit cost in an economy of scale (Chandler 1990).

In addition, these complementary assets make an appropriation regime, "that governs an innovator's ability to capture the profits generated by an innovation" (Teece 1986, p. 287). When market competition requires significant investments in a wide range of complementary assets it is simply very hard for a newcomer to disrupt the barrier and translate a competing design into a competitive product. It is argued that this mechanism ranges beyond the stability of dominant designs and can help incumbent, product developing organization to appropriate the value also of radically new technology. At the heart of such reasoning we find the idea that "incumbent industry performance improves if the new technology can be commercialized through [existing] specialized complementary assets" (Rothaermel and Hill 2005, p. 52). Consequently, such assets make a valuable, strategic instrument for most organizations in a product innovation regime. Preserving this value over time is a key challenge for most product developing organizations.

On the one hand, modularity can be viewed as an abstract and "very general set of principles for managing complexity" (Langlois 2002, p. 19). However,

applied to product architecture it becomes highly concrete as it defines how a system can be separated and recombined (Schilling 2000). Properly implemented a modular architecture opens up for a palette of variants, still preserving the overall structure of the system and the interfaces between components. The extent to which a product can be viewed as modular is reflected in "the tightness of coupling between components and the degree to which the 'rules' of the system architecture enable (or prohibit) the mixing and matching of components" (Schilling 2000, p. 312). Properly exercised a modular architecture allows an organization to increases the value of complementary assets as products can be assembled at the same line, by the same people, using the same tools and the same basic components. This simply gives a manufacturer the opportunity to depreciate investments across significantly larger volumes, eventually leaving larger margins and higher profit. However, this leaves the manufacturer with an intricate architectural challenge, inevitably enforcing a perspective on modularity where the physical structures are in focus; a whole range of different products have to be decomposed and aggregated on the same basic premises, yet delivering variety. Modular architectures offer this, yet without exploding in complexity. The capability to encapsulate information and functionality in hierarchical structures of components, while serving simple external interfaces, is a critical aspect of modular product architectures since complex solutions are more difficult to assemble, require more expensive tools, and tend to be weaker in terms of quality. Even more important, increasing complexity may hamper product change over time. This is critical since the value of complementary assets is not only relying on the generic capability to support a range of products, but also on its resilience over time. The value of tools, materials, and processes decreases dramatically if they continuously have to be adjusted in order to align products with a changing market context. Modularity allows for a range of variants and product generations to share a temporally stable architecture. This offers "reduced uncertainty over product design [which in turn] provides an opportunity to amortize specialized long-lived investments" (Teece 1986, p. 288). Synthesizing our discussion so far, one can argue that with a modular strategy to product architecture organizations may reinforce and preserve the value of complementary assets across specific product offers and generations of designs.

Let us now focus our attention on the role of modularity in reusing components. Approaching this topic we need to make a slight detour, discussing design processes. As we know, a product innovation regime exercises linear models of product development. This prevailing model of innovation can be traced to a strong need to reduce ambiguity about the physical structure of the product (Godin 2006). Relying on formal control modes and strictly linear development processes organizations have to

change their locus, from functional design to physical design, at an early stage. By the time a design is released for production functional properties are inevitably frozen (Baldwin and Clark 2000). Clearly, the deployment of functional structure to physical structure is a critical moment, defining how a product can be changed both within the life cycle and across generations (Ulrich 1995). As a consequence, the interplay between functional structure and physical structure is highly visible in the architectural thinking in product innovation literature. One of the most cited definitions of product architecture is phrased by Karl Ulrich in a Research Policy paper from 1995. He defines product architecture as "the scheme by which the function of the product is allocated to physical components" (Ulrich 1995, p. 419). Elaborating this condensed statement, he offers a detailed discussion on how this overall definition translates into "the arrangement of functional elements", "the mapping from functional elements to physical components", and "the specification of the interfaces among interacting physical components" (p. 420).

Ulrich's definition of architecture is seemingly distant from the perspective I have outlined in the introduction of this subsection. After all, he identifies the architecture as a bridge between a functional domain and a physical domain, rather than an evolutionary bridge between generations of designs. However, let us recall that it is in the deployment of functional structure to physical structure that coupling appears, at least the kind of coupling that messes up the assembling of components into products. The product architecture "determines which functional elements of the product will be influenced by a change to a particular component, and which components must be changed to achieve a desired change to a functional element of the product" (Ulrich 1995, p. 426). Thereby, it defines evolutionary properties of a product. "A modular architecture increases the likelihood that a component will be commonly useful". The ultimate modular architecture maps functional elements to components one-to-one, meaning that "each component implements one and only one function" (Ulrich 1995, p. 431). At a practical level, such one-to-one mappings make improvement of a particular functional property a lot more likely, since it does not requires different component suppliers to synchronize and align their efforts.

Even more important in a context where functionality is frozen early in design processes, the product architecture defines "the degree to which a system's components can be separated and recombined" (Schilling 2000, p. 312). Loosely coupled components are simply significantly easier to reuse and reconfiguration for new purposes. In fact, modularity exponentially increases the number of possible configurations achievable from a given set of inputs, which greatly increases the flexibility of a system (Arthur 2009;

Schilling 2000). This makes an almost priceless capability for product developing firms to moderate variation and change without redefining a system solution or, ultimately, even the components of the system. As we have discussed, coupling may favor functional performance in the short run, but complicates change and adaptation over time. To what extent it is desirable to reduce coupling by modularization is largely given by context. However, product innovation regimes normally face markets with substantial dynamics and harsh competition over price. In such an environment, decoupling translates into vital strategic flexibility, facilitating adaptation to context and, eventually, improving competitive advantage.

Concluding this section, a product innovation regime translates the reuse of plants, production tools, processes, and components into competitive advantage. The architecture of products has proved to be of significant importance when building such capability. In general, product developing organizations architect their products according to the principles of modularity. The near decomposability of such architectures gives them significant flexibility to differentiate products over a range of variants and across generations of designs, yet commoditizing critical assets.

2.1.3.2 Software Architecture: Generative Designs and the Reuse of Ideas

Software engineering is a young discipline. So is the notion of architecture in the context of software. A historical expose in a 2006 special issue of IEEE Software (Kruchten et al. 2006) traced the concept of software architecture back to an early conference on software engineering techniques in Rome 1969 (Buxton and Randell 1970). The conference hosted a whole range of researchers, such as Tony Hoare, Edsger Dijkstra, Alan Perlis, Per Brinch Hansen, Friedrich Bauer, and Niklaus Wirth, later making the backbone of the upcoming software engineering discipline. In relation to the concept of software architecture Ian P. Sharp made a statement which diverted from established thinking and paved the way for deeper theoretical contributions in this area. Arguing that "architecture is different from engineering", he wanted to point out that an architecture is not the same as a design and the act of architecting is not the same thing as designing. Sharp wanted to put attention to the consequences of seeing specifications of software purely as functional specifications.

> *We only talk about what it is we want the program to do. It is my belief that anybody who is responsible for the implementation of a piece of software must specify more than this. He must specify the design, the form; and within that framework programmers or engineers must create something. No engineer or programmer, no programming tools, are going*

> *to help us, or help the software business, to make up for a lousy design (Buxton and Randell 1970, p. 9).*

Although being ahead of his time, Sharp certainly helped seeding the idea that architecture is relevant to the software industry and something that may support the reinforcement of sound and coherent software systems over time. Yet, over the coming two decades "the word 'architecture' was used mostly in the sense of system architecture (meaning a computer system's physical structure) or sometimes in the narrower sense of a given family of computers' instruction set" (Kruchten et al. 2006, p. 23).

However, shifting into the 90th, the concept of software architecture attracted enough attention to form a distinct discipline. In 1991 Royce and Royce (1991) published a seminal paper positioning software architecture explicitly between technology and process. This is also the period when it became increasingly accepted to claim that this subtle concept could mean different things, depending on the observer. In the "4+1 view model" Philippe Kruchten (1995, p. 1) proposes a new way of "describing the architecture of software-intensive systems, based on the use of multiple, concurrent views". He argues that "the use of multiple views allows to address separately the concerns of the various 'stakeholders' of the architecture: end-user, developers, systems engineers, project managers, etc., and to handle separately the functional and non functional requirements".

I consider this a critical period in the history of software architecture. This is when a wider audience accepts the idea that software architecture plays out at many different levels, beyond pure technology. It is possible to see it as "the structure or structures of a system, which comprise elements, their externally visible properties, and the relationships among them" (Clements et al. 2003, p. 471), yet discuss inherently different perspectives. Studying a system from an end-user perspective, 'elements' and 'structure' may refer to functional building blocks. For a programmer the architecture may be the guide to sound real-time behavior and the hardware designers is primarily interested in the deployment of code to physical components. However, the key to sound and competitive products is found in the capability to *combine* different perspectives.

In modern literature on software architecture we see several different branches. I would argue that, on a general level, we can tell them apart by the way they stress different architectural views. Some schools, in particular the early ones, tend to approach architecture primarily from the perspective of design processes and production of software systems. They are clearly inspired by the engineering techniques that successfully improved flexibility and efficiency in product development. Therefore, they emphasize structures with impact on the realization of software systems, rather than on functional

design. Modular software design (MSD) (Parnas 1972; Parnas et al. 1985), as an example, offers a design technique where information hiding is reinforced through a hierarchically structured code base, much similar to how modularity is applied in product development. Another branch of software engineering, highly intertwined with Parnas' ideas on modular design, is often labeled component-based software engineering (CBSE) (Crnkovic 2001; Heineman and Councill 2001). As for modular design, proponents of CBSE underline the separation of concerns in a software system. Software components are seen as autonomous, independent elements, defined by their interfaces. The main idea is that software should be componentized – that is built from prefabricated components – much similar to the fields of electronics or mechanics. Further, software product lines (SPL) (Clements and Northrop 2001; Pohl et al. 2005) is a contemporary movement also inspired by manufacturing industries, where software systems are created from a shared set of software assets using common methods, tools, and techniques for production. On a general level, SPL takes the concept of mass customization (Pine and Davis 1999) to the domain of software.

As illustrated, SPL, CBSE, and MSD approach architecture from the perspective of software design. Investments in architecture pay off through efficient work processes, flexible software systems, and reusable code bases. However, with the emergence of object-oriented programming (OOP) and object-oriented analysis (OOA) (Booch et al. 1991; Mathiassen et al. 2000) we see a gradual shift in architectural thinking across the software engineering discipline. Proponents of OOP/OOA stress the need to model real-world phenomena. They argue that an artifact design has to emerge in coherence with an improved understanding of context. They distance themselves from a practice where context is squeezed into specifications at an early stage, after which the whole attention is focused on the design of an artifact. Instead, context and system has to be modeled together. This put what is today widely recognized as the logical view more in center of attention. In contrast to CBSE, OOP/OOA methodology seeks to create the "verbs" and "nouns", readable to humans, rather than structures of reusable assemblages of software. This challenges the taken for granted distinction between system and context applied in traditional product development. To some extent, OOP/OOA includes the context in the design process, rather than building on a static, pre-fabricated stance of it.

With architecture increasingly associated with the functional structures of software in context, the concept was gradually loaded with a new meaning. Over the last two decades service-oriented computing (Allen 2006; Papazoglou and Georgakopoulos 2003), pattern-oriented software architecture (Buschmann et al. 2008; Gamma et al. 1995), and other

theoretical perspectives have reinforced the idea that architecture is not just a set of tools for the structural transformation of the software system as an artifact, but a strategic tool guiding the gradual transformation of functionality.

Service-oriented computing (SOC) "uses services to support the development of rapid, low-cost, interoperable, evolvable, and massively distributed applications. Services are autonomous, platform-independent entities that can be described, published, discovered, and loosely coupled in novel ways" (Papazoglou et al. 2007, p. 38). Obviously, SOC does not approach software or software systems as something that is up-front defined. Rather, services are software functions that are reusable in new configurations, and for new purposes. A Service-oriented architecture (SOA) is a set of flexible design principles used for designers to navigate in a volatile and changing environment. Therefore, in SOC, a software system is something that emerges over time.

Pattern-oriented software design (POSD) (Buschmann et al. 2008; Gamma et al. 1995) emphasizes similar values as SOC, although it does not to the same extent engage in the realization of services or business processes. Instead, this branch stresses that patterns "document existing best practices built on tried and tested design experience. Patterns are not invented or created artificially just to be patterns" (Buschmann et al. 2008, p. 8). Rather, they "distill and provide a means to reuse the design knowledge gained by experienced practitioners," so that developers familiar with an adequate set of patterns "can apply them immediately to design problems without having to rediscover them" (Gamma et al. 1995, p. 1).

Most people argue that the concept of patterns, as applied in software engineering, can be traced back to the work of Christopher Alexander (Alexander 1964; Alexander 1979; Alexander 2002; Alexander et al. 1977). To Alexander a pattern "describes a problem which occurs over and over again in our environment [...] and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" (Alexander et al. 1977, p. x). However, Alexander did not stay with the pattern concept in isolation, but argued that the potential in patterns is uncovered when studying how they relate to each other. Much like the words of a language, patterns make sense together through vocabulary, syntax and grammar. The vocabulary – a set of patterns – is a collection of solutions to well defined problems. The syntax show how a specific pattern fit with other patterns in a larger design. Finally, the grammar describes in what way the pattern solves a problem.

Alexander developed his ideas around pattern languages from observations of "certain generative schemes" in the building of houses that exist in traditional cultures. Trying to make sense of his work as an architect in the eyes of software engineers, Alexander explains that:

> *These generative schemes are sets of instructions which, carried out sequentially, will allow a person or a group of people to create a coherent artifact, beautifully and simply. The number of steps vary: there may be as few as half a dozen steps, or as many as 20 or 50.When the generative scheme is carried out, the results are always different, because the generative scheme always generates structure that starts with the existing context, and creates things which relate directly and specifically to that context. Thus the beautiful organic variety which was commonplace in traditional society could exist because these generative schemes were used by thousands of different people, and allowed people to create houses, or rooms, or windows, unique to their circumstances (Alexander 1999, p. 81).*

Obviously, Alexander does not seek structures that make software systems (or in his case buildings) homogeneous and uniform. On the contrary, he seeks the structures that allow for new solutions to emerge in harmony with context, yet taking historical wisdom and best practice into account. Representing key ingredients of "living structure" a pattern language will act as a sifter in a sandbox as evolution gradually reinforces sound ideas in a series of "structure-preserving and structure-enhancing transformations" (Alexander 1999, p.79).

This detour into the details of pattern languages is relevant to illustrate a slow and gradual, yet clear shift of perspective in the discipline of software architecture over the last two decades. On the one hand, the different perspectives I have reviewed agree that decoupling is a key property in handling complexity in software systems. On the other hand, the locus of attention seems to shift from the complexity of artifacts to the complexity of problems. With a growing focus on problems, software architects increasingly recognize that in order "to study and analyse a problem you must focus on studying and analysing the problem world in some depth, and in your investigations you must be willing to travel some distance away from the computer" (Jackson 2000, p. 9). Distancing themselves from the computer, many designers and architects see in software what Alexander saw in buildings; good design is an emergent phenomena. Context is certainly not static, neither are the problems defined by context. Therefore, it is increasingly emphasized that architectural design is less about the identification of generic structures of software systems *per se*, but rather a matter of identifying, describing, and using the generative schemes helping us to create what Alexander refers to as "living structure", valid across

contextual barriers. As our ideas evolve, an architecture has to allow for designs to be re-factored and code to be reshaped and transformed. So "rather than looking for complex design tools with the hope of creating the ultimate design, we should continue to seek out practices, techniques, and tools that support a sustainable software design process and adaptable, habitable designs" (Wirfs-Brock 2009, p. 7).

Finally, seeing design as an emergent process is largely incompatible with a reductionist perspective on complexity. In contrast to a traditional product development setting, software architects increasingly find themselves not knowing exactly what they are architecting for. A software platform, such as Android, certainly offers a whole range of generic elements, yet we have no clear idea of how these elements will be used and combined to form the applications of tomorrow. Therefore, the question of how the whole is to be described through its parts may be hard to answer, or ultimately meaningless, for a software architect. Rather, they ask how things assemble themselves. How do new patterns emerge from existing elements? Software architecture seems to increasingly adopt a bottoms-up approach to complexity, as it is described in complexity theory (Anderson 1999; Holland 1992a; Holland 1996), rather than the mechanistic, reductionist perspective taken for granted in product development.

Then, how does this perspective on software architecture play out in contemporary practice? How does an emergence-oriented stance, seeing software as a complex adaptive system (Holland 1992b), resonate with the many other aspects we have discussed of a digital innovation regime? So far, I have argued that digital innovations evolve in networks, centered on a shared platform that makes a tool to orchestrate a variety of heterogeneous knowledge in the harsh competition over attention. Such networks – or ecosystems – are generally not up-front assembled to support a specific purpose or a given product. Rather they emerge in response to opportunities offered by a general platform (Katz and Shapiro 1994; Selander et al. 2010; Tiwana et al. 2010). Referring to the work of Zittrain (2006; 2008), we can argue that a platform able to trigger "voluntaristic and spontaneous innovation" in "large, heterogeneous and uncoordinated crowds of people" (Remneland et al. 2011, p. 210) holds generative capacity. Obviously, such generative capacity is a phenomenon playing out in the interplay between technology and social structures. Still, there has to be some inherent properties of the technology enabling generative practices. Can we identify these properties? How does the shift in philosophy among software architects favor the emergence of generative technology?

Approaching these questions, let us first note that digital technology is inherently intertwined with the stored-program concept. In product

innovation the physical artifact – the product – mediates value and guarantees revenue for the innovator. In contrast, a digital innovation regime feeds functionality which is not inscribed in products, but carried by software, decoupled from the physical artifact. This fundamental separation of hardware and software derives from the stored-program concept, manifested as a processing unit, executing digitally encoded instructions, and a storage unit holding both instructions and data. The programmability following from such von Neumann architectures (Burks et al. 1963; Goldstine and Von Neumann 1963) or Turing machines (Turing 1937) allows for technological progression to occur without entering a new loop of artifact design and production. Essentially, digital technology allows for new ideas to materialize without marginal cost. This replicability dramatically reduces the role of price in technological change. Progression is not constrained by a need to cover the cost of plants, production tools, and supply chains across product variants and generations. Reuse is not motivated by depreciation of economic investments, at least not to the same extent as in traditional product innovation.

Still, reuse is a central aspect in digital innovation. It is just not about the reuse of material things, such as tools or physical components. If we listen to the proponents of pattern-oriented design it is not even about the reuse of software components or code, it is about the reuse of ideas. The shared platforms, making a center of gravity in successful innovation ecosystems, represent a pattern language. This pattern language offers both a way to identify the core design problems of a particular application domain and replicable rules and building blocks for their solutions (Steenson 2009). We can see these shared platforms as a common framework for collaboration and a set of axiomatic resources to be used in innovation. Obviously, the platform is a manifestation of reuse. It offers a whole range of reusable resources that make life easier for software designers. Yet, it can be argued that its main advantage is not found in the capability to facilitate a given work process by offering precompiled bodies of code, but in its capability to shape applications – potentially a whole domain of applications – over time. Following Alexander (1999), such platforms constitutes structure-preserving and structure-enhancing elements, shaping coherent and sound, but different designs over time. On the one hand, it facilitates unbounded innovation and technological progression. On the other, it embodies an innovation regime and may constrain and hamper change. A “thin” and too general platform may be unable to hold an ecosystem together as it offers minor support in innovation. A “thick” and too specific platform, on the other hand, may prevent designers to realize ideas as it enforces predefined, standard solutions to solve known problems.

Although Christopher Alexander has not explicitly foregrounded the notion of generativity *per se*, I think it is rather straight-forward to say that it is a central concept in his work. I would even argue that his core message to software engineering is that the creation of generative technology is one of the main challenges for the discipline in general and for software architecture in particular (Alexander 1999). Turning to recent writings on generativity (Remneland et al. 2011; Zittrain 2006; Zittrain 2008), a software platforms should be architected with five principle factors in mind; capacity for leverage, adaptability, ease of mastery, accessibility, and transferability.

Capacity to leverage denotes the degree to which a technology enables "valuable accomplishments that otherwise would be either impossible or not worth the effort to achieve" (Zittrain 2006, p. 1981). The more effort a software platform saves, the more generative it is. *Adaptability* refers to "the breadth of a technology's use without change and the readiness with which it might be modified to broaden its range of uses" (p. 1981). A software platform allowing for hundreds of different applications to emerge is simply more generative than a platform tailored to the needs of a particular branch. Further, *ease of mastery* "reflects how easy it is for broad audiences both to adopt and to adapt it" (p. 1981-1982). Essentially, it is a measure of the magnitude of skills necessary for a designer when making use of a technology's leverage capacity. From this perspective a generative software platform should offer low cognitive barriers of entrance for designers, while at the same time being malleable in the sense that it does not prescribe a specific use. *Accessibility* of a technology is a measure of how "readily people can come to use and control a technology" (Zittrain 2006, p. 1982). Software platform without economic barriers of entrance tend to be more generative than those requiring significant up-front investments. Similarly, a limited number of legal barriers tend to open a platform for a wider audience of potential users. Finally, the level of *transferability* "indicates how easily and accessible changes and updates in the technology are distributed among its users" (Remneland et al. 2011, p. 210). As an illustration of this dimension, it is suggested that an open source platform may be more generative than a proprietary correspondence, simply since "contributions are open for a wide community to modify and change".

For product developing organizations it is crucial to design products so that it is possible to reuse plants, production tools, processes, and organization structures to meet future challenges. In general, they architect their products according to the principles of modularity so that these massive investments can be covered by a range of variants and across generations of designs. As a contrast, a digital innovation regime increasingly recognizes ideas, solutions,

patterns, and functions as key elements for reuse in a combinatorial evolution of technology. Competitive advantage grows from the capability to explore and exploit these elements internally as well as externally (Chesbrough 2006). In a digital innovation regime this pushes organizations to architect generative software systems, allowing essentially unrelated and unaccredited audiences to build and distribute code and content (Zittrain 2006).

2.2 Research Challenges in Digital Product Innovation

Innovation is about change. More precisely, the technological change we associate with innovation arises by *combination* of existing technologies. We can view an innovation regime as the ground rules of this process. It defines how technology over time "bootstraps itself upwards from the few to the many and from the simple to the complex" (Arthur 2009, p. 21). As we have seen, the literature offers distinctly different perspectives on product innovation and digital innovation. As illustrated in Table 1, a digital innovation regime emphasizes some elements, structures, and logics, while product innovation emphasizes other. As digital technology is increasingly integrated in tangible products we can expect these differences to translate into tensions, making strong forces in the change of established innovation practices.

Table 1. Salient dimensions of product innovation and digital innovation.

	Product Innovation	*Digital Innovation*
Organizing logic	Linear processes	Non-linear processes
	Behavioral control	Output control
	Vertical industries	Horizontal industries
	Teleology	Evolution
	Flexibility	Agility
	Firm-centricity	Network-centricity
Market Dynamics	Direct sales	Two-sided markets
	Competition over price	Competition over attention
	Marginal cost	Fixed cost
	Economy of scale	"a mass of niches"
	Dominant Designs	Shared Platforms
Architectural Design	Physical structures	Functional structures
	Complexity of artifacts	Complexity of problems
	Reuse of assets	Reuse of ideas
	Hierarchy	Network
	Change at the level of details	Change at the level of specifics
	Reductionism	Emergence
	Modular designs	Generative designs
	Early binding	Late binding

Do we see these tensions? Can we even argue that digital technology is widely integrated in tangible products? As an illustration of the increasing importance of digital technology in product development, the software content of a modern car now exceeds 10 million lines of code (Broy et al. 2007). Further, as much as 80% of all car innovations can be traced to digital technology (Hardung et al. 2004; Leen and Heffernan 2002). No doubt, the appropriation of new capabilities (King and Lyytinen 2005) following from digital innovation has improved the functionality of cars significantly in many application areas, ranging from climate control and infotainment to engine, braking, and transmission systems.

Still, the momentum we see in digitization of complex manufactured products seems to be powered by arguments from a product innovation regime. The miniaturization of hardware, increasingly powerful microprocessors, inexpensive and reliable memory, broadband communication, and efficient power management simply offers extra-ordinary opportunities to improve complex manufactured products (Yoo 2010; Yoo et al. 2010a). Searching for industry-related evidence, literature offers a variety of explanations to the ongoing digitization of manufactured products.

A recurring argument is that digital technology "is an important enabler of new and increasingly complex functions. Using software and networking it is today possible to create *new functionality* (italics added), such as an [in-car] anti-skid system, that was considered unfeasible, both with respect to cost and functionality, some ten years ago" (Axelsson et al. 2004). Further, it offers new approaches to *systems integration,* moving complexity from a physical domain to the logical domain of software elements, interconnected over digital networks (Eklund et al. 2005; Racu et al. 2007). Such networks allow manufacturers to "replace the numerous cables and harnesses and thereby reduce the number of connection points, cost and weight" (Axelsson et al. 2004). It is also stressed as a new opportunity to handle *variability*. While such "variability has typically been addressed on a case-by-case basis in late development phases" digital infrastructures now allow manufacturers to adopt a more "systematic approach to the ever-increasing number of variants" (Thiel and Hein 2002, p. 66). Furthermore, digitization of products and processes has opened up for the adoption of *model-based design* methods, making "a more rigorous approach to system development compared to the current state of practice" (Cuenot et al. 2007).

Indeed, product developing organizations change in the wakes of digitalization. Given the documented impact on functionality, systems integration, variability, design methodology, etc, it is probably an

understatement to say that these new technologies changes innovation practices significantly. However, change largely seems to align with the path prescribed by a product innovation regime. We see little change in organizing logic and markets seem to remain relatively homogeneous, centered on a few dominant designs. In short, the tensions of Table 1 do not fully play out in practices. As illustrated well by today's most salient industry-wide software initiative in the automotive industry – the AUTomotive Open System Architecture (AUTOSAR) – the ongoing domestication of digital technology is essentially directed towards the number one challenge of a product innovation regime; the balancing between commoditization and diversification.

> *Reductions of hardware costs as well as implementations of new innovative functions are the main drivers of today's automotive electronics. Indeed more and more resources are spent on adapting existing solutions to different environments. At the same time, due to the increasing number of networked components, a level of complexity has been reached which is difficult to handle using traditional development processes. […] To achieve this, AUTOSAR defines a methodology that supports a distributed, function-driven development process and standardizes the software-architecture for each ECU in such a system (Fennel et al. 2006).*

Even though we can find digital technology essentially wherever we look in product development, the industry seems to argue that tangible, complex products are fundamentally different from IT. In a car context, "any software architecture must first recognize the automobile industry's myriad unique pressure and ad hoc design practices" (Simonds 2003, p. 8). Therefore, "one should be very careful to uncritically apply technical solutions from one industry in another", even when they are closely related (Fröberg et al. 2005). The evidence reported in literature suggests that these organizations approach digital technology with an ambition to shoehorn it into existing models for innovation. Rather than seeing the generative aspects of digital technology, it is made a tool to reinforce a traditional product innovation regime. It simply seems as if they are using digital technology to solve problems associated with former generations of technology, not as an opportunity to identify new paths of innovation, exploiting novel angles on product development practices, organizational configurations, or business models.

Clearly, product developing industries need to develop new capabilities to release the potential of digital technology (cf. Henfridsson et al. 2009a; Jonsson 2010). Although mainstream innovation follows a well known trail,

we see genuinely new initiatives, such as the GENIVI alliance[5], and launches of new concepts, such as Ford Sync, BMW ConnectedDrive, Saab IQon, and Fiat Mio. It can be argued that these initiatives break with the logic of established product innovation practices and are framed to benefit from the generative capabilities of IT. Essentially, the literature is silent on this ongoing adaptation of product innovation practices. At least, we do not see a scholarly discussion on how to combine product innovation and digital innovation, allowing software to be increasingly disconnected from hardware, while at the same time recognizing that complexity keeps playing out across both hardware and software.

Motivated by the upcoming initiatives in industrial practice and the apparent gap in literature, I embark on a study of how product developing firms build new innovation practices, combining the logics of product innovation and digital innovation. In seeking a better understanding of how digital technology shapes new innovation practices in product developing organizations I focus my attention on a salient phenomenon, present all throughout my distinction of product innovation and digital innovation; software separates the meaning and functional behavior of a product from the product in itself. Up until recently this basic property of IT has been exploited at a technical level to leverage functional improvements, and efficient design practices, but largely ignored outside R&D departments. To benefit more widely from the decoupling of functionality and hardware organizations have to develop new ground rules for how to moderate technological change. Therefore, the main focus of this thesis is to study how the two innovation regimes portrayed in this thesis are conceptualized and combined in architectural designs and architectural thinking. The research question is: *how do product developing firms architect digitized products to leverage the generative capability of IT?*

This section has contrasted a product innovation regime and a digital innovation regime. Such a portrait is not unambiguous and can be done from many different perspectives. My compilation is done to illustrate the challenges a product innovation regime faces when traditionally non-digital products are increasingly digitized. However, it is also done to uncover a gap in the literature, justifying this thesis. Next section explicates the theoretical lenses used when approaching the empirical context. First, I outline a perspective on digital materiality. I have no ambition to make a generally applicable contribution to this extensive topic. Rather, this seeming detour is motivated by a need to clarify some basic differences between digital and

[5] http://www.genivi.org

analog technology. This discussion is then applied in the development of two distinct architectural frames, corresponding to the two innovation regimes.

3 Theoretical Framework

Product innovation and digital innovation represent different modes of innovation. They bring forward different organizing logic, feed different market dynamics, and cultivate different approaches to architectural design. The notion of *regime* underlines that these different modes of innovation are shaped by a whole range of different actors – human and material – together forming a web of forces, pulling in different directions. An innovation regime is constituted by a reasonably stable state – equilibrium – where the different forces play in concert.

By viewing an innovation regime as a particular form of interplay between humans and technology I align with a central discourse in the information systems discipline; the continuous debate on the relationship between information technology and organizations. As framed by Leonardi and Barley (2008), technology is shaped by negotiations (Constantinides and Barrett 2006; Howcroft and Wilson 2003; Orlikowski 1992), human agency (Boudreau and Robey 2005; Poole and DeSanctis 2004; Vaast and Walsham 2005), and personal interest (Kling 1992; Markus and Benjamin 1996; Scott and Wagner 2003). At the same time, it is widely recognized among researchers that organizations emerge in an interaction between people and machines (Mohr 1971; Thompson and Bates 1957), social and technical subsystems (Barley 1990; Scott et al. 1998), or social and material practices (Orlikowski 2002; Schatzki 2005). Essentially, information systems researchers agree that "information technology and organizations both arise at the intersection of social and material phenomena" (Leonardi and Barley

2008, p. 160). Still, we see a wide range of different perspectives on the epistemological and ontological nature of the relationship between the social and the material. These perspectives range from techno-centric determinism to human-centric relativism. The former extreme sees human action largely as a response to technological change, while the latter emphasizes that humans have free will and shape their environments to achieve particular goals.

Although truly fascinated by the different facets of this discourse, my current engagement is motivated by a rather precise observation; as tangible products are increasingly digitalized, the relative stability of traditional product innovation regimes is disrupted. Software, digital networks, integration with external digital infrastructure, etc inject new opportunities in innovation. At the same time, it challenges established processes, structures, and logics, which introduces new tensions and, eventually, seeds new paths of change in product innovation.

I will not engage in a deeper discussion on different socio-material perspectives and their respective benefits. However, addressing the research question I want to make two statements, positioning my research in the continuum between hardcore determinists and extreme relativists. First, I see digital technology as inherently different from the non-digital technology of tangible products. It introduces material properties allowing people to do what they already do in novel ways, but also to do things they could not do before. Thereby, I distance myself towards the kind of research arguing that technology holds a subordinate position in organizational change. By illustrating tensions between somewhat idealized innovation regimes in product development and IT settings I seek a demonstration of how digital technology, in itself, constitutes a powerful force in shaping new practices.

At the same time, I reject the idea that a new technology superimposes an inevitable path of change. Rather, an innovation regime is formed over time by an intricate set of contradictory forces, gradually mangling out new practices. This mangling is far from deterministic and can be seen as a threesome dance of agency (Svahn et al. 2009) where the affordances of a new technology are subjected to human agents, experiencing resistance from established socio-technical structures. This model underlines that technology is not a progressive force *per se*, but can make powerful resistance to change when embedded in organizational structures, routines, and practices. Consequently, we will not see automotive manufacturers and other industrial actors translate from orthodox product innovation to digital innovation, as we know it from software settings, as they assimilate digital technology. Their unique path will emerge over time in reasonable concert with existing practices.

In summary, I advocate a perspective recognizing the transformative power of technology, while at the same time not resorting to determinism. This makes a realist perspective in that it recognizes that technology exists independently from observers. At the same time, it recognizes that our knowledge about the world is socially constructed. In addressing the research question, this critical realist view makes a compass directing my attention to material agency and affordances, but at the same time reminds me of the need to understand and conceptualize human reasoning exercised when people try to make sense of digital technology in an essentially non-digital setting. Trying to be loyal to this perspective, I have compiled a theoretical framework in three steps, explicating (1) how digital technology subjects new opportunities, (2) implications on established innovation practices as these opportunities are exercised, and (3) a perspective on how to conceptualized digital products in order to bridge the gap between product innovation and digital innovation.

First, I develop a perspective on materiality and material agency, grounded in the concept of affordances. This perspective emphasizes the performativity of a material. In doing so it turns our attention from physical characteristics, such as weight, plasticity, and hardness, to qualities of an object that allows an individual or organization to perform an action. It is a general lens, widely applicable in innovation, yet seeing materiality through the lens of affordances is particularly rewarding in context of digital innovation. Such innovation environments largely play out in a virtual world of representations, where physical characteristics are pointless. Without shifting focus, from physical properties to subjected possibilities, the notion of materiality is increasingly marginalized in context of digital technology. The concept of affordances is simply a way to give material agency a concrete face in digital innovation.

Second, I outline a model for understanding how novel affordances associated with digital technology transform product innovation. The model is centered on two key barriers that significantly contributes to the character of traditional product innovation, but which are largely not present in digital innovation. Essentially, the dismantling of these two barriers can be traced to two specific digital affordances; *programmability* and *replicability*. The programmability of digital products largely eliminates the rationale behind early binding of functionality to physical artifacts, effectively destroying the taken-for-granted barrier between functional design and physical design. The replicability coming with digital technology, in turn, effectively destroys the barrier between design and production as software essentially is a manifestation of both.

Third, I introduce the concept of *architectural frames*. This concept makes a tool for understanding how designers approach complexity as product innovation is increasingly digitalized. The theoretical model is manifested as two idealized representations of a complex product's architecture. The *hierarchy-of-parts* frame is centered on the physical structure of components and emphasizes decomposition with subsequent aggregation as the core principle for managing complexity. It largely reflects the thinking of Herbert Simon, or at least how his thinking is interpreted in the product innovation domain. The *network-of-patterns* frame is centered on the structure of problems and solutions, rather than the structure of artifacts. It emphasizes generalization with subsequent specialization as a complementary approach to complexity. This frame is derived from the work of Christopher Alexander and his forward-looking, emergent, and open-ended approach to design and architecture. The architectural frames model contributes at several layers. It (1) helps us understand the complexity of architecting digitized products in general. More specifically, it makes an instrument I will apply in my empirical investigation to (2) analyze how the introduction of digital technology makes impact on architectural design in product developing organizations. It helps me explicating how the core ideas of product innovation regimes and digital innovation regimes are represented, combined, and realized in the architecture of digital products.

3.1 Digital Affordances

Material agency is an accepted concept in many scientific disciplines, but it is particularly well discussed in the field of information systems (IS). Generally, the idea of assigning agency to non-human elements is rooted in the observation that technological change emerges from the interplay between artifacts and people. Steel, glass, or plastics have distinct material properties coloring artistic work and innovation. There are apparent reasons behind their respective use in car chassis, windows, and bags. In fact, the material properties of steel, glass, or plastics define how they can be used, making a particular force of change in the complex interplay between people and organizations. As pointed out by Paul Leonardi (2010) "it sounds rather odd to say that digital artifacts – like software – have material properties because people generally think of materials or materiality as physical substances such as wood, steel, and stone". Still, that is exactly what researchers do when they increasingly talk about the materiality of digital artifacts, thereby giving the concept a broader scope than just matter.

On the one hand, materiality plays out in "the world of things and objects" (Pinch 2008). It is in this physical realm, where we hear, see, touch and smell that the concept has an explicit meaning. At the same time, it is

suggested that the physical matter out of which objects are constructed is not all that important when defining materiality of digital artifacts (Leonardi 2010; Pinch 2008). Rather, the adjective "material" seems to refer to some property of the technology that provides users with the capability to perform some action. "The how" seems to be more relevant than "the what" (Westergren 2011). Calling something material emphasizes its *performativity* – the notion that it provides people with capabilities that they can use to accomplish their goals (Pickering 2001).

With this perspective materiality seems to be more closely related to *affordances* than matter. When introducing the notion of affordances James Gibson (1979; 1977) wanted to put attention on qualities of an object (or an environment) that allows an individual to perform an action. On a general level, he defined affordances as "action possibilities". A key argument characterizing Gibson's writings is that "the meaning is observed before the substance and surface, the color and form, are seen as such" (Gibson 1979, p.134). While he originally developed the concept of affordances in context of visual perception, I find this reasoning applicable and highly relevant to the broader context of innovation as well. Physical properties, such as plasticity, elasticity, and hardness, certainly play an important role in everyday engineering as they facilitate and constrain the realization of products. Still, that is not what we see, as human beings, when elaborating an artifact or a material for new purposes. We do not explicitly perceive the hardness of a diamond, but we know it affords us the possibility to cut glass. Using the words of Gibson (1979, p.134), "what we perceive when we look at objects are their affordances, not their [physical] qualities".

Affordances cannot be derived from an object in isolation. Rather, "affordances are properties taken with reference to the observer" (Gibson 1979, p.143). A stroller affords sleeping to the baby and walking to the parent. Thereby, "an affordance is neither an objective property nor a subjective property; or it is both if you like" (Gibson 1979, p.129). It is objective in that its existence does not depend on value, meaning, or interpretation. Yet it is subjective in that an actor is needed as a frame of reference (McGrenere and Ho 2000). Although unfolding in relation to an observer, it is important to note that "the affordance of something does *not change* as the need of the observer change" (Gibson 1979, p.138). Drawing on situated knowledge the user of a navigation system exercises different affordances in different contexts (Svahn 2004; Svahn and Henfridsson 2009). At some point, real-time traffic information allows for re-routing to avoid jam, while another situation calls for precise guidance to reach an unknown destination. However, the system affords routing and guidance whether the user needs it in a given situation or not.

Discussing materiality through the lens of affordances resolves a problem I find increasingly critical as innovation turns digital; digital innovation largely plays out in a virtual world of representations where physical characteristics, such as length, weight, hardness, and plasticity fade into the background. Without shifting focus, from physical properties to subjected possibilities, the notion of materiality is increasingly marginalized in context of digital technology. The concept of affordances is simply a way to give material agency a concrete face in digital innovation.

Let me, as an illustration, contrast adaptive noise cancellation (ANC) technology (cf. Widrow et al. 1975) with traditional solutions, based on insulation materials. ANC uses a set of microphones to detect vibrations in the body of e.g. an aircraft or car. It applies a digital model of the physical vehicle to estimate how these vibrations propagate in the body and eventually transforms into sound waves. Finally, these estimations are used to generate inverted sound waves, transmitted through speakers, cancelling out the noise. This example illustrates how the locus of innovation is shifted, from the physical domain of insulation materials to a representational domain of software and digital models of the physical world. Ideally, ANC affords exactly the same thing as a traditional solution – noise reduction. However, while the latter can be precisely characterized in terms of attenuation[6], the performance of the former follows from the capability to model the propagation of sound in a particular vehicle body.

The ANC example illustrates that digital technology cannot be reduced to material properties, measurable in the physical realm of our everyday lives. Instead, the materiality of digital technology unfolds from the virtual world of representations, where physical properties make no sense. The idea of virtual materials is not new. Photos and texts are everyday examples of such virtual materials, affording things to people that cannot be reduced to physical properties of the particular book or picture. Following the view on virtuality proposed by Deleuze and Guattari (1980), books and pictures are artifacts carrying aspects of reality that it not material, but nonetheless real. Inspired by Bergson, Deleuze later suggested that we can conceive of the virtual as a kind of potentiality that eventually becomes fulfilled in the actual (Deleuze 1988).

I find this Deleuzian angle generally interesting and useful in a conversation about materiality. In fact, it resonates well with the idea of affordances – action possibilities. When resolving an affordance, its potential is fulfilled in

[6] Attenuation is the gradual loss in intensity of any kind of flux through a given medium.

the actual. This underlines that meaning is not created at the time of production, but at the time of consumption and use. We perceive music when playing a record, not when pressing discs. However, Deleuze's view on the virtual is particularly interesting in context of digital technology and digitalization.

As we have seen in section 2, digital technology challenges the temporal sequencing of design and production – the conceptual and the physical – characterizing product innovation. When functionality is mediated by software it is possible to cross this barrier, back and forth, returning to a design state without being constrained by production processes that enforces the product to be assembled as a whole. Consequently, the affordances of a digital product can be readily changed, making it significantly more malleable than a non-digital product. An ANC system originally built for an Airbus can afford noise cancellation in a Boeing aircraft, given new digital models of the aircraft body. In the words of Deleuze, new software opens up for a new potentiality to be fulfilled in the actual, without changing any physical properties of the ANC system.

Clearly, the possibility to reprogram a given product blurs taken-for-granted boundaries between design and production. However, digital technology does not only offer the opportunity to alter affordances of a product by giving it new software. Digital products are often able to revise their affordances autonomously in that they are adaptable. Turning back to the by now familiar ANC example, a new aircraft engine will change the characteristics of noise significantly. Still, the ANC system will be able to afford noise cancellation by using the digital model of the aircraft to adapt its output in response to measured noise characteristics. Similarly, mobile phone infrastructure affords energy saving by continuous adaptation of terminal transmission power to meet the specific needs at every moment. Another example is the auto-break functionality afforded by modern cars, where potential collisions are foreseen by an extrapolation of motion trajectories of surrounding vehicles and pedestrians. Affording noise cancellation for unforeseen sources of interference, power saving in unknown settings, and auto-break in just any traffic context relies on a capability to represent and adapt virtual models of the physical world hosting the system. Therefore, digital technology does not only blur the boundaries between the representational domain of abstractions and the physical domain of concretions – it clearly displaces it. In context of digital technology, the transition between design and production does not mark the scrapping of abstractions. Rather, a key advantage of digital technology is its inherent capability to draw on the representational realm in a use context, subjecting novel affordances in the physical realm available to our senses.

Then, why does digital technology to such a considerable extent redefine the boundaries between the representational and the physical realm? What explains that a digital product serves us, not one potentiality to be fulfilled in the actual, but virtually an indefinite number of potentialities? Seeking the answers on these questions it is necessary to elaborate the fundamental differences between analog and digital.

The phrase analog refers to a specific property of the relation between an original and a copy (cf. Poster 2001, p. 79). The density and distribution of silver salt crystals in a photo *resembles* the characteristics of the original scene. The same applies to the grooves on a vinyl record in relation to the air waves of sound. An analog representation establishes an isomorphism between real world objects and their representation, although manifested in different material forms. As a consequence of this isomorphism, time and space are inevitably inscribed in the representation. Separation of objects in space is reflected in the photograph, and the causality of the music is replicated on the record.

Digital representations do not hold this property of resemblance. The microscopic pattern on a CD (representing zeros and ones) do not in any way look like the sound it stands for. Instead an algorithm relates the zeros and ones to the characteristics of the sound at discrete points in time. Without knowing this algorithm (here embodying sampling time, resolution, coding, etc) the numbers simply make no sense. Consequently, time and space cannot be considered part of the data, but rather of the mechanism generating it. This essential attribute of digital technology holds major implications in that it fundamentally changes the representational form. Algorithm and data are separated, yet deeply intertwined, making sense to the representation only as a whole. Two printed posters of, let us say a car, may differ substantially in terms of color, shading or pattern. Yet, the digital, vectorized representations, used to produce the posters, differ only in terms of a few parameters in a ray tracing algorithm. A digital representation is not only one potentiality, fixed in space-time, but an infinite number of potentialities. *Space and time is essentially decoupled from the representation.*

Clearly, digital technology affords very different things to designers and organizations. Representations decoupled from time and space can take a new form without the physical constraints of analog technologies. They can be moved, duplicated, refined, changed or combined with other representations without “visiting” the physical realm. An increasing number of papers discuss this new materiality of digital technology (cf. Jonsson et al. 2009; Leonardi and Barley 2008; Svahn et al. 2009; Yoo et al. 2010d). Some even propose well defined sets of material properties (cf. Yoo 2010).

In the next section, I elaborate two particular properties of digital technology and their critical impact on product innovation. First, software allows for instant *replication*, without fixed or marginal cost (Benkler 2006; Shapiro and Varian 2000). In an environment where organizing logic, market dynamics, and architectural designs largely have emerged as a response to the efforts and costs of production, this property makes a significant force in changing innovation practices. Since the design of software essentially *is* the product, this unbounded replicability destroys the barrier between design and production, giving a product innovation regime its distinct character.

Second, digital products are *programmable*, which largely detaches functionality from physical artifact. To exercise the affordance programmability organizations have to break with linear models of development, prescribing temporal sequencing of functional design and physical design. Similarly, they have to establish new business models, allowing for recurring sales across the lifetime of a product. As software separates the meaning and functional behavior of a product from the product itself it destroys the barrier between functional design and physical design.

3.2 Programmability and Replicability

The digitalization of tangible products can take different forms. Affordances may be exercised so as to comply with an existing innovation regime and, thereby, avoid many tensions calling for new practices. However, reaching a point of digitalization where a product can be given inherently new functional properties across its lifecycle by changing its software, established innovation practices are confronted with massive pressure for change. Supported by the literature review in section 2 I argue that the digitalization of tangible products destroys, or at least challenges, two key barriers that significantly contributes to the character of traditional product innovation, but which are largely not present in a digital innovation regime.

First, product innovation is characterized by a considerable *barrier between a design and its realization as a physical product*. Largely, we can trace this barrier to the substantial fixed and marginal costs associated with the production of tangible products. The production of cars or airplanes requires massive investments in specialized assets, such as tools, supply chains, and plants. To stay competitive a product developing firm has to depreciate these fixed costs across large volumes of product, enforcing an economy of scale. Similarly, every single unit is associated with a marginal cost of materials, explicitly translating into product price.

As we have seen, yet not expressed in terms of barriers, a product innovation regime is highly colored by production. Products are often architected for

producibility, rather than functional supremacy. Markets are relatively homogeneous as competition plays out across dominant designs. These dominant designs allow firms to fine tune production processes, while offering variation in the details. We can also trace the rationale behind organizational forms to production. The centralized organization structures, behavioral control modes, and linear development processes of a product innovation regime all make sense in light of production. They are not primarily instruments for the delivery of a particular functionality, but ensure that different parts are at the same place at the same time, that they fit together and can be easily assembled into a product.

As a whole, a product developing organization is a highly teleological machinery. There is an up-front plan for design, sourcing of components, systems integration, verification, assembly, and shipping. Competitive advantage emerges out of a capability to make this machinery work in concert. Inevitably, this enforces a temporal sequencing, normally implemented as waterfall models of design and production (Boehm 1976; Royce 1970). Such temporal sequencing raises another fundamental *barrier between functional designs and physical designs of the artifact mediating functionality*. When the teleological machinery has turned its design attention to production aspects, functionality is largely frozen. It is simply very hard and costly to override prescribed processes in order to reconsider once agreed on functional specifications. When reaching the point of production, functionality is inevitably inscribed in the artifact.

To illustrate the barriers and their implications on innovation I propose a simple model (Figure 1), centered on the distinctions between abstractions and observable phenomena, on the one hand, and technology and its context, on the other hand. First, I argue that innovation is about the continuous redefinition of meaning taking place in practice (right half-plane). As artifacts evolve on open markets, they are filled with new meanings, eventually resulting in new practices. Using the language of Wenger (1999), designer reifications, making the foundation of specific products, may be overridden by the participation of users, applying the artifact for new purposes. What is pushed to the market for one particular purpose may be pulled by users for another purpose.

Second, innovation is in general a matter of interplay between practice and our capability to understand practice through models, theories, and various abstractions (lower half-plane). Innovation is to a large extent an act of creativity that relies on our capability to make such abstractions of everyday life and elaborate them for specific purposes. Weather forecasting, as an example, has improved in coherence with our theoretical understanding of meteorological phenomena. To make sense in a product innovation context,

I have described this as the interplay between functional design and practice. Observe that this refers to the domain of problems, independently of technological manifestation. As an illustration navigation functionality can be captured in its details with a human co-driver in mind, rather than an in-car device. Routing and guidance still make perfect sense and can be understood independently of technology.

Third, we find innovation at the borderline between functional design and physical design (left half-plane). On the one hand, designers have to be skilled in the conceptualization of problems in context. At the same time, they need a solid language for how to understand the technology they work with and the opportunities and constraints coming with it. Successful innovation grows from the overall capacity to mangle these perspectives together. Essentially, this entails a genuine capability to rethink both the domain of technology and problems, seeing them from different angles where they make sense together. A one-sided approach may enrich our thinking, but does not translate into innovation. Consider, as an example, the case of teleportation[7]. From a philosophical point of view the idea is highly intriguing and it is relatively straight forward to engage in its functional design. However, when considering its manifestation in contemporary technology it is reduced to a marginal curiosity, simply because we cannot imagine the technology mediating teleporting functionality and even less so envisioning a conceptual design of it. At the level of designers, innovation is a process where the design of tangible artifacts is guided by functional specifications, but at the same time guides the conceptualization of these functions. This interplay injects new ideas in the design process, exposes affordance, and identifies constraints.

Finally, product innovation plays out in the transition from physical design to product (upper half-plane). A blueprint represents a conceptual model of a product, mediating a particular function. However, it is also the schema guiding the assemblage of different parts into a product. Successful innovation emerges from the simultaneous capability to produce artifacts for particular functional purposes and architect products for the specific processes of mass production. Reducing the number of different bolt types, while preserving the functionality of a design offers larger volumes in sourcing bolts and fewer tools in production. Reducing the total number of bolts reduces the total cost and simplifies assembly as the number of operations fall.

[7] Teleportation normally refers to instantaneous transfer of matter over long distances.

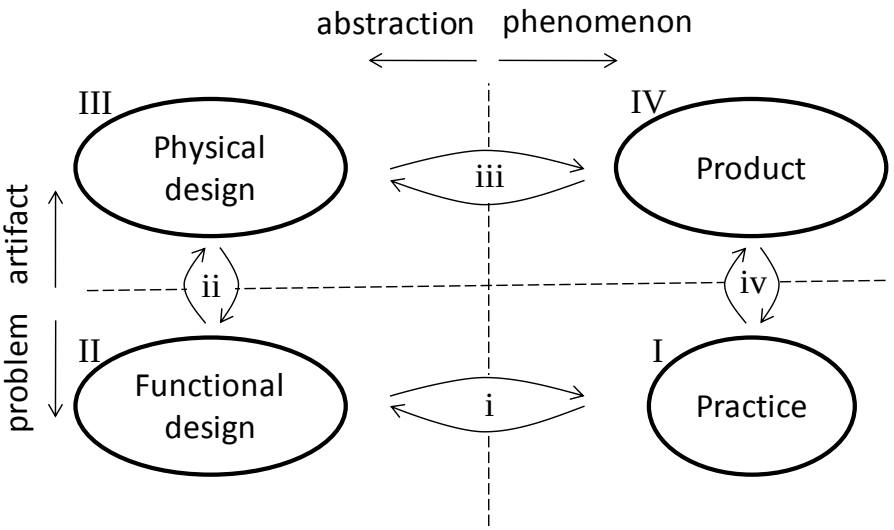

Figure 1. A model of product innovation.

The separation of design and production, on the one hand, and functional design and physical design, on the other, constitute barriers that are highly characterizing for product innovation. However, my attention on these barriers does not primarily serve the purpose of illustrating a product innovation regime, but to make a basis for understanding digitalization of tangible products. In fact, these barriers essentially do not exist in a digital innovation regime. There is certainly fixed cost associated with software. However, this cost derives from design and not production. As we have seen, software affords instant *replication*, without fixed or marginal cost (Benkler 2006; Shapiro and Varian 2000). While the transition between design and production is fundamental to a product innovation regime, it is largely meaningless in digital innovation. Essentially, the design *is* the product. Once a detailed design is in place, there is virtually no time lag before the product can be distributed to users.

Similarly, the stored-program concept separates functional logic from the physical hardware that executes it. Therefore, a physical artifact can perform new functions across its life time if equipped with new instructions or programs. There is certainly a dependency between software-enabled functionality and the hardware executing it. Yet, the temporal sequencing of product innovation, inscribing functionality in physical designs has no correspondence in digital innovation. Digital technology affords *programmability*, largely detaching functional design from the physical design of artifacts.

Indeed, programmability and replicability are properties of digital technology with the potential to radically transform product developing organizations and industries. At the same time transportation, heating, lawn mowing, etc, will remain highly physical values even as cars, heat pumps, and mowers are increasingly digitized. Digital technology will not enforce a digital innovation regime on product developing organizations, but clearly we will see new forms of innovation emerge. How digitalization plays out in established innovation processes is determined by the extent to which firms and industries are prepared to rethink organizational forms, design processes, governance frameworks, business models, etc. As demonstrated in manufacturing over the last two decades, it is certainly possible to benefit from digitization at the level of components, without really changing the logic of innovation. Software and digital technology has proved successful in reducing cost of components as well as improving functionality at the margin. However, until now we have seen relatively few examples where product developing organizations have used the programmability and replicability of digital technology to set up an innovation practice with a genuinely new flavor. In the following section I outline a theoretical framework for how to conceptualize digital products at the boundary between digital innovation and product innovation.

3.3 Architectural Frames

Whether studying biological or artificial systems, progression is about *combination* (Alexander 2002; Arthur 2009; Simon 1996). In nature such combinatorial progression "assumes no teleological mechanism. The complex forms can arise from the simple ones by purely random processes" (Simon 1962, p. 471). Artificial systems are clearly designed. Still, all novel technologies arise by combination of existing technologies (Arthur 2009), whether making incremental or radical impact on practice. Those performing better are selected for future growth and development. Technology simply "bootstraps itself upwards from the few to the many and from the simple to the complex" (p.21).

In order to understand this bootstrapping process, shaping new paths of innovation, we somehow need to appreciate the intricate interplay between humans and technology. As product developing environments are increasingly digitalized, we have to pay attention to the novel affordances of digital technology that, exercised by humans, translate into new opportunities (3.1). We also have to consider the impact of such new affordance on established innovation practices. That is, to what extent these affordances challenge established processes, structures, and logics (3.2). Still, we will not be able to paint a credible portrait of digital product

innovation unless we can explain how designers and managers conceptualize digital products to mindfully combine new and old, digital and non-digital. That is a theoretical perspective explaining how product developing organizations exercise the opportunities of digital technology in reasonable harmony with its legacy.

Organizations manifest their strategies for technological change in product architectures. In contrast to design, which we often view as forward-looking, aiming for the solution of a particular problem or challenge, architecture can be described as retrospective. It represents some kind of guideline and best practice for how to combine and reconfigure existing elements for new purposes. Thereby, the magnitude of the concept becomes visible across generations of designs. Architecture is structure-preserving and structure-enhancing (Alexander 2002), passing sound solutions on from design to design and generation to generation. In connecting historical achievements with future potentialities, architecture is a key instrument for path creation, helping firms to create competitive advantage over time.

As technology "bootstraps itself upwards" (Arthur 2009) it turns increasingly complex. Combination inevitably feeds larger and larger systems, constituted by a growing number of elements and rapidly increasing interaction between elements. In a complex system, "the whole is more than the sum of the parts", meaning that "given the properties of the parts and the laws of their interaction, it is not a trivial matter to infer the properties of the whole" (Simon 1962, p.468).

Clearly, the management of complexity is a critical challenge in architecting products. When designers do not understand how elements of a system form a whole and make sense together they cannot work effectively with each other. In the long run, they cannot mediate the evolution of the system as they do not know the effects of restructuring it. When change in one part of the system is likely to generate unexpected side-effects in another designers will simply be very careful changing working solutions. Without a sound architecture, giving access to key structures of the system while backgrounding subordinate aspects, historical achievements are likely to turn into ballast rather than assets.

The management of complexity is primarily a matter of identifying and exercising appropriate representations. As underlined by Herbert Simon, complexity is not an invariant aspect of technology. Rather, "how complex or simple a structure is depends critically upon the way in which we describe it" (Simon 1996, p. 215). Consequently, a given product can be complex in the eyes of one observer and simple for another. To make a successful link between historical achievements and future potentialities an architecture has

to offer a perspective on a product or system that can be shared among a wider audience. Therefore, at its most general level, *an architecture is constituted by a shared way of thinking*.

As we have seen product innovation and digital innovation represent two radically different forms of architectural thinking. In product innovation change emanates from the center of organizations that exercise formal control to improve functionality and reduce cost of products subject to dominant designs. In this environment architectures are framed to foreground physical structures shared between variants. This framing is grounded in a need to depreciate substantial investments in plants, tools, and processes across a range of planned models and several generations of the product. Thereby, the architecture is an instrument offering *reuse of critical assets*, eventually securing reasonable variability, change, and attractive pricing at the level of end-users.

Digital innovations regimes exercise quite different architectural thinking. In these software-centric environments, innovation takes place in loosely coupled networks where a shared platform makes a tool to orchestrate a variety of heterogeneous knowledge in the harsh competition over attention. The architecture cannot be viewed as the common parts of a range of known products. Rather, it is focused on the structure of functions or problems, offering a collection of best practice tools and inherent support for the *reconfiguration and reuse of existing ideas*. Making a catalyzer for open-ended innovation in ecosystems of rich and heterogeneous knowledge the architecture is a key instrument for firms building generative capability.

Clearly, product innovation and digital innovation represent two different traditions of architectural thinking. To understand complexity in a context where tangible products turn increasingly digital I will now present a theoretical model, centered on the concept of architectural frames. This model is designed to contribute at three different levels. First, it connects the architectural practice of product innovation and digital innovation to two different stream of intellectual thinking on complexity. Second, the model helps explicating the distinctions and differences between the two regimes. Finally, it makes a theoretical tool for understanding how architectural thinking can be combined in digital product innovation.

On a general level, architectural frames are *schemas for thinking about and representing a complex product's architecture*. Thereby, architectural frames can be conceived of as cognitive processes crystallizing as particular ways of managing complexity in the design of products. It is worth noting that this model uses the notion of frames somewhat differently than the seminal works of Bijker (1987), Gioia (1986), and Orlikowski (Orlikowski

and Gash 1994). Here frames do not refer to "built-up repertoire of tacit knowledge" (Gioia 1986, p.56), or a "subset of members' organizational frames that concern the assumptions, expectations, and knowledge they use to understand technology in organizations" (Orlikowski and Gash 1994, p.178). Instead, the substance of respective frame is grounded in the history of intellectual thinking on complexity, rather than in the everyday practice of technology stakeholders. The frames can be seen as invariant and theoretically grounded bases in a space of architectural thinking. Together, the frames span this space and can be combined to inform practice in different phases of digitalization.

The model distinguishes between two frames, the *hierarchy-of-parts* frame based in the thinking of Herbert Simon (Simon 1962; Simon 1973; Simon 1996; Simon 2002) and the *network-of-patterns* frame based in the thinking of Christopher Alexander (Alexander 1964; Alexander 1979; Alexander 1999; Alexander 2002; Alexander et al. 1977). In what follows, I derive and explicate the characteristic features of these frames (summarized in Table 2). Reflecting the thinking of Herbert Simon and Christopher Alexander is, of course, an almost impossible task. It is particularly challenging since they belong to different disciplines and have different points of departure in their respective work. To make reasonable justice to their ideas, yet focus my attention where their lifetime achievements overlap or relate, I will now engage them in a fictitious dialogue (3.3.1) on complexity, structure and change. As far as possible this dialogue is compiled using direct quotes (italic) from a few seminal writings. Drawing on the dialogue I then outline a set of conceptual constructs, characterizing the hierarchy-of-parts frame (3.3.2) and network-of-patterns frame (3.3.3) correspondingly. Rather than giving literal justice to original intellectual sources, these constructs together manifest an ideal frame, "formed by the one-sided accentuation of one or more points of view" (Weber 1949, p. 90). Finally, I examine the relationship between the frames and how they interact during design of digitized products (3.3.4).

Table 2. Core dimensions of architectural frames.

Dimension	Hierarchy-of-Parts	Network-of-Patterns
Complexity	Decomposition-aggregation	Generalization-specialization
Structure	Hierarchy	Network
Element	Part	Pattern
Relation	Interface	Inheritance

3.3.1 A Dialogue between Simon and Alexander

Making a fair portrait of someone's thinking is a challenging task. Trying to contrast the legacy of two great thinkers is even more demanding. To give life to the core ideas of Herbert Simon and Christopher Alexander, yet keep attention on the concept of architecture, I have engaged them in a fictitious dialogue. This dialogue is compiled on the basis of direct quotes, marked in italic. As demonstrated in other settings (cf. Steiner 2009) this approach enables original sources, while at the same time offering the reader a fast track, designed for a particular purpose. To set up such a fast track, this dialogue is moderated to center attention on topics of relevance for this thesis. At the same time it is complicated to extract quotes from a lifetime achievement to make certain points. To avoid a biased view this fictitious dialogue is composed on the basis of a few seminal sources and specifically designed to reflect Simon's and Alexander's views on complexity, structure and change. This said, let us give the floor to the moderator.

[Moderator] Today we are offered an exciting opportunity to plunge into the ideas of two very influential thinkers. Herbert Simon is a Nobel laureate in economics with seminal contributions in several scientific domains, including artificial intelligence, decision-making, organization theory, and complex systems. His work on nearly decomposable systems is generally considered to be the core concept of modularity, being immensely influential in modern mass production. Christopher Alexander is an architect who devoted his career to the de-professionalization of design, trying to restore users at the center of creative processes. His concept of pattern languages was created to empower anyone to design and build at any scale. Being somewhat an outsider in his own discipline, Alexander's work has been highly influential in software industries.

We are here today for something of a debate. It is grounded in the observation that the works of these two gentlemen are sometimes put on par and sometimes described as fundamentally different. The aim of this dialogue is to bring light to the different perspectives from which our guests have derived their frequently cited discussions on complexity, structure, and change.

Let us warm up with a brief discussion on the concept of design. What is design? And what major challenges do you see in contemporary design? Prof Simon?

[Simon] Well, that is a very general question. Let me try to give you a general answer. I use to say that *everyone designs who devises courses of action aimed at changing existing situations into preferred ones*. That makes design a very broad concept. *The intellectual activity that produces*

material artifacts is no different fundamentally from the one that prescribes remedies for a sick patient or the one that devises a new sales plan for a company or a social welfare policy for a state. Design, so construed, is the core of all professional training; it is the principle mark that distinguishes the professions from the sciences (Simon 1996, p. 111).

[Moderator] Prof Alexander?

[Alexander] I think Prof Simon just made a very good point here; Design is everywhere around us. We are all designers. In my work as an architect, I have spent a lot of time studying traditional societies where people largely created their own environment to meet their own particular needs. I think – as a response to your question about challenges – this perspective, to some extent, has disappeared. *In our own time, the production of environment has gone out of the hands of people who use the environment (Alexander 1999).* The feeling that *people have been robbed of their intuitions by specialists* (Alexander 1979, p. 246) has been an ever-present inspiration in my work with patterns and pattern languages. Sorry, let us not drift away from the topic.

[Moderator] No, please. Feel free to give us an introduction to patterns. We have plenty of time.

[Alexander] Well, we originally derived the conceptual foundation of pattern languages from *certain generative schemes that exist in traditional cultures. These generative schemes are sets of instructions which, carried out sequentially, will allow a person or a group of people to create a coherent artifact, beautifully and simply. The number of steps vary: there may be as few as half a dozen steps, or as many as 20 or 50. When the generative scheme is carried out, the results are always different, because the generative scheme always generates structure that starts with the existing context, and creates things which relate directly and specifically to that context. Thus the beautiful organic variety which was commonplace in traditional society could exist because these generative schemes were used by thousands of different people, and allowed people to create houses, or rooms, or windows, unique to their circumstances (Alexander 1999, p. 81).*

[Moderator] So, to some extent, pattern languages empower the people?

[Alexander] Yes, that is way to frame it.

[Moderator] Would you then say that modern man is powerless? Do we generally expect authorities and major corporations to design environments for us?

[Alexander] Well, at least there was a time, not long ago, when *people believed that a town had to be planned by a planner who made a plan or a*

blueprint. It was said that if the order of the town is not created from above, there will just not be an order in the town. And so, even in spite of the most obvious evidence of all the beautiful towns and villages built in traditional societies without master plans, this belief has taken hold, and people have allowed themselves to give up their freedom. However, I think we might witness the pendulum swinging back, to some extent. It is increasingly accepted *that the structure of a town can be woven much more deeply, more intricately, from the interaction of its individual acts of building within a common language, than it can from a blueprint or a master plan - and that indeed, just like your hand, or like the bush outside my window, it is best generated by the interaction of the rules which govern the construction of the parts (Alexander 1979, p. 499).*

[Simon] I like your examples and fully agree on your observations. I just want to add that they are valid far beyond your professional discipline. Over the last century, w*e have become accustomed to the idea that a natural system like the human body or an ecosystem regulates itself.* Following Darwin and his disciples*, we explain the regulation by feedback loops rather than a central planning and directing body. But somehow, untutored intuitions about self-regulation without central direction do not carry over to the artificial systems of human society (Simon 1996, p. 33).*

[Moderator] Now, you both emphasize that design is an emergent phenomenon, something that evolves over time under the influence of different people. How does that resonate with the empowerment of people?

[Simon] I am probably drifting away from your question, but, clearly, a major strength of human kind is our capability to collaborate. That is our capability to create – design if you want – things together. My actions are not independent from yours. Neither are they independent from what people did yesterday. They are not even independent from what people do tomorrow. I think, the point we are trying to make here is that we generally tend to underestimate how both our historical legacy and our expectations on the future influences design. We are rational beings and are expected to act rationally when engaging in design. However, since *the consequences of many actions extend well into the future, correct prediction is essential for objectively rational choice.* That is not always easy to achieve. *In simple cases uncertainty arising from exogenous events can be handled by estimating the probabilities of these events, as insurance companies do - but usually at a cost in computational and information gathering. An alternative is to use feedback to correct for unexpected or incorrectly predicted events. Even if events are imperfectly anticipated and the response to them less than accurate, adaptive systems may remain stable in the face of severe jolts, their feedback controls bringing them back on*

course after each shock that displaces them (Simon 1996, p. 35). Returning to your question, one could of course argue that an adaptive perspective allows for good ideas to be absorbed in the evolution of products or towns, as they come, whether they derive from a firm's internal R&D activities or the man in the street. In that sense, an evolutionary perspective on design is more efficient, simply since it is less likely to reject good ideas, just because they do not match our original expectations and assumptions.

[Moderator] I guess the obvious question then is: How do we create technology that is naturally adaptable?

[Simon] Indeed. How do we build technology which can be changed in its parts without falling apart over time into something useless? To me this is, first and foremost, about complexity and the management of complexity. *Roughly, by a complex system I mean one made up of a large number of parts that have many interactions. In such systems the whole is more than the sum of the parts in the weak but important pragmatic sense that, given the properties of the parts and the laws of their interaction, it is not a trivial matter to infer the properties of the whole (Simon 1996, p. 183-184).*

Looking around in nature as well as our artificial environment, it is striking how often complexity takes the form of hierarchy – *the complex system being composed of subsystems that, in turn, have their own subsystems, and so on (Simon 1962, p. 468).* Hierarchies are interesting for many reasons, but, in particular, it offers very special conditions for evolution. *The time required for the evolution of a complex form from simple elements depends critically on the numbers and distribution of potential intermediate stable forms (Simon 1962, p. 471; Simon 1996, p. 190).* In fact, these *stable intermediate forms exercise a powerful effect on the evolution of complex forms that may be likened to the dramatic effect of catalysts upon reaction rates and steady-state distribution of reaction products in open systems (Simon 1996, p. 192).*

[Moderator] But, what makes these stable intermediate forms of a hierarchy so powerful in change processes? Could you elaborate a bit on that?

[Simon] Let us consider human problem solving, as an illustration. Here, *a partial result that represents recognizable progress toward the goal plays the role of a stable sub-assembly (Simon 1962, p. 472).* If we can approach a problem, piece by piece, we are much more likely to find a solution. When our partial results are not overturned as we approach the next sub-problem, we can benefit effectively from our historical achievements as we interconnect intermediate solutions in a hierarchy, eventually forming a solution for the overall problem. This example underlines that *it is not*

assembly from components, per se, but hierarchic structure produced either by assembly or specialization, that provides the potential for rapid evolution. The claim is that the potential for rapid evolution exists in any complex system that consists of a set of stable subsystems, each operating nearly independently of the detailed processes going on within the other subsystems, hence influenced mainly by the net inputs and outputs of the other subsystems. If the near-decomposability condition is met, the efficiency of one component (hence its contribution to the organism's fitness) does not depend on the detailed structure of other components (Simon 1996, p. 193).

[Moderator] Then, would you say that near decomposability is a fundamental condition for stable intermediate forms to arise?

[Simon] Yes, I would say so, at least in the context of complex systems. *Hierarchies have the property of near decomposability.* In such a nearly decomposable system, *intracomponent linkages are generally stronger than intercomponent linkages. This fact has the effect of separating the high-frequency dynamics of a hierarchy – involving the internal structure of the components – from the low-frequency dynamics involving interaction among components (Simon 1996, p. 204).* Consequently, *in a nearly decomposable system the short-run behavior of each of the component subsystems is approximately independent of the short-run behavior of the other components (Simon 1996, p. 198).* Such components are stable intermediate forms in the sense that they perform relatively independent of other components. Is that a reasonably clear illustration of how hierarchy and stable intermediate forms play out in evolution? One way to see it is that nearly decomposable systems are able of limping along, even if some subsystems are incomplete or damaged. Therefore, *among possible complex forms, hierarchies are the ones that have the time to evolve (Simon 1996, p. 196).*

[Moderator] If we turn to the literature, you are sometimes cited together, to make the same argument. I think we find the most salient example in different writings on modularity. Does that mean you essentially share the same perspective on complexity and the structure of complexity? Prof Alexander?

[Alexander] Well, first of all, I am an architect. I have never claimed broad validity of my theories, beyond this domain. That is done by others. That said, I can see both similarities and differences in our respective work. I certainly approve of Prof Simon's work. You have provided us with a whole range of beautiful theories, allowing us to better understand processes of change. At the same time, we clearly have adopted different perspectives. To

use your words, I have found a lot of my inspiration in the simple observation that we see so **little** of stable intermediate forms in our physical environment. At least in those environments we see as whole and living. I think *it is very puzzling to realize that the 'elements', which seem like elementary building blocks, keep varying, and are different every time that they occur. For among the endless repetition of elements we see almost endless variation. Each church has a slightly different nave, the aisles are different, the west door is different … and in the nave, the various bays are usually different, the individual columns are different; each vault has slightly different ribs; each window has a slightly different tracery and different glass.* Still, we have no problems recognizing a church. I find this truly fascinating. *If the elements are different every time that they occur, evidently, then, it cannot be the elements themselves which are repeating in a building or a town: these so-called elements cannot be the ultimate "atomic" constituents of space. Since every church is different, the so-called element we call "church" is not constant at all. Giving it a name only deepens the puzzle. If every church is different, what is it that remains the same, from church to church that we call 'church'? (Alexander 1979, p. 84-91).* If there are stable intermediate forms, where do we find them?

Let us *look more carefully at the structure of the space from which a building or a town is made, to find out what it really is that is repeating there. We may notice first that over and above the elements, there are relationships between the elements which keep repeating too, just as the elements themselves repeat. Beyond its elements each building is defined by certain patterns of relationships among the elements.* For example, *in a gothic cathedral, the nave is flanked by aisles which run parallel to it. The transept is at right angles to the nave and aisles; the ambulatory is wrapped around the outside of the apse,* and so on. There are countless examples illustrating that our ability to recognize a gothic cathedral is hidden in the arrangement of elements – how they relate to each other – rather than the elements in themselves. *Evidently, then, a large part of the 'structure' of a building or a town consists of patterns of relationships.* Medieval churches as well as cities get their characters from these repeating patterns of relationships (Alexander 1979, p. 85-87).

When we look closer, we realize that these relationships are not extra, but necessary to the elements, indeed a part of them. The aisle is constituted by its relationships to the nave and other elements around it. As an element in isolation it is pointless and empty. *Once we recognize that much of what we think of as an 'element' in fact lies in the pattern of relationships between this thing and the things in the world around it, we then come to the second even greater realization, that the so-called elements is itself nothing but a*

myth, and that indeed, the element itself is not just embedded in a pattern of relationships, but is itself entirely a pattern of relationships, and nothing else. I hope this detour makes some sense to you? That it, to some extent, illustrates my view on evolution and change. Whatever object or system we study, it cannot be described in isolation. To get a grip of the structures repeating themselves over time, we need to capture the relations between systems and context. We need to refocus our minds – tune in for relationships, rather than element. Given this, *the things which seem like elements dissolve, and leave a fabric of relationships behind, which is the stuff that actually repeats itself, and gives the structure to a building or town (Alexander 1979, p. 88-89).*

Finally, to conclude this monologue, *the patterns are not just patterns of relationships, but patterns of relationships among other smaller patterns, which themselves have still other patterns hooking them together - and we see finally, that the world is entirely made of all these interhooking, interlocking nonmaterial patterns (Alexander 1979, p. 91).*

[Moderator] And this is where the pattern language comes in? The structure you refer to is a web of interlinked patterns?

[Alexander] Yes, that is right. Let us focus on a particular pattern, labeled A. Let us say it is the aisles of a gothic church. *If we make a picture of all the patterns which are connected to the pattern A, we see that the pattern A sits at the center of a whole network of patterns, some above it, some below it. Each pattern sits at the center of a similar network. And it is the network of these connections between patterns which creates the language.* It is worth noting that *the links between the patterns are almost as much part of the language as the patterns themselves (Alexander 1979, p. 313-314).* Therefore, *the structure of the language is created by the network of connections among individual patterns: and the language lives, or not, as a totality, to the degree these patterns form a whole (Alexander 1979, p. 305).*

[Moderator] So, this structure – the pattern language – is a knowledge base, guiding, for example, the construction of a gothic cathedral to be characteristic, yet unique?

[Alexander] Yes, if you like. Ultimately, *a person with a pattern language can design any part of the environment. He does not need to be an 'expert'. The expertise is in the language. He can equally well contribute to the planning of a city, design his own house, or remodel a single room, because in each case he knows the relevant patterns, knows how to combine them, and knows how the particular piece he is working on fits into the larger whole. And it is essential that the people shape their surroundings for themselves (Alexander 1979, p. 353-354).* As you probably have noticed this

is a cornerstone in my approach to design and architecture. Wholeness cannot be achieved by remote specialists, detached from the local environments. That is why I would prefer another twist of your original question; I do not think the challenge is to create technology that is naturally adaptable. We do not just want new technology, at an increasing pace of change, do we? We want technology creating wholeness and harmony in our everyday life, right? So, to me a central challenge is how to create such living structure. That is structure, allowing for variety, yet preserving the wholeness across generations of designs. *It turns out that these living structures can only be produced by an unfolding wholeness. That it, there is a condition in which you have space in a certain state. You operate on it through things that I have come to call 'structure-preserving transformations', maintaining the whole at each step, but gradually introducing differentiations one after the other. And if these transformations are truly structure-preserving and structure-enhancing, then you will come out at the end with living structure (Alexander 1999, p. 78-79).*

[Moderator] It strikes me that structure is a central concept in your respective writings, yet you refer to quite different things, do you not? When you talk about structure, Prof Simon, you seem to refer to the configuration of a well defined system, an observable phenomenon. It may be a church, a car, or an organization, yet it is something quite concrete. It is the schema interconnecting the whole and the parts. Some structural forms allow for the system to be changed and manipulated more easily than others. Therefore, seen across generations, the structure makes a basic condition, deciding the pace of change. In contrast, the structures we find in Prof Alexander's pattern languages refer to something quite different. With your discussions in mind, I see it is a map, defining how different solutions – patterns – make sense together. A pattern language does not capture the structure of a particular gothic church, yet it holds the soul of any gothic church. When applying a pattern language in a design process, the designer is gradually carving out a path through the network of interconnected patterns making up the language. It does not reflect the decomposition of a system into parts, but rather how general patterns can form specific solutions in an indefinite number of ways.

[Simon] I think I see the distinction you want to make. In my work, I have been theorizing on *the relation between the structure of a complex system and the time required for it to emerge through evolutionary processes*. Specifically, I have been arguing that *hierarchic systems will evolve far more quickly than non-hierarchic systems of comparable size (Simon 1962, p. 468).* Thereby, the system is at the center of attention in my writings. I

know it is easy to associate 'system' with hard things – cars and airplanes – but, keep in mind; the idea on near decomposability is equally valid for social systems and human problem solving.

[Moderator] Still, the hierarchy of a nearly decomposable system – social or technical – is a structure of a real, concrete system, while a networked pattern language is an abstract structure, used to *generate* such specific instances. They play out at different levels. Am I wrong? Drawing inspiration from nature and biological systems, Prof Simon serves us with a perspective where historical legacy is carried from generation to generation by the system in itself. You, Prof Alexander, serve us a perspective, where legacy is embedded in our collective mental models for how to solve problems.

[Alexander] Well, clearly a building is an instance. Ultimately, each building is unique, tailored for its particular context. Yet, there is a relationship between buildings, unfolding over time. To me it is rather clear that we will never understand this relationship – and thereby not the basic forces of evolution – if we try to dismantle the buildings. The blueprints of every church ever built will not help us understand why they are all churches, yet very different. Rather, we have to understand *that every place is given its character by certain patterns of events that keep on happening there (Alexander 1979, p. 55)*. That is where to look for structure. To me *the world does have a structure just because these patterns of events which repeat themselves are always anchored in the space*. You do certain things in a church. The structures of a church will not unfold, unless you understand why and how people practice their different ceremonies. *I cannot imagine any pattern of events without imagining a place where it is happening. I cannot think of sleeping, without imagining myself sleeping somewhere. Of course, I can imagine myself sleeping in many different kinds of places – but these places all have at least certain physical geometrical characteristics in common (Alexander 1979, p. 69)*. What I mean is that the structure we look for is found at the level of problems and solutions, rather than at the level of physical objects. That is at the level of actions and processes. *What seems at first sight like a static thing is in fact a constant flux of processes (Alexander 1979, p. 356)*. This is where we need to focus our attention. It is the processes we need to understand. And this is where the pattern language comes in. *Every act which helps to shape the buildings and the towns and their activities is governed by the pattern language people share – and governed above all by just that portion of the language which is especially relevant to that especial act (Alexander 1979, p. 358)*.

[Moderator] You might find me obsessed with the idea of distinguishing your work, but I simply find it intriguing that you from certain angles seem

to say the same thing, while from other angles show remarkable difference. One early observation I did when first trying to get familiar with your work was that you, Prof Simon, obviously have found a lot of inspiration in nature and biological evolution, while Prof Alexander, being an architect, is very design oriented. That first led me to believe that hierarchic structures and near decomposability are dominant in change processes where path dependency is strong and short-term, human agency is weak, while patterns languages prosper in design-centric change, characterized by human will. I can see now that this is a very narrow view and essentially incorrect. As we know, modularity is a key concept in product design that has radically transformed industrial productivity.

[Alexander] With respect for your observations, but I think it would be quite misleading to distinguish between us by attributing evolutionism to Prof Simon and design centrism to me. I would argue that we have both adopted an emergence-centric perspective on change. The role model of such a perspective is, of course, nature and the change processes we see in nature.

As a collective, humans are exceptionally successful. Yet, compared to nature we are mediocre designers in some respects. Why? Nature makes use of time. It does not create things in a single act of mindful creativity. At least to this date, *the great complexity of an organic system which is essential to its life cannot be created from above directly; it can only be generated indirectly (Alexander 1979, p. 162)*. In my work, I have argued that the same basic mechanisms apply to artificial system, meaning that *towns cannot be made, but only generated, indirectly, by the ordinary actions of the people, just as a flower cannot be made, but only generated from the seed (Alexander 1979, p. 162)*. I truly believe that this applies to any complex system which we would see as living and whole. This is actually the theoretical underpinning of our work with pattern languages – *we wanted to generate the environment indirectly, just as biological organisms are generated, indirectly, by a genetic code (Alexander 1999, p. 73)*.

[Simon] Listening to Prof Alexander I can only agree. At this level we have a very similar approach to change. I have seldom used the notions of indirect generation or generativity, but I have often said that *among possible complex forms, hierarchies are the ones that have the time to evolve (Simon 1996, p. 196)*. Near decomposability, unfolding from hierarchy and stable subassemblies, simply makes a powerful and efficient bridge between generations. Nature clearly shows that this architectural feature *accelerates the rate at which the fitness of organisms possessing it increases over time through the standard processes of genome change and natural selection (Simon 2002, p. 588)*. It applies to artificial systems as well. Using the language of Prof Alexander, near decomposability allows a designer to

generate new systems from proven building blocks, working together in a proven configuration, without being exposed to its full complexity.

[Moderator] You certainly do not make life easy for me, trying to find areas of disagreement! Let me try a different angle, before closing down this very rewarding session. You are both turning to biological evolution and genetics when demonstrating your views on change. I know it is a metaphoric use, yet it illustrates well the crucial role of an unbroken, common thread across generations. That is what genes do; they carry a legacy from generation to generation. Progression is about exploiting this legacy in a beneficial way, yet not being a hostage of it. Although you largely share the same view on the overall mechanisms behind change processes, I would say that you suggest quite different manifestations of this thread.

[Simon] Could you elaborate that a bit more? I am not sure I follow.

[Moderator] Well, as you have both pointed out, time is a key aspect when it comes to complexity and the management of complexity. Modular architectures and pattern languages are containers of knowledge, experience, insights, wisdom – whatever we want call it – sparing us the full complexity of a system up-front. They are instruments interconnecting generations of designs in a particular way. Thereby, they define a particular perspective on path dependency. It just strikes me that you promote rather different structures and, thereby, different models for understanding path dependency.

Let us take the automotive industry as an example. Here, modularity is a central concept, present from the first sketches, through design and development, to the final car. In practice, this means that the entire organization is shaped to fit a representation of the final product – the *outcome* of the design process. In these modular design practices it is primarily the decomposition of the car that is documented, not the rationale behind that decomposition. The function of a particular component is largely described in terms of interfaces, telling us how it relates to other components. When I listen to you, Prof Simon, I can see that this is a key point, explaining why nearly decomposable systems evolve faster; when we have learnt how to produce a particular component it makes a stable subassembly. As long as we can continue producing this subassembly we can focus our full attention on other issues. The history behind it is, to some extent, expendable.

[Simon] Indeed, that is certainly true.

[Moderator] As a result, these stable subassemblies cause rather strong path dependency. If you like, historical legacy is carried from generation to generation by the artifacts – components and car. A designer responsible for

the navigation system knows the decomposition of his components in deep detail, but he knows very little about the original problems facing the original designers. In this context, it is a rather bold decision to make significant changes. With a modular system designers can predict how changes translate through the hierarchic system, i.e. how other components are influenced, but blueprints and component specification give little support in understanding how these changes play out at a social or practical level.

[Simon] OK, I think I see where you are going.

[Moderator] Prof Alexander argues that wholeness cannot be achieved by standard components. To create living structure every building has to be designed uniquely in harmony with its local context. Therefore, beyond an elementary level, you see few stable subassemblies, shared between all buildings of a class. However, you underline that there are standardized *processes*, able to generate such coherent and morally sound buildings. Pattern languages, whether formalized or tacit, are the structures passing knowledge about these processes on from one generation of carpenter or architects to another. Seeking inspiration in the design of a new house it is a waste of time trying to decompose an existing building in its different physical parts. Instead, we have to see through the physical structures to understand the problems addressed and solutions offered by a particular design. This way, path dependency manifests itself as structure-preserving and structure-enhancing transformations, generating buildings which offer solutions to generic, shared human needs, yet in harmony with local context.

[Simon] You certainly have a point in that we discuss structure at different levels. I am just not sure these discussions are incompatible.

[Moderator] Oh, that is not what I am suggesting. I just seek to clarify the distinctions in your respective writings.

[Simon] OK. Let me then suggest that these related, but still slightly different discussions follow from different initial questions. Maybe it is that simple? After all, I have studied *the structure of a complex system and the* ***time*** *required for it to emerge through evolutionary processes (Simon 1962, p. 468).* Clearly, this is not what has motivated Prof Alexander in his work with pattern languages. You have sought the evolutionary roots of wholeness and living structure.

[Alexander] True.

[Simon] I have used nature as a source of inspiration to understand the basic premises of change in artificial system. I have studied *pace* of change, but I have not engaged in a discussion on whether change is appropriate or desirable. That is, on the other hand, what your pattern language is all about,

Prof Alexander. You have used nature as a role-model to understand how sound designs, offering value to people in different context, can be preserved and enhanced over time.

[Moderator] That was an excellent closing of this dialogue. Rather than seeing Simon's nearly decomposable hierarchy and Alexander's pattern language as competing views we can see them as complementary approaches to the complexity of artificial systems. That is indeed a very encouraging perspective. I think we all have learned a lot and will walk out of this room with a lot of inspiration. Thank you all for coming here today.

3.3.2 Hierarchy-of-Parts

In his seminal work on complex systems, Herbert Simon has studied "the relation between the structure of a complex system and the time required for it to emerge through evolutionary processes" (Simon 1962, p. 468). Specifically, he argued that "hierarchic systems will evolve far more quickly than non-hierarchic systems of comparable size". The "potential for rapid evolution", coming with such hierarchic systems can be explained in that it "consists of a set of stable subsystems, each operating nearly independently of the detailed processes going on within the other subsystems, hence influenced mainly by the net inputs and outputs of the other subsystems" (Simon 1996, p. 193). If this so called "near-decomposability condition is met, the efficiency of one component (hence its contribution to the organism's fitness) does not depend on the detailed structure of other components".

Simon's work is centered on the structure of systems. It identifies such structures as the link between generations of the system. The stable subassemblies of a hierarchy allows for historical efforts and investments to be efficiently reused and managed across generations. In the moment an in-car navigation system or a kidney proves functional, its genesis turns irrelevant. As long as it remains functional, the subassembly can be pushed to the background, and forces of change can be directed elsewhere.

As suggested by the name, the hierarchy-of-parts frame draws on this Simonian thinking by prescribing ***hierarchy*** as the core structure of design. Such hierarchic structure emerge as designers recurrently practice ***decomposition*** and ***aggregation*** in the design of products. With the decomposition of products into ***parts*** designers seek to establish and preserve stable, loosely coupled subassemblies. Such stable subassemblies hide complexity and delivers functionality through well defined ***interfaces***. In the aggregation of parts into products designers take the opposite perspective and seek new configurations of parts. This is a bootstrapping process, where simpler structures are recursively combined over time to

form increasingly complex ones. This process allows for new value to emerge, yet while preserving parts and configurations of parts to take full advantage of design legacy.

Thereby, the hierarchy-of-parts frame makes a distinct architectural approach to path dependency. On a general level, when decomposing a product, designers seek to maximize the pay-off on investments. A smart decomposition makes an asset for coming generations of the product. It is an investment in the future. Aggregation, on the other hand, seeks to leverage existing designs for new purposes. It is a way to exercise opportunities in design legacy. Drawing on this I suggest following definition of the hierarchy-of-parts frame:

> **Definition:** *The hierarchy-of-parts frame views a product's architecture as a hierarchy of loosely coupled parts, emerging from the recursive application of decomposition and aggregation as designers seek to connect historical achievements with future potentialities.*

As a scheme for allocating product functionality to physical components (Ulrich 1995, p. 419), the hierarchy-of-parts frame relies on early binding of functionality to physical configurations. The advantage of relying on this principle during the design process is that it encourages designers to develop architectures that support subsequent production. Some flexibility is achieved during the design process by assigning design parameters that can take on different values within a particular range to individual parts (Baldwin and Clark 2000). This allows the designer to distribute functionality to specific physical components while still defining generic conceptions of a product. Later, the designer can then freeze a specific solution by selecting particular values for each design parameter (Iansiti 1995). These considerations motivate the following:

> *The hierarchy-of-parts frame helps designers of digitized products converge towards a physical product through separation of concerns and early binding of functionality to physical components.*

3.3.3 Network-of-Patterns

Christopher Alexander has devoted a long and successful career "trying to learn how to produce living structure in the world". From the perspective of an architect "that means towns, streets, buildings, rooms, gardens, places which are themselves living or alive" (Alexander 1999, p. 73). Two aspects seem particularly important to Alexander when seeking to explain the formation of living structure. First, he underlines that we cannot create

coherent, morally sound objects, unless we do it in deep harmony with the local environment. "The characteristics of any good environment is that every part of it is extremely highly adapted to its particularities" (Alexander 1999, p. 74). Therefore, in environments which we perceive as whole and living, many of "the 'elements', which seem like elementary building blocks, keep varying, and are different every time that they occur" (Alexander 1979, p. 84). To produce such living structure, "it is essential that the people do shape their surroundings for themselves" (cf. Alexander 1979, p. 74). Clearly, this breaks with the mass-production/mass-consumption logic of our contemporary world. In our time "the production of environment has gone out of the hands of people who use the environment" (Alexander 1999, p. 74). Alexander even argues "people have been robbed of their intuitions by specialists" (Alexander 1979, p. 246).

Second, Alexander states that "that life cannot be made, but only generated by a process" (Alexander 1979, p. 74). He argues that it is highly misleading to "think of works of art as 'creations' conceived in the minds of their creators" (Alexander 1979, p. 160). The great complexity characterizing living structure "cannot be created from above directly; it can only be generated indirectly" (Alexander 1979, p. 162). Following this line of reasoning, the main asset of a successful artist or engineer is not the capability to make detailed up-front constructions, but rather the application of generic and relatively simple skills in a responsive dialogue with materials and context. "The brush stroke becomes beautiful, when it is visible only as the end product of a process – when the force of the process takes over the cramped will of the maker" (Alexander 1979, p. 160). This is when "the maker lets go of his will, and lets the process take over".

The legacy of Christopher Alexander is embodied by the concepts of patterns and pattern languages. A pattern "is a three-part rule, which expresses a relation between a certain context, a problem, and a solution" (Alexander 1979, p. 247). As an example, the star-shaped torx pattern for screw heads addresses the problem of cam-out[8] in assembly processes and offers a more precise application of torque and increasing life-span of tools. It is widely adopted in manufacturing settings, but largely inadequate in consumer contexts, simply because the tool is not part of standard household equipment.

> *"As an element in the world, each pattern is a relationship between a certain context, a certain system of forces which*

[8] Cam-out is the process by which a screwdriver slips out of the head of a screw, once the torque required to turn the screw exceeds a certain limit.

> *occurs repeatedly in that context, and a certain spatial configuration which allows these forces to resolve themselves. As an element of language, a pattern is an instruction, which shows how this spatial configuration can be used, over and over again, to resolve the given system of forces, wherever the context makes it relevant. The pattern is, in short, at the same time a thing, which happens in the world, and the rule which tells us how to create that thing, and when we must create it. It is both a process and a thing; both a description of a thing which is alive, and a description of the process which will generate that thing" (Alexander 1979, p. 247).*

As indicated, "the structure of a pattern language is created by the fact that individual patterns are not isolated" (Alexander 1979, p. 311). The torx pattern comes with significant material strain, stressing driver bits. Therefore, it relies on subordinate patterns, solving the problem of material damage in this particular context. When realized, such a subordinate pattern may be manifested as an appropriate alloy. Consequently, "each pattern then, depends both on the smaller patterns it contains, and on the larger patterns within which it is contained" (Alexander 1979, p. 312). In fact, "each pattern sits at the center of a network of connections which connect it to certain other patterns that help to complete it" (Alexander 1979, p. 313). "It is the network of these connections between patterns which creates the language" (Alexander 1979, p. 313).

Clearly, a pattern language can be viewed from different perspectives. On the one hand, it gives an alternative angle on "some of the physical structures that make the environment nurturing for human beings" (Alexander 1999, p. 73). A pattern langue helps us capture and describe the character of a place "by certain patterns of events that keep on happening there (Alexander 1979, p. 55). It is a formal representation, telling us why a church is a church, even though we cannot find two instances looking the same. However, a pattern language can also be viewed as a way to describe sound design practices in a particular field. Representing key ingredients of "living structure" a pattern language will act as a sifter in a sandbox as evolution gradually reinforces sound ideas in a series of "structure-preserving and structure-enhancing transformations" (Alexander 1999, p.79). The language is a generative scheme of instructions which, "carried out sequentially, will allow a person or a group of people to create a coherent artifact, beautifully and simply" (Alexander 1999, p. 81). Ultimately, "a person with a pattern language can design any part of the environment. He does not need to be an 'expert'. The expertise is in the language" (Alexander 1979, p. 353). Thereby, a pattern language makes a link, interconnecting contemporary design activities with yesterday's experiences and efforts. Together a network of patterns, constituting a pattern language, allows for historical problem solving to be

reused and managed by laymen as they design artifacts adapted to their own local environment, unique in time and space.

As suggested by the name, the network-of-patterns frame draws on this Alexandrian thinking by prescribing ***network*** as the core structure of design. That is a network spanning a problem-solution space, rather than the structure of a particular artifact. Such network structure emerges as designers recurrently practices ***generalization*** and ***specialization*** in the design of products. Generalization is a way to manage complexity, where designers seek increasingly generic representations of the functionality associated with an artifact. These representations, or ***patterns***, are distinct solutions for particular problems, defined by a given context. In the process of generalization patterns are repeatedly disassembled into increasingly generic elements, relating to each other through ***inheritance***. We can view inheritance as a barrier, hiding complexity as it separates the more general aspects of a solution from the more specific. Thereby, the resulting web of patterns makes a map of a problem space, simultaneously offering different levels of granularity. Exercising specialization designers take a bottoms-up perspective, seeking to extend the application of generic patterns by reusing them for new purposes. While generalization largely is a matter of increasing granularity, this process extends the problem space in that it generates new specialized patterns from existing, more general ones. It is the process of specialization that produces network structure. While generalization largely generates hierarchy, as patterns are gradually broken down, the multiple reuse of patterns produces many-to-many relationships, forming network structure.

The network-of-patterns frame makes a distinct architectural approach to path dependency. When exercising generalization, designers seek to identify and describe principal elements[9] of a given function. This time-consuming work can seldom be motivated in context of an isolated product, mediating well defined functionality. Rather, the increasingly generic patterns emerging from generalization of high-level functionality has to be viewed as an asset for coming generations. To a significant extent, generalization is an investment in the future. Largely, these investments are exercised in the act of specialization. This is when designers reuse general patterns to shape new specialized patterns, resulting in new functions, products, and, eventually, revenue. Specialization is about exercising the embedded opportunities in design legacy. In this vein, specialization is associated with generativity. Drawing on this I suggest following definition of the network-of-patterns frame:

[9] Compare principal component analysis in mathematics.

> **Definition:** *The network-of-patterns frame views a product's architecture as a network of interrelated patterns, representing functionality at different levels of granularity. This network emerges from the iterative application of generalization and specialization as designers seek to connect historical achievements with future potentialities.*

As a scheme for maintaining functional fit between a product and its use environment, the network-of-patterns frame relies on delayed, or fluid, binding of functionality to physical configurations. The advantage of relying on this frame during the design process is that it encourages designers to develop architectures that support "living structures" or generativity (Alexander 1999; Zittrain 2006). Flexibility is achieved by adopting a pattern language that allows connections to other patterns specified by other designers, making the space of possible solutions profound over time. These considerations motivate the following:

> *The network-of-patterns frame helps designers of digitized products share design ideas through general solution patterns and delayed binding of functionality to physical components.*

3.3.4 Interaction between Frames

The hierarchy-of-parts frame and network-of-patterns frame are not disparate or incompatible perspectives. To some extent all designers have to shape structures allowing them to reuse pre-existing components as products evolve. Modern cars apply standard engines across variants and generations. Ancient villagers used the same standard bricks, from the local brickyard, to build all the different houses in a village. Although playing out at completely different levels of abstraction and complexity engines and bricks are both examples of stable physical subassemblies, possible to combine with other parts, given strict compliance with some overall principles. Similarly, all designers have to shape structures allowing them to reuse the knowledge embedded in existing processes for new purposes. The star-shaped torx pattern for screw heads has been adopted to solve a countless number of problems and has improved assembly and reliability of cars substantially. Once again turning to ancient builders, the diffusions of half-timbering technique[10] allowed 16th century builders to improve strength, material consumption, construction time, and insulation of houses. Therefore I suggest that:

[10] Half-timbering – or timber framing - is the method of creating structures using heavy timbers jointed by pegged mortise and tenon joints (Wikipedia).

> *The hierarchy-of-parts frame and the network-of-patterns frame are both represented when ever organizations architect digitized products.*

Although operating on different levels, the two frames complement each other and contribute in distinct ways to reduce design complexity. The hierarchy-of-parts frame creates concepts and structures centering on the standardization and reuse of components, allowing for streamlined production of related, yet differentiated artifacts. The network-of-pattern frame creates concepts and structures zooming in on the standardization and reuse of solution patterns, allowing for novel functionality to emerge from the knowledge embedded in existing processes. Therefore:

> *The hierarchy-of-parts frame and the network-of-patterns frame offer complementary schemes to architect digitized products.*

As tangible products are digitized, the network-of-patterns frame is increasingly relevant in architectural practices. It is not just anecdotal evidence, such as the immense popularity of Alexander's pattern theory in the software engineering community, suggesting that product developing firms have to complement traditional approaches. This increasing relevance can be traced to the generative capability of digital technology, where programmability enables artifacts to perform new functions after production and replicability support inexpensive production. With its focus on general solutions to problems, the network-of-pattern frame leverages these capabilities of digital technology, concentrating on the product's functionality rather than its production. In this regard, the two frames complement each other, enabling the product-developing firm to exploit its tangible products as platforms for a recurring digital business. This seemingly minor change opens up for a completely different model, where revenue is distributed across the lifecycle of the product.

In short, it can be argued that firms need to stay focused on the hierarchy-of-parts frame to produce cheap, powerful, and scalable hardware, creating significant installed base. At the same time, they have to cater for the network-of-patterns frame to make this hardware an attractive, generative platform, representing unlimited opportunities in the eyes of external actors. The network-of-patterns view is essential to mobilize external creativity and innovation, while the hierarchy-of-parts perspective allows for reasonable control over the innovation process and appropriation of some of the value created outside the boundary of the firm. Therefore:

> *By exercising the network-of-patterns frame when architecting digital products organizations build capability to*

> *leverage the generative capability of digital technology across the entire product lifecycle.*
>
> *By exercising the hierarchy-of-parts frame when architecting digital products organizations reinforce control over hardware, in turn, allowing them to appropriate value from largely uncoordinated innovation processes.*

Although the two architectural frames are represented in any design process, there is no doubt that the emphasis differ with the context. The ideas behind near decomposability have transformed manufacturing industries dramatically. Cars, airplanes, and home appliances are inherently modular, using near decomposability to create variety and rapid pace of change. At the same time patterns and pattern languages have influenced the software industry significantly over the last two decades. Today software designers find themselves spending more time on logical modeling of problems than on coding. How can we explain this obvious difference? How do product-developing organizations, with stakes in hardware as well as software, combine frames?

Addressing this question I suggest that we focus our attention on the barrier between a design and its realization as a physical product, derived in section 3.2. Clearly, the production of tangible products is associated with substantial fixed and marginal costs, while software products are not. The production of cars or airplanes requires massive investments in specialized assets, such as tools, supply chains, and plants. To stay competitive a product developing firm has to depreciate these costs across large volumes of the product, enforcing an economy of scale. Relying on the hierarchy-of-parts frame, product developing firms bridge the barrier between design and production by a shared product structure, allowing specialized assets to be reused across variants and generations of products. One can argue that *physical products tend to be architected for producibility*.

If we think about it, this reasoning applies also to biological systems, so frequently used as role models by Herbert Simon. Nature has spent billions of years shaping sophisticated production machineries able to form living organisms out of genetic blueprints. These specialized production assets, shaped since the beginning of time at earth, are invaluable. The hierarchic structure we find in living organisms allows for these assets to be reused, yet opens up for adaptation at the level of parts. Any change in nature that renders reproduction mechanisms useless is destined to become a dead end in evolution. Living organisms are also primarily architected for producibility.

In software engineering the transition between design and production is relatively uninteresting. To some extent, we can argue that the design *is* the product. There are few specialized assets involved in the production, enforcing a shared structure across variants and generations. Therefore, the structure of a software product is not colored by production. There is simply little pressure for uniformity in the realization of a software product. Instead, competitive advantage emerges from the capability to reuse general ideas of a design to constantly spinoff new products to the market. We can argue that *software products tend to be architected for generativity.*

If we turn to the many examples from traditional societies, used by Christopher Alexander in his writings, we will find that these settings are characterized by moderate barriers between design and production as well. Alexander keeps coming back to this setting since it is distinguished by a layman practice, where buildings were designed and constructed by the man at the street. It was not conditioned by expensive machines, exotic materials, or extreme skills. A house was reasonably associated with significant cost, but it was built using highly generic tools, such as saw, hammer, screwdriver, etc. It did not require specialized assets, enforcing uniformity between different buildings.

In summary, design-production barriers determine the locus of architectural frames. Innovation environments characterized by substantial barriers – involving a lot of costly, specialized assets – view product architecture as a hierarchy of parts. In contrast, settings with low design-production barriers favor practices centered on the network-of patterns frame. Therefore:

> *Inherent design-production barriers determine how the hierarchy-of-parts frame and network-of-patterns frame can be combined when architecting digital products*
>
> *Seeking generative capability, organizations need to reduce design-production barriers, allowing for an increased emphasis on the network-of-patterns frame when architecting digital products.*

4 Methodology

Digitalization is an ongoing, contemporary phenomenon. History gives limited advice in understanding how digital technology irreversibly changes our lives. Moreover, we have limited opportunities to test and elaborate in controlled environments. Digitalization is an emergent phenomenon which unfolds in a complex interplay between social structures and technology. Often we have no option but to study it in real-world settings and in real-time. Therefore, in researching how "product developing firms architect digitized products to leverage the generative capability of IT" (p. 57) I have used qualitative in-depth case study as a basis of inquiry (Eisenhardt 1989; George and Bennett 2005; Gerring 2007). The case study "is preferred in examining contemporary events, but when the relevant behaviors cannot be manipulated" (Yin 2003, p. 7). Further, the case study offers a unique strength in its "ability to deal with a full variety of evidence – documents, artifacts, interviews, and observations" (p. 8).

As discussed in section 3, the philosophical underpinning of this research has a critical realist stance (Archer et al. 1998; Bhaskar 1998; Easton 2010; Sayer 1992). It adopts a realist view in that it assumes observable, material properties of technology which exist independently from observers. Such properties translate into affordances, allowing designers, managers, and organizations to perform different actions. At the same time, it recognizes that our knowledge about the world is socially constructed. How people exercise affordances cannot be understood by studying technology in isolation. Instead, the assimilation of a new technology in an established practice has to be viewed in light of existing, socially constructed frames of

reference. With such a critical realist outlook this thesis approaches digital product innovation by combining realist ontology with interpretive epistemology (Archer et al. 1998). While this philosophical perspective resonates well with my personal character it is also suggested by several researchers as an important method to enrich IS research in general (Bygstad 2010; Mingers 2004; Smith 2006) and longitudinal case studies in particular (Dobson 2001; Easton 2010).

Seeking to explain how product developing organization build generative capability the case study was tuned to uncover new cognitive frames for thinking about and representing digital technology. In that sense the research was interpretive (Klein and Myers 1999; Walsham 1993) in nature. At the same time, the critical realist outlook suggests that some ways of conceptualizing digital technology make more accurate representations of external objects than others. The assumption is simply that some schemas for thinking about and representing a complex product's architecture are better off in delivering generative capability than others. Seeking to identify and represent such schemas the case study was specifically tuned to capture the interplay between actors and technology over time (Langley 1999; Markus and Robey 1988). This framing was supported in that the case study was longitudinal in its character, spanning a timeframe of approximately one decade. Even more important, it covered four different projects in four different phases of digitalization. The four projects were centered on different technologies, making it an embedded case study (Scholz and Tietje 2002; Yin 2003), involving different units of analysis within the same case.

The case study was conducted at CarCorp, a product developing firm in the automotive industry that develops, produces, markets, and sells cars on the global market. CarCorp is known for its eccentric designs and innovative features, resulting in a small but devoted customer base primarily in Europe and the US. At the turn of the century, CarCorp became fully owned by GlobalCarCorp, a major global automaker. Seeking scale advantages GlobalCarCorp enforced shared product platforms across its different brands, causing CarCorp's product innovation to be tightly integrated with its parent's global organization. In this integration CarCorp was given a leading role in designing in-car infotainment systems for the different brands in GlobalCarCorp's palette of global automakers.

4.1 Data Collection

The empirical study, used as a basis for this thesis, was not originally framed towards architecture. Rather, the process started as a business model and value chain analysis of nomadic device integration. In a collaborative project, involving three major automotive manufacturers, consultant firms, and

content suppliers, the first phase of this research initially aimed to identify and describe complementary business strategies for an infotainment system based on nomadic device integration. As the study unfolded, however, it became increasingly clear that each identified strategy was tightly connected to a distinct architectural view on the system. Launching a new research project with CarCorp, explicitly framed towards architecture, the study was reoriented to investigate the relationship between design flexibility and architecture. As CarCorp gradually adopted an open perspective on infotainment, encouraging external development of applications, the study focus was slightly shifted again. Rather than centering on design flexibility it unfolded as a research study of architectural implications on generativity.

Eventually, the study spanned four different CarCorp projects, making up four embedded cases (Figure 2). Each project focused on a distinct technology; MOST, SoftCluster, nomadic device integration, and the Android platform. Together, these projects extend over a period of approximately one decade. Data was collected in three phases. The first phase (January 2007 – November 2007) centered on contemporary nomadic device integration and its historical roots at CarCorp. The primary data sources were interviews (29) and project meetings (16). However, the phase also included focus groups (4) at the three automakers and one content supplier. The second data collection phase (October 2007 – December 2008) was part of a smaller, architecture-centric research initiative, largely implemented on site at CarCorp. Over the most intensive period I spent 2 full days a week making participant observations as an embedded researcher in different infotainment projects. Being on site at CarCorp opened up for access to a wide range of documents (37), such as technical specifications and project descriptions, but also gave access to internal project meetings (47). As a complement to these in-practice data sources the study included a range of interviews (27) and a few focus groups (5). Finally, the third phase (April 2008 – March 2011) was carried out in tight connection with CarCorp's open platform initiative. In particular, this phase was framed towards the intricate interplay between ecosystems and platforms. The primary data source was project meetings (39), but the phase also included interviews (11), documents (6), and one focus group.

All together, the three data collection phases embraced five different data sources; interviews, project meetings, focus groups, participant observations, and documents. First, the study includes 67 interviews. The scope of these interviews amounted to a total recorded time of 76 hours and a transcription word count of 697.170 words. All in all, 102 project meetings were attended at an estimated total time of 316 hours. That makes a mean length of approximately three hours, although they varied from less than an hour to a

full working day (σ=1 hours 55 min). Moreover, 10 different focus groups were completed, with a total time of more than 15 hours. Finally, the three phases included approximately 250 hours of on-site participant observation and 37 selected documents.

In distinguishing between different styles of researcher involvement in case studies, the data of this thesis was collected by an "involved researcher" rather than an "outside researcher" (Walsham 2006). Although close involvement comes with certain risks, I find it highly unlikely that this research could have been conducted on the basis of a more distanced approached. Automakers are considerable organizations. Technology is developed in a collaborative manner, on a global scene. Tasks are distributed, not only across buildings and departments, but across nations and continents. In such a setting it is very difficult to study the interplay between technology and people as an outsider. There is no local, well defined practice to examine. Therefore, to make sense of events and actions in a local project team or design group it is of outmost importance to understand the bigger picture in which these people exist. Unless the researcher is able to adopt their perspective, in-depth access to that picture will slip between his fingers.

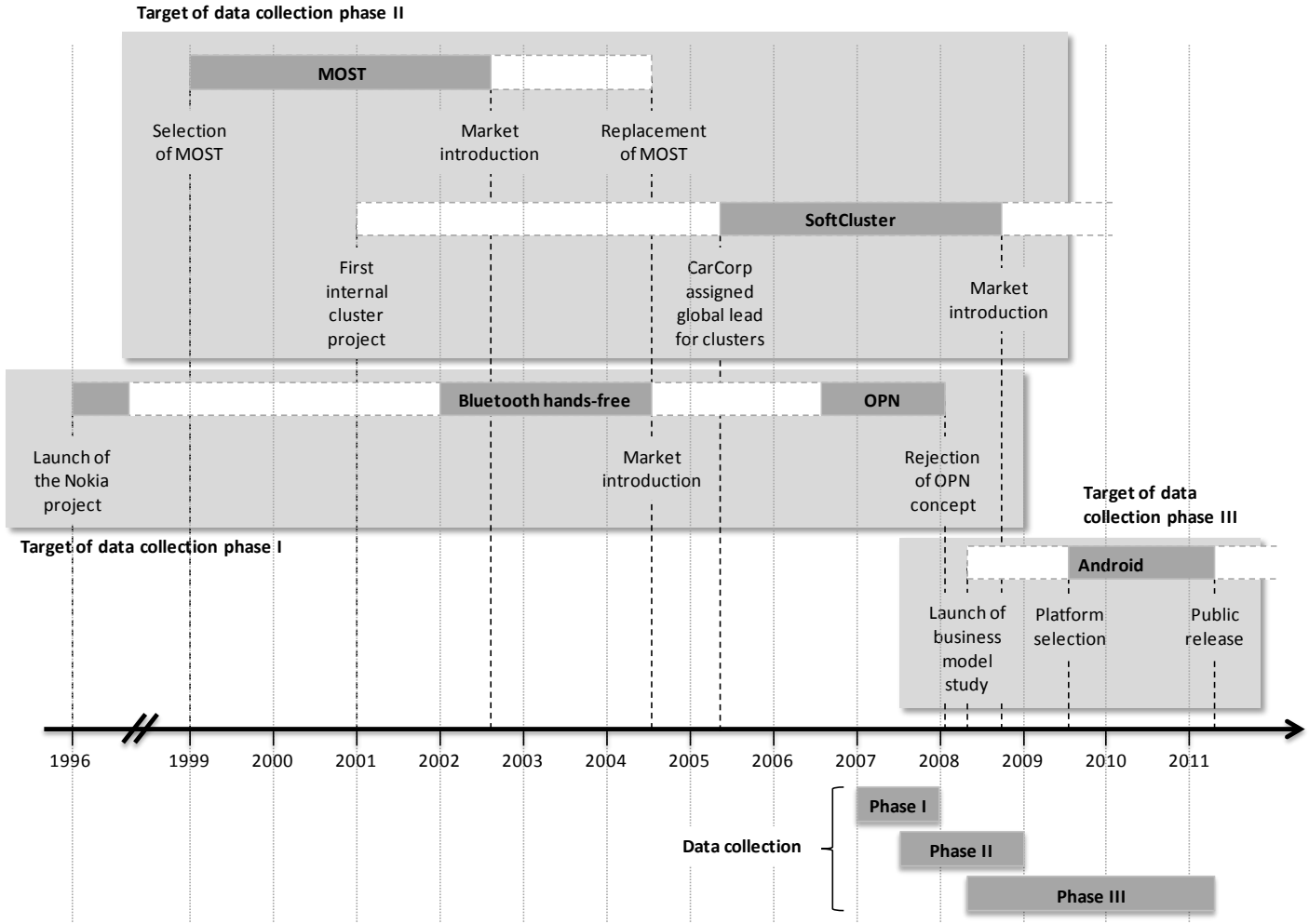

Figure 2. Timeline of empirical setting.

Table 3. Data collection statistics.

Data Source	Ph. I	Ph. II	Ph. III	Total	Comments
Interviews	29	27	11	67	μ: 1:08:33 σ: 0:23:15 word count: 697.170
Focus groups	4	5	1	10	μ: 1:40:25 σ: 0:34:08
Project meetings	16	47	39	102	μ: 3:06:08 σ: 1:55:18
Participant observation (hours)		248		248	
Documents		37	6	43	Project descriptions Technical specifications Sales forecasts Press releases

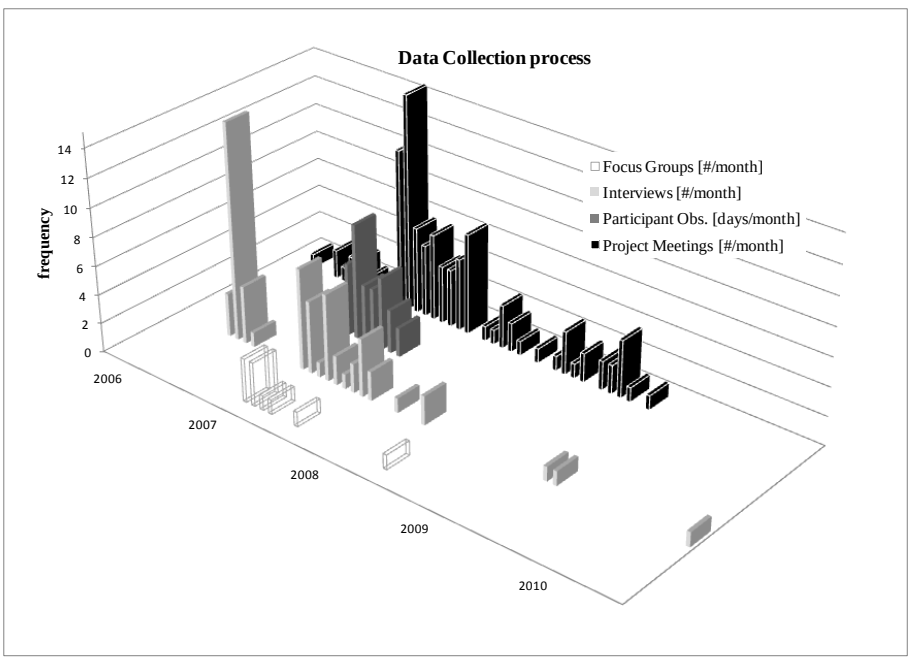

Figure 3. Overview of data collection activities.

4.2 Data Analysis

The analysis of empirical data was conducted to create distance to the research setting I had been so deeply involved with over a longer period of time. Therefore, in order to reduce the effects of biased judgment the initial coding process was conducted in a highly open-ended manner (Strauss and Corbin 1998), aiming to construct "analytic codes and categories from data, not from preconceived logically deduced hypotheses" (Charmaz 2006, p. 5). The various data sources were repeatedly read and coded to identify key themes from major events, activities, and technology choices that emerged over time (Langley 1999). The open coding process was organized using the ATLAS.ti software for qualitative data analysis. All in all, the open coding process generated 181 primary documents, 1217 unique quotes, and 327 codes.

Next, to reduce overlap and weed out clearly irrelevant concepts, the code base was reviewed again, in a more reflective manner. In the process of comparing the concepts generated by the open coding, preliminary definitions of properties and dimensions were inferred. This eventually resulted in a reduced list of 273 mutually exclusive concepts and 41 descriptive memos, interlinked with the code base.

Up until this point the analysis had been conducted bottoms-up, with deliberate distance to the practice generating the data, but also disconnected from the selected theoretical framework. To filter out the data resonating with the framework, the identified categories were mangled with the key concepts of architectural frames. Experimenting with different approaches co-occurrence analysis turned out to be an effective technique for mapping underlying structure of the data material to the framework. Co-occurrences reveal links between codes in that they share the same quotes. This analysis was conducted by frequent application of the Query Tool in ATLAS.ti, combined with the network view. This phase of the data analysis can be viewed as focused coding in that the objective was to "synthesize and explain larger segments of data" (Charmaz 2006, p. 57).

In particular, co-occurrence analysis provided a relevant angle on the data material. It traced concepts such as reuse, modularity, and complexity to data sources and organizational settings. However, it also resulted in four distinct high-level categories that emerged out of this process; platform, architecture, organization, and software development practice. Together, these categories (Figure 4), weeded out codes and quotes of particular importance when analyzing the application of architectural frames in digitalization at CarCorp.

At the same time, the analysis of data turned out to be a challenge in that the different concepts of architectural frames are not explicitly used in industrial practice. In other words; the distance between theoretical constructs and the applied language used by practitioners was considerable. Further engagement in more or less instrumental coding of the data material seemed difficult and not particularly rewarding. Instead, to widen the scope and apply the broad lines of the framework, without resigning precision, the refined model of the data material was repeatedly revisited from three well-defined angles. These angles aimed to:

1. Identify, describe, and contrast network-of-patterns thinking and hierarchy-of-parts thinking at CarCorp. Since concepts such as specialization, generalization, or aggregation were not explicitly represented, this phase was directed to single out representative, implicit patterns in the data material. Eventually, this analysis resulted in a validation of the theoretical framework and a demonstration of its ontological significance (6.1).
2. Review the locus of architectural frames in respective embedded case and shifts in architectural thinking across cases. Largely, the purpose of this phase was to derive evidence for the proposition that generative capability follows from the cultivation of network-of-patterns thinking. Given this point of departure, temporal shifts in architectural thinking across the four embedded cases were at center of attention. This phase eventually resulted in a detailed portrait of how network-of-patterns thinking propagates across an organization (6.2).
3. Identify tensions and contradictions between architectural frames. In particular, this part of the analysis was conducted with a dialectical stance to uncover incompatible aspects of the frames and their consequences in innovation practice. In synthesizing the four embedded cases I finally derived a theoretical perspective which explains why the governance model entailed by traditional hierarchy-of-parts practices is fundamentally incompatible with the governance prescribed by network-of-patterns (6.3).

Approximately ten years ago CarCorp started a journey characterized by gradual assimilation of a new kind of architectural thinking, which I have referred to as network-of-patterns. This journey was paved by challenges and disappointments, but also by radical advancement and great success. In what follows I present the story of digital product innovation at CarCorp over the period of the period of 1998 to 2011.

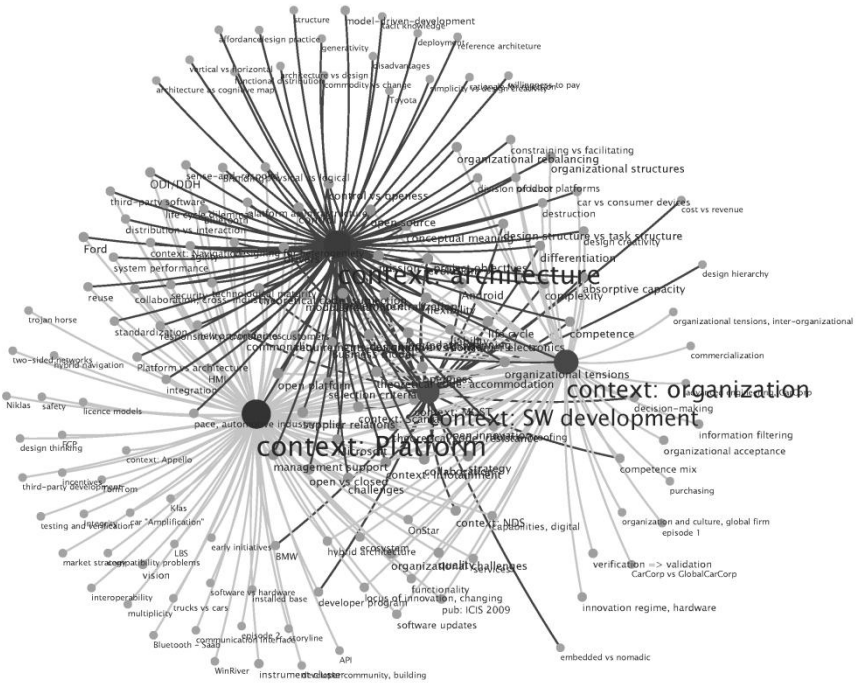

Figure 4. Illustration of high-level categories and their underlying codes (MapEquation.org).

5 Digital Product Innovation at CarCorp

At the turn of the century, the automotive industry could look back on an exceptional growth of infotainment-related functions. In just a few years rapid digitalization had opened up for in-car phones, navigation, telematics, TV, CD, and rear-seat entertainment where there used to be just a radio. However, while enabling such exceptional growth, digitalization had now reached a point where it seemed to challenge established innovation practices. It was more and more obvious to designers at CarCorp that complexity increased dramatically as coupling between components skyrocketed. Speakers, displays, controls, and various sensors simply had to be shared over the full range of infotainment applications to support coherent and progressive end-user functionality as well as leveraging economy of scale. However, the dominant architectural frame did not offer proper structures to handle this complexity. While seeing the system as a modular hierarchy of parts, with well defined interfaces in between, the design practice remained centered on components. Functionality was specified at the level of components and suppliers were contracted to deliver components. At the same time, intensified coupling between amplifiers, radios, CD players, and navigation systems made infotainment ambiguous for designers as well as customers as functionality could simply not be fully understood at the level of components.

Following the deep-rooted logic of modularity CarCorp responded to this increasing complexity by defining groups of components to hide the increasing interdependence within sub-systems. In retrospect, we can see

this as a first step towards recognition of a new architectural frame in that it accepted that the structure of functionality did not match the structure of components. Sub-systems were explicitly defined to encapsulate functionality, distributed over several components. While initially boosting functional growth, this strategy soon turned into a serious burden for the automakers. With modularity being a central element in enforcing hierarchical control over suppliers and sub-contractors, CarCorp and its competitors largely found themselves being in the hands of the suppliers. Amplifiers, radios, CD players, etc, remained separate physical entities, but highly intertwined through various proprietary and, from the automakers' perspective, largely unknown networks, protocols, and harnesses from a few major suppliers. In practice, this created monolithic, highly static solutions, hard to change after the time of production since functionality was bounded not only to a component, but to a particular configuration of components. R&D staff perceived decreasing control of system design, product planners of upcoming functionality, and purchasers of the sourcing process. The rapidly increasing coupling simply had to be addressed in a new way to reclaim control and secure future growth; CarCorp had to develop new fundamental principles for how to architect infotainment products.

In what follows, I present CarCorp's struggle with the identification and domestication of such a new architectural frame over a period of approximately ten years. The story covers four embedded cases, together making up a detailed narrative of how CarCorp transformed architectural thinking in response to digital technology.

5.1 MOST: The Recognition of a New Architectural Frame

Studying the automotive frontier as well as other manufacturing industries, infotainment designers at CarCorp realized that the unconditional encapsulation of software in hardware components was at the heart of their problems. To some extent, software afforded suppliers the opportunity to quickly change functionality at the component level, while this advantage did not play out at all at the system level. Specialized and complex interfaces effectively prevented reorganization of the system or redeployment of functionality. Somehow they had to build infotainment systems where software-enable functionality could exist without being forever inscribed in particular components. Such decoupling between hardware and software was expected to bring much needed flexibility to change functionality and, at the same time, increase the freedom when decompose the system. At the turn of the century there was one automotive initiative with enough

momentum to be hailed as a solution to this challenge: MOST[11]. A senior systems architect at one of CarCorp's competitors, later consultant for CarCorp, recall the early discussions promoting MOST as an interesting general purpose network concept, supporting the domain specific requirements and thereby further growth:

> *We all saw the transformation of infotainment. It was a remarkable change, and growth, and new lifecycles of the products. We needed an infrastructure to support this. MOST was [already] selected by BMW, with others talking about it. Somehow it should support this domain, with needs beyond body [electronics] and powertrain.*

At a physical level, the MOST architecture was constituted by a fiber-optical bus network, providing bandwidth far beyond hitherto established solutions. This network interconnected the different components through a generic, non-functional interface in a ring topology. In such a ring topology components are not nested to hide complexity. Instead, all components are found at the same level, whether functionally interconnected or not. Seen as a layer in a higher-level hierarchy – e.g. a car – such a system is flat, having a wide span (Simon 1962) at that level. However, the MOST architecture was also constituted by an object-oriented, event-driven application framework – the so called function blocks. These blocks could be viewed as instantiations of functional patterns at different levels of specificity. As an example, navigation core functionality, such as routing and guidance could be associated to one block, while traffic information, GPS positioning, and map data access could be associated to other. A pattern could be inherited by one or several other patterns, forming a network of patterns. Since MOST do not enforce any particular deployment strategies, the network of patterns can be reconfigured without touching the hardware setup.

Clearly, MOST offered two complementary architectural views on an infotainment system. On the one hand, it offered a new way to structure and interconnect physical components. The ring network encouraged a flat structure, with many components. This would allow engineers to mount components just about anywhere in a car, as long as it was possible to connect a tiny fiber-optical wire. Further, it offered a new way to *interconnect* physical components. As illustrated by another external MOST specialist, with experience from CarCorp and other automakers, the generic interface was a key argument behind MOST:

> *[With MOST] we saw an opportunity to bring things together. To get control [data], signals, audio, and, as we expected, also*

[11] Media Oriented Systems Transport.

> *video into the same bus concept. This in contrast to a mess of different harnesses and cables. MOST would simplify the [physical] system dramatically for us. You can compare it to a computer that you are plugging into the wall; you don't have one network for control signals, one for streaming audio, and one for streaming video. You've got ONE Ethernet connector. MOST takes this kind of thinking to the car.*

In addition to this generic interface, the fiber optics used in MOST technology offered significant communication bandwidth compared to established solutions. Extrapolating the functional growth of the 90th, this was standing out as a critical issue, partly contributing to increasingly monolithic systems. Demanding applications, involving streaming audio or video, could not be distributed across several components unless supported by a separate interface. The MOST architecture seemed to remove bandwidth requirements as a constraint in the decomposition of systems. CarCorp's former MOST project manager recalls that:

> *We had remarkable ambitions. We planned for video screens in the back seat and support for external video sources, delivering services such as park assistance. It should be a pretty high level of functionality. And when we looked at the different things customers should be able to do concurrently – it was a concept work I guess – we found that CAN[12] wouldn't do. We needed a really powerful bus concept to survive that. It should be able to support graphics, while simultaneously transmitting a burst of navigation data. These were key arguments [behind the selection of MOST].*

However, with the function block framework MOST foregrounded the network-of-patterns frame, offering a new way to conceptualize functionality independently from the hardware structure. In the past, the different functions of the infotainment system had been specified in relation to a component. An interface specification defined the functional relationship between components and a functional requirement specification all the details guiding the inner design of components. In contrast, the MOST architecture was largely centered on the so called "MOST Function Catalog". This specification identified and described all the functional patterns available in the systems, without any assumptions on where these functions eventually should be deployed. More specifically, it described how patterns were to be instantiated in software, including all the details on how to call a

[12] The Controller Area Network is a standardized vehicle bus network, designed to allow microcontrollers and devices to communicate with each other within a vehicle without a host computer.

particular function. Complementary specifications described how different functional patterns related to each other, i.e. how general functions were iteratively combined to form more specific functionality. A senior systems architect reflects back on the early impressions of MOST's approach to functional architecture:

> *I think we all realized – at least the people involved in [architecting] infotainment – that somehow this was the future. We needed to focus on the system, solving problems at the system level. We could not remain in the hands of suppliers, making stand-alone components. Instead, we had to make these suppliers part of a larger whole. […] I think, at the heart of MOST, we find a kind of system level thinking that is not component-oriented. Instead, it centers on the structure of logical elements or functionality.*

The architect underlines that the separation of functionality and components, applied in the MOST' architecture, was necessary to address the challenges of increasing interdependencies between infotainment components. After all, these problems were rooted in an inability to rebalance the system over time, as new functional requirements emerged.

> *We had to prepare ourselves in order to get access to that kind of flexibility [offered by MOST]. That is probably an interesting concept here – flexibility to produce information anywhere in the car and consume it somewhere else, in a simple way.*

This flexibility derives from the fact that MOST introduced a formalized network-of-patterns frame that remained relevant across generations of product design. As new functional requirements emerged, MOST promised the opportunity to deploy function blocks differently to physical infotainment components. The MOST architecture simply offered the automakers increasing freedom in combining architectural frames. In practice, this translated to a whole new *freedom in decomposing systems*.

Deeply rooted in component-based innovation, most people outside the R&D departments saw this freedom as a way to *reinforce modularity*. With frustration CarCorp product planners had observed how increasingly monolithic hardware structures destroyed attractive business models. Instead of an attractive list of options, enabling customer unique combinations, these interdependent systems had forced the automakers to bundle functionality in a few predefined offerings. This did not just cripple the new sales, but complicated the lucrative aftermarket business. The chief systems architect at CarCorp recalls that MOST's inherent capability to support a distributed system, opening up for a wide range of combinations, was recognized and highly appreciated.

> *The aftermarket business was a key concern at the time when we decided for MOST. We saw an opportunity to change and modify components on the aftermarket, at a low customer cost. It would be possible to add components over time and it would be possible to upgrade systems. […] With a distributed architecture [made up of more but smaller components] you get a cost penalty on the system, but you can still argue that 'it is so important to be able to offer customers the opportunity to extend their audio system with new amplifiers or additional speakers'. Then, he doesn't have to throw something away that he has already paid for.*

Rooted in the hierarchy-of-parts view, automotive organizations saw another great opportunity in extensive decomposition of infotainment systems, supported by MOST: *standardization*. Significant adoption of a standardized technology was considered a great potential. With the traditional, proprietary system solutions CarCorp could possibly benefit from competition at the time of sourcing, but not over the product life cycle. Major investments in systems integration effectively prevented re-sourcing of components, causing lock-in effects where the manufacturer had no option but to stick with existing suppliers. Considering another reflection from the external MOST specialist, it is clear that standardization was an important argument in promoting MOST. With standardized components CarCorp and other automakers saw a potential to dramatically increase competition, with lower thresholds for re-sourcing.

> *This idea about common specifications on functions and interfaces, that's a major benefit. More or less being able to buy a component [off the shelf], like a radio tuner, developed for one manufacturer, but applicable to another since it's a common interface specification.*

Clearly, the freedom to decompose infotainment in relatively independent physical components made MOST an attractive opportunity to increase revenue as well as reduce cost. Still, architects and designers could see that complexity would not disappear just because software-enabled infotainment functionality could be distributed across components. Inherent emphasis on the network-of-patterns frame would certainly facilitate change by transferring complexity to a digital domain, but it would not *per se* address the accelerating interaction between components. Somehow, the new infrastructure also had to address the root cause of increasing interdependency between components: *reuse*. At the time, there was neither technical nor organizational support for the reuse of resources or functions. As prescribed by the logic of modularity, problems were solved at the component level, by component engineers having the specific functional requirements of this component in mind. Consequently, sensors, displays, or

control buttons were designed as solutions to specific problems associated with the component. Appropriating such solutions for other purposes was time consuming, expensive, and unnecessarily complicated. A new infotainment architecture also had to support the idea of generalization, where reuse is an integral part of creative processes. It had to support designers and engineers in creating generic solutions that could be easily inherited by colleagues working with similar problems. This would not just reduce complexity of the system, it would also translate to a much needed alignment of infotainment functionality. Reflecting back, a senior systems architect at one of CarCorp's competitors, later consultant at CarCorp, argue that this opportunity to create a shared platform of common functional resources was – and still is – a key argument in favor of MOST.

> *There are many reasons to question a lot when it comes to MOST, but its fundamental principles remain sound and viable. That's what I think. It brings an object-oriented approach to "standardization" of [elements in] systems.*
>
> *I tend to see the logical perspective in the original MOST philosophy. I see some kind of elements that you reuse in different configurations. These [functional] elements can [for example] be a media player or GPS. They are not supposed to be too specific, with manufacturer unique information, related to HMI[13] or something. They should be pretty basic, to take care of well-defined tasks. On top of this you build your system, defining how these elements are used in different [specific] applications.*

As we shall see next, CarCorp's appropriation of the MOST architecture provided little change in business practices, marginal benefit from standardization, but a radically different infotainment system. In this MOST-based infotainment system the structure of every function was visible and well described in UML[14]. However, it was not just described, but generalized in the sense that general elements were up-front designed to be shared among specific functions. As a consequence, the new infotainment system offered functional alignment and coherence far beyond what earlier systems had been able to provide.

Appropriating MOST

With the decisions to adopt MOST for the coming generations of infotainment CarCorp entered a rather painful path, unfolding in the clash of

[13] Human-Machine Interface

[14] Unified Modeling Language

two architectural frames. MOST's service-oriented approach, largely dealing with functionality decoupled from hardware, reinforced the network-of-patterns frame significantly. Where the specification of functionality had been a relatively informal activity, at most supported by a shared word document template, MOST prescribed a seemingly unambiguous way to create, structure, and maintain the different functional patterns of the infotainment system. However, this frame was about to be applied in a domain deeply characterized by the component-based modularity of a product innovation regime. With innovation processes, organization structures, and products shaped by the architectural thinking of a hierarchy-of-parts frame the automakers had a difficult journey ahead, trying to set up a new interplay between frames.

The new way to structure components in a ring topology, implemented as a fiber-optical network, caused considerable concerns to CarCorp. At the same time, these troubles largely played out at a rather practical level. As an example, the novel integrated circuits that enable access to the optical network were not yet stable, thus causing major trouble to most manufacturers. Although learning how to manage fiber optics in an automotive context was demanding, these challenges had little to do with the conceptual principles behind MOST and, therefore, did not translate beyond the engineering level. Instead, the long term challenges related to MOST's new architectural perspective. With MOST, the notion of architecture became blurred in the eyes of designers, and gradually loaded with new meaning. The traditional rationale behind architectural work – hiding complexity, division of labor, etc – was extended with a new, partly incompatible logic. With software-based functionality distributed over a range of physical components, other properties became salient. The new infotainment architecture became an *enabler of functionality*, largely defining the shape and form of this distributed computing environment. Systems architects turned into platform designers. The architecture came to manifest a design philosophy and generic system level services, rather than a structure of components.

Although this transition was highlighted in the original MOST concept, the automakers underestimated the challenges of discovering, understanding, and implementing this design philosophy. It was simply far from obvious how specific infotainment functionality should be composed from general functional elements. A senior systems architect at CarCorp remembers his disappointment, when discovering that the architectural concept was far from solid:

> *They [MOST cooperation] promoted MOST as a new system-level model, a new kind of thinking, a new philosophy for*

> *design. But this model was never written down. It was BMW and Becker running it, but not in public. […] we could see how it was designed, I mean the result of the MOST interface definition, but we never understood the [deeper] thinking, and how they intended to evolve it. That made many of us, implementing at the time, doing extensions of our own, tweaking around, and creating solutions which probably did not align with the visions.*

On a general level, systems architects and designers were trapped between two architectural frames, without solid ideas on how to combine them. On the one hand, they had to reinforce the network-of-patterns frame to launch a more service-oriented approach to infotainment development. There was consensus among systems architects that the established component-based modularity would not be able to secure future growth for this family of increasingly changing applications. On the other hand, they were still embedded in a product development context that is tightly entangled with hardware-centric component-based modularity. A massive body of existing requirements was derived from the architectural thinking associated with a hierarchy-of-parts frame. Further, both suppliers and the automakers' own purchasing were reluctant to adopt software-driven business models. So were the product planners, showing marginal interest in software as a future revenue generator. The component seemed to remain the center of gravity for everyone, except designers and architects trying to reform infotainment. With such a range of path-dependent forces, the lack of clear and unambiguous design vision became highly problematic.

The automotive industry's component-based modularity, refined over a hundred years, is tightly intertwined with strict hierarchies both in product and organization structures. Product structures are hierarchical, with horizontal independency between components. In the same way, organizations are hierarchical, dividing relatively independent branches of labor. In order to govern such design hierarchies CarCorp followed strictly linear innovation processes, with a dynamics powered by waterfall models (Boehm 1976; Royce 1970) of product development. In practice, requirements were gradually broken down alongside the design hierarchy. Business objectives, general system topics, and overall functional properties were managed by the manufacturers, while the design of components and detailed functionality was assigned to highly autonomous suppliers, further down the hierarchy. As witnessed by a consultant, deeply involved in CarCorp's MOST project as systems engineer, this traditional hierarchy-of-parts approach did not change when sourcing the new infotainment system.

> *They thought the traditional model would work, where each [supplier] had responsibility for his own function, embedded in*

> *his own component. […] Down the road, they saw the flip side. It didn't work since the whole system – end-to-end – was so incredibly distributed.*

Attracted by the opportunities of a distributed system, CarCorp had decomposed their systems widely, resulting in significantly more components than earlier solutions. At the same time, they had invested considerable efforts in generalization, trying to build a solid infotainment platform where shared functional patterns were consistently reused by higher level, more specific functions. At the time of deployment, when functional patterns were allocated to physical components, the clash between the two architectural frames became obvious; functions and components did not match anymore. The remote "islands of innovation" did not perform anymore, when a specific function, such as navigation, was distributed across several components in that it inherited general functionality deployed to other components. Suppliers were contracted to design and produce components, not software. However, relying on the deep rooted practices of component-based modularity, these suppliers were formally made liable to functionality that was distributed across a range of other components, outside their immediate control.

Neither suppliers nor manufacturers were comfortable with this situation. Without dedicated software suppliers, taking full responsibility for component-spanning functions, innovation would most likely slow down. CarCorp saw no other option than bridging the gap between suppliers themselves by specifying not only interfaces between components, but also the system level behavior of all component-spanning functions. As illustrated by a project manager at CarCorp, this transition of responsibility increased the automakers' stakes in functional design dramatically.

> *You are taking a [new] responsibility as a manufacturer, when specifying this stuff. It becomes… I mean, they [suppliers] CANNOT even do anything! When I think about it, it's not them rejecting responsibility; it's us taking it from them. Yes, that's what it is. We are telling them that "the only thing you're about to do is to support this [our solution]. […] Earlier, when things were more component-oriented, they had an opinion of their own on things, they had tested it – possibly with other manufacturers – and knew what was good and what was bad. With this approach [MOST] we more or less lost such feedback.*

Clearly, these problems were grounded in an emerging and fundamental mismatch between the existing organizational structures and MOST's approach in conceptualizing software-enabled functionality. Taking the network-of-patterns frame seriously, designers had to increasingly

background the physical hardware. At the same time, these designers remained organized to match the hardware structure of the system.

Knowing that this mismatch could not be easily resolved, the automaker initiated two different measures to smooth the implementation of a MOST-based infotainment solution. First, they reorganized the workforces at a local level to meet the new commission. The management realized that the notion of component was less important with the new technology and architecture. Therefore, the local organizational unit, related to infotainment, where tuned for an increased need to exercise system level control and specify overall functionality. The acting project manager for MOST industrialization reflected on this topic:

> *Originally, it was a component-oriented group. They were expected to work with functional specifications as well. Later on, this didn't work out, so they invited some people working with functions only. They needed more and more such people and, eventually they were a group of their own. Probably 10-12 [persons], maybe even more. Most of them were consultant since it was running so fast, and we wanted it implemented. We underestimated the efforts significantly.*

Rather than obliterating the hierarchical structure, the manufacturer rebalanced the workforce, with old roles and levels of the hierarchy essentially remaining the same, while the locus of design activities moved upwards in the waterfall model, from the component level to the functional level.

Second, as designers reinforced the network-of-patterns frame, they had to break with the strictly linear models of innovation associated with component-based modularity. The new situation pushed new forms for collaboration and new relations – some temporary and some more permanent – between actors that were not supported by the official hierarchy. Moreover, with functionality becoming a system-level issue, it was necessary to adopt iterative approaches to innovation. While the official development processes stated very few recursions, each resulting in the production of a pre-series car, the new way of designing infotainment seemed to call for an endless series of iterations. While the reorganization was formally approved by management, solutions to these challenges emerged bottom-up, from designers' daily need to make progress. When specifications were ambiguous to suppliers, workshops were initiated with relevant stakeholders. When supplier implementations failed due to various misconceptions, the automakers built extensive system-level test environments to identify and solve problems collectively. When progress was too slow, the number of iterations increased dramatically, sometimes

exceeding one software release a week for individual components. Such figures are in stark contrast to the official development process, stating just a handful of releases for an entire 3-4 year car project.

Struggling with an appropriate model to combine architectural frames, CarCorp gradually found a reasonably stable way forward. On the one hand, the product innovation regime associated with component-based modularity remained. Formal specifications were written, broken down to a component level and, eventually, sourced to various suppliers according to existing principles. On the other hand, much of the critical system level and functional work was performed in a fluid structure of more or less temporary, cross-organizational design teams. Relations between actors and arenas for collaboration were established and destroyed according to project needs. Together these informal teams and processes made up a network-based model for innovation, augmented to the formal hierarchy.

Balancing these two, partly incompatible forms of collaboration was highly challenging to designers and architects. To support the network-oriented daily work, the automakers had to create new design practices, improving the collaborative visibility. At the same time, to enforce the formal hierarchies they had to find new practices for the deployment of the growing functional designs to physical components.

Systems architects at CarCorp had studied new design practices from the software industry even before the introduction of MOST. Since they had already seen increasing interdependencies with the low bandwidth CAN networks, they were attracted by the ideas of service-orientation and the ontological separation between software and hardware. With the decision to adopt MOST technology, bringing object-orientation and event-driven design, such ideas became legitimate and apparently useful.

As a first step to reinforce the new architectural frame CarCorp revised the definition of architecture. In the architecture specification for the new MOST-based infotainment system, they revised the notion of components, now referring to them as either logical entities or physical nodes. On the basis of this extended notion, they defined architecture as

> *the structures of the components of the system, their interrelationships, and principles and guidelines governing the design and evolution over time.*

In contrast to prior architectural approaches which more or less addressed the decomposition of systems in independent parts, this definition significantly changed the locus of architectural work. In including the dynamics of interconnected components and principles for development, it made system architecture a matter for designers in their daily work.

Further, with the network-of-patterns view on systems design in place, CarCorp adopted new CASE[15] tools supporting a model-based approach to functional design. They decided to use the unified modeling language (UML) as a basis for modeling. These new tools provided significant support in the process of deployment. Linking functional designs to physical design, they allowed for smooth generation of component level specifications and interface specifications. Clearly, this model-based approach played an important role in shaping how CarCorp *combined* architectural frames. As the new practices emerged, the role of component engineers transformed radically. Their prior role, interpreting information and compiling specifications, was essentially reduced to editorial work, including various non-functional requirements. Therefore, this approach supported not only the cognitive aspects of system design, but also the more organizational challenges of rebalancing the workforces.

Over time CarCorp realized that the mismatch between hierarchy-of-parts thinking, reflected in established organizations, and network-of-patterns thinking, reinforced by MOST, materialized at the time of deployment. It was the *allocation* of functional elements to different components that, in the end, enforced new relationships between component suppliers, without proper support in traditional processes. Therefore, it became increasingly clear that deployment could not be done on technological premises alone. Consequences on organizations and innovation processes were equally important when deciding how to distribute functionality across different components.

In response to this challenge CarCorp tried to allocate functionality characterized by high pace of change to just a few components. This strategy was expected to give a malleable infotainment system that could be effectively changed, without exercising the intricate tensions between different suppliers. As described in CarCorp's architecture specification, this strategy was centered on user interfaces, considered to be the most volatile part of infotainment.

> *The infotainment system is a user interactive and user intensive system (application) with continuously changes in the user interface but with core functionality that in some degree is defined as stable. Therefore it is a good idea to split the core functionality from the user interface.*

However, splitting the more general and durable "core functionality" from specific user interface functionality was not just a matter of making

[15] Computer-aided software engineering

appropriate deployment; it required an up-front conceptualization of functionality that *allowed* for such deployment to take place. It was simply necessary to have this strategy in mind when deriving general functions from specific. Further, it called for a *shared* approach to generalization. A scenario where navigation, telematics, and media player derived generic functional patterns on different premises would not just increase complexity; it would most likely result in different user interface logics and, eventually, confusion at the level of end-users. Therefore, as described in CarCorp's architectural specification, all designers had to adopt the same strategy when applying generalization to their respective functionality.

> *In many cases there exist design issues that does not map onto a single component, neither physical nor logical. These issues are more general in nature and must be addressed and expressed in form of strategies that must be followed by all designers involved in the design of the infotainment system family.*

Specifically, CarCorp had to set up a strategy addressing the imminent need to separate changing interface functionality from more stable base functions.

> *Based on this insight the decomposition of the infotainment system is based on the well-known architectural pattern Model-View-Controller.*

With the model-view-control (MVC) pattern guiding generalization, designers were encouraged to break functionality apart in a very precise way; the so called *model* objects corresponded to basic functionality, such as navigation routing or digital music decoding. *View* objects implemented the user interface, while *control* captured the dynamic properties. Further, the automaker implemented the observer pattern (sometimes labeled publisher-subscriber) to facilitate event-driven interaction between increasing amounts of distributed objects. Basically, this pattern identified controllers and views as subscribers of events at the models, creating a hierarchy between objects.

Summary and Epilogue

MOST introduced a new architectural frame at CarCorp. This new architectural thinking allowed them to structure physical and functional parts of the infotainment system more independently from each other. In order to reinforce the logic of modularity the automakers decomposed the physical system quite extensively, resulting in a wide range of components. Guided by hierarchy-of-parts thinking, the general, non-functional interface promised unbounded configuration of these components, translating to rich business opportunities. At the same time the automaker invested considerable efforts in generalization of infotainment system functionality. Released from the grip of components, systems architects and designers

inferred a range of general elements, to be reused for different specific functions. As a result MOST brought coherence between functions and, therefore, significant harmonization of the infotainment system.

However, facing an organizing logic fully defined by hierarchy-of-parts thinking, CarCorp found that the new architecture destroyed established innovation logic. With specific functionality intertwined through shared, general functional elements infotainment became ambiguous from the perspective of suppliers. Still responsible for physical components, they largely rejected responsibility for functionality with the valid argument that it was outside their control. Distributing a specific function across several components, delivered by different suppliers, essentially prevented any of these suppliers from taking over all responsibility. The only way to resolve this problem was for the automakers to increase their stakes in functional design. As a result, the locus of innovation moved upstream, from suppliers to the manufacturers. CarCorp had to specify functionality in detail, deploy it to components and derive concrete component-level requirements for each supplier. In practice, this resulted in *earlier* binding of functionality and even less opportunities to adapt to changing needs.

Addressing this weakness the automaker tried to deploy functionality to physical components so that expected changes would be isolated to just a few components. Knowing that a vast majority of infotainment change requests were related to user interfaces it was reasonable to direct attention to HMI. Therefore, the infotainment system was deployed to concentrate specific user interface functionality to one or two components, while the underlying functionality was distributed across the system. As noted by CarCorp's MOST project manager, this had some brutal consequences for the system as a whole; it accelerated the coupling between components dramatically.

> *Well, we did not make the ideal MOST implementation – it was hyper interactive. […] There was massive communication between the user interface and [e.g.] the audio manager, who needed to be involved. Then, when it had decided how to respond, it resulted in massive communication with the connection master and, then, the connection master with everyone else to set up new channels. So, yes, it was hyper interactive.*

In retrospect, CarCorp domesticated the network-of-patterns frame with MOST. The organization learned to use this complementary frame to engage in generalization, resulting in much needed and appreciated harmonization of infotainment functionality. At the same time, the attempts to combine architectural frames did not match established innovation logic and forced them to take full responsibility for functional design. With limited supplier

innovation, even earlier binding of functionality, and little focus on specialization of general functional patterns for new purposes the new, sophisticated MOST systems offered essentially no new specific functionality. On top of this, extensive distribution of functionality caused strong coupling between components, largely preventing low-end implementations of the MOST system. Whether configuring the system for base functionality or high-end application, most components turned mandatory. As expressed by CarCorp's former MOST project manager, this made the infotainment system far too expensive and, in particular, very hard to scale for a range of car models.

> *Considering our different levels… For high-end, with navigation, MOST was competitive. It delivered more functionality to a lower cost than any other solution within GlobalCarCorp. Unfortunately the low-end system became more expensive than corresponding GlobalCarCorp solutions.*

At CarCorp this eventually resulted in the rather remarkable decision to abandon MOST and turn back to existing solutions.

> *It happened fast and I think it had to do with the fact that GlobalCarCorp had a solution we could use. If we had been on our own, we might have given it another chance. Then we might have designed a cheaper, more centralized, and simpler system.*

Although few automakers followed CarCorp's quite dramatic decision to abandon MOST, it was generally questioned and criticized. As witnessed by a senior systems architect at one of CarCorp's competitors, the introduction of MOST systems at markets was painful for managers and designers at most automakers.

> *MOST was associated with failure. Then, it obviously was very costly. […] It is not an easy thing to stand up and defend a solution that just drained the wallet and caused a lot of trouble. It is not an easy thing for a manager to do. I know several manager who got a lot of blame, some were even close to losing their job.*

5.2 SoftCluster: Rethinking Platforms

The MOST architecture and other related projects in the late 90th had opened up a Pandora's Box in the automotive industry. These projects had introduced a radically new architectural thinking, foregrounding the structure of functionality, rather than the structure of physical artifacts. The new architectural frame had encouraged designers and architects to identify and reinforce functional patterns that were or could be shared within the system. When reusing such patterns for different purposes this generalization paid off as coherence between specialized infotainment

functions and, as a whole, a significantly more harmonized system. No doubt, the new frame was there to stay. At the same time, designers and architects were painfully aware that they had accomplished little progress in terms of new, specialized end-user functionality. Despite substantial investments they had simply not delivered anything new and exciting in the eyes of the vast majority, not seeing the system from the inside.

One legitimate explanation to this lack of novelty is of course that CarCorp had to focus their attention on short term challenges. They had to learn a radically new technology and find new ways of collaborating with suppliers. They simply did not have the time and energy to push end-user functionality. However, as indicated by CarCorp's chief systems architect there was a significantly more important challenge to tackle – a challenge that would not disappear as technology matured. Being a manifestation of the network-of-patterns frame, MOST's function block framework offered more or less unlimited freedom in designing functionality, without being constrained by the established logic of modularity. At the same time, this freedom disappeared at the moment when the functional designs were deployed to physical components. At this point in time the hierarchy-of-parts frame took over and suppliers considered specifications frozen.

> *I would say the main challenge was to handle this variability from an aftermarket and production perspective. We created [design-time] flexibility, but it was a problem to preserve this flexibility [...] That's often where we end; resolving technical challenges is one thing, to resolve all the different organizational challenges, about infrastructure and maintenance and support, that's a lot worse.*

As illustrated by CarCorp's architect, the inability to preserve the network-of-patterns frame could be explicitly inferred from a mismatch between the new architectural thinking and established organizational structures. Infotainment was about to be a matter of software. With software increasingly disconnected from hardware and powerful communication infrastructures in the cars, the new digital technology aligned well with the new architectural frame. There were simply few technological arguments to cut functionality into pieces and deport these pieces to isolated components. However, the organization, with its embedded routines, practices, and other structures, was built around the design and production of well defined components, with minor dependencies to other parts. These structures essentially forced CarCorp to background the network-of-patterns frame when leaving the implementation of specifications in the hands of suppliers. Suppliers were still contracted to deliver components. Purchasers were scanning the domain of components to locate the best offer. Quality assurance was centered on the testing of components.

CarCorp could see that the established product innovation regime did not bring proper incentives for suppliers to engage in software-centric innovation. At the bottom line, they would be accounted for a component, not a piece of software. Still, they were expected to design and deliver the increasingly important software, making up the infotainment system. In addition, the clash between a new architectural frame and established organizational structures largely had transferred responsibility for functional design from suppliers to CarCorp. In summary, suppliers cared for components, CarCorp for functionality, but no one really cared for the software.

At the end of the 90th CarCorp started to see that software was at risk of falling through the cracks. Although unclear *how* it would affect the organization, it seemed necessary to reinforce competence in this area. As narrated by today's software manager, it was more or less insignificant at the time:

> *From a software perspective CarCorp was a disaster when I came here 1998. You can't even imagine. I was offered a position as software engineer, but realized that I, more or less, was the only one with software competence. Coming from Ericsson, it made a huge contrast. Over there, software was at the heart of what we did.*

Lost in this hardware-oriented organization he struggled a lot with how to contribute in the new organization. In contrast to his earlier positions, he ended up quite far from the practice of software design and implementation.

> *I had major problems understanding what they wanted me for – what are their intentions? I spent a lot of time working with processes. You know, what does a software process look like at CarCorp? What is our role and what is the role of suppliers?*

However, while working in frustration with these high-level questions, pretty far from hands-on software development, coincidence played him in the hands. A new car model was in a critical phase of development when the supplier of the instrument cluster proved alarmingly weak on software competence. For several reasons CarCorp was stuck with the supplier and in a highly unorthodox manner, management decided to offer the relatively new software engineer to put together a small team and make the software in-house.

> *This supplier proved to be weak on software. Really weak. They were excellent on mechanical things, on hardware, but they couldn't handle software. So we said, OK, let's make the software. We had backing from management, although I doubt they understood what we were about to do. With this support we*

> *put together a group of 4-5 [designers]. We did the software on the basis of a platform that we had made ourselves and [eventually] we were able to take it to production. […] It was very successful. We did it fast and with quality.*

Looking back on this period the software manager sees a milestone. These achievements brought the issue of software up on CarCorp's public agenda and indicated that it could be rewarding to engage in the actual production of it. With in-house development of software the automaker did not have to rely on the linear innovation processes where specifications were frozen too early, for deployment in different components. Instead, software development and functional design could co-evolve, essentially keeping the network-of-patterns frame alive and relevant across the entire development cycle. With in-house development of software, it suddenly seemed possible to achieve variability that was not grounded in the reconfiguration of components. As pointed out by the chief systems architect, such variability was particularly valuable for HMI-related functionality.

> *By tradition, suppliers offered a low [component] price, knowing that change orders would feed them down the road. These changes always turned out to be 'small and simple' HMI changes.*

As CarCorp's successful initiative raised attention at its owner, the rationale behind internal development of instrument cluster software was further reinforced. With several brands and a lot of different car models, GlobalCarCorp also saw an opportunity to commoditize hardware. With the traditional hierarchy-of-parts perspective on products, differentiation tended to drive cost, simply since it manifested itself as new component variants. More variants gave lower volumes and less opportunity to benefit from an economy of scale. Commoditization, on the other hand, tended to make designs inflexible. Making a component fit in the design hierarchy of several different cars was a challenging task, even with minor dependency to other parts. Changes would trigger re-validation across the entire range of vehicles. Therefore, commoditization came with no less than tough compromises between the values of different market segments and brands. Ripping up a settled deal to introduce new functionality would trigger a new round of painful negotiation. GlobalCarCorp now saw an opportunity to escape from this seemingly unavoidable contradiction; CarCorp's approach promised one shared cluster hardware, powered by unique and easily changeable software for each model.

2005 GlobalCarCorp launched a new strategy, where instrument cluster software was classified as 'strategic software' to be developed in-house. The task of developing this software for GlobalCarCorp's all brands and models

was assigned to CarCorp. The official acknowledgment of software competence opened up for a new organizational setup. Above all, this allowed for significant professionalization and growth. CarCorp's software manager remembers that:

> *[2005], about three years ago, the first [real] software team was put together. That was when the organization started to see software. The team was approved and people with dedicated software skills were hired. This is also when we were trusted with the software development for all [instrument] clusters within GlobalCarCorp. […] Down the road we have hired more people working with structure and architecture and all the different aspects making a software organization.*

Developing the SoftCluster platform

Over a relatively short period of time, the new team put together what was later recognized as the SoftCluster platform. Designing the architecture of this new platform, CarCorp had one primary objective; to destroy barriers of change. It was increasingly obvious that software-centric innovation was an emergent phenomenon, where functionality could not be designed up-front, as prescribed by established processes and structures. As underlined by one of the HMI designers, later using the software platform, it was designed for recurring specialization, where general elements were reused and recombined to form genuinely new functionality.

> *Well, flexibility was a key argument. To be able to implement [new] HMIs down the road and modify them quite extensively.*
>
> *It's like playing with LEGO. You've got a particular set of bricks. They've got their limitations, but you can build a whole lot of different things with them. And it's simple.*

To make the SoftCluster platform a truly flexible toolbox of such LEGO-like bricks, the software team had to find a solution to an intricate challenge. On the one hand, *instrument cluster functionality is inherently distributed.* A display in the cockpit mediates information on average speed, fuel consumption, outdoor temperature, radio station frequency, and many other things. The instrument cluster system collects, aggregates, and presents all this rich information deriving from remote sensors, encapsulated in a whole range of different components. No doubt, these remote components had to remain stand-alone components, primarily for cost reasons. A low-end car would have a radically different setup compared to a high-end car. In that sense, the SoftCluster platform had to recognize the hierarchy-of-parts frame and support the decomposition of the system into different components.

On the other hand, the legacy of MOST projected strong arguments *not to distribute instrument cluster software* across different components. In order to take the network-of-patterns frame seriously it was necessary to avoid the destructive clash between frames that had eventually caused the exit of MOST technology at CarCorp. The only reasonable way to achieve this in an organization dominated by hierarchy-of-parts thinking was to deploy the SoftCluster platform to one key component. If change could be isolated to one physical node there would be little need for continuous synchronization of different parties through static component specifications that, eventually, would prevent variability.

Trying to set up an architecture resolving this inherent contradiction between the hierarchy-of-parts frame and the network-of-patterns frame, the new software team concentrated their effort on three distinct measures:

First, they agreed on a specific design rule to be applied when making functional designs; the system should be generalized in such a way that all information sources could be handled as independent, simple, and general patterns, logically decoupled from more specialized functions. As described in one of the specification for the SoftCluster platform, such a "functional unit (FU) defines what [information] content it is able to display", but essentially nothing more. This would allow for a decomposition of the system in one master component, hosting all specialized functionality, and a range of slave units that could be configured to meet particular model requirements. On the whole, this design rule was introduced to assist the combination of architectural frames. It would help preserving some of the flexibility offered by modularity, while at the same time allowing for functionality to evolve as software could be easily changed at the master component, with minor implication at the slave nodes.

Second, to cater for changes in FUs that after all would occur, the SoftCluster platform architecture introduced a new end-to-end communication strategy. This strategy was manifested as an XML interface – the OpenXMLInterface (OXL) – allowing for retrospective changes in communication between the master component and remote information providers without impact on the system level. As described in one of the system specifications the general CAN network in the car, used by the SoftCluster platform, allowed for restructuring of bus messages. However, such changes entailed modification of the central signaling database, essentially enforcing revalidation of the entire electrical system in the car.

> *The [network] handler architecture allows architecture designers to change both parameter IDs and message framing [...], but this ambiguity is in conflict with the component sharing and interoperability strategy for components using the OXI*

> *interface [...] Therefore, all OXI messages shall have fixed parameter IDs, startbyte, and startbit relationships.*

With OXL instrument cluster designers could rely on fixed CAN network messages, while augmenting an XML structure on top. Changes could be rapidly introduced in the ends, without involving the rest of the organization. In practice, the system could evolve at the network-of-patterns frame, while preserving the hierarchy-of-parts frame intact. OXL was an important measure to avoid the lethal clash between architectural frames, causing so much damage to the former MOST-based infotainment system.

Third, with the SoftCluster platform CarCorp introduced a macro-oriented approach to HMI development. Similar to the logic of a web browser, the new concept made use of a markup language to specify layout and look-and-feel. The macro was stored in a database and interpreted in real-time by a standardized software component – the so-called HMI engine. Consequently, the new concept commoditized not only the hardware, but also considerable pieces of the software. Reflecting on the long-term consequences of architectural work, a senior engineer at the consultant firm co-developing the SoftCluster platform with CarCorp touches on the main rationale behind this radical approach; the architecture is explicitly reflected in processes and, therefore, it defines how a product can be changed over time.

> *In some sense, it's when you break the system into pieces that you really see the architecture. That's when it is most important. It defines the processes for distribution, purchasing, verification, and things like that.*

Even though CarCorp had decided to develop instrument cluster functionality in-house, the design team feared that software would be inscribed in hardware at the time of production. The macro-oriented HMI (MOH) reinforced the separation between functionality and the tangible product one step further. With MOH CarCorp saw an opportunity to create and, in particular, maintain truly differentiated HMI solutions for the full range of vehicles within GlobalCarCorp, without being constrained by part number administration, system re-verifications, or other organizational burdens, hampering change. Specifications clearly state that the MOH is grounded in:

> *...a need to support vehicle brand differences within the GlobalCarCorp family such as difference in graphics, layout and menu structures without having to change operational software in any ECU.*

This reasoning is developed in some detail by the chief systems architect, when trying to summarize the rationale behind the selected architecture of the SoftCluster platform:

> *With the new generation [instrument clusters] we wanted it to be perceived as uniquely designed for each market, with its language and culture, still relying on a shared [HMI] engine, handling everything. With this solution we wouldn't have to verify the software for every market. The code was shared, it was decoupled from the look-and-feel. So, we truly separated presentation from logic and [general software] application.*

Summary and Epilogue

With the SoftCluster platform CarCorp reinforced the network-of-patterns frame significantly. Earlier initiatives, such as the MOST project, had focused on generalization, providing coherence between functionality and increasing harmony in using in-car systems. With the new cluster project CarCorp shifted focus, from generalization to specialization, launching an architecture with one overall objective; to assist rapid and smooth evolution of instrument cluster HMIs. This architecture was grounded on three distinct elements:

1. It separated volatile and specialized HMI functionality from relatively stable and generic functional units, collecting and aggregating basic information. With this separation CarCorp could combine architectural frames in a new way. Modular strategies could be exercised to vary hardware setup between car models, while at the same time concentrating the evolution of software-centric HMI functionality to one component.
2. It introduced end-to-end communication capability, allowing for distributed functionality to change with insignificant implications on the system level.
3. It introduced a macro approach to build specialized functionality. This allowed designers to launch new functionality without recompiling software, which reduced risk, validation efforts, end eventually cost.

As illustrated by an excited HMI designer, the SoftCluster platform was exceptionally successful in reducing barriers for change. He argues that the platform completely dissolved the gap between design and product. In principle, a new idea could be pushed all the way to customers, without involving suppliers, software engineers, or even test teams.

> *You know, lead times are usually very long in this industry. But it's fascinating [with this new concept], because now I can be*

> *part of a design discussion, trying to plan for a change that feels pretty challenging, and people ask me "how long does it take to implement it?" And I can tell them "it's already done." With this architecture I can make some design changes really, really fast. But I think there are very few realizing it.*

The designer underlines that the MOH concept introduced a set of fundamental limitations for creative work, but given compliance with the offered framework it provided substantial freedom in designing new specialized solutions:

> *There are restrictions [in the SoftCluster platform], but as long as you follow them I see no problems. I could put something together, like… I mean now we are in the automotive industry, [working] with radios and phones, but give me two hours and I have designed a solution for a washing machine. Give me another two hours and I have the HMI ready. I don't know [right now] what it's going to look like, but give me a few sketches and it's done. And then I mean up and running.*

Although many designers did not think the new instrument cluster solution was used to its full potential, the SoftCluster platform has to be described as a success. It entered production 2008 and is still a strategic tool in GlobalCarCorp's maintenance of instrument cluster functionality. It is applied to 5 different families of clusters, spanning 23 different languages. All in all, GlobalCarCorp have shipped more than 10 million cars using the SoftCluster platform.

Given this success, the software team at CarCorp was requested by GlobalCarCorp management to study how their experiences could be applied in a new infotainment platform. Fall 2007 the software manager put together a review team of employed engineers and external specialists to set up the guidelines for such a new software-centric infotainment platform, intended to be shared by all the brands within GlobalCarCorp. After several months of intensive brainstorming, technology reviews, and state-of-the-art analysis it was clear that the SoftCluster concept could not be transferred to infotainment unless it was extensively modified. The main reason was that infotainment evolves according to a logic which is substantially different from an instrument cluster.

The SoftCluster concept was based on the assumption that variety and change plays out on the most specialized level, while the underlying, more general functionality is static. HMI design ideas which complied with the markup language, defined by MOH, could be realized with minor efforts. However, ideas that could not be implemented on the basis of these pre-defined, general building blocks were destined for rejection. Similarly, ideas reusing and recombining existing FUs could be set up with little trouble,

while the need for new information sources would trigger painstaking development of new physical components.

The review team found that the rigid SoftCluster platform, essentially relying on fixed general patterns, was unrealistic for infotainment. Somehow, an upcoming infotainment platform had to be adaptable also at the level of general patterns; generalization could not be seen as a one-off activity, performed at the time of original platform design, but had to be considered a recurring activity. Moreover, it was increasingly clear that the pace of recurring generalization and specialization could not be defined by the automotive industry. Over a few years engineers at CarCorp and other manufacturers had witnessed how infotainment was increasingly colored by the rapid progression in consumer electronics. Customer expectations were more and more defined by standards of mobile phones, handheld computers, and PNDs[16] As illustrated by CarCorp's software manager, it seemed inevitable to involve external parties to cope with this challenge.

> *[For example] We don't see these [increasingly important] advanced graphic engines as a core competence at software & control. We simply don't think we can get state-of-the-art user interfaces if we decide to do them ourselves.*

With consumer electronics actors engaged in the development of an infotainment platform, CarCorp saw an opportunity to share general functional patterns across industry boundaries, eventually assisting them to keep up with its higher pace. The software managers emphasized that such strategic collaboration would not just have implications on HMI, but play out more widely and give:

> *…the opportunity to actually offer marketing, product portfolio, and design what they really want. […] Requirements we get from design are very inspired by the iPod and the iPhone. That's the kind of functionality they would like to see in our infotainment system. And we can't do that in-house.*
>
> *So, what is new here? Well, I think we need to focus a whole lot more on flexibility and pace of change. We need [for example] to keep up with new trends for connectivity. Seeing WiFi connectivity coming for iPod we need meet that quickly with a new solution. So, by being in control of software, we can be fast and make sure there are [general] software functions supporting whatever it is we see coming. I think that's the main challenge here.*

[16] Portable Navigation Device.

The overall challenge for CarCorp and other automakers was how to find an appropriate way to combine architectural frames so that infotainment could, on the one hand, exist in a hierarchy of parts characterized by one pace of change, while, on the other hand, allow for functionality to evolve at pace defined by consumer electronics. As we shall see, CarCorp addressed this challenge by two different initiatives; nomadic device integration (5.3) and the open Android platform (5.4).

5.3 Nomadic Device Integration: Bridging Pace Barriers

While the MOST-enabled infotainment system and the SoftCluster platform largely were responses to internal technological progression and accelerating functional growth, the automotive industry also faced increasing external pressure for change. Between 1998 and 2002 the number of mobile phone subscribers in the developed world increased from 25% to 65%[17]. Similarly, portable navigation devices had a remarkable commercial breakthrough a few years later, illustrated by TomTom's 375% increase of sales between 2004 and 2005[18]. The roll-out pattern of portable music players is similar to navigation, although diffusion figures are even more overwhelming. In September 2009 Apple announced that the cumulative sales of iPods exceeded 220 million units[19], with a significant breakthrough at the end of 2004.

Over just a few years the design challenge in vehicle infotainment changed dramatically as an emerging consumer electronics market offered an attractive alternative to integrated systems under the control of auto manufacturers. Whether CarCorp liked it or not, they had to relate their own products to the new competing systems. Every new release of in-car infotainment products would be measured by consumer electronics standards. Even more problematic, the high pace of change in consumer electronics made CarCorp's in-car infotainment systems seem outdated in months or years, while the car, hosting these systems, had a significantly longer life time. Over time designers feared that the long development cycles in the automotive industry would turn these systems obsolete from the first day of sales.

On the one hand, new initiatives for systems design and new perspectives on architecture had introduced technology and knowledge, allowing CarCorp to

[17] ICT statistics from the International Telecommunication Union (ITU).

[18] TomTom press release February 14, 2006

[19] Apple press release September 9, 2009.

decouple software from hardware. The network-of-patterns frame, reinforced with MOST and the SoftCluster concept, had offered significant design-time flexibility. At the same time, this flexibility had vanished at the time of production, when architectural frames were irreversibly combined. With supplier revenue streams triggered by the delivery of high quality components, not superior design processes, the distribution of functionality across several components created unpleasant ambiguity. Similarly, the automaker centered its verification and quality assurance on component tests, not design reviews, further reinforcing a component-centric view on the products. Finally, the hierarchy-of-parts frame was made permanent as suppliers were made liable for components over time on the basis of warranties. Retrospective introduction of new software would not just bring additional properties to a system; it would potentially disrupt the stability of the system.

One can argue that the MOST project introduced generalization at CarCorp, while the SoftCluster initiative established specialization. However, the automaker had not been able to close the loop, allowing the two mechanisms to survive beyond the time of production. No doubt, this would be necessary in order to keep in-car infotainment up-to-date with the progression in consumer electronics. However, a more software-centric perspective, where the network-of-patterns frame could survive the transition between design and production pushed for a radically new perspective on products as well as business models and seemed distant. It would come with no less than the dissolution of inherent institutional structures.

Seeing the massive challenges in this, designers and managers working with infotainment saw an option in direct integration with mobile phones, PDAs, navigation systems, and other nomadic devices[20]. With this approach infotainment functionality could still be viewed through the lens of a component, included in the overall hierarchy making up a car. However, for a simple reason this functionality would not be restrained by the hierarchy-of-parts thinking dominating the automotive industry; nomadic devices were designed in another industry and intended for a different market. Over time they would change at a pace defined by the consumer electronics industry, not the automotive industry. Nomadic device integration seemed to offer a combination of architectural frames where automakers could keep considering the product as a hierarchy of parts, while the consumer electronics industry would ensure that functionality was not forever

[20] Nomadic device is a term used widely to refer to a handheld wireless device, such as a PDA or smartphone.

inscribed at the time of production. Nomadic device integration (NDI) would enforce consumer electronics life cycles on the automotive industry.

As early as 1996 CarCorp initiated collaboration with the mobile phone manufacturer Nokia. The initiative rested on a cradle-based vision, enabling convenient use of at least one of Nokia's phone models in the car setting. In-car resources, such as speakers, microphones, and controls would give new opportunities to adapt the off-the-shelf phone to a driving context. The functional design of the phone would be untouched, while CarCorp planned to introduce more specialized patterns, inheriting general nomadic functionality to deliver a dedicated in-car user experience. However, the project was terminated in early stages as Nokia changed the interface for accessing the phone several times during the project. Nokia did not consider the potential gain in car-related cell phone sales attractive enough to stay with the initial interface. It would have slowed down its own product innovation. Sales of cell phones exceeded that of cars many times, making cooperation with CarCorp marginal to Nokia's business proposition. The Nokia project had, once again, demonstrated that the hierarchy-of-parts frame relies on a stable structure, with fixed interfaces. Functionality can evolve within components, but the decomposition of the product is unlikely to change.

Despite these discouraging experiences CarCorp renewed the efforts to create a system for nomadic device integration 2002. The sales of integrated phones would not take off and it was now recognized as a dead end. It was simply too expensive for most users and the solution was hopelessly out-of-date in that it did not support the functionality offered by a typical cell phone. In addition, it was costly to maintain and modify. It was time for CarCorp to reconsider the idea of nomadic device integration.

With the Nokia project in mind, designers realized that nomadic device integration would remain nothing but a vision unless they found a solution that allowed phone functionality to evolve, without impact on the in-car system. Addressing this issue, they formulated two guiding principles for the upcoming NDI initiative. First, it had to overcome interoperability problems caused by ever-changing physical characteristics. Instead of a cradle, tightly connected to phone designs, the new solution would make use of a wireless interface. Second, integration between vehicles and nomadic devices should not rely on vendor-specific, proprietary technology. Stability would come with no less than a public standard, not controlled by a specific actor.

Although there were a few interface options available at the time, the escalating momentum of Bluetooth technology draw most of CarCorp's attention. The fact that Bluetooth was not created by or for the automotive

industry complicated the process. Yet, it seemed to resolve the problem causing the Nokia project to fail. As illustrated by the infotainment product manager, Bluetooth was seen as the missing link, allowing nomadic functionality to evolve, while preserving the critical interfaces in the overall hierarchy of parts.

> *We have overcome the barriers associated with proprietary standards in mechanics, electronics, buses, and so on. General standards such as the Bluetooth protocol now exist, making us believe that this will actually work, also beyond a particular phone model's lifecycle.*

One of the engineers made a similar statement:

> *Using a standardized interface means that we can both lower our development costs and increase customer value. […] It is much easier when you follow a standard rather than trying to develop a standard or a proprietary technology as we have done up to now.*

2004 CarCorp launched the first solution for nomadic device integration, based on Bluetooth technology. It was a handsfree-kit, allowing seamless transfer of phone calls between mobile and automotive contexts. Without the need for docking cradle, the user could leave the phone in the pocket or a bag, while interacting on the basis of dashboard controls and in-car audio resources.

From a functional perspective the new system performed well. However, as a means of handling the inherent life cycle differences between consumer electronics and automotive industries the selected solution confronted significant and unanticipated challenges. First, the Bluetooth standard proved not to be a standard, at least not up to CarCorp's expectations. Cell phone manufacturers interpreted and implemented the Bluetooth protocols differently, leaving significant interoperability problems for CarCorp. As a result, the NDI solution could only be developed, certified, and tested for a limited range of cell phones. This was a major disappointment for the people involved in the project. An infotainment manager ironically reflected upon the unanticipated problems:

> *If you were an early adopter, you ran into troubles. CarCorp was a really early adopter [of Bluetooth] in automotive…. Standard proved not to be standard. There was a very complex relationship between devices across brands and models, which made the process rather tricky.*

Second, the Bluetooth standard proved not to be stable in time, at least not up to automotive norms. Instead, it evolved in harmony with new phones entering the market. With a repurchase time of less than 18 months

Bluetooth enhancements were pushed to consumers through new devices. With vehicle life cycles of approximately 25 years, this strategy was inherently closed for CarCorp. Suddenly, the idea of keeping the system up-to-date over time, as customers bought new cell phones seemed hopeless. As illustrated by one of the infotainment designers, the fundamental differences in product life cycles turned out as the key challenge in creating a sustainable solution to nomadic device integration:

> *Sadly, we don't support the latest cell phones. We are working on it but we are facing a tough automotive reality. We have not been able to change our processes. It takes very long time to introduce software updates. The software has to be validated as part of a system. This is related to safety, and the fact that we must guarantee the endurance and quality of our systems over time. Getting a component into a car takes one year. When it is supposed to talk to another system in the car, it involves a major validation process. At CarCorp today, a new piece of software means a new validation process of the entire system. That's why we can't keep pace with new devices coming.*

At this point the NDI proponents were lost and frustrated. They gradually realized that Bluetooth would not be the solution to their vision. It did not allow them to support a wide range of mobile devices and, more important, it did not allow them to support future devices, not yet designed at the production of the car. Despite exceptional efforts they had not been able to appropriate Bluetooth to bridge the gap between architectural frames. In practice, functionality was still inscribed in the system solution.

At the same time, the rapid consumer uptake of cell phones, portable music players, and navigation devices kept building up pressure from the consumer electronics industry. As illustrated by an infotainment project manager, NDI proponents expected CarCorp's traditional business cases to break down as a consequence of this external pressure:

> *We are a couple of people who think that [selling embedded navigation and CD-changers] won't be possible in the future.... When you have navigation in your pocket, why have an integrated navigation system in the car? You will not have a CD-changer in the car AND an mp3-player in your pocket. We believe that this type of car equipment won't be there in the future – that the market will disappear for us.*

At the same time he underlines the underdog situation by noting that:

> *Now, I should not presume that this is the company's official stance. I get a lot of shit for saying this, especially from our marketing people... they don't believe in this, they don't think it's reasonable to think like this. They still believe that it's going to*

> *be possible to sell integrated navigation in large volumes, and that it still will be possible to sell CD changers.*

Consequently, the infotainment group faced not only a significant technological challenge, but also minor support from management and the rest of the organization. On the one hand, the absence of management attention was problematic. On the other hand, it nurtured a skunkwork attitude within the increasingly tight group. Reflecting back on this period one of the strongest NDI proponents noted that:

> *We were rebels. We have always worked on ideas and solutions that have been difficult to appreciate from an automotive perspective. We have always seen ourselves as outsiders in view of the mainstream automotive designer.*

In this tolerant environment the design team approached the life cycle dilemma again. They realized that the NDI vision would remain a vision unless they were able to shift their mindset. As one designer commented, the idea of a fixed interface had proven fundamentally misleading:

> *The car has a long life cycle and a slow development life cycle. We therefore need a flexible software-based connection for nomadic devices that can adapt the car to modern technology after the point of sale. We need this in order to offer new applications in a flexible and agile way*

Another designer noted that:

> *What we should try to do is to introduce leeway in the interface between our slow cycle and a much faster cycle, and still create customer value. So, the objective must be to identify the magical interface that enables us to adapt to a world that moves so much faster*

Synthesizing previous experiences they abandoned the idea of enforcing the pace of one industry on another, which essentially was the idea behind using a standardized interface. Instead, they started to promote the idea that integration between in-car resources and nomadic device could be enabled and maintained through a gateway component. This solution would allow for interoperability between a static vehicle environment and evolving functionality on nomadic devices. With a simple software patch the CarCorp customer would be able to buy a new cell phone or music player, while preserving in-car support of the new device.

This approach reinforced the network-of-patterns frame. Although seemingly simple at the level of technology, the idea represented a major deviation from established architectural thinking. Following the traditional logic of manufacturing, CarCorp applied a “fire-and-forget strategy” across its whole range of development. In practice, functionality was mangled out in

the design phase, but fixed at the moment of production. New ideas and solutions, born after start of production had to wait for a facelift or a new car model. Therefore, the new gateway concept for NDI had to challenge the traditional design-production barrier. It would keep the network-of-patterns frame alive and relevant beyond the time of production. This would be particularly problematic without a solid management support.

2006 CarCorp launched an advanced engineering project to demonstrate these new ideas. The gateway was framed as an open platform for interaction with nomadic devices (OPN). At this point the notion of *openness* essentially reflected the envisioned capability to stay tuned with external consumer electronics, while *platform* emphasized the hardware-software distinction – the gateway would be malleable to external environmental changes through software updates only. The former manager of the Nokia project, now appointed project manager for the gateway project, reflected upon the new concept and the road ahead:

> *We are envisioning a software design that boosts the car's capacity to handle the digital world. The solution must enable us to follow the technical development in telecommunications during both the construction and production time of the car, which, taken together, is around seven years.*

With a vision calling for device-independency and the idea of standardized interfaces left behind, infotainment designers had to identify new mechanisms for the integration of vehicle and nomadic devices. Following traditional product innovation logic it would be CarCorp's responsibility to identify new interfaces, specify them, and make sure the corresponding software was designed and installed on the gateway. Consulting earlier experiences they saw the absurdity in such a practice. CarCorp did not have intelligence capability to indentify proper candidates for integration. It simply did not know consumer electronics and telecommunication industries well enough. Further, automotive pace would effectively prevent quick turnaround of software drivers. Rigid automotive processes would delay the introduction of a new interface to the point where it was no longer interesting. Struggling with this challenge designers turned their attention to the successful USB technology. A senior software architect argued that:

> *We should mimic the plug-in flexibility offered by USB. It is the device that is responsible for providing the relevant driver. This enables an end-to-end architecture for making the systems operate together… As a third-party vendor, you'll supply this opportunity by installing the driver on our open platform.*

At this point the concept of openness was filled with a slightly different meaning in the eyes of designers. With this new perspective 'open' did not

just refer to the flexibility enabled by technological integration with external devices. Instead, it recognized integration between the vehicle and nomadic *software*, essentially disconnected from the device. A third-party developer would be able to provide software, relevant for in-car usage, without being in the hands of device manufacturers. At this moment, the architectural locus shifted significantly in favor of the network-of-patterns frame. It became increasingly clear that the functional structure of the infotainment system could change independently from the physical hierarchy of parts. Third party actors could introduce or revise general functional patterns, allocated to nomadic devices, for use in specialized in-car functions, allocated to in-car components under CarCorp's direct control.

With enthusiasm the NDI proponents saw the potential and beauty in this new perspective. With a general API[21], the functionality of nomadic devices could evolve at its own pace, without being constrained by the physical decomposition of the infotainment system. Essentially, it would be up to external, third party actors to secure compliance with the car. Nomadic device software would not be accessible in the car unless it could be inherited by these specific patterns, adapting functionality to a car context. This seemed to be an extraordinary opportunity to resolve the life cycle problem.

With the mobile navigation provider Appello as partner in the project, the first benchmark was more or less given. The NDI gateway should demonstrate how Appello's mobile phone-based off-board navigation[22] solution could be enabled for specialized use in CarCorp's cars. From an architectural point of view Appello's existing solution represented a network of patterns, hosted by the nomadic device. This network had to be extended with a new layer of specialized functional patterns, solving problems that were unique to the car environment. In order to preserve Appello's existing solution as much as possible these specialized patterns had to be deployed to the NDI gateway. In practice, the software designers approached the challenge by looking upon the car as an extended interface to the cell phone. Audio, video, and control signals were directed to the vehicle for presentation according to car-specific requirements.

The role model for this approach was found in the almost forgotten work of AMI-C[23], completed a few years earlier (cf. Guglielmetti 2003). Although the

[21] Application Programming Interface

[22] In off-board navigation key features (e.g. map data access and routing) are remote services, deployed at a server, while other features (typically guidance and HMI) are deployed to the mobile client.

[23] Automotive Multimedia Interface Collaboration.

original AMI-C protocol had to be complemented (e.g. to support the vector graphics needed for the transfer of moving images), the solution performed virtually as expected. With this so called "streaming approach" Appello's base functionality was preserved untouched, while the HMI was adapted to the car context.

The validation of the streaming approach revealed a few technical weaknesses. The transmission of vector graphics across a wireless serial interface was a weak link. The solution faced an inherent latency issue that could not be easily resolved. Further, it effectively prevented customizations for the vehicle environment, using in car resources. However, the most important conclusion drawn from the demonstrator was not technical to its nature; it was increasingly clear to everyone involved that a non-functional interface, of the type used in the streaming approach, would transfer responsibility unconditionally to external actors. On the one hand, this was at the heart of the original idea; external actors would keep up pace of change. At the same time, the AMI-C-based interface would gladly relay any information for presentation in the car, as long as it complied with some basic specifications. Without influence over the nomadic device there would be no technical barriers, whatsoever, preventing a third party vendor from introducing functionality disapproved by CarCorp. In some sense, it would be possible to hijack the car.

Such an aggressive strategy would be exceptionally provocative to a conservative automaker. For several reasons the automotive industry is centered on explicit control for governance. One motive is related to production – the act of assembling components. Significant control of component interfaces ensures compatibility at the time of production. This aspect would not be compromised by the new perspective on openness. Another strong argument behind the dominant control agenda in the automotive industry is related to liability. Although a car is assembled of components from a wide range of suppliers, customers look upon it as a coherent product. Distinct hierarchy-of-parts thinking, manifested as modularity, is the established way to exercise control over this organizational hierarchy. To implement the new perspective on openness CarCorp simply had to give up this kind of architecturally enforced control and identify other mechanisms to govern largely independent actors. From this moment in time, the openness-control dilemma made a dominant discussion in the project. The ambivalence is demonstrated by the project manager:

> *We see great promise in the idea of developing a general API that gives third-party developers the opportunity to develop in-car applications. By definition, the problem is that we won't know what will happen. What applications will be developed?*

> *There are major stakes involved in openness; they involve huge uncertainty, ignorance, and some fear about which direction this will take.*

The idea of giving up control of the design process was obviously highly challenging to CarCorp. By tradition, suppliers are influential in the design of in-car functionality. Still, the automakers exercise significant control through architecture. Together, the decomposition of systems into components and interfaces between components define how products can evolve over time. Functionality may be designed at Denso, Delphi, and Harman Becker[24], but according to CarCorp's overall agenda. Opponents argued that the proposed model for openness would put CarCorp in a reactive position, rather than a proactive.

However, while losing control of design was problematic to CarCorp, losing control over the *product* was highly alarming. It was obvious to proponents as well as opponents that application software residing at an external device would be completely outside of CarCorp's control. Furthermore, the nomadic device would be malleable across the vehicle life time, while the car was essentially fixed. Consequently, there was no mechanism *binding* a particular configuration of nomadic software to a particular vehicle. Consequently, functionality would evolve over time and seek new meanings. Meanings over which CarCorp had no influence, what so ever. As reflected in a later discussion with GlobalCarCorp's top infotainment managers, this kind of openness was largely unthinkable:

> *At least in the United State we have something called product liability and, if we think that people could create something that they're gonna put in our vehicle and that is a distracting or somehow interferes with the primary task, then to some extent we are, we're liable because we've kind of opened the door to that.*

This reasoning suggests that customers would make CarCorp liable for any disloyal functionality developed after time of production. Again, a promising idea to solve the life cycle problem seemed to fail.

Summary and Epilogue

Nomadic device integration posed a new challenge to CarCorp; the product had to be architected for change, not just between generations of the product, but across the lifecycle of products. Architectural frames had to be combined in such a way that in-car infotainment functionality could evolve at a pace defined by the consumer electronics industry. Up until now,

[24] Major suppliers of automotive infotainment systems.

CarCorp had approached the network-of-patterns frame with the intention to generalize (MOST) and specialize (SoftCluster), but without closing the loop from the perspective of a given product. With NDI they envisioned an infotainment system where general functional patterns were continuously supplied by consumer electronics and easily provided to in-car users in a specialized form. With NDI infotainment would be able to evolve continuously, not just at discrete occasions constituted by the release of new car models.

In an early attempt, CarCorp experimented with an accessory-like approach to this new challenge. A state-of-the-art Nokia mobile phone was integrated with a car on the basis of existing physical and electrical interfaces. This hierarchy-of-parts approach to NDI relied on the same basic idea used when customers buy new wheels; given a fixed, modular decomposition of the system and permanent interfaces between parts the end-user is free to pick the wheels of his or her own choice and upgrade when appropriate. However, the experimental setup demonstrated to CarCorp that nomadic device did not offer stable interfaces. Manufacturers used different solutions and, even more problematic, these manufacturers continuously changed these interfaces.

Wide adoption of Bluetooth technology in consumer electronics injected new hope in CarCorp's NDI vision. With a standardized, well diffused, and non-physical interface, it once again seemed possible to approach nomadic devices as accessory parts that could be changed at personal preference. Bluetooth promised that general functionality, hosted by nomadic devices, could be inherited for specialized in-car usage. However, once again the idea of rigid structure of physical parts, preserved by stable interfaces turned out to be naïve. Bluetooth was a standard. Still, whether CarCorp liked it or not, it evolved at a pace defined by the consumer electronics industry. CarCorp was able to ship NDI solutions to customers, but after just a few years it could not support the latest devices.

CarCorp designers realized that modularity did not offer a durable solution for nomadic device integration. In perspective of automotive lifecycles, mobile phones, portable navigation devices, and other nomadic devices turned out to be far from the stable subassemblies prescribed by a hierarchy-of-parts frame. Further, such stability could not be enforced through standards, such as Bluetooth. Change could simply not be isolated to nomadic devices. In order to make use of the momentum in consumer electronics CarCorp had to find a way to deal with changing interfaces, not defined by them. With OPN, specialized functional patterns, hosted by the cars, would be able to evolve in harmony with generic functionality at the

nomadic devices. In practice, this meant that OPN software had to be updated across the life time of the vehicles.

In a hierarchy-of-parts frame stable parts are bootstrapped into ever more complex configurations. Decomposition defines interfaces between parts and, in turn, how parts can be aggregated into new products. To some extent, the original setup of specific patterns is preserved in the interfaces. Therefore, in a hierarchy-of-parts frame, the physical break-down of a product is a manifestation of control. In context of NDI, CarCorp was inherently deprived of this traditional mechanism to exercise control. In a network-of-patterns frame generic functional patterns are bootstrapped into ever more complex configurations. Generalization defines how patterns can be inherited and, in turn, how these general patterns can be reused in increasingly specific functions. In this architectural frame, control is exercised through general patterns. Performance improvements in positioning will translate to the more specific navigation function, while the opposite is false. Suddenly, CarCorp designers found themselves in a situation where they neither had control over interfaces nor over general functional patterns. It became increasingly obvious that the OPN solution would be rejected for security and safety reasons, but also since it clearly lacked governance mechanisms allowing CarCorp to capitalize on its investment.

Triggered by these insights, Appello's software designers highlighted that porting could be accomplished simply by running their cell phone application at the gateway. At the time, cell phones were open to third-party software under the limited premises given by the Java sandbox[25], provided by most manufacturers. Appello had successfully exploited this opportunity to leverage a device independent navigation solution. A similar Java environment at the gateway would offer a solution to CarCorp's problems. Smooth porting of *existing* applications, originally developed for cell phones and other devices, would allow CarCorp to appropriate value from consumer electronics. At the same time, it would give reasonable control over innovation processes. Hosting the Java platform in cars, rather than nomadic devices, would give CarCorp control over general functional patterns. Such control would not just reduce liability issues by bring influence over specific application software, but also allow for efficient integration with the rich set of resources offered by the car. It would make an opportunity to govern external innovation, rather than just follow it.

[25] In computer security the sandbox metaphor refers to a mechanism for separating running programs.

At this moment, CarCorp designers started to rethink the role of the gateway. They had used the notion of platform when discussing the gateway for a longer period of time, but essentially to point out that it would support a range of different nomadic devices. From now, they started to see it as a coherent enabler of hardware and software resources allowing for the execution of a wide range of applications.

Consequently, they complemented the original demonstrator, turning the gateway into a full blown Java platform. What emerged as the "host solution" successfully demonstrated a high-performing port of Appello's navigation, executed at CarCorp's in-car platform. Latency issues vanished and software designers easily modified the software to align with the interaction resources provided by the car.

When at the end of the project reflecting upon the transition between gateway and software platform, project members saw both opportunities and challenges. On the one hand, running the software at CarCorp's platform enabled a set of new tools to handle the intricate liability issue. While controlling the platform, it suddenly seemed possible to dissolve the hitherto distinct boundary between wide open and closed. A smart platform strategy could reasonably be used to enforce CarCorp's agenda, while still not internalizing the process of developing applications. As one of the consultants involved in the project noted, it would allow for a gradual transition between the traditionally closed model and a truly open one.

> *CarCorp must start by offering services and applications that they control. It'll be extremely difficult to open up the system to everything…. The first step will be to release some of the control and to work with third-party application providers that can offer some new services.*

At the same time, the NDI proponents began to see that this new combination of architectural frames would pose a whole range of new challenges to the organization. An open platform under CarCorp's control would enforce new perspectives on product planning, purchasing, production, marketing, and sales. It would come with new product offers, new forms for supplier collaboration, and new business models. Shortly, it would disrupt the existing organization structure. At this time no one could see a way to implement this transition smoothly. In the next section (5.4) we follow CarCorp's progress in seeking novel solutions to these substantial challenges.

5.4 Android: Designing for Generativity

2007 CarCorp had reached a point in its transition of innovation practices where the concept of open innovation was recognized and to some extent

accepted as a solution to the challenges facing infotainment. This journey can be viewed as a gradual uncovering of the network-of-patterns frame. Projects such as MOST had introduced *generalization* (5.1). By identifying, specifying, and reusing a range of general functional elements, across different components, CarCorp had designed an infotainment system where applications made better sense together and offered significantly more harmonized functionality. Later, the SoftCluster initiative demonstrated that such general and reusable elements, when released from the grip of components, had impact on innovation far beyond streamlining of functionality (5.2). It turned out that a platform making general functionality available and accessible accelerated creativity in design teams. In an act of recurring *specialization* new functions could be generated continuously, as general patterns were easily combined in new ways. Finally, CarCorp's commitment in nomadic device integration showed that generalization and specialization are *intertwined* phenomena (5.3). Unless the general patterns of the platform and the many specialized applications using it evolve together the generative capability will inevitably decline. Drawing on the many experiences from nomadic device integration CarCorp started to see how such evolution could be governed; it was critical to take control of the platform. The general patterns of a platform are inherited by specific applications. This inheritance creates a unilateral relationship, where specific patterns rely on general, while general patterns can be described independently of specific. In practice, this translates to an opportunity for platform owners to exercise control over application development. Although seemingly evident, this offered a distinctly different perspective on governance. Traditionally, CarCorp and other automakers specified *specific* functionality, while leaving the functional breakdown in the hands of suppliers. Largely, they governed innovation processes through the structural decomposition of the system into components.

As illustrated by a statement of the director of controls and software engineering at GlobalCarCorp, designers and engineers considered traditional linear innovation processes outdated for infotainment:

> *This idea of being five years ahead to predict what future our customers are going to be in only means [that] what we deliver is irrelevant.*

The only way to keep up with consumer electronics would be to actually get involved, reduce existing barriers, and encourage the consumer electronics community to keep the car up-to-date on their own premises. Coming infotainment systems could not be up-front designed; they had to emerge in a continuous interplay between CarCorp and external actors. An open platform, under the control of CarCorp seemed to be a way forward. Given a

capability to draw attention, it could take the creativity and multiplicity of consumer electronics to an automotive setting. The only thing was that the notion of openness and the concept open platform were fuzzy phenomena. Neither the software team nor the organization as a whole shared a view on them or a language to discuss them. Although blurred in its contours most people agreed that the value of an open platform was largely manifested in its capability to boost uncoordinated, creative processes. As underlined by a GlobalCarCorp strategist, such value is tightly connected to multiplicity and diversity. Unless an open platform is able to generate such multiplicity it is essentially useless.

> *the way you establish value for this open platform is this idea that you have to be able to look at hundreds of ideas, and then you're going to see the value. The minute you limit what you're going to look at, by the nature of the beast, you have basically eliminated your value.*

With brutal precision the statement emphasizes that the purpose of an open platform is to generate options – *digital options*. Unless platform owner, developers, and end-users could find a way to do this together, an open platform approach would fail. To succeed they had to find a model where designers' creative leeway could be balanced towards CarCorp's need for influence and control.

Despite the many promises it was increasingly clear to managers and designers that an open innovation approach to infotainment would require not only R&D staff to reconsider the hierarchy-of-parts frame, but essentially the whole company. Rather than placing well defined *orders* on tier-1 suppliers, CarCorp would make *offers* to independent developers. This would enforce new perspectives on product planning, purchasing, production, marketing, and sales. It would come with new product offers, new forms for supplier collaboration, and new business models. Shortly, it would disrupt the existing organization structure.

Although CarCorp did not know how to address all these challenges, it was increasingly clear that an open innovation practice would require them to close the loop between generalization and specialization. The platform and its wide range of applications had to evolve together, in reasonable harmony, while at the same time preserving revenue generation for GlobalCarCorp. In architecting such an infotainment system they predicted two main challenges. First, generalization could not be seen as one-off activity, taking place in isolation from application development. Instead, the platform had to be architected for continuous adaptation to developers' shifting needs. Second, application development would not occur out of nowhere; it would be necessary to set up an attractive innovation ecosystem, hosting

specialization of infotainment functionality on CarCorp's platform. Consequently, CarCorp launched two advanced engineering initiatives; one project to identify and specify the next generation infotainment platform and another to delve into the non-technical aspects of the open innovation concept, with particular focus on business models and developer ecosystems.

Designing the "Next Generation Infotainment Platform"

The task to identify and design a new infotainment platform was formally assigned to CarCorp by its parent, GlobalCarCorp. Originally, it was framed as a study of how to apply the successful SoftCluster platform to infotainment (see p. 130). Fall 2007 the software manager put together a review team of employed engineers and external specialists to set up the guidelines for such a new software-centric infotainment platform, intended to be shared by all the brands within GlobalCarCorp. It soon turned out that the SoftCluster concept could not be transferred to infotainment unless it was extensively modified. It allowed for easy modification of the specific look-and-feel in instruments clusters, but only given the fundamental rules defined by the platform. It turned out that the macro-oriented approach to HMI development was far too rigid for an infotainment context. Inheriting the SoftCluster architecture would essentially prevent the platform from evolving over time. Therefore, the team found themselves facing the challenge to develop a whole new platform concept, with little possibility to reuse existing solutions. This new platform had to be malleable to the changing functional requirements in automotive as well as consumer electronics far beyond what could be offered by SoftCluster. It soon became clear that this process would be both painful and difficult.

To define the limits of CarCorp's assignment, systems architects at GlobalCarCorp made an outline of how the new infotainment system would fit with the rest of the car. With some dismay the team found out that GlobalCarCorp's architectural outline was a traditional, modular breakdown of the system. It assumed a hierarchy of components, allowing the automaker to scale the infotainment system, from the most basic low-end solution to premium configurations in the high-end segment.

In a first workshop, trying to understand and make sense of the architectural outline, one of the team's hired software specialists underlined that flexibility to adapt functionality does not resonate well with a distributed solution, where functionality is inscribed in components. With such a hierarchic setup of the system retrospective adaptation of functionality tends to increase coupling between components, inevitably making the system increasingly monolithic.

> *When you make drawings it makes sense using many components. That's how you make architecture – by drawing. That's how we all do. But we must not fool ourselves. It is tempting to distribute systems until you have a whole range of [physical] components, consuming huge amounts of resources. Then, suddenly you find yourself in a situation where even low-end cars require a full configuration for things to work out. Then you're screwed.*
>
> *This problem is a lot easier to handle [with a software solution]. How do we most effectively handle variants? By hardware or by software? That's an important question [for GlobalCarCorp]. We need to keep in mind that this [document] is a draft. It makes a set of more or less spontaneous ideas.*

CarCorp's team leader makes a similar statement, underlining that the new infotainment system must be architected on new premises.

> *Scalability can cause a lot of damage to software architecture if we end up with a lot of variants. [...] Scalability and cost optimizations will not give us the best architecture. It is something different. I'm not sure these criteria [at all] apply to software architecture.*

Over time it became increasingly clear to the new software team that a traditional hierarchy-of-parts thinking stood in opposition to the kind of malleability they envisioned. GlobalCarCorp's hierarchy-of-parts approach would give them a range of well-defined, stable subassemblies, each streamlined for a specific functional purpose. While this would allow them to differentiate the offer across a range of different car models and brands, it would effectively prevent the system from changing over time. Given the overall vision to make infotainment increasingly open to external innovation it was increasingly clear that CarCorp had to give up some of the advantages offered by a hierarchy-of-parts frame in order to benefit from network-of-patterns thinking. In practice that meant a position *against* distribution of the system. As far as possible the functionality of the new infotainment system had to be deployed to one component – a component hosting the software platform.

Redefining the Scope

With GlobalCarCorp's attempt to define the scope of the project in mind, the software team decided to make an aggressive move and define the limits of the project themselves. In order make up a solid guide in their work, without damaging creativity and bold ideas, they developed a "project one-pager". This brief project outline summarized a vision, critical aspects, and key enablers. From an architectural perspective there are several statements worth mentioning. First, the team established that the mission was to "build

a platform, not an implementation." This seemingly uncontroversial statement marked that the project did not focus on the development of specific infotainment functionality. Instead, the objective was to engage in generalization and build generative capability that, in the hands of internal and external designers, could make the basis for independent and relatively unconstrained innovation. From an automotive perspective that was a major break with traditional, linear processes, always starting with functional specifications. Being careful about using the notion of openness to describe envisioned innovation practices, the team established that the platform should offer "support for plug-in software". Such so called plug-in software was defined rather broadly, ranging from "CarCorp managed" to "3rd party aftermarket developed software".

It is worth emphasizing that although CarCorp had engaged in generalization before (e.g. the MOST project) it was now done on different premises. The objective was not primarily to build a coherent and harmonized system. Instead, as described, the upcoming infotainment platform aimed for innovation practices, where internal and external designers could engage in specialization independently from infotainment experts at CarCorp. To reinforce this position the one-pager declared that "[design] decisions and [project] focus should be business-case driven, not technology driven." Thereby, it was critical to design a platform that allowed for CarCorp to *appropriate value* from increasingly independent innovation practices. Drawing on recent failures (the open platform for nomadic devices was eventually turned down) CarCorp had arrived at the conclusion that distributed, software-centric innovation, of the kind they envisioned for infotainment, could not be governed with less than significant influence over the general elements of a system. Therefore, they argued; "To be in control of [platform] SW is and will be very important". To once again emphasize that such control did refer to the enforcement of a functional agenda, the one-pager stated that "control does not mean doing-it-all-yourself".

Finally, it is important to show that the software team, from the beginning of the project, recognized that generalization had to be a recurring, continuously ongoing activity. Envisioned innovation practices would neither emerge nor persist without a living and fertile interplay between generalization and specialization. As described in the one-pager, the platform had to evolve in reasonable harmony with innovation practices.

> *To be able to match and to interface to quickly developing consumer electronics, the system must be able to mature, both between model updates and in the after-market.*

Searching for Platform Concepts

In relative agreement on the task ahead the software team initiated what they referred to as a concept selection process. On a general level, they expected this process to result in an open platform strategy. In a first phase they agreed on a range of evaluation criteria; cost, technical challenge, business challenge, quality, liability and responsibility, flexibility and malleability, incentives and motivation, finally, what they labeled suitability for automotive applications. Second, they initiated a long discussion on how to identify and describe credible and realistic concept alternatives to evaluate towards each other. It was clear that the critical axis of tension would be found in the interplay between openness and control. The platform had to make an attractive offer, allowing for relatively unconstrained innovation to take place. At the same time, GlobalCarCorp would require influence and, ultimately, the right to veto inappropriate applications. It was also clear that control had to be exercised through the platform, since the opportunities to set up legal agreements would decline in an open environment.

Over time the team found two dimensions, guiding the work to identify concept alternatives. First, they discussed intensively whether the platform should be "public" or not. Being programmers they used the notion of public (in contrast to private) to identify whether a platform was unconditionally open to external parties or not. In practice, such a public platform would allow for external actors to get full access to the platform to develop software application without CarCorp's approval. From an architectural perspective this translates to the question of exercising hierarchy-of-parts control or not. With the software platform deployed to a physical component in the car CarCorp had, in a very practical sense, the key needed to unlock the software platform for external development. Therefore, the hot topic was whether they should make use of this opportunity, keep the key, and actively gate keeping introduction of new applications or release the key in public and allow for functional evolution outside GlobalCarCorp control.

Second, the team argued intensively whether the upcoming infotainment system should be grounded on a GlobalCarCorp platform or an established off-the-shelf platform. Designing the platform internally would offer great opportunities to exercise network-of-patterns control as the process of generalization then would be in their own hands. With control over the general functional elements offered by the platform, CarCorp would preserve influence over applications, even though development went increasingly public. On the other hand, such a strategy would leave CarCorp with the task to continuously align with volatile requirements of the consumer electronics community. In addition, it was increasingly clear to the team that installed base was a critical concept in open innovation environments. The only way

to build multiplicity and generate digital options would be to offer a well diffused platform. Whether GlobalCarCorp would be able to build such a significant installed base on its own was a major question mark.

Given these two dimensions, the software team singled out four concept alternatives (Figure 5), described in the document "Next Generation Infotainment Platform: Proposals for Concept Selection"; A "GlobalCarCorp 'open' API platform", an "industry standard 'open' integration platform", an "industry standard 'open' public platform", a GlobalCarCorp specific 'open' public platform".

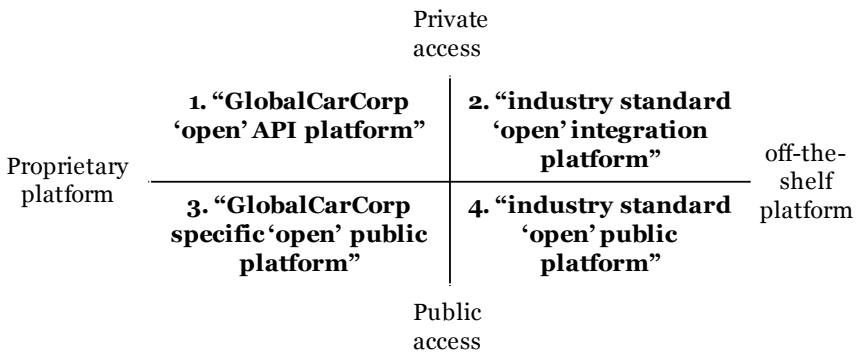

Figure 5. Key dimensions in platform benchmarking.

1. The "GlobalCarCorp 'open' API platform" would enable a fast follower approach, where the automaker could domesticate successful consumer electronics initiatives and make them fit with a platform essentially designed for automotive industry needs. It would allow CarCorp e.g. to give the platform APIs "to a navigation engine supplier and source them for delivering an adopted [and diffused] navigation engine ready to integrate in the GM infotainment platform".
2. The "industry standard 'open' integration platform" would take the follower approach a number of steps further by actually adopting an established platform, frequently used by consumer electronics communities. It would allow CarCorp to "buy and integrate off-the-shelf software components".
3. The "GlobalCarCorp specific 'open' public platform" would enforce another stance on openness. Essentially, this approach was grounded in the idea that a public platform would trigger new, external innovation processes, feeding GlobalCarCorp with novel applications. In practice, they would "create a public and open GlobalCarCorp-specific run-time environment similar but different to Symbian, MIDP and .NET environments". With this approach the automaker would "rely on and encourage software module suppliers to develop and create [automotive

related] business on this GlobalCarCorp platform". Consequently, this class of platforms was not just a tool in a follower approach, but a way to inject new momentum in automotive innovation.

4. Finally, the "industry standard 'open' public platform" was seen as an initiative to merge automotive and consumer electronics innovation. It was "a platform approach aiming for adoption of widespread consumer electronics frameworks". In this vein, the main purpose was "to minimize entrance barriers for CE actors, interested in porting their applications to the automotive environment". The team envisioned two different models for such innovation; either the platform would be used to "support co-branding under competition, where partnership with a strong brand is used to strengthen the vehicle brand". In such a scenario GlobalCarCorp would be in control of the partners invited to competition. In the other, more aggressive approach GlobalCarCorp would agree "to fully open the platform for third party development and distribution of software". In practice, it would mean that "the customer can download standard SW modules with minimum or no integration work".

Getting Management Support

The four concept alternatives can be viewed as a way to maneuver in the minefield of tensions unfolding as the team started the demanding process of translating ideas into practice. On the one hand, they were committed to implement an architecture that could support more open innovation practices. On the other hand, they were embedded in an organization that would resist attempts to introduce openness in many different ways. Launching the four concept alternatives was an attempt to balance the different aspects and opinions that would inevitably materialize in the wakes of an open platform. In order to accomplish any change at all it would be critical to build support in the organization.

Although the project had started in minor, with GlobalCarCorp's attempts to enforce a hierarchy-of-parts perspective, it suddenly took an unexpected turn and continued in major. Unexpectedly, the director of controls and software engineering at GlobalCarCorp showed great personal commitment to the project. He gave his full support to the idea of rejecting decomposition of the system and focus on a software platform, deployed to one component.

However, he also recognized other aspects with explicit implications on how to exercise and balance architectural frames. In particular, he could see and articulate the tensions between the automaker's internal need for control, grounded in hierarchies-of-parts thinking, and the need to create deregulated, open, innovation environments, able to attract creative people and organizations, seeking to realize their ideas and dreams. On the one

hand, CarCorp had to reinforce control over the software platform to exercise influence over innovation processes. On the other hand, a proprietary CarCorp platform would draw little attention and build limited installed base, offering little chance to accommodate the changes in consumer electronics. The platform had to be open, yet, at the same time, making the basis for the automaker's influence over external application development. The director of controls and software engineering argued that the only credible way to resolve this contradiction would be to adopt an existing open source platform.

> *I would say we need to take control through an open source [platform] initiative and by our contribution to it, approve it. Because I think take control of it in a proprietary sense is still not going to create the crowd.*

No doubt, this position was exceptionally controversial and provocative in an automotive setting. To most people in the industry open source was an almost bizarre phenomenon. How would a relationship with an open source community be manifested? Clearly, traditional governance logic would not apply. As anticipated by the director, this kind of movement would trigger strong reactions by established institutional structures.

> *…we have standard terms and conditions that everybody signs, because those are readily acceptable in the industry. They are fair terms, right. Those terms do not apply here, and the fact there is no really readily acceptable standard terms in the software industry, so everything is a negotiation. You have to start from a decent place, and then you got to negotiate teeth-to-teeth, and our legal guys don't even get that concept. We have our standard ones – you sign it or go away. That's not going to work. You have to have different ones that you start with and negotiate, so take that, that's just buying commercial proprietary cuts, and tell these guys we want go open source. You've totally ripped the foundation that they are standing on from underneath them, but it is exactly what we need to do.*

Although anticipating rock-solid opposition, in particular from legal departments, GlobalCarCorp's director of controls and software engineering injected hope and commitment in an open source agenda.

> *in the end – here's the kicker – in the end, the lawyers don't run the company, right, they only make recommendations.*

In an attempt to clarify and reinforce his position, the director identified an open source platform as the only credible alternative for an automaker to engage with community-based innovation, being increasingly important to contemporary consumer electronics.

> *I absolutely believe this is where we have to go – this is the game changer. And this is the thing that solves the problem of how do you stay relevant in infotainment telematics in a company that fundamentally operates at a speed that will make you irrelevant, right. And that is, you have to separate it, right. I mean you've got to get out to the communities that moves fast, which is open source, and you got to install a platform in the vehicle that can accommodate that innovation, and you got to kind of separate the life cycles, right.*

By repeatedly returning to and elaborating the *role* and *meaning* of the platform in practice, rather than its technical properties, the director increased the attention on a question that CarCorp had just started to study in another project; how could an automaker initiate, moderate and maintain a productive developer community, centered on the platform?

Forming a Platform Ecosystem

In parallel with the platform assignment, CarCorp had initiated a project with specific focus on new business and innovation practices. This project was rooted in the late experiences of integrating nomadic devices (5.3). In retrospect, managers and designers could see a breakthrough in the struggle with nomadic devices, although the open platform for nomadic devices did not translate into a commercial product. First, they had internalized a totally new view on governance, where the general patterns of a platform unfolded as critical elements in governing application development. However, they had also realized that such a platform would be essentially useless unless it would draw attention and collect crowds of developers. Without diffusion of the platform there would be little multiplicity and, eventually a limited range of digital options for GlobalCarCorp to capitalize on.

Therefore, a new project was initiated late 2007, with an explicit ambition to study the business conditions in more open innovation environments. As illustrated by a senior consultant it was increasingly clear to the inner circle of infotainment designers and managers that the major challenges of launching an open platform would be "organizational and on the business side". The critical questions would be:

> *How should we sell this? How do we market our applications? How do we earn money? This is the fundamental issue!*

The original project application, presented to get internal funding, declared in a somewhat vague manner that:

> *a business strategy for Open Source Software/open API and open innovation for automotive applications will be developed. This would require an open platform, probably the infotainment system should be targeted.*

According to the project descriptions, the main motivation behind such a business strategy related to two critical questions:

> *How can we increase the speed and flexibility of implementing new functionality and features?*
>
> *How can we increase the capability of detecting, exploiting, and developing new use cases?*

Largely, the participants of this exploratory project spent their limited time to figure out how an open innovation approach could help them address these questions in an automotive context. In order to provide concrete illustrations, they derived and described four alternative strategies (Table 4).

Table 4. Proposed strategies for open innovation at CarCorp.

Strategy	Description
Enhanced Crowdsourcing	This approach was framed as a way to reduce burdens of R&D. By balanced involvement in open source projects CarCorp had a potential to leverage the possibility of mass collaboration for reducing cost and improve software quality.
Software Accessories	The "software accessories" approach referred to a strategy focusing on building an aftermarket business around infotainment software for connectivity personalization, and new functionality. Rather than creating new revenue streams, this strategy aimed to reinforce existing ones. Therefore, the primary objective of this strategy was to generate rich and potentially free complements, increasing the value of the core business – selling infotainment systems.
Maintenance through Open Source	This strategy suggested that proprietary application software should be donated for open source governance as new generations emerged. It would allow CarCorp to focus resources on core development and capitalize on investments, while securing customer satisfaction over time. In its ambition to reduce cost and, at the same time, preserve customer loyalty it can be view as a mix of the two former.
Semi-Open Competition	Finally, the "semi-open competition" was launched as a strategy to assign development of core applications to external 3rd party suppliers. Rather than reinforcing the CarCorp brand by offering proprietary infotainment applications this approach recognized that that the value of established and acknowledged brands, owned by external actors, could be leverage through strategic partnerships. In other words, these strategic

	partnerships would be used to reinforce the CarCorp brand. As the project emerged, it became increasingly clear that this approach would redefine the automaker's core business and force them to capitalize on complementary assets, rather than core applications.

Together, the four strategies made clear to CarCorp that open innovation was not a well defined phenomenon, ready to be uncovered in different advanced engineering projects. Instead, it was a concept with margin for interpretation. The tricky thing would be to identify a strategy which successfully could exploit external creativity to expand horizons and generate a wide range of applications, but without challenging established innovation practices too much. It would simply be a bad idea to create too many enemies by questioning the rationale of the organization.

Trying to launch a reasonably aggressive but realistic agenda, the project foregrounded the potential in partnering with new actors, deeply anchored in consumer electrics. Such partnerships would enable critical competence, in a reasonably controlled form. It would be controlled in the sense that it would be possible to apply traditional and familiar governance instruments, such as legal contracts. However, the project team could not overlook the potential in a more radical approach to open innovation. Therefore, as described by the project manager, the team agreed to propose a complementary strategy, aiming for novelty and originality.

> *We foresee two models for how to introduce new applications. One option is to actively seek partnerships where brands and products reinforce each other [without cannibalizing]. The other one is [unconstrained] open innovation, where you do not really know what is going to happen, where you allow yourself to be surprised by people and their ideas. Together, these two models will pave the way for great products.*

Clearly, the more unconstrained view on open innovation called for a new playing field. Studying other successful initiatives, it was rather obvious that developers would not just gather around a platform. As underlined by GlobalCarCorp's director of controls and software engineering, being interview by project members, the platform would not deliver open innovation on its own. The aggregate potential in platform and developers would not unfold without a community.

> *We [also] need a community that is willing to develop on it, right. We need to be part of that community, and in some cases try to out-innovate, and we need to be attractive [enough] for the innovators to come to us and say we would like to get this in your vehicle and we have a more predictable, guaranteed safe*

> *distribution mechanism than just downloading of the open source.*

At this point, the locus of the project changed as resources were redirected to delve deeper into the concept of developer communities, later recognized as innovation ecosystems. Developer incentives became top priority as well as trying to understand the interplay between developers and platforms. Almost reluctantly CarCorp started to realize that ecosystems would feed a quite different perspective on platforms, at least compared to the view adopted by their own engineers. Engaging in specialization independent developers would interact with the platform through a software development kit (SDK), consisting of a whole range of tools allowing for the creation of software applications. Further, they would draw on application programming interfaces (API) to reuse and recombine general patterns offered by the platform. Largely, the complex software stack, making up the platform in the eyes of CarCorp's designers, would be invisible to external developers. Suddenly, the disconnection between platform and applications became very real; it would be fully possible to design infotainment applications to CarCorp without real-life contact with car or its different physical parts. Specialization and generalization would be interlinked through a few critical boundary resources.

These findings played an important role when CarCorp 2008 pushed the project into a new phase, aiming for industrial implementation of the ideas in practice. Leaving the exploratory character behind, the team adjusted and refined the objectives of the project to focus on a few critical issues; the constitution of a developer program and the identification of critical boundary resources. In short, the objective of the second phase of the project was:

> *To develop an open innovation concept for next generation infotainment systems including a developer program, platform-community boundary resources, and process innovation.*

While the first phase was implemented on a skunkwork basis, involving a very limited group of people, the second phase was rolled-out widely across the entire organization. To get maximum support the project manager engaged people from different parts of R&D, marketing, sales, product planning, and aftermarket. In addition, she involved two key players from Sony Ericsson with long term experience from partnerships programs and community management.

Over a period of approximately one year CarCorp's conceptual studies of innovation ecosystems and open innovation platforms continued side by side, while eventually merging as the company made a bold decision; the

next generation infotainment system – including in-car platform and innovation ecosystem – should be based on Android[26].

Platform Selection

In February 2010 CarCorp's executive management board decided to adopt Android as a basis for infotainment in the upcoming mid-sized car, with an ambition to apply it for all models in the longer run. It was indeed a bold decision, with many unresolved question marks in the margin. Still, it was not a reluctant or doubtful board approving a major investment in Android – a platform they did not control. On the contrary, these executive managers expressed their enthusiasm and strong support. In retrospect, the infotainment people could see that this massive support followed from the simple fact that although the proposed solution was flawed by lack of precision it gave credible answers to a whole range of critical questions. Top management could see through weaknesses and shortcomings since the proposal offered a coherent solution to a whole range of management and business challenges. It made sense together, as a whole.

Judging by retrospective statements (see Table 5), CarCorp management paid particular attention to Android's capability to:

- Enable a ***recurring infotainment business*** in the form of aftermarket applications. In practice, this opportunity was grounded in a decoupled relationship between the car, hosting the Android platform, and application development. Infotainment functionality would be able to evolution continuously, without considering car lifecycles.
- Generate ***multiplicity and diversity*** by drawing on an established developer community and an exceptional installed base in various consumer electronics devices.
- Secure ***sense-and-respond capability***, allowing for quick turn-around on ideas and, thereby, enable state-of-the-art infotainment to CarCorp customers.
- Support ***domain-specific extensions***, being a basic condition for car-specific innovation. In fact, the capability to extend the platform with car-specific, general patterns would be crucial for CarCorp to appropriate value from an Android community and, therefore, necessary to secure revenue streams and profitability.

[26] Android is a Linux-based operating system for mobile devices such as smartphones and tablet computers. It is developed by the Open Handset Alliance led by Google (Wikipedia).

- Support a versatile ***governance*** model, where CarCorp could balance between openness and control in a pragmatic way. Without a clear strategy for how to address fundamental security and safety threats open innovation would stay a beautiful vision in the automotive industry.

While reflecting back on the long process that eventually made CarCorp go for broke on Android, the two main protagonists emphasized the same aspect; without combining the business oriented ecosystem project and the techno-centric platform project they would have followed a different path. Without understanding the interplay between community-based specialization and platform-based, continuous generalization of functionality the promises of open innovation would have remained a distant, unreachable vision.

> *I think the ecosystem project made a huge difference. It changed our way of thinking. That kind of thinking has to be around for a while to understand. You need to hear it over a longer period of time to craft a clear vision. If not, it is easy to do what we first did; pick a platform [on technical premises] and see what happens. Without the ecosystem project I'm pretty sure we wouldn't have taken the step to go for Android. Then, we would have had some other kind of open source platform, with a vague idea that such platforms can handle reuse. I think it is pretty interesting. I mean, how it actually influenced our way of thinking. […] Now, we have rock-solid support for this. This is it!*

The project manager for the ecosystem project agreed with her colleague, but underlined that the new model for how to view infotainment was not just an outcome of these two contemporary projects, but had emerged from many different projects, implemented over several years.

> *I think this journey has been incredibly important for our company. We have now sanctioned the project throughout the organization and received a great response… This wouldn't have happened without the early efforts. I don't think that the company has been mentally prepared to make this journey until now… It's fantastic. Sometimes I have to pinch my arm, confirming that I'm not dreaming. So many years, so much fighting, and suddenly it happens and everything works out - it feels very strange!*

Table 5. Key argument behind the selection of Android at CarCorp.

Recurring business	Multiplicity and Diversity	Sense-and-respond	Domain-specific extension	Governance
"[This solution] will allow infotainment services to constantly evolve during the lifetime of a car's product cycle, unlike current in-car systems which are fixed some years before a car goes on sale and then remain static."[27]	"The number of already existing [Android] applications is a huge advantage. It's extremely efficient in terms of development effort. It takes the focus away from technical development to business development."[28] "[With our Android-based system] there are no limits to the potential for innovation. […] We will be inviting the global Android developer community to use their imagination and ingenuity."[27]	"[This solution] provides a faster, more efficient and more flexible alternative to the conventional, in-house development of vehicle infotainment services."[29] "Our open innovation strategy, using the Android operating system, will keep the provision of in-car infotainment up to date."[27]	"CarCorp will issue third-party developers with a vehicle application programming interface (API) providing access to more than 500 signals from different sensors in the vehicle."[29] "CarCorp's 'open innovation' strategy offers the global developer community access to the full bandwidth of car communications — infotainment, telematics, systems monitoring and diagnostics."[29]	"To ensure [that] high driving safety and quality standards are maintained, programs from software developers and application providers will be evaluated and approved by CarCorp before they are made available to customers."[29]

[27] Director at CarCorp Aftersales.
[28] Infotainment project manager.
[29] Written press material

6 Discussion: Generative Product Design

In this thesis I set out to explore how product developing firms build new innovation practices to leverage the generative capability of digital technology. I have approached this task from a technological change perspective. In one way or the other new technologies arise from combination of existing technologies. While this process of combination is powered by forward-looking visions and a desire to accomplish new goals, it is also highly characterized by its legacy – the genesis of a particular technology largely defines how it can be reused for new purposes. The legacy simply makes some directions of progression "much more compelling of attention than others". Often "advance seems to follow advance in a way that appears almost inevitable" (Nelson and Winter 1982, p. 258). I have used the concept of innovation regime to give a concrete face to the inherent logics defining how a physical component, a piece of software, or an algorithm can be reused and recombined with other artifacts. The "ground rules" of an innovation regime defines how a particular technology over time "bootstraps itself upwards from the few to the many and from the simple to the complex" (Arthur 2009, p. 21).

Existing literature gives solid evidence for the idea that physical products and digital technology change according to different logics. Therefore, as tangible products are increasingly digitized, distinct innovation regimes clash into each other. In reality, this clash is manifested as tensions between the legacy of established practices and existing technology, on the one hand, and the potential in upcoming digital solutions, on the other. To leverage the

generative capability of IT organizations have to resolve or at least manage these tensions.

In this thesis I have studied how product developing organizations use and develop architecture and architectural thinking to cope with these tensions. Broadly, organizations engage in architectural design to manage complexity. It is worth pointing out that complexity is not an invariant aspect of technology. Rather, "how complex or simple a structure is depends critically upon the way in which we describe it" (Simon 1996, p. 215). Still, such descriptions make rigid templates for how technology can be combined. Thereby, the architecture makes a link between historical achievements and future potentialities. Architecture is a strategic tool that, properly exercised, can be used to gradually reinforce sound ideas in a series of "structure-preserving and structure-enhancing transformations" (Alexander 1999, p.79). In other words, architecture is an instrument for path creation, but, at the same time, a shackle of path dependency. Whether product developing firms will be able to transform innovation practices and leverage the opportunities of IT relies, to a significant extent, on their capability to fertilize new architectural perspectives, resonating with the opportunities of digital technology.

To give a distinct perspective on how product developing firms architect digital products, I have developed and applied a theoretical framework that culminates in the concept of architectural frames. First, this framework takes off from the assumption that digital innovation cannot be understood unless we shift focus from physical properties to affordances. While the potter's creative leeway is defined by the plasticity of his clay, the creative work of a software designer plays out in a virtual realm of representations. An algorithm affords the designer to solve a particular problem, but cannot be traced to physical quantities in any form.

Second, the framework offers a model for understanding the implications of digitalization on product development through two distinct affordances; programmability and replicability. Replicability affords instant replication, without exercising marginal cost, while programmability affords separation of meaning and functional characteristics from the physical artifact. The presented model put emphasis on two distinguishing barriers in product innovation, largely destroyed by programmability and replicability; the transition between functional design and physical design and the transition between design and production.

In product innovation, the transition between design and production triggers substantial marginal cost. Harsh competition over price force product developing firms to develop economies of scale, where massive investments

in specialized assets, such as tools, supply chains, plants, etc, pays off in terms of low unit cost. At the same time, these firms have to differentiate their products to stay competitive. Consequently, it is critical to architect products in such a way that all these specialized assets can be reused across variants and generations of the products. Products have to be architected for producibility. Modularity offers a solution, based on well-defined, highly autonomous components, forming scalable systems on the basis of a predefined, hierarchical template. This template prescribes how to recombine these components to leverage functional variation. As a consequence, this architectural strategy causes a barrier between functional design and physical design since modularity requires the overall functionality to be defined at an early stage for deployment to dedicated components. The critical task to design components cannot be initiated unless the whole picture is acknowledged. In practice, this unidirectional transition marks an irreversible shift in focus, from solving functional problems to the challenge of designing an artifact that can realize and mediate that functionality in a competitive manner.

To some extent, programmability and replicability pull the rug from under the feet of traditional product innovation. The opportunity to design and produce functionality without considering the physical wrapping or the cost of implementing it opens up for genuinely new paths of technological change. With functionality increasingly detached from the artifact in itself, complexity does not primarily play out in the structure of products, being at the same time functional and producible. The architectural challenge is increasingly less a matter of reusing assets for new configurations of a product. As a consequence the introduction of digital technology in product development redefines the role of architecture and triggers a shift in architectural thinking.

The third part of the framework introduces the concept of architectural frames as a way to view architecture in digital product innovation. With this theoretical model, making up two idealized representations of a complex product's architecture, I suggest that digital product innovation rests on two architectural pillars. On the one hand, cars, airplanes, and washing machines will remain physical artifacts that have to be architected for reuse of assets, with producibility in mind. At the same time, accelerating digitalization blurs the functional boundaries of these artifacts; meanings and perceived experiences are not inscribed in the products, but can emerge over time. To reinforce such generativity digital products have to be architected for reuse of ideas.

To shed light on this duality, the architectural frames model extends the established architectural thinking in product innovation, largely grounded in

Herbert Simon's near decomposability, with a complementary view, based on the legacy of Christopher Alexander. In perspective of technological change, Alexander takes a position which assumes that progression cannot be fully understood if we seek to explain the genesis of an artifact through its predecessors. He rejects the idea that price/performance ratio is a dominant selection mechanism. Cheap, mass-produced products fulfilling a set of market-standard requirements do not always win in the sense that they make basis for coming generations. If they did, the world would be a lot more repetitive, uniform, and dull. Instead he argues that "the characteristics of any good environment is that every part of it is extremely highly adapted to its particularities" (Alexander 1999, p. 74). Following Alexander, it is not stable subassemblies, manifested as physical artifacts, that translate from generation to generation, but the *ideas* generating these adapted artifacts. When solving a particular problem, in a given context, our first question is; how did others solve similar problems in other settings? We do *not* start looking for existing building blocks to play with.

According to Alexander, technological change has to be understood from the perspective of processes. The genesis of an artifact unfolds when we zoom in on the processes, generating the palette of highly adapted artifacts we can see around us. The task of architecture is to map, represent, and enable our problem solving heritage to designers, creating new artifacts. Thereby, architecture plays out in a problem-solving domain, rather than in a domain of physical things. Architecture is about the structure of problems and solutions, not about the structure of artifacts. Following this reasoning, we can argue that legacy is not carried from generation to generation by artifacts in themselves, but by the functionality they deliver.

The architectural frames model makes an analytical tool for understanding and articulating a shift in architectural thinking; digitalization push product developing organizations from a product-centric to a process-centric perspective. However, the model does not assume a transition from one end to the other. Instead it makes a tool for reasoning about a new balance point, where product developing organizations can identify a sound and rewarding interplay between producibility and generativity.

Specifically, the model consists of two idealized frames; *hierarchy-of-parts* and *network-of patterns*. The hierarchy-of-parts frame, derived from the legacy of Herbert Simon, is centered on the physical structure of components and emphasizes *decomposition* with subsequent *aggregation* as the core principle for managing complexity. Drawing on the concept of near decomposability and stable subassemblies, this frame resonates well with the incentives of traditional product innovation, with strong focus on the reuse of assets. The network-of-patterns frame is derived from Christopher

Alexander's work in architecture, also frequently applied in contemporary software engineering practices. This frame is centered on the structure of problems and solutions, rather than the structure of artifacts. It emphasizes *generalization* with subsequent *specialization* as a complementary approach to complexity. Drawing on the concepts of patterns and pattern languages, this frame resonates with software-centric innovation practices, focusing on reuse of ideas.

Addressing the research question of this thesis I have applied the theoretical framework to digital product innovation practices at CarCorp over approximately one decade. This longitudinal case story ranges from the automaker's first staggering attempt to generalize infotainment with MOST, to the adoption and integration of Android, recognized as a highly generative platform in contemporary consumer electronics. In applying the architectural frames model to digital product innovation I have been able to:

1) Demonstrate the ***ontological significance*** of the architectural frames model. Although being a theoretical model, with the ambition to explain digital product innovation at an abstract, aggregate level, the concepts forming the two frames have proven deeply anchored in observable real-world phenomena. The model offers a language for understanding people's view on technology and thereby their rationale for engagement.
2) Show that permanent generative capability relies on extensive ***organizational support*** for the network-of patterns frame. With the assimilations of this frame people increasingly view products as enablers and catalyzers of new, yet unknown functionality, rather than carriers of pre-fabricated functions.
3) Show that inherent tensions and contradictions between architectural frames force organizations to develop new ***governance models***. The CarCorp case story bears significant evidence of such tensions, unfolding when a product developing organization seeks to leverage the generative capability of IT. To develop practices based on a sound and effective interplay between architectural frames these inherent tensions have to be avoided.

As demonstrated in this thesis a product developing firm faces several challenges when appropriating the network-of-patterns frame. These challenges become particularly salient when reaching a point where the new frame is exercised to build generative capability, rather than just internal design flexibility. Largely, generative capability is about encouraging external creativity, but also about benefiting from such creativity. This calls for a new perspective on products. Rather than viewing their products as carriers of pre-fabricated functionality, the organization has to embrace a

view where products are enablers and catalyzers of new, yet unknown functionality (6.1). Unless such a view informs all the different actions and decisions across the organization a product developing firm has little chance to build permanent generative capability (6.2). Further, this thesis suggests that a product developing firm cannot build generative capability unless adopting a distinctly different governance model. Generative capability relies on unconstrained freedom to create new specific functions. Such freedom clashes hard into established modular governance models, where decomposition of products is guided by specific functionality. Unless product developing firms find ways to govern innovation through general patterns rather than specific they have little chance to build permanent generative capability (6.3).

6.1 Ontological Significance

Hierarchy-of-parts and network-of-patterns are theoretical constructs, derived with an epistemological concern. At the same time, the application of these architectural frames to digital product innovation practices has disclosed a strong ontological significance of the model; the different concepts constituting architectural frames translate well into observable phenomena. The model does not just help us theorize digital product innovation, but makes a concrete language for representing, describing and discussing different perspectives on technology present in digital product innovation. Therefore, an architectural frame is an ontology in the sense that it offers a distinct model for understanding the different mechanisms powering an innovation regime.

I argue that a significant strength of the architectural frames model is its capability to bridge the gap between micro level and macro level, concrete and abstract, ontological issues and epistemological concerns. It is valuable at an abstract level not despite of, but because it makes a lot of sense at the concrete level. As summarized below, the four embedded cases show that the architectural frames model offers a language for understanding how people view technology and thereby their rationale for engagement. In some sense, this makes an entry ticket for theory development.

> **MOST.** The MOST architecture was adopted as a response to accelerating complexity of infotainment systems. To meet future challenges in a progressive manner CarCorp had to make these systems more malleable. It had to be significantly easier to adapt the system to new premises without entering a new loop of artifact design and production. MOST offered an answer to this challenge. However, as summarized in Box 1, there were two distinctly different outlooks on *how* the MOST architecture would enable such malleability.

Box 1. Hierarchy-of-parts and network-of-patterns as distinct perspectives on the adoption of MOST.

Hierarchy-of-parts

With MOST fiber optics entered the car. This new technology made an exceptionally simple *interface*. In fact, the same, standardized interface would apply to all the different *components* constituting the system. With such a clean and simple template for how to build the *physical structure* of an infotainment system CarCorp saw a great opportunity to reinforce *modularity*. MOST promised exceptionally flexible *decomposition* of the system into a wide range of components, each enabling a well defined piece of functionality. This would not just preserve the existing *hierarchy* of tier-1 and tier-2 suppliers, but would also allow them to *aggregate* the system aggressively to differentiate the product portfolio and launch a range of new, attractive offers.

Network-of-patterns

MOST introduced the concept of function blocks. These function blocks are concrete solution *patterns* for how to solve different problems in the context of infotainment. This new concept opened up for *functional structure* to be designed independently from the physical realization of the system. Drawing on this capability, CarCorp saw an opportunity to engage in *generalization* of the system. Such generalization would enable a whole range of shared general patterns that could be *inherited* by specific applications. Consequently, it would pay off in the act of *specialization*, when functions such as navigation, telematics, and audio playback could *reuse* the same general solutions for volume control or positioning to deliver a coherent and more harmonized end-user experience.

SoftCluster. The SoftCluster initiative was triggered by a need to reinforce commoditization while, at the same, allowing for flexible functional design and competitive functional diversity. Serving a range of brands, CarCorp had to find a way to share instrument clusters across car models while, at the same time, avoiding the rigidity and uniformity associated with standard solutions. The SoftCluster architecture was designed as a response to these challenges. An important property of this solution was its underlying assumptions on functional deployment; specific functionality, with high likelihood for change, was concentrated to one key component. Still, as outlined in Box 2, there were two different perspectives on how SoftCluster would enable a better balance between commoditization and differentiation.

Box 2. Hierarchy-of-parts and network-of-patterns as distinct perspectives on SoftCluster.

Hierarchy-of-parts

The SoftCluster architecture rested on two key concepts; the open XML *interface* (OXI) connecting the cluster hardware with remote *components* and a macro-oriented approach to HMI design (MOH). Together, these two concepts promised a competitive approach to commoditization. It allowed CarCorp to *decompose* the system into a *hierarchy* where the top-level cluster component hosted most of the functionality, while remote components turned functionally trivial data suppliers. Thanks to MOH the cluster could be commoditized in hardware as well as software, since functional designs were defined by macros, interpreted in run-time. Further, this top-loaded *modularity* allowed for smooth *aggregation* of the system in that the shared cluster could be easily configured to support just any combination of slave nodes, from low-end cars to extreme high-end.

Network-of-patterns

SoftCluster offered a condensed and well defined platform for *specialization* of cluster functionality. This platform was defined in a careful act of *generalization*, defining all general *patterns* of an instrument cluster. These patterns were then offered to designers in the form of a macro language, making a template for how to *reuse* and recombine general patterns. By applying the language in a specific macro, designers could *inherit* functional patters at a relatively detailed level to realize end-user applications with minor efforts. In practice, this new *functional structure* of an instrument cluster allowed for exceptional turn-around on ideas. As long as the macro language remained untouched, designers could engage in recurring specialization without paying attention to underlying hardware or software.

Nomadic device integration. The automotive industry is generally characterized by linear development processes, where functionality is defined up-front. At some point, such processes could not provide for competitive infotainment solutions, keeping up with the clockspeed demonstrated by consumer electronics. Portable navigation devices and mobile phones simply outperformed in-car solutions far too early in their long car lifecycles. Nomadic device integration seemed to offer an answer to this challenge. However, as summarized in Box 3, there were two distinctly different outlooks on *how* it would address this lifecycle problem.

Box 3. Hierarchy-of-parts and network-of-patterns as distinct perspectives on nomadic device integration.

Hierarchy-of-parts	***Network-of-patterns***
Nomadic device integration seemed to offer CarCorp the opportunity to solve the painful lifecycle dilemma by exercising an aftermarket approach to infotainment. Properly *decomposed*, the system could be designed to allow for an external *component* to fit with the established *hierarchy* of internally developed parts. Given an accepted and diffused *interface* this solution would offer customers the opportunity to upgrade the system simply by replacing their nomadic devices. Such *aggregation*, leaving a new configuration of components, could be done independently of the car lifecycle and at customer expenses.	With nomadic device integration CarCorp saw an opportunity to draw on the largely uncoordinated creativity in consumer electronics. The creation of a *layered architecture*, where nomadic functionality could be enabled as *general services*, promised a *process focus* where functionality could emerge independently from car lifecycles. By engaging in a process of *specialization* such general services could be *inherited* and continuously adapted to car specific use. Besides enabling new tools for CarCorp's designers, this new approach was expected to bring interest for external, non-automotive developers to engage in infotainment design.

Android. At a general level, Android was launched as a way for CarCorp to get access to rich, distributed innovation without becoming passive followers. Over almost a decade of experimentation with nomadic device integration they had identified a set of critical challenges. In particular, NDI did not allow an automaker to influence design processes. In practice, such solutions left for CarCorp and other car makers to be followers, adapting application already in place. By taking control over the software platform they saw an opportunity to reinforce their own influence over design processes. Still, as outlined in Box 4, there were two different perspectives on *how* the Android platform would allow for such influence.

Box 4. Hierarchy-of-parts and network-of-patterns as distinct perspectives on the adoption of Android as a basis for infotainment.

Hierarchy-of-parts	***Network-of-patterns***
Android had the potential to provide CarCorp with unprecedented variety. However, to get access to such variety it was critical to build significant installed base. The traditional approach to break installed base apart in different hardware configurations, adapted to different carlines, would not work well. Therefore, CarCorp *decomposed* the system carefully and planned for one standardized infotainment *component*, to be mounted in all models. Although Android was a software platform ultimately controlled by Google, the new infotainment system offered an important advantage compared to previous NDI solutions; the hardware would be in the hands of CarCorp. Being in control of a physical component ultimately seemed to offer power of access to that component. It would offer the opportunity to govern software development by certification, providing necessary rights to install and run software at a given hardware.	With Android CarCorp had access to a developer community of substantial proportions. This community made a huge innovation resource, waiting to be explored by progressive automakers. In such a context, the launch of car-specific services and API:s made an invitation for Android developers to engage in *specialization* of the rich automotive environment. From a technical point of view, such extensions seemed relatively smooth. Android offered not only a rich *network* of pre-existing *patterns*, but also an established model for how to extend the platform with new, reusable services. In contrast to NDI, reliance on Android would allow CarCorp to engage in *generalization*. As a key ingredient of a new governance strategy, explicit power to define general patterns made a template for how these patterns could be *inherited* when independent developers engaged in specialization. Ultimately, this made a bottoms-up strategy, leaving for unbounded innovation within a marked path.

6.2 Organizational Support

The application of architectural frames to digital product innovation at CarCorp has provided evidence for the idea that digitalization causes a shift in architectural locus. In its different parts the case story illustrates such a shift at an individual level, where people gradually reconsider their perspective on infotainment in light of new, digital technology. However, together the four embedded cases also provide rich evidence on how new architectural thinking propagates across an organization. Table 6 is a snapshot of selected quotes, illustrating how the network-of-patterns frame rolled out across CarCorp.

With MOST CarCorp uncovered and learned the basics of generalization. This had major impact on systems architecture. MOST provided tools for architects to reason about an infotainment system in terms of logical or functional elements, rather than components (i). It allowed them to focus their attention on detailed functional problems and, eventually to architect a

coherent system hosting increasingly harmonized functionality. At the same time, specialization has a minor role in the story about MOST. The reason is simple; CarCorp stayed with a traditional component-based approach to distribute design assignments. Rather than inviting to development of software-based infotainment functionality, CarCorp engaged existing tier-1 suppliers to design and produce components according to a traditional hierarchy-of-parts template. Purchasing expected that MOST would bring off-the-shelf components (xx), introducing much need competition between automotive suppliers. Similarly, product planning saw an opportunity to draw on the simple physical interface in MOST and break the infotainment system apart in a wide range of components. This would open up for differentiation and better business cases (xii). However, the conservation of traditional hierarchy-of-parts logic in design practices created problems, when generalized functionality should be deployed to a highly distributed system (vi). The clash between architectural frames became obvious, when suppliers found out that their components were not functionally independent anymore, but deeply intertwined with other components, outside their control. In making commitments on the basis of components, suppliers had no option but to reject much of the functional responsibility they normally had. Rather than designing components on the basis of overall property requirements, they needed complete interface specifications, falling out of the deployment of generalized functionality on distributed components. In summary, MOST introduced generalization at CarCorp and brought a radically new perspective on architecture, but seeded little new thinking outside the inner circle of systems architects. Without proper organizational support for this new thinking MOST turned yet another illustration that "implementation of technology intended to reinforce organizational control can instead cause organizational disorder and drift from intended purposes" (Sandberg 2010).

With SoftCluster CarCorp discovered specialization. The new macro-oriented approach disconnected functional design from platform software as well as physical properties of the system. Making a precise and clear template for how to reuse general patterns the macro language defined the creative leeway offered to designers (viii). It paid off quickly in exceptionally fast turn-around on ideas (vii). Rather than making a tool for smooth realization of existing ideas, SoftCluster turned into a generative platform. In the hands of designers, it produced novel functionality that no one had planned for up-front. These new design practices rested on a successful generalization of cluster functionality. By taking in-house control over platform design (ii) CarCorp was able to set up a macro language that gave structure and support in design processes, while at the same time allowing designers to exercise creativity. While network-of-patterns thinking had now changed design

practices at CarCorp, this new perspective was far from rooted outside R&D. Product planning, purchasing and many other actors recognized that SoftCluster brought different design practices, but largely they understood its potential from a hierarchy-of-parts perspective. Rather than seeing its generative capability they emphasized the unique opportunity to exercise functional differentiation while, at the same time, capitalizing on scale advantages of both commoditized hardware and software (xiii). Similarly, SoftCluster's well defined, straight-forward and relatively simple design process was attractive from a purchasing perspective. It seemed to make a solution to recurring and expensive change requests on component functionality, which tended to ruin project budgets (xxi). In summary, the SoftCluster story gives evidence on significant assimilation of generalization (platform design) as well as specialization (functional design). However, the many people not explicitly involved in technology largely understood SoftCluster from a hierarchy-of-parts perspective.

Nomadic device integration was triggered by the increasingly obvious difference in clock speed between the automotive industry and consumer electronics. Somehow the comparatively slow automakers had to bridge this gap not to be marginalized in infotainment (xiv), (xv). Nomadic device integration seemed to offer a solution where the automotive industry could tap in to the prospering developer communities forming around consumer electronics. Seeking architectural solutions for nomadic device integration CarCorp gradually understood that generalization and specialization are intertwined phenomena that cannot be disconnected from each other. As illustrated by their early Bluetooth enabled phones, specialization will inevitably decline and eventually stop if general patterns are not continuously adapted to new premises of application development. In response to this challenge CarCorp and its project partners researched many different concepts for how to make the car malleable to changes in consumer electronics (iii). At the R&D department engineers and designers were aware that a nomadic device strategy, grounded in a dynamic interplay between platform development and external application development would challenge established hierarchy-of-parts logic. Down the road, services would be a matter for external actors rather than tier-1 suppliers, delivering on CarCorp specifications (ix). However, the obvious threat to existing business models also pushed product planners to reconsider their view on infotainment. Although making an extraordinary idea with unclear implications, they started to accept that it would be possible to capitalize on such externally created functions and services (xvi). Slowly they recognized the logic of two-sided markets, where specialization materializes in the interplay between developers and end-users, powered by a platform that offers attractive and continuously revised general patterns. It was not clear

exactly how, but it seemed be possible to make money on these kinds of solutions. While this new position made a bridgehead for network-of-patterns thinking outside R&D, most people at CarCorp still tended to apply hierarchy-of-parts thinking to make sense of nomadic device integration. With this perspective, a nomadic device made an accessory that could be added to an existing solution. It was a component that, like a trailer hitch, could be added at a pre-defined position in a design hierarchy to improve end-user functionality. From a cost perspective, however, this accessory perspective made sense; these components would not be associated with any cost for CarCorp, even though it made a key part of the infotainment system. Customers would buy phones and other nomadic devices independently from car investments. This opened up for a new segment of low cost infotainment solutions which CarCorp had not been able to offer before (xxii). In brief, the work with nomadic device integration furthered network-of-patterns thinking at CarCorp. Generalization and specialization were increasingly seen as intertwined phenomena which, at its most concrete level, mean that platform development is a recurring activity that has to be exercised in harmony with application development. Even though this perspective was largely accepted at R&D and slowly taking root in product planning practices, the organization as a whole understood nomadic device integration from a hierarchy-of-parts perspective.

With Android the network-of-patterns view on infotainment had its breakthrough at CarCorp. A smaller group of people at R&D had worked for many years on new architectural solutions in order to benefit from the digitization of infotainment. Their efforts now made a solid foundation for a new perspective on infotainment. Still, the generative capability of these solutions largely remained unresolved, since the organization consistently put emphasis on benefits making sense with a traditional hierarchy-of-parts lens. With Android this changed. There was simply no rationale for adopting this platform under a hierarchy-of-parts paradigm. To make this bold idea fly it was necessary to get unconditional support from top management. It would prove surprisingly simple to get that support. With a long track record, credibility, and solid arguments, the core infotainment group got Android sanctioned by the executive management team and in a couple of weeks supported by the organization as a whole (xxiv). Now supported by top management, network-of-patterns thinking spread across the organization. Aftermarket saw great opportunities in a recurring infotainment business, decoupled from car sales (xxiii). Marketing, product planning, and other sales oriented parts of the organization suddenly accepted that developers were to be considered a new type of customer and thereby in their interest. The car became an offer, not just addressing traditional end-users, but also independent developers (xix). This offer

would be delivered through different APIs, enabling rich and interesting content to third party developers (xviii). With this perspective, CarCorp slowly transformed its view on value. Rather than associating value with the embedded functionality of a component, the decision to adopt Android helped people to see that the value of a generic software platform is found in its capability to generate rich and varied content (xvii). A rich and continuously changing functional offer also stood out as a solution to the lifecycle dilemma, where cars always tended to be outdated (x). Drawing on an open community, CarCorp would be able to respond quickly to external change. While it was not fully clear what such a community would look like, managers at infotainment outlined different approaches. These approaches made complementary, but coexisting models for open innovation (xi). One of the main arguments behind Android was its tight connection to established communities. With Android CarCorp would have an architecture which adapted continuously to specialization practices (iv). At the same time, an open source solution would break significantly with established governance models. The idea to rely on public source code, which could not be modified unilaterally, seemed awkward and confusing. Slowly, CarCorp understood that influence would require engagement. In order to avoid being passive observers they had to get actively involved in the open source community maintaining the platform. Influence over the platform would be related to contributions (v). To take reasonable control over an automotive fork of Android CarCorp would have to give away valuable things. Being one of the first automotive actors showing interest in Android, there was a window of opportunity – an opportunity with the potential to make CarCorp a leading actor in automotive.

Together, the four embedded cases provide a detailed narrative of how network-of-patterns thinking may propagate across an organization. Starting with systems architects and designers CarCorp gradually rolled-out a new perspective on their infotainment products over the studied period of approximately one decade. At the end of this period the organization viewed these products as enablers and catalyzers of new, yet unknown functionality, rather than carriers of pre-fabricated functionality. Although this study does not cover a commercial introduction of the Android platform it suggests that this view on digital products is a basic premise for generativity. Unless such a view informs all the different actions and decisions across the organization a product developing firm has little chance to break with traditional innovation logic to build permanent generative capability.

Table 6. The propagation of network-of-patterns thinking at CarCrop, illustrated by selected quotes from the case stories. A few quotes are not represented in the case stories and therefore marked with formal positions.

	MOST [1999-2003]	**SoftCluster[2002-]**	**Nomadic Device Integration [2004-2007]**	**Android [2008-]**
Systems Architecture	(i) *"I think we all realized – at least the people involved in [architecting] infotainment – that somehow this was the future. We needed to focus on the system, solving problems at the system level. […] I think, at the heart of MOST, we find a kind of system level thinking that is not component-oriented. Instead, it centers on the structure of logical elements or functionality."*	(ii) *"by being in control of software, we can be fast and make sure there are [general] software functions supporting whatever it is we see coming. I think that's the main challenge here."*	(iii) *"We should mimic the plug-in flexibility offered by USB. It is the device that is responsible for providing the relevant driver. This enables an end-to-end architecture for making the systems operate together.[…] As a third-party vendor, you'll supply this opportunity by installing the driver on our open platform."*	(iv) *"you've got to get out to the communities that move fast, which is open source, and you got to install a platform in the vehicle that can accommodate that innovation."* (v) *"we need to take control through an open source [platform] initiative and by our contribution to it, approve it. Because I think take control of it in a proprietary sense is still not going to create the crowd."*
Design	(vi) *"They thought the traditional model would work, where each [supplier] had responsibility for his own function, embedded in his own component. […] Down the road, they saw the flip side. It didn't work since the whole system – end-to-end – was so incredibly distributed."*	(vii) *"With this architecture I can make some design changes really, really fast. But I think there are very few realizing it."* (viii) *"It's like playing with LEGO. You've got a particular set of bricks. They've got their limitations, but you can build a whole lot of different things with them. And it's simple."*	(ix) *"We have realized that we don't have the capability to define all those upcoming services, to understand what people want. It might not even be our job anymore. [...] The pace of telecom is at the heart of this. The automotive industry can't handle this rapid pace. Things are too old when they come to the car! […] At some point we decided not to care about services. They can emerge on their own premises. Our task is to offer connectivity [between nomadic device and car]." (project manager, CarCorp)*	(x) *"[This solution] provides a faster, more efficient and more flexible alternative to the conventional, in-house development of vehicle infotainment services."* (xi) *"We foresee two models for how to introduce new applications. One option is to actively seek partnerships where brands and products reinforce each other [without cannibalizing]. The other one is [unconstrained] open innovation, where you do not really know what is going to happen, where you allow yourself to be surprised by people and their ideas. Together, these two models will pave the way for great products."*

Product Planning	(xii) *"We saw an opportunity to change and modify components […] It would be possible to add components over time and it would be possible to upgrade systems."*	(xiii) *"a need to support vehicle brand differences within the GlobalCarCorp family such as difference in graphics, layout and menu structures without having to change operational software in any ECU".*	(xiv) *"We are a couple of people who think that [selling embedded navigation and CD-changers] won't be possible in the future."* (xv) *[Today] we make money off of our current portfolio of entertainment products, and pretty soon we won't, because none of them are gonna be viable, you know." (product manager, GlobalCarCorp)* (xvi) *"It is not easy to get acceptance for the ideas that we are going to develop something that other actors may capitalize on or that we may generate revenue [on services] at the aftermarket. We might see a breakthrough here." (product manager, CarCorp)*	(xvii) *"the way you establish value for this open platform is this idea that you have to be able to look at hundreds of ideas, and then you're going to see the value."* (xviii) *"CarCorp will issue third-party developers with a vehicle application programming interface (API) providing access to more than 500 signals from different sensors in the vehicle."* (xix) *"CarCorp's 'open innovation' strategy offers the global developer community access to the full bandwidth of car communications — infotainment, telematics, systems monitoring and diagnostics."*
Purchasing, Aftersales	(xx) *"This idea about common specifications on functions and interfaces, that's a major benefit. More or less being able to buy a component [off the shelf], like a radio tuner, developed for one manufacturer, but applicable to another since it's a common interface specification."*	(xxi) *"By tradition, suppliers offered a low [component] price, knowing that change orders would feed them down the road. These changes always turned out to be 'small and simple' HMI changes."*	(xxii) *"According to customer surveys we have done regarding this kind of connectivity [with nomadic devices] a 300 dollar option, including a connected 'color screen radio' would attract almost 85% [of our customers]. If we can make such a system cheap enough – which we can – it'll increase our margins. And if we stay down there [in the low-end segment] we won't cannibalize on the premium products. It' makes a complement, which is good. So from a techno-strategical, but also commercial perspective, this is going to be important!" (senior manager, GlobalCarCorp)*	(xxiii) *"[This solution] will allow infotainment services to constantly evolve during the lifetime of a car's product cycle"* (xxiv) *"We have now sanctioned the project throughout the organization and received a great response"*

6.3 Governance Models

As we have discussed, architectural frames are not mutually exclusive. An innovation practice is not based on *either* hierarchy-of-parts *or* network-of-patterns. On the contrary, both frames are represented in any innovation regime. Still, this thesis has demonstrated that it is not an easy task to shift architectural locus. Tensions and contradictions always play an important role in the introduction and assimilation of new information technology (cf. Wimelius 2011). Such tensions and contradictions are particularly salient as organizations seek new combinations of architectural frames to release the generative capability of IT. Some of these tensions are easily resolvable while others turn out to be more fundamental contradictions of dialectical character. They are dialectical in the sense that they uncover incompatible applications of the two frames. Such contradictions identify areas where the they offer different "possibilities and one of them has to be made" (Benson 1977, p. 18).

In seeking a better understanding of how to build generative capability in product developing firms, contradictions between architectural frames play a critical role. Not surprisingly, it is a lot easier to extend a familiar way of doing something with new ideas, than replacing it. Dialectical tensions are not easily resolved and hold the potential of radical change. They enforce a new path, which makes them difficult for practitioners and interesting for researchers. Although the empirical study of this thesis demonstrates that CarCorp developed new approaches to combine architectural frames, each embedded case also bare witness of significant tension between frames. Let us, in an attempt to derive a useful theoretical perspective on this, zoom in on the most prominent contradictions of each embedded case.

> **MOST.** Drawing on net-work-of patterns thinking the MOST architecture afforded CarCorp a new innovation practice, where general function blocks and specific infotainment applications could emerge together in a productive manner, relatively disconnected from hardware. At the same time, its exceptionally simple physical interface afforded extensive decomposition of the system, reinforcing a hierarchy-of-parts practice. While commanding absolute compliance with an interface, such practices engage suppliers in relatively unbounded component innovation.
>
> Trying to exercise both perspectives CarCorp uncovered a strong tension between the two frames at the point when function blocks were deployed to physical components. In some sense, one can argue that suppliers were denied the creative leeway they had traditionally had by CarCorp's intervention in design of general functionality, but

without being offered something in exchange. As a consequence, this contradiction pushed them to adopt a defensive strategy, largely leaving for CarCorp to define how to improve navigation, radio, and other infotainment functions. As we know, CarCorp's MOST-based infotainment system had a lot of potential. The problem was that this potential was largely unresolved. Traditional applications, such as navigation, telematics, and media playback, could be repackaged in a coherent and more harmonized manner, but the substantial investment in MOST did not pay off in any new end-user functionality. Network-of-patterns thinking had offered the automaker new opportunities, but to the price of a crashed innovation model.

SoftCluster. In creating the SoftCluster platform CarCorp exercised network-of-patterns thinking. The macro language made a rich network of patterns that could be reused for new specific applications, not planned for up-front. At the same time, SoftCluster was a response to a strong need for commoditization. Therefore the SoftCluster architecture was designed to allow the wide range of components feeding the instrument cluster with information to evolve according to the hierarchy-of-parts logic. Petrol fuel measurement, speed, or cockpit temperature made well defined, relatively simple functions that could fit a traditional hierarchy-of-parts practice. The macro-oriented approach to cluster HMI design seemed to offer a working combination of a network-of-patterns practice, where specific functionality could emerge in an open-ended manner, and a hierarchy-of-parts practice, where underlying components could be incrementally improved, given the constraints of fixed system decomposition and rigid interfaces.

The SoftCluster project did not experience serious contradiction between architectural frames until designers tried to adapt the concept. For several reasons it turned out to be a hopeless mission. The established interplay between a hierarchy-of-parts practice and a network-of-patterns practice rested hard on the invariability of the macro language. Applying SoftCluster more widely, as a basis for the entire domain of infotainment, turned out to be very challenging. Apparently, the rigidity of SoftCluster would be increasingly problematic to cluster design as well. In practice, specialization would decline as it could not be supported by recurring generalization.

Nomadic device integration. Nomadic device integration was for many years viewed from a pure hierarchy-of-parts perspective, where mobile phones and handheld computers could be easily integrated with the car, using standardized interfaces. This would afford automakers tremendous design flexibility and give direct access to innovation in consumer electronics communities. At the same time, several years of experience with nomadic device integration had fostered a complementary view. Rather than centering on components, this network-of-patterns perspective recognized the functionality offered by nomadic devices. It viewed mobile phones as platforms, offering general patterns available for specialization in a car context.

Being relatively close to industrialization of an open platform for nomadic devices, CarCorp uncovered a fundamental contradiction that was rooted in a clash between the two perspectives. On the one hand, they recognized and encouraged the innovation taking place in nomadic components. The problem was that they had little influence over this process, since they could not unilaterally define interfaces. At the same time, they recognized and encouraged innovation associated with specialization, i.e. adaptation of nomadic functionality to an automotive context. Gradually they understood that this process was governed by general patterns, hosted by nomadic devices. CarCorp was about to launch an infotainment system where they had very few tools for governing the innovation process.

Android. The adoption of Android was grounded in a wide-spread belief that network-of-patterns thinking would be better off in providing rich infotainment experiences than a traditional hierarchy-of-parts practice. By extending the well diffused platform with complementary general patterns, offered as car-specific API:s, CarCorp saw an opportunity to exercise significant influence over distributed and uncoordinated innovation processes. At the same time security issues and driving safety remained critical topics. Leaving the platform wide open for any kind of software was not an option. Drawing on hierarchy-of-parts thinking strong voices promoted the idea that CarCorp should use its unlimited control over the hardware, hosting the Android platform, to weed out undesirable applications.

Trying to predict how an upcoming development community would act, CarCorp could see a significant contradiction between the two perspectives. Launching an Android-based infotainment solution

> would most likely force them to exercise both, but unless it was done with extreme care and vigilance the whole initiative would fail. An open innovation environment would not survive without multiplicity and niche applications. Enforced CarCorp control would most likely drive important actors away from the ecosystem. Practicing lock-out of applications would send such a message to the community.

Let us, when reflecting on the four embedded cases, recall that innovation is about combination. It plays out in a continuous interplay between existing building blocks and upcoming visions. Successful innovation practices offer predefined building blocks that assist production of new ideas, rather than enforce old ideas. Therefore, they are characterized by a sound and rewarding balance between creative leeway and rigid support for realizing new ideas. They offer, at the same time, freedom to design and well defined, solid structures supporting such design.

Synthesizing the four embedded cases of this thesis I argue that practices with an architectural locus on hierarchy-of-parts offer one template for this balancing, while a network-of-patterns-centric practice offers another. These templates are not always compatible. As the cases demonstrate, a product developing organization cannot enjoy the benefits of both.

- Hierarchy-of-parts, as we know it from modular practices in product development, requires specific functionality to be up-front defined, while enabling significant freedom for independent creation of new general functionality at the level of components.
- An architectural locus on network-of-patterns hampers change of general patterns, reused across a wide range of applications, while allowing for an almost unbounded freedom to create new specific functionality.

Not surprisingly, the predefined building blocks, paving way for creative leeway, also defines how an innovation process can be governed. With a hierarchy-of-parts perspective technological progression is primarily an outcome of the interplay between decomposition and aggregation. Applying modularity, as exercised at CarCorp and other product developing industries, the decomposition of systems is *governed by specific patterns* – the agreed overall functionality of the different products expected to be realized by the system. Such decomposition recursively brings a rigid and visible hierarchy of physical parts that makes a shared view on system characteristics, division of labor, production, etc. At the same time, this recursive process reduces complexity by hiding functional structure. Largely, the functional structure behind an interface is hidden at the system level. By detaching the functional

interior of different components a manufacturer builds remote islands of innovation. Suppliers are given significant freedom to design the interior of components as long as they obey the constraints defined by the system decomposition and, ultimately, the interfaces falling out of this decomposition.

In contrast, an innovation practice centered on network-of-patterns thinking is defined by the interplay between generalization and specialization. As practiced e.g. in the Android community, such innovation is open-ended in that there is no up-front plan for specific functionality to be delivered. Instead, such specific functionality emerges over time in a never-ending, iterative process, continuously extending and enriching a network of publicly available patterns. Still, this iteration does not make a random walk. The process is *governed by general patterns* – the different resources offered to developers by a platform owner. By offering a complementary Android API, giving access to diagnostic data, CarCorp will seed one path of innovation, while access to break data will seed another. Therefore, in a network-of-patterns-centric innovation practice creative leeway follows from direct access to the full functional structure of a system, allowing ideas to be iteratively reused for every new specific function. At the same time, it applies a layered architecture where designers can select the level of granularity. In particular, a layered software platform reduces complexity by hiding physical structure. In practice, it allows developers to implement applications on the basis of general software services, drawing on sensors and actuators, without ever seeing or working with the underlying hardware.

Again, a hierarchy-of-parts practice offers creative leeway at the level of components by defining *specific* functionality, while a network-of-patterns practice offers unbounded freedom to create new specific functions by defining *general* functionality of the system. A product developing organization cannot foster innovation practices extracting the benefits of both. Specifying both specific and general functionality will inevitably kill creativity, while specifying none leaves the manufacturer without influence.

One way to illustrate this inherent contradiction between architectural frames is to present a hierarchy-of-parts practice as *recursive*, while a network-of-patterns practice is *iterative*. These concepts are seemingly equivalent, both referring to a repetitive behavior. However, with this distinction I want to emphasize that decomposition-aggregation generates nested functional structure, hidden to an external observer, while generalization-specialization generates visible functional structure,

observable as a whole[30]. Recursion is not easily represented. As an example, a flow chart is able to illustrate loops, but not the logic of recursion, where a function calls itself over multiple instances. Therefore, in an attempt to make a simple illustration of why hierarchy-of-parts practices generate nested functional structure that cannot be unfolded and studied as a whole I have composed a few lines of pseudo code (Box 5 and Box 6). Although excessively simplified Box 5 demonstrates that hierarchy-of-parts practices generate hierarchical structure by repeated inscription of functionality in physical components. Typically, a product developing firm defines a first level of components, critical to production (03-06). Next, tier-1 suppliers are contracted to make the detailed designs of components. This involves a mapping and break-down of the component's specific functionality into more general functional elements (11-13), followed by an assignment of these elements to sub-components, potentially provided by tier-2 suppliers (15-18).

Box 5. Pseudo code demonstrating the recursive materialization of hierarchic structure in hierarchy-of-parts practice.

```
01 main ()
02 (
03  for each specific-function(i) of product
04    define component(i)
05    decompose(component(i),specific-function(i))
06  loop
07 )
08
09 decompose(component-to-decompose,function-to-instantiate)
10 (
11   do
12     define general-function(i) from function-to-instantiate
13   loop
14
15   for each general-function(i)
16     define component(i)
17     decompose(component(i),general-function(i))
18   loop
19 )
```

The purpose of the equally simplified pseudo code in Box 6 is to demonstrate that network-of-patterns practices do not embed functionality in components. Instead, functionality – the network of patterns – is a visible and available structure, evolving over time as different stakeholders extend

[30] Network-of-patterns do not enforce open source practice, where publicly available software code represents functional structure. However, to remain relevant over time it has to include mechanisms for making good solutions available beyond original settings.

its boundaries. Let us see the network of patterns as the different software elements of a digital product, from the lowest layers of a platform to end-user applications. When the product is launched this network of patterns is given an initial state (01). However, it evolves in an asynchronous, yet interdependent mangle between specialization and generalization. Typically, developers extend the network as they repeatedly reuse existing functional patterns of the platform to create new, specific applications (07-09). At the same time, platform owners seek to extend the platform by continuously developing new general functions with wide application across different contexts (11-13).

Box 6. Pseudo code demonstrating the iterative evolution of networked structure in network-of-patterns practice.

```
01 #define network-of-patterns
02
03 main()
04 (
05   do in parallel
06   A:
07     do
08       specialize(network-of-patterns)
09     loop
10   B:
11     do
12       generalize(network-of-patterns)
13     loop
14   end-do
15 )
```

Given that the governance model associated with hierarchy-of-parts seems to be largely incompatible with a network-of-patterns practice, how did CarCorp resolve this conflict as products turned increasingly digital? The short answer is that they did not resolve it. With MOST, the deployment of functional patterns to physical components marks a point in time when architectural locus shifted rapidly from network-of-patterns to hierarchy-of-parts. In the early phase designers exercised generalization to build coherent an aligned functionality. This functional design was performed largely independently from physical dimensions. However, in the later phase this perspective was marginalized and progression was guided by the structural constraints defined when allocating patterns to components.

In contrast to MOST, it can be argued that the other three embedded cases preserved network-of-patterns thinking, at different levels, beyond deployment and production. The two frames seem to have coexisted. All three cases account for software-based innovation practice which existed relatively independent from the processes of designing and developing hardware. At the same time, these innovation practices were embedded

within a traditional hierarchy-of-parts practice. Functionality of clusters and nomadic devices was allowed to emerge according to a new logic, but only within well defined boundaries. This "sandbox" was defined in an act of decomposition, when making the overall systems architecture of cars. Cluster development, nomadic functionality, and Android-based infotainment could be changed on new premises, but only within a given component. The rest of the car evolved according to a hierarchy-of-parts logic. Therefore, the two architectural frames co-existed, but largely without interaction.

The problem with this approach is that physical structure is created on the basis of an assumption of the functionality of the system. That assumption will forever constrain innovation and prevent the firm from releasing the full potential in generative capability. Generalization can be exercised for functionality associated with a given component, but all remote functionality, residing at other components, is created for specific purposes, beyond range for generalization. In practice, all the sensors, actuators and data sources of cars and other complex products hold enormous innovation potential which is very hard to release since they all make pieces in a pre-defined puzzle.

A question that remains unanswered is why product developing firms have not left hierarchy-of-parts thinking behind? Given that innovation processes cannot combine recursive, top-down governance and bottoms-up governance, powered by control over general patterns, what prevents them from releasing their grip of specific functionality? After all, architecture is not an inherent property of technology, but "a shared way of thinking" (p. 73) and architectural frames are "schemas for thinking about and representing a complex product's architecture" (p. 73). Would it not be possible for these firms to concentrate their efforts on network-of-patterns thinking, which does not *per se* hide new ideas deep down in nested hierarchies? Instead, such a network of patterns, manifested as platform services, APIs, tools in an SDK, code examples, community discussions, documentation, etc, represents an open and accessible pool of best practice. It makes a generative scheme of instructions which, "carried out sequentially, will allow a person or a group of people to create a coherent artifact, beautifully and simply" (Alexander 1999, p. 81).

Together, the embedded cases give a clear indication that the answer is no. Product developing firms cannot abandon hierarchy-of-parts in their struggle to build generative capability. Although increasingly digitalized, physical products have to be architected for producibility (see section 3.2). The production of all the different systems making up a car remains associated with substantial fixed and marginal costs. This insists on massive

investments in specialized assets, such as tools, supply chains, and plants. To stay competitive a product developing firm has to depreciate these costs across large volumes of the product, enforcing an economy of scale. The hierarchy-of-parts frame allows product developing firms to bridge the barrier between design and production by a common product structure, allowing specialized assets to be reused across variants and generations of products.

A question for future research is whether it is possible to architect digital products for producibility *and* generativity, without exercising the inherent clash between governance models. I will elaborate this issue further in section 7.

6.4 Summary

The architectural frames model is designed as a tool for understanding technological change in digital product innovation, where different innovation regimes clash into each other. It underlines that innovation processes are deeply colored by the way we conceptualize products. Architecture and architectural thinking largely defines change across generations of products. Being a link between historical achievements and future potentialities, the architecture is an instrument for path creation as well as a shackle of path dependency. However, the model also emphasizes that tangible products and software tend to be architected for different purposes. When architecting tangible products, product developing firms center on the physical structure. Modular designs allow for efficient *reuse of assets*, such as production tools and machineries. In contrast, software tends to be architected for efficient *reuse of ideas*. Therefore, as proposed in the theoretical framework (see e.g. section 3.3.4) and later illustrated by the four embedded cases, hierarchy-of-parts is largely associated with *producibility*, while network-of-patterns thinking is linked to *generativity*. This research suggests that generativity is not an explicit objective when product developing firms engage in software-centric design practices. Rather, it emerges along with a new kind of architectural thinking, triggered by new affordances of digital technology. Programmability and replicability disrupts taken-for-granted barriers between design and production, allowing functionality to be elaborated and adapted on a recurring basis, independently from hardware. Therefore, network-of-patterns thinking centers on processes, rather than products. It associates architecture with the structure of problems and solution, not structure of physical products.

The architectural frames model is demonstrated in product development over several embedded cases and a temporal extension of one decade (6.1). This demonstration shows that hierarchy-of-parts and network-of-patterns

are not just theoretical constructs. They also have ontological significance in that they offer concrete, frequently applied views on digital products. Further, we have seen that the generativity associated with network-of-patterns thinking has implications across the entire innovation process. While, it is certainly possible to draw on the affordances of digital technology to increase internal flexibility (e.g. SoftCluster), "the generative capacity for unrelated and unaccredited audiences to build and distribute code and content" (Zittrain 2006, p. 1975) follows from a new organizational perspective on existing products (6.2). Rather than viewing their products as carriers of pre-fabricated functionality, the organization, as a whole, has to embrace a view where products are enablers and catalyzers of new, yet unknown functionality. Finally, we have seen that hierarchy-of-parts and network-of-patterns are associated with distinctly different governance models that cannot be combined (6.3). Generative capability relies on unconstrained freedom to create new specific functions. Applying network-of-patterns philosophy, such practices can be governed by general patterns, offered to developers as platform services, APIs, SDK, code libraries, etc. At the same time, a hierarchy-of-parts practice offers creative leeway at the level of components by defining specific functionality. A product developing organization cannot build innovation practices drawing on both models. Specifying both specific and general functionality will inevitably kill creativity, while specifying none leaves the manufacturer without influence.

This thesis offers a theoretical perspective for understanding digital product innovation. It also discloses a range of challenges facing a product innovation organization trying to build generative capability. Two key lessons are that (1) generativity requires organization-wide support for network-of-patterns thinking and (2) a bottoms-up governance model, rooted in control over general patterns. CarCorp found a way to implement these substantial changes. However, they did it by setting up two innovation regimes in parallel. Infotainment embraced open innovation, inviting external software developers, while chassis, body electronics, and most other functional areas preserved a traditional hierarchy-of-parts practice. In some sense, CarCorp created an isolated playground, contained by specific components, where software-based functionality could evolve on different premises.

The problem of this approach is that sensors, actuators, and data sources outside this isolated environment remain pieces in a pre-defined puzzle. They cannot be easily generalized since they are created for a specific purpose. Therefore, a major challenge for future research and practice is to find a way to *combine* innovation regimes. How could digital products be architected, as a whole, for both producibility and generativity? Is there a

way to avoid the inherent contradiction between recursive hierarchy-of-parts development and iterative network-of-patterns practices? Can hardware be developed without exercising top-down governance, inscribing specific functionality in system solutions at an early point? In the following section I will elaborate these questions in some detail.

7 Implications and Future Work

This research provides new insights into how product developing firms adapt architectural thinking in response to digital technology. While products must continue to be architected for producibility, managers and designers are increasingly aware that architecture also is an instrument for building generative capability; it defines the creative processes providing yet unknown functionality and content for tomorrow's products.

To understand and elaborate this shifting view on products and product development this thesis develops a complementary perspective on the concept of architecture. Rather than seeing architecture as "the scheme by which the function of a product is allocated to physical components" (Ulrich 1995, p. 419) or "the structure or structures of a system" (Clements et al. 2003, p. 471) this perspective centers on the inherent capability of architecture to define change processes. It views architecture as structure-preserving and structure-enhancing in the sense that it may be used to pass sound solutions on from design to design and generation to generation. With this view, architecture connects historical achievements with future potentialities, making it a key instrument for path creation. With this perspective follows a different approach to complexity. Rather than seeing the architecture as an answer to the question *how is the whole described through its parts*, it triggers architects to ask: *how do things assemble themselves? How does new functionality emerge from existing elements?*

Contributions to Research

Drawing on this view on architecture, this thesis contributes to existing research in several ways. Most important, it develops and demonstrates a distinct theoretical lens – architectural frames – allowing for the new perspective on architecture to be applied in empirical studies of digital product innovation. This framework makes a contribution to several bodies of literature. On a general level, it can be viewed as a response to different calls in IS literature seeking to regain focus on the IT artifact (cf. Benbasat and Zmud 2003; Lyytinen and Yoo 2002; Orlikowski and Iacono 2001). In this discourse, architectural frames offer a new way for understanding and conceptualizing the mutual entanglement of technology and human action, frequently studied in contemporary IS research (Jones 1998; Kallinikos 2006; Latham and Sassen 2005; Monteiro and Hanseth 1995; Orlikowski 2007; Orlikowski and Scott 2008).

More specifically, this framework contributes to an increasingly vital subset of this literature, discussing the materiality of IT (cf. Jonsson et al. 2009; Leonardi and Barley 2008; Leonardi 2010; Svahn et al. 2009; Yoo 2010; Yoo et al. 2010d). While recognizing that "it may seem odd to say that information technologies have material properties" (Leonardi and Barley 2008, p. 162) this stream of research draws attention to the performativity of IT (Barad 2003; Pickering 1995). It is argued that the notion of materiality remains relevant for digital technology as long as we refer to properties of the technology that provides users with the capability to perform some action (cf. Leonardi 2010). As we have discussed in section 3.1 such an affordance perspective defines materiality in relation to an observer (Gibson 1979). The theoretical framework contributes to this stream of literature in that hierarchy-of-parts and network-of-patterns make distinct perspectives on such relations. One the one hand, consistent application of an architectural frame changes products over time. It is structure-enhancing in the sense that it over generations of designs reinforces selected material properties of a product or technology. On the other hand, an architectural frame gradually changes how people and organization conceptualize and makes sense of products. It reinforces an organization's capability to act by offering a shared, cognitive model, specifically tuned for selected material properties. This touch on an intricate question, leaving a research opportunity for the discourse of materiality in IS; is an affordance independent of an actor's experience and culture, as claimed by Gibson (1979), or are we better off conceptualizing it from a cognitive perspective, as resulting from "the mental interpretation of things, based on our past knowledge and experience" (Norman 1988, p. 14)?

However, this thesis also contributes to product architecture literature (Baldwin and Clark 2000; Henderson and Clark 1990; Robertson and Ulrich 1998; Sanchez and Mahoney 1996; Simon 1962; Sosa et al. 2004; Ulrich 1995), providing much of the theoretical foundations of this work. Clearly, modularity is the dominant view on architecture in product innovation. Simon's work on near decomposability (Simon 1962; Simon 2002) is a given point of departure when theorizing on modularity and the application of modularity in product development. However, a significant body of product innovation research points to the similarities in Simonian and Alexandrian thinking (Baldwin 2008; Langlois 2006; Murmann and Frenken 2006; Schilling 2000; Ulrich and Eppinger 2004; Von Hippel 1990). Opposing such a view, this thesis suggests that the increasing digital content of physical products (Andersson et al. 2008; Lenfle and Midler 2009; Yoo 2010; Yoo et al. 2010b) calls for consideration of their differences.

Finally, this thesis contributes to an emerging discourse on generativity in the context of IT and digital technology. The term generativity has been described as a *technology's* capacity to enable voluntaristic and spontaneous innovation driven by large, heterogeneous an essentially uncoordinated crowds (Remneland et al. 2011; Zittrain 2006). At the same time, there are strong voices arguing that generative capability is primarily an attribute of a *person*, which "refers to one's ability to reframe reality and subsequently to produce something ingenious or at least new in a particular context" (Avital and Te'eni 2009, p. 345). This thesis seeks to avoid the traditional wrestling match between techno-centrism and human-centrism by shifting focus from attributes and properties of technology and social structure to change processes. In doing so it aligns with the traditional use of generativity in behavioral sciences, where the concept is inherently associated with transformation processes (Erikson 1963; Kotre 1984; McAdams and de St Aubin 1992; Schön 1979). A salient example from this literature is Erik H. Erikson's (1963) discussion in "Childhood and Society" on adolescence and the interplay between generations:

> *The fashionable insistence on dramatizing the dependence of children on adults often blinds us to the dependence of the older generation on the younger one. Mature man needs to be needed, and maturity needs guidance as well as encouragement from what has been produced and must be taken care of. Generativity, then, is primarily the concern in establishing and guiding the next generation, although there are individuals who, through misfortune or because of special and genuine gifts in other directions, do not apply this drive to their own offspring (Erikson 1963, p. 266-267).*

Applying this basic reasoning to digital product innovation, this thesis suggests that generativity emerges from product architectures that can "pass sound solutions on from design to design and generation to generation" (p. 35). In the words of Erikson, generative, then, is primarily the architectural concern in establishing and guiding the next generation of yet unknown functionality and content for tomorrow's products.

The empirical study of digital product innovation in the automotive industry generates some distinct insights on the challenges facing an organization as it seeks to develop generative capability. Essentially, these challenges derive from the appropriation and adoption of network-of-patterns thinking. In building generative capability firms seek to encourage voluntaristic and spontaneous creativity, but also new models for appropriating value from such creativity. The study discloses that such capability entails a new perspective on products. Rather than viewing their products as carriers of pre-fabricated functionality, the organization has to embrace a view where products are enablers and catalyzers of new, yet unknown functionality (6.1). Unless such a view informs all the different actions and decisions across the organization a product developing firm has little chance to build permanent generative capability (6.2). Further, this thesis suggests that a product developing firm cannot build generative capability unless adopting a distinctly different governance model. Generative capability relies on unconstrained freedom to create new specific functions. Such freedom clashes hard into established modular governance models, where decomposition of products is guided by specific functionality. Unless product developing firms find ways to govern innovation through general patterns rather than specific they have little chance to build permanent generative capability (6.3).

Contributions to practice

This thesis also contributes to industrial practice. Applying architectural frames as a lens to digital product innovation in the automotive industry, it illustrates how the concept of architecture is gradually loaded with a new meaning in product developing industries. Rather than being a tool for producibility it slowly turns into to an instrument for generative capability. Uncovering the network-of-patterns frame, reflecting Christopher Alexander's view on architecture, product developing firms increasingly center their attention on how products are created, rather than how they are decomposed. With this view, the architecture affords reuse of solutions to recurring problems, rather than reuse of physical components across models and over generations of products. This research suggests that an increasing locus on the network-of-patterns frame may increase competitive advantage for product developing organizations in several ways.

First of all, the network-of-patterns frame opens up for *proactive rather than reactive architectural strategies*. Its inherent focus on solutions and problems makes the architecture an instrument to cultivate new ideas and exercise new business opportunities, rather than a tool for cost savings. With recognition of the network-of-patterns frame designers can motivate *not* to streamline solutions for particular, pre-defined purposes. When the architecture plays out in an ever-expanding space of functionality it is, on the contrary, a bad idea to minimize memory size or processor capacity to meet the needs of a given function.

Second, the network-of-patterns frame *enables a new strategic asset* as it turns the spotlight from specific functionality to general functional patterns. In a hierarchy-of-parts practice, products are architected to support a range of well-defined, specific functions, such as navigation, telephony, or audio playback in cars. General elements, such as positioning, routing, or decoding are largely irrelevant from an architectural point of view, simply because they are embedded in components and, thereby, a headache for a particular supplier. Recognition of the network-of-patterns frame, to some extent, turns this equation up-side-down; products are architected to supply flexible generic functions to distributed ecosystems, in turn, providing a range of specific functions, far beyond what a firm can do in isolation. Suddenly, high-precision positioning, enabled by integration of GPS and ABS[31] sensors, is a strategic asset for manufacturers, allowing them to appropriate value from in-car navigation solutions supplied by other independent actors.

Third, the network-of-patterns frame allows for *appropriation of value across the product life cycle*, rather than just at the time of sales. This is not primarily a consequence of increasingly software-based functionality, but rather an implication of the new frame. Network-of-patterns do not enforce inscription of specific functionality in the physical structure of products. Thereby, the meaning of a particular product is not up-front defined, but can evolve over time. Specialization can occur independently from hardware design, generating a constant flow of new functions.

Limitations and Future Work

There are certainly limitations in this work. Given the explicit ambition to study how product innovation regimes and digital innovation regimes are conceptualized and combined (p. 57), it can rightly be argued that the case story provides poor evidence of a truly revised architectural practice, combining hierarchy-of-parts and network-of-patterns. Although providing a

[31] Anti-lock breaking system.

relatively rich and nuanced portrait of how product developing organizations take in and adapt to network-of-patterns thinking, this new architectural philosophy remained somewhat an exception at CarCorp over the studied period. With the Android platform still not in production, hierarchy-of-parts remained the dominant view on the car, providing a well defined logic for product design, organizations, and business models. As discussed in section 6.3 the two architectural frames co-existed, but with limited interaction. To some extent, network-of-patterns thinking made a complement to established practices, which largely remained untouched.

For several reasons this approach is likely to fail over time. Allowing software-based functionality to evolve in a bottoms-up manner, while specifying the hardware, hosting that software, according to a traditional hierarchy-of-parts logic cripples generativity. It simply prevents the platform owner from exercising effective generalization over time, continuously feeding developers with new, interesting patterns that cannot be instantiated independently from hardware. Therefore, to build sustainable generative capability it is critical to form a working interplay between hierarchy-of-parts and network-of-patterns. Physical structures of a product have to be designed in careful dialogue with functional structures. Therefore, the diffusion of network-of-patterns thinking in product development comes with implications on hardware design and physical architecture. Together, researchers and practitioners have to develop a revised practice, approaching the hierarchy-of-parts frame from a slightly different angle. That is an angle where Simon's concept of stable subassemblies remains relevant, but not primarily as a way to define remote islands of innovation in a rigid and fixed overall structure. Instead, such a view has to emphasize the close link between stable subassemblies and producibility. This leaves several opportunities for future research on product architecture. Let us briefly reflect on a few of them.

First, there is an opportunity for future research to find a better balance point between the proven benefits of modularity and the emerging opportunities of digital technology by developing new *aggregation strategies*. Modularity allows an organization to form variants from a pre-defined setup of components, designed to meet particular functional needs (Schilling 2000). This way an organization can balance differentiation and commoditization (Robertson and Ulrich 1998). Low-end variants are simply based on fewer components. However, a generative environment, cultivating functional variety relies on the accessibility offered by considerable installed base (Zittrain 2006). To make niche applications interesting from a business perspective it is necessary to create significant audience (Davenport and Beck 2001). Against this backdrop, it is important for future research on

product architecture to study new aggregation strategies. To increase generative capability, physical products need to be architected on new premises, where variants remain compatible from a software perspective. Let us note that shared hardware may increase cost of low-end products. One way to preserve scale advantages in such a scenario is to consider standardized, off-the-shelf hardware. Such solutions are not tailored to the particular context, but allows for large series without pressure to reuse production assets over time.

Second, upcoming research may address product architecture by developing new *decomposition strategies*, considering breaking down digital product into physical parts on other premises than functionality. A modular product is decomposed to create stable subassemblies that can make up predefined physical element of a system (Simon 1996). Such stable subassemblies are autonomous in the sense that their interior can be changed without influencing the rest of the system. This is a fundamental aspect of traditional product innovation since it preserves the overall functionality of a product, while allowing for continuous price/performance improvements (Clark 1985). However, low coupling between components is achieved by decomposing the system from the perspective of functionality (Ulrich 1995). This effectively inscribes functionality in the physical structure of the system, which prevents reuse for new purposes. Therefore, a critical question for future research is to identify alternative premises for decomposition. How can physical products be decomposed for low coupling without inscribing functionality in the structure? As a suggestion, such decomposition may be guided by producibility issues, rather than functionality.

Third, future research may examine how *hierarchic span* correlates to generativity in the context of digitized products. Modularity prescribes hierarchy as the dominant structure (Simon 1962; Simon 1996). In established product innovation practices, such hierarchic structure follows from recursive innovation processes, handling complexity by hiding functional structure in components (Parnas 1972). Essentially, the different general patterns used to realize the functionality of a particular component are hidden for an external observer. Clearly, the modular approach to complexity does not resonate well with the holy grail of generativity; unconstrained reuse of old solutions to solve new problems. Therefore, alternative structure for physical products is an important topic for future research is to study. One question is whether there are better or worse hierarchies. Let us recall that recursion is at the heart of the problem. It is reasonable to believe that a deeper hierarchy is more rigid and hard to change. The more nested levels, the more hidden functionality. This line of argument suggests that a key measure to extend the generative capability of

digital products is to increase the hierarchic span (Simon 1996, e.g. p. 202). In practice, this will result in a less deep structure, where components exist, side by side, without a nested structure hiding them from each other.

Fourth, future research may address the question of appropriate *specificity of interfaces* in digital product innovation. In a modular innovation practice interfaces define the roles of components in a system (Schilling 2000). They reveal how different parts of a system interact and how functionality flows from component to component. To reduce complexity and coupling, but also to exercise precise control over innovation, interfaces tend to be tailored for the specific purpose falling out of decomposition (Ulrich 1995). Making a cornerstone in the specifications used to engage suppliers, interfaces tend to be rigid and hard to reconsider. This makes a stark contrast to the principles of generalization, being so central for generative capability. An important topic for future research is to derive best practice for how to reduce specificity of interfaces. While such general interfaces cannot be justified in a specific design process, they make an investment in the functionality and content of tomorrow's products.

8 References

Abernathy, W., and Clark, K. 1985. "Innovation: Mapping the Winds of Creative Destruction," *Research Policy* (14:1), pp 3-22.

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

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

Alexander, C. 1964. *Notes on the Synthesis of Form*. Harvard Univ Pr.

Alexander, C. 1979. *The Timeless Way of Building*. Oxford University Press, USA.

Alexander, C. 1999. "The Origins of Pattern Theory," *IEEE Software* (16:5), pp 71–82.

Alexander, C. 2002. *The Nature of Order: An Essay on the Art of Building and the Nature of the Universe*. Center for environmental structure.

Alexander, C., Ishikawa, S., and Silverstein, M. 1977. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, USA.

Allen, P. 2006. *Service Orientation: Winning Strategies and Best Practices*. Cambridge University Press.

Amram, M., Kulatilaka, N., and Association, F.M. 1999. *Real Options: Managing Strategic Investment in an Uncertain World*. Harvard Business School Press Boston.

Anderson, C. 2006. *The Long Tail: Why the Future of Business Is Selling Less of More*. Harper Collins.

Anderson, P. 1999. "Complexity Theory and Organization Science," *Organization Science* (10:3), pp 216-232.

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

Andersson, M., Lindgren, R., and Henfridsson, O. 2008. "Architectural Knowledge in Inter-Organizational IT Innovation," *Journal of Strategic Information Systems* (17:1), pp 19-38.

Andreasson, L., and Henfridsson, O. 2009. "Digital Differentiation, Software Product Lines, and the Challenge of Isomorphism in Innovation: A Case Study," *ECIS2009*, Verona, Italy.

Archer, M.S., Bhaskar, R., Collier, A., Lawson, T., and Norrie, A. (eds.). 1998. *Critical Realism: Essential Readings*. London: Routledge.

Arthur, W. 1989. "Competing Technologies, Increasing Returns, and Lock-in by Historical Events," *The Economic Journal* (89), pp 116-131.

Arthur, W.B. 2009. *The Nature of Technology: What It Is and How It Evolves*. Free Pr.

Avital, M., and Te'eni, D. 2009. "From Generative Fit to Generative Capacity: Exploring an Emerging Dimension of Information Systems Design and Task Performance," *Information Systems Journal* (19:4), pp 345-367.

Axelsson, J., Fröberg, J., Hansson, H., Norström, C., Sandström, K., and Villing, B. 2004. "A Comparative Case Study of Distributed Network Architectures for Different Automotive Applications," in: *Handbook on Information Technology in Industrial Automation*. IEEE Press.

Bagozzi, R.P. 1986. *Principles of Marketing Management*. Chicago, IL: Science Research Associates

Baldwin, C. 2008. "Where Do Transactions Come From? Modularity, Transactions, and the Boundaries of Firms," *Industrial and Corporate Change* (17:1), p 155.

Baldwin, C.Y., and Clark, K.B. 2000. *Design Rules: The Power of Modularity Volume 1*. Cambridge, MA, USA: MIT Press.

Baldwin, C.Y., and Clark, K.B. 2003. "Managing in an Age of Modularity," in: *Managing in the Modular Age: Architectures, Networks, and Organizations,* R. Garud, A. Kumaraswamy and R.N. Langlois (eds.). Blackwell Publishing, pp. 149-160.

Barabba, V., Huber, C., Cooke, F., Pudar, N., Smith, J., and Paich, M. 2002. "A Multimethod Approach for Creating New Business Models: The General Motors Onstar Project," *Interfaces* (32:1), pp 20-34.

Barad, K. 2003. "Posthumanist Performativity: Toward an Understanding of How Matter Comes to Matter," *Signs* (28:3), pp 801-831.

Barley, S.R. 1990. "The Alignment of Technology and Structure through Roles and Networks," *Administrative Science Quarterly* (35:1), pp 61-103.

Barrett, M., Davidson, E., Prabhu, J., and Vargo, S. 2010. "Call for Papers - MISQ Special Issue on Service Innovation in the Digital Age," in: *MIS Quarterly*.

Basole, R.C. 2009. "Visualization of Interfirm Relations in a Converging Mobile Ecosystem," *Journal of Information Technology* (24:2), pp 144-159.

Baudrillard, J. 1998. *The Consumer Society: Myths and Structures*. Sage Publications Ltd.

Baum, J.A.C., Korn, H.J., and Kotha, S. 1995. "Dominant Designs and Population Dynamics in Telecommunications Services: Founding

and Failure of Facsimile Transmission Service Organizations, 1965-1992," *Social Science Research* (24:2), pp 97-135.

Benbasat, I., and Zmud, R. 2003. "The Identity Crisis within the IS Discipline: Defining and Communicating the Discipline's Core Properties," *MIS Quarterly* (27:2), pp 183-194.

Benkler, Y. 2006. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press.

Benson, J. 1977. "Organizations: A Dialectical View," *Administrative Science Quarterly* (22:1), pp 1-21.

Bharadwaj, A., El Sawy, O., Pavlou, P., and Venkatraman, N. 2010. "Call for Papers - MISQ Special Issue on Digital Business Strategy: Toward a Next Generation of Insights," in: *MIS Quarterly*.

Bhaskar, R. 1998. *The Possibility of Naturalism: A Philosophical Critique of the Contemporary Human Sciences*. Psychology Press.

Bijker, W. 1987. "The Social Construction of Bakelite: To-Ward a Theory of Invention," in: *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology,* W. Bijker, T. Hughes and T. Pinch (eds.). Cambridge: MIT Press, p. 159.

Boehm, B. 1976. "Software Engineering," *IEEE Transactions on Computers* (100:25), pp 1226-1241.

Boland, R., Lyytinen, K., and Yoo, Y. 2007. "Wakes of Innovation in Project Networks: The Case of Digital 3-D Representations in Architecture, Engineering, and Construction," *Organization Science* (18:4), pp 631-647.

Booch, G., Maksimchuk, R., Engle, M., Young, B., Conallen, J., and Houston, K. 1991. *Object-Oriented Analysis and Design*, (1st ed.). Addison-Wesley.

Boudreau, M.-C., and Robey, D. 2005. "Enacting Integrated Information Technology: A Human Agency Perspective," *Organization Science* (16:1), pp 3-18.

Brooks, F.P. 1975. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley.

Brown, S.L., and Eisenhardt, K.M. 1997. "The Art of Continuous Change: Linking Complexity Theory and Time-Paced Evolution in Relentlessly Shifting Organizations," *Administrative Science Quarterly* (42:1), pp 1-34.

Broy, M., Krüger, I., Pretschner, A., and Salzmann, C. 2007. "Engineering Automotive Software," *Proceedings of the IEEE* (95:2), February, pp 356-373.

Brynjolfsson, E., Hu, Y.J., and Smith, M.D. 2010. "Research Commentary: Long Tails Vs. Superstars: The Effect of Information Technology on Product Variety and Sales Concentration Patterns," *Information Systems Research* (21:4), pp 736-747.

Brynjolfsson, E., and Smith, M.D. 2003. "Consumer Surplus in the Digital Economy: Estimating the Value of Increased Product Variety at Online Booksellers," *Management Science* (49:11), pp 1580-1596.

Burks, A.W., Goldstine, H.H., and Von Neumann, J. 1963. "Preliminary Discussion of the Logical Design of an Electronic Computing

Instrument," in: *John Von Neumann Collected Works,* A.H. Taub (ed.). New York: The Macmillan Co.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. 2008. *Pattern-Oriented Software Architecture: A System of Patterns*. Wiley-India.

Buxton, J., and Randell, B. 1970. "Software Engineering Techniques: Report on a Conference Sponsored by the Nato Science Committee," NATO Science Committee; available from Scientific Affairs Division, NATO.

Bygstad, B. 2010. "Generative Mechanisms for Innovation in Information Infrastructures," *Information and Organization* (20:3-4), pp 156-168.

Carlton, D.W. 1979. "Vertical Integration in Competitive Markets under Uncertainty," *The Journal of Industrial Economics* (27:3), pp 189-209.

Chandler, A.D. 1977. *The Visible Hand*. Cambridge, MA: Harvard University Press.

Chandler, A.D. 1990. *Scale and Scope : The Dynamics of Industrial Capitalism*. Belknap Press.

Chandler, A.D. 1997. "The Computer Industry: The First Half-Century," in: *Competing in the Age of Digital Convergence,* D.B. Yoffie (ed.). Boston, MA, USA: Harvard Business School Press.

Charmaz, K. 2006. *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*. Sage Publications Ltd.

Chesbrough, H. 2006. "Open Innovation: A New Paradigm for Understanding Industrial Innovation," in: *Open Innovation: Researching a New Paradigm*. Oxford University Press, USA, pp. 1-12.

Christensen, C.M. 1997. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*. Harvard Business School Press.

Christensen, C.M., Suárez, F.F., and Utterback, J.M. 1998. "Strategies for Survival in Fast-Changing Industries," *Management Science*), pp 207-220.

Clark, K. 1985. "The Interaction of Design Hierarchies and Market Concepts in Technological Evolution," *Research Policy* (14:5), pp 235-251.

Clements, P., Garlan, D., Bass, L., Stafford, J., Nord, R., Ivers, J., and Little, R. 2003. *Documenting Software Architectures: Views and Beyond*. Pearson Education.

Clements, P., and Northrop, L. 2001. *Software Product Lines*. Addison-Wesley.

Coase, R. 1937. "The Nature of the Firm," *Economica* (4:16), pp 386-405.

Cohen, W.M., and Levinthal, D.A. 1990. "Absorptive Capacity: A New Perspective on Learning and Innovation," *Administrative Science Quarterly* (35:1).

Constantinides, P., and Barrett, M. 2006. "Negotiating ICT Development and Use: The Case of a Telemedicine System in the Healthcare Region of Crete," *Information and Organization* (16:1), pp 27-55.

Crnkovic, I. 2001. "Component Based Software Engineering—New Challenges in Software Development," *Software Focus* (2:4), pp 127-133.

Cuenot, P., Chen, D., Gerard, S., Lonn, H., Reiser, M.-O., Servat, D., Sjostedt, C.-J., Kolagari, R.T., Torngren, M., and Weber, M. 2007. "Managing Complexity of Automotive Electronics Using the East-Adl," in: *ICECCS '07: Proceedings of the 12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*. Washington, DC, USA: IEEE Computer Society, pp. 353-358.

Cusumano, M.A., Mylonadis, Y., and Rosenbloom, R.S. 1992. "Strategic Maneuvering and Mass-Market Dynamics: The Triumph of VHS over Beta," *The Business History Review* (66:1), pp 51-94.

Davenport, T.H., and Beck, J.C. 2001. *The Attention Economy: Understanding the New Currency of Business*. Harvard Business Press.

David, P. 1985. "Clio and the Economics of QWERTY," *The American Economic Review* (75:2), pp 332-337.

Deleuze, G. 1988. *Bergsonism*. Zone Books.

Deleuze, G., and Guattari, F. 1980. *A Thousand Plateaus: Capitalism and Schizophrenia*. Ed. de Minuit.

Demil, B., and Lecocq, X. 2006. "Neither Market nor Hierarchy nor Network: The Emergence of Bazaar Governance," *Organization Studies* (27:10), p 1447.

Dixit, A., Pindyck, R., and Davis, G. 1994. *Investment under Uncertainty*. Princeton University Press Princeton, NJ.

Dobson, P.J. 2001. "Longitudinal Case Research: A Critical Realist Perspective," *Systemic Practice and Action Research* (14:3), pp 283-296.

Dosi, G. 1982. "Technological Paradigms and Technological Trajectories: A Suggested Interpretation of the Determinants and Directions of Technical Change," *Research Policy* (11:3), pp 147-162.

Du, X., Jiao, J., and Tseng, M.M. 2001. "Architecture of Product Family: Fundamentals and Methodology," *Concurrent Engineering* (9:4), pp 309-325.

Easton, G. 2010. "Critical Realism in Case Study Research," *Industrial Marketing Management* (39:1), pp 118-128.

Economides, N., and Katsamakas, E. 2006. "Two-Sided Competition of Proprietary Vs. Open Source Technology Platforms and the Implications for the Software Industry," *Management Science* (52:7), p 1057.

Eisenhardt, K. 1985. "Control: Organizational and Economic Approaches," *Management Science* (31:2), pp 134-149.

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

Eisenmann, T., Parker, G., and Van Alstyne, M. 2006. "Strategies for Two-Sided Markets," *Harvard Business Review* (84:10), p 92.

Eklund, U., Askerdal, Ö., Granholm, J., Alminger, A., and Axelsson, J. 2005. "Experience of Introducing Reference Architectures in the Development of Automotive Electronic Systems," *ACM SIGSOFT 2005*, pp. 1-6.

El Sawy, O.A., Malhotra, A., Park, Y.K., and Pavlou, P.A. 2010. "Research Commentary: Seeking the Configurations of Digital Ecodynamics: It

Takes Three to Tango," *Information Systems Research* (21:4), pp 835-848.

Erikson, E.H. 1963. *Childhood and Society*, (2nd ed.). New York: WW Norton & Co., Inc.

Ethiraj, S.K., and Levinthal, D. 2004. "Modularity and Innovation in Complex Systems," *Management Science* (50:2), pp 159-173.

Fennel, H., Bunzel, S., Heinecke, H., Bielefeld, J., Fürst, S., Schnelle, K.-P., Grote, W., Maldener, N., Weber, T., Wohlgemuth, F., Ruh, J., Lundh, L., Sandén, T., Heitkämper, P., Rimkus, R., Leflour, J., Gilberg, A., Virnich, U., Voget, S., Nishikawa, K., Kajio, K., Lange, K., Scharnhorst, T., and Kunkel, B. 2006. "Achievements and Exploitation of the AUTOSAR Development Partnership," in: *Proceedings of SAE Convergence 2006*.

Ferrier, W., Holsapple, C., and Sabherwal, R. 2007. "Call for Papers: Digital Systems and Competition," *Information Systems Research* (18:2), pp 228-230.

Fichman, R. 2004. "Real Options and IT Platform Adoption: Implications for Theory and Practice," *Information Systems Research* (15:2), pp 132-154.

Fine, C.H. 1999. *Clockspeed: Winning Industry Control in the Age of Temporary Advantage*. Perseus Books.

Fixson, S.K., and Park, J.-K. 2008. "The Power of Integrality: Linkages between Product Architecture, Innovation, and Industry Structure," *Research Policy* (37:8), pp 1296-1316.

Frenken, K., Saviotti, P.P., and Trommetter, M. 1999. "Variety and Niche Creation in Aircraft, Helicopters, Motorcycles and Microcomputers," *Research Policy* (28:5), pp 469-488.

Fröberg, J., Sandström, K., and Norström, C. 2005. "Business Situation Reflected in Automotive Electronic Architectures: Analysis of Four Commercial Cases," in: *SEAS '05: Proceedings of the second international workshop on Software engineering for automotive systems*. St. Louis, Missouri: ACM Press, pp. 1-6.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. 1995. *Design Patterns: Elements of Object-Oriented Software Architecture*. Addison-Wesley Reading, MA.

Garlan, D., and Shaw, M. 1994. "An Introduction to Software Architecture," School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

George, A.L., and Bennett, A. 2005. *Case Studies and Theory Development in the Social Sciences*. The MIT Press.

Gerring, J. 2007. *Case Study Research: Principles and Practices*. Cambridge Univ Pr.

Ghazawneh, A., and Henfridsson, O. 2011. "Micro-Strategizing in Platform Ecosystems: A Multiple Case Study," *Int. Conf. on Information Systems*, Shanghai, China.

Ghazawneh, A., and Henfridsson, O. forthcoming. "Balancing Platform Control and External Contribution in Third-Party Development: The Boundary Resources Model," *Information Systems Journal* (:).

Gibson, J. 1979. *The Ecological Approach to Visual Perception*. Houghton Mifflin Boston.

Gibson, J.J. 1977. "The Theory of Affordances," in: *In Perceiving, Acting, and Knowing: Toward an Ecological Psychology,* S. R. and B. J. (eds.). Lawrence Erlbaum, pp. 67-82.

Gioia, D.A. 1986. "Symbols, Scripts, and Sensemaking: Creating Meaning in the Organizational Experience," in: *The Thinking Organization,* H.P. Sims and D.A. Gioia (eds.). San Fransisco, CA: Jossey-Bass Inc. Pub., pp. 49-74.

Godin, B. 2006. "The Linear Model of Innovation: The Historical Construction of an Analytical Framework," *Science, Technology & Human Values* (31:6), p 639.

Godoe, H. 2000. "Innovation Regimes, R&D and Radical Innovations in Telecommunications," *Research Policy* (29:9), pp 1033-1046.

Goldstine, H.H., and Von Neumann, J. 1963. "On the Principles of Large Scale Computing Machines," in: *John Von Neumann Collected Works,* A.H. Taub (ed.). New York: The Macmillan Co.

Guglielmetti, L. 2003. "Standardizing Automotive Multimedia Interfaces," *IEEE multimedia* (10:2), pp 76-78.

Gupta, A., Tesluk, P., and Taylor, M. 2007. "Innovation at and across Multiple Levels of Analysis," *Organization Science* (18:6), p 885.

Hagedoorn, J., Carayannis, E., and Alexander, J. 2001. "Strange Bedfellows in the Personal Computer Industry: Technology Alliances between IBM and Apple," *Research Policy* (30:5), pp 837-849.

Hardung, B., Kölzow, T., and Krüger, A. 2004. "Reuse of Software in Distributed Embedded Automotive Systems," in: *EMSOFT '04: Proceedings of the 4th ACM international conference on Embedded software*. Pisa, Italy: ACM Press, pp. 203-210.

Heineman, G., and Councill, W. 2001. *Component-Based Software Engineering: Putting the Pieces Together*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

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* (35:1).

Henfridsson, O., and Lindgren, R. 2005. "Multi-Contextuality in Ubiquitous Computing: Investigating the Car Case through Action Research," *Information and Organization* (15:2), pp 95-124.

Henfridsson, O., Mathiassen, L., and Svahn, F. 2009a. "Reconfiguring Modularity: Closing Capability Gaps in Digital Innovation," *Sprouts Working Papers on Informations Systems* (9:22).

Henfridsson, O., Mathiassen, L., and Svahn, F. in review. "Managing Technological Change in the Digital Age: The Role of Architectural Frames," *submitted to international journal* ( ).

Henfridsson, O., Yoo, Y., and Svahn, F. 2009b. "Path Creation in Digital Innovation: A Multi-Layered Dialectics Perspective," *Sprouts Working Papers on Informations Systems* (9:20).

Hill, C., and Rothaermel, F. 2003. "The Performance of Incumbent Firms in the Face of Radical Technological Innovation," *The Academy of Management Review* (28:2), pp 257-274.

Holland, J.H. 1992a. *Adaptation in Natural and Artificial Systems*. MIT Press.
Holland, J.H. 1992b. "Complex Adaptive Systems," *Daedalus* (121:1), pp 17-30.
Holland, J.H. 1996. *Hidden Order: How Adaptation Builds Complexity*. Basic Books.
Hounshell, D.A. 1984. *From the American System to Mass Production, 1800-1932: The Development of Manufacturing Technology in the United States*. Johns Hopkins Univ Pr.
Howcroft, D., and Wilson, M. 2003. "Paradoxes of Participatory Practices: The Janus Role of the Systems Developer," *Information and Organization* (13:1), pp 1-24.
Hughes, T.P., Bijker, W., and Pinch, T. 1987. "The Evolution of Large Technological Systems," in: *The Political Economy of Science, Technology, and Innovation*. pp. 51–82.
Huizingh, E.K.R.E. 2011. "Open Innovation: State of the Art and Future Perspectives," *Technovation* (31:1), pp 2-9.
Iansiti, M. 1995. "Technology Integration: Managing Technological Evolution in a Complex Environment," *Research Policy* (24:4), pp 521-542.
IEEE Std 610.12. 1990. "IEEE Standard Glossary of Software Engineering Terminology," in: *IEEE Std 610.12-1990*.
Jackson, M. 2000. *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.
Jiao, J., and Tseng, M.M. 2000. "Fundamentals of Product Family Architecture," *Integrated Manufacturing Systems* (11:7), pp 469-483.
Jones, M. 1998. "Information Systems and the Double Mangle:Steering a Course between the Scylla of Embedded Structure and the Charybdis of Material Agency.," in: *Information Systems: Current Issues and Future Challenges,* T. Larsen, L. Levine and J. DeGross (eds.). Laxenburg: International Federation forInformation Processing, pp. 287-302.
Jonsson, K. 2010. "Digitalized Industrial Equipment: An Investigation of Remote Diagnostics Services," in: *Department of Informatics*. Umeå, Sweden: Umeå University.
Jonsson, K., Holmstrom, J., and Lyytinen, K. 2009. "Turn to the Material: Remote Diagnostics Systems and New Forms of Boundary-Spanning," *Information and Organization* (19:4), pp 233-252.
Kallinikos, J. 2006. *The Consequences of Information: Institutional Implications of Technological Change*. Cheltenham, UK: Edward Elgar Publishing.
Karlsson, C., and Sköld, M. 2007. "Counteracting Forces in Multi-Branded Product Platform Development," *Creativity and Innovation Management* (16:2), June, pp 133-141.
Katz, M.L., and Shapiro, C. 1994. "Systems Competition and Network Effects," *The Journal of Economic Perspectives* (8:2), pp 93-115.

Khazam, J., and Mowery, D. 1994. "The Commercialization of RISC: Strategies for the Creation of Dominant Designs," *Research Policy* (23:1), pp 89-102.

King, J.L., and Lyytinen, K. 2005. "Automotive Informatics: Information Technology and Enterprise Transformation in the Automobile Industry," in: *Transforming Enterprise: The Economic and Social Implications of Information Technology,* W.H. Dutton, B. Kahin, R. O'Callaghan and A.W. Wyckoff (eds.). pp. 283-333.

Kirsch, L. 1996. "The Management of Complex Tasks in Organizations: Controlling the Systems Development Process," *Organization Science* (7:1), pp 1-21.

Klein, B., Crawford, R.G., and Alchian, A.A. 1978. "Vertical Integration, Appropriable Rents, and the Competitive Contracting Process," *Journal of Law and Economics* (21:2), pp 297-326.

Klein, H., and Myers, M. 1999. "A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems," *MIS Quarterly* (23:1), pp 67-93.

Klepper, S. 1997. "Industry Life Cycles," *Industrial and Corporate Change* (6:1), p 145.

Kling, R. 1992. "Audiences, Narratives, and Human Values in Social Studies of Technology," *Science, Technology & Human Values* (17:3), p 349.

Kotre, J. 1984. *Outliving the Self: Generativity and the Interpretation of Lives*. Baltimore: Johns Hopkins University Press

Kruchten, P. 1995. "Architectural Blueprints: The '4+1' View Model of Software Architecture," *IEEE Software* (12:6), November, pp 42-50.

Kruchten, P., Obbink, H., and Stafford, J. 2006. "The Past, Present, and Future for Software Architecture," *Software, IEEE* (23:2), pp 22-30.

Langley, A. 1999. "Strategies for Theorizing from Process Data," *Academy of Management Review* (24:4), pp 691-710.

Langlois, P., and Richard, N. 1992. "Networks and Innovation in a Modular System: Lessons from the Microcomputer and Stereo Component Industries," *Research Policy* (21:4), pp 297-313.

Langlois, R. 2006. "The Secret Life of Mundane Transaction Costs," *Organization Studies* (27:9), pp 1389-1410.

Langlois, R.N. 2002. "Modularity in Technology and Organization," *Journal of Economic Behavior & Organization* (49:1), pp 19-37.

Latham, R., and Sassen, S. 2005. *Digital Formations: IT and New Architectures in the Global Realm*. Princeton, NJ: Princeton University Press.

Leen, G., and Heffernan, D. 2002. "Expanding Automotive Electronic Systems," *Computer* (35:1), pp 88-93.

Lenfle, S., and Midler, C. 2009. "The Launch of Innovative Product-Related Services: Lessons from Automotive Telematics," *Research Policy* (38:1), pp 156-169.

Leonardi, P., and Barley, S. 2008. "Materiality and Change: Challenges to Building Better Theory About Technology and Organizing," *Information and Organization* (18:3), pp 159-176.

Leonardi, P.M. 2010. "Digital Materiality? How Artifacts without Matter, Matter," *First Monday* (15:6-7).

Lessig, L. 2004. *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*. Penguin.

Levina, N., and Vaast, E. 2005. "The Emergence of Boundary Spanning Competence in Practice. Implications for Implementation and Use of Information Systems," *MIS Quarterly* (29:2), Jun, pp 335-363.

Lichtenthaler, U. 2011. "Open Innovation: Past Research, Current Debates, and Future Directions," *The Academy of Management Perspectives* (25:1), pp 75-93.

Liebowitz, S.J., and Margolis, S.E. 1995. "Path Dependence, Lock-in, and History," *JL Econ. & Org.* (11:1), p 205.

Lindgren, R., Andersson, M., and Henfridsson, O. 2008. "Multi-Contextuality in Boundary-Spanning Practices," *Information Systems Journal* (18:6), pp 641-661.

Luecke, R., Staff, H., and Katz, R. 2003. *Harvard Business Essentials: Managing Creativity and Innovation*. Harvard Business School Pr.

Lyytinen, K., and Yoo, Y. 2002. "Research Commentary: The Next Wave of Nomadic Computing," *Information Systems Research* (13:4), pp 377-388.

Markus, L.M. 2007. "The Governance of Free/Open Source Software Projects: Monolithic, Multidimensional, or Configurational?," *Journal of Management and Governance* (11:2), pp 151-163.

Markus, M., and Robey, D. 1988. "Information Technology and Organizational Change: Conceptions of Causality in Theory and Research," *Management Science* (34:5), pp 583-598.

Markus, M.L., and Benjamin, R.I. 1996. "Change Agentry - the Next IS Frontier," *MIS Quarterly* (20:4), pp 385-407.

Marples, D.L. 1961. "The Decisions of Engineering Design," *Engineering Management, IRE Transactions on* (8:2), pp 55-71.

Mathiassen, L., Munk-Madsen, A., Nielsen, P.A., and Stage, J. 2000. *Object-Oriented Analysis and Design*. Marko Publishing.

McAdams, D.P., and de St Aubin, E. 1992. "A Theory of Generativity and Its Assessment through Self-Report, Behavioral Acts, and Narrative Themes in Autobiography," *Journal of Personality and Social Psychology* (62:6), p 1003.

McGrenere, J., and Ho, W. 2000. "Affordances: Clarifying and Evolving a Concept," Citeseer, pp. 179-186.

Mingers, J. 2004. "Real-Izing Information Systems: Critical Realism as an Underpinning Philosophy for Information Systems," *Information and Organization* (14:2), pp 87-103.

Mohr, L.B. 1971. "Organizational Technology and Organizational Structure," *Administrative Science Quarterly* (16:4), pp 444-459.

Monteiro, E., and Hanseth, O. 1995. "Social Shaping of Information Infrastructure: On Being Specific About the Technology," in: *Information Technology and Changes in Organisational Work,* W. Orlikowski, G. Walsham, M. Jones and J. DeGross (eds.). Chapman & Hall, pp. 325-343.

Morein, J.A. 1975. "Shift from Brand to Product Line Marketing," *Harvard Business Review* (53:5), pp 56-64.

Murmann, J., and Frenken, K. 2006. "Toward a Systematic Framework for Research on Dominant Designs, Technological Innovations, and Industrial Change," *Research Policy* (35:7), pp 925-952.

Nelson, R., and Winter, S. 1982. *An Evolutionary Theory of Economic Change*. Belknap Press.

Norman, D.A. 1988. *The Psychology of Everyday Things*. Basic books.

O'Mahony, S. 2007. "The Governance of Open Source Initiatives: What Does It Mean to Be Community Managed?," *Journal of Management and Governance* (11:2), pp 139-150.

Orlikowski, W. 1992. "The Duality of Technology: Rethinking the Concept of Technology in Organizations," *Organization Science* (3:3), pp 398-427.

Orlikowski, W. 2007. "Sociomaterial Practices: Exploring Technology at Work," *Organization Studies* (28:9), p 1435.

Orlikowski, W., and Iacono, C. 2001. "Research Commentary: Desperately Seeking The" IT" In IT Research-a Call to Theorizing the IT Artifact," *Information Systems Research* (12:2), pp 121-134.

Orlikowski, W., and Scott, S. 2008. "Sociomateriality: Challenging the Separation of Technology, Work and Organization," *The Academy of Management Annals* (2:1), pp 433-474.

Orlikowski, W.J. 2002. "Knowing in Practice: Enacting a Collective Capability in Distributed Organizing," *Organization Science* (13:3), pp 249-273.

Orlikowski, W.J., and Gash, D.C. 1994. "Technological Frames: Making Sense of Information Technology in Organizations," *ACM Trans. Inf. Syst.* (12:2), pp 174-207.

Orton, J.D., and Weick, K.E. 1990. "Loosely Coupled Systems: A Reconceptualization," *The Academy of Management Review* (15:2), pp 203-223.

Ouchi, W. 1979. "A Conceptual Framework for the Design of Organizational Control Mechanisms," *Management Science* (25:9), pp 833-848.

Papazoglou, M., and Georgakopoulos, D. 2003. "Service-Oriented Computing " *Communications of the ACM* (46:10), pp 24-28.

Papazoglou, M.P., Traverso, P., Dustdar, S., and Leymann, F. 2007. "Service-Oriented Computing: State of the Art and Research Challenges," *Computer* (40:11), pp 38-45.

Parnas, D.L. 1972. "On the Criteria to Be Used in Decomposing Systems into Modules," *Communications of the ACM* (15:12), pp 1053-1058.

Parnas, D.L., Clements, P.C., and Weiss, D.M. 1985. "The Modular Structure of Complex Systems," *Software Engineering, IEEE Transactions on* (11:3), pp 259-266.

Perry, D.E., and Wolf, A.L. 1992. "Foundations for the Study of Software Architecture," *SIGSOFT Softw. Eng. Notes* (17:4), pp 40-52.

Pickering, A. 1995. *The Mangle of Practice: Time, Agency & Science*. University of Chicago Press.

Pickering, A. 2001. "Practice and Posthumanism: Social Theory and a History of Agency," in: *The Practice Turn in Contemporary Theory*, T.R. Schatzki, K. Knorr-Cetina and E. Von Savigny (eds.). New York: Routledge, pp. 163-174.

Pinch, T. 2008. "Technology and Institutions: Living in a Material World," *Theory and Society* (37:5), pp 461-483.

Pine, B.J., and Davis, S. 1999. *Mass Customization: The New Frontier in Business Competition*. Harvard Business School Pr.

Pohl, K., Böckle, G., and Van Der Linden, F. 2005. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer-Verlag New York Inc.

Poole, M.S., and DeSanctis, G. 2004. "Structuration Theory in Information Systems Research: Methods and Controversies," in: *The Handbook of Information Systems Research,* M.E. Whitman and A.B. Woszczynski (eds.). Idea Group, pp. 206-249.

Porter, M. 1985. *Competitive Advantage: Creating and Sustaining Superior Performance*. Free Pr.

Poster, M. 2001. *What's the Matter with the Internet?* University of Minnesota Press Minneapolis.

Powell, W. 1990. "Neither Market nor Hierarchy: Network Forms of Organization," *Research in Organizational Behavior* (12), pp 295-336.

Racu, R., Hamann, A., Ernst, R., and Richter, K. 2007. "Automotive Software Integration," in: *DAC '07: Proceedings of the 44th annual conference on Design automation*. San Diego, California: ACM Press, pp. 545-550.

Reinhardt, A. 2006. "Nokia's Magnificent Mobile-Phone Manufacturing Machine," in: *Businessweek Online*.

Remneland, B., Ljungberg, J., Bergquist, M., and Kuschel, J. 2011. "Open Innovation, Generativity and the Supplier as Peer: The Case of Iphone and Android," *International Journal of Innovation Management* (15:1), pp 1-26.

Robertson, D., and Ulrich, K. 1998. "Planning for Product Platforms," *Sloan Management Review* (39), pp 19-32.

Robertson, P.L., and Langlois, R.N. 1995. "Innovation, Networks, and Vertical Integration* 1," *Research Policy* (24:4), pp 543-562.

Rosemann, M., Andersson, M., and Lind, M. 2011. "Digital Complementary Assets," *Int. Conf. on Information Systems*, Shanghai, China.

Rosenberg, N. 1982. *Inside the Black Box: Technology and Economics*. Cambridge Univ Pr.

Rosenkopf, L., and Nerkar, A. 1999. "On the Complexity of Technological Evolution," in: *Variations in Organization Science: In Honor of Donald T. Campbell,* J.A.C. Baum and B. McKelvey (eds.). Thousand Oaks, CA: Sage Publications.

Rothaermel, F., and Hill, C. 2005. "Technological Discontinuities and Complementary Assets: A Longitudinal Study of Industry and Firm Performance," *Organization Science* (16:1), pp 52-70.

Royce, W.E., and Royce, W. 1991. "Software Architecture: Integrating Process and Technology," *TRW Quest* (14:1), pp 2–15.

Royce, W.W. 1970. "Managing the Development of Large Software Systems," in: *Proceedings of IEEE WESCON*. p. 9.

Sahal, D. 1985. "Technological Guideposts and Innovation Avenues," *Research Policy* (14:2), pp 61-82.

Sambamurthy, V. 2010. "Editorial Notes," *Information Systems Research* (21:4), pp 661-664.

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.

Sambamurthy, V., and Zmud, R.W. 2000. "Research Commentary: The Organizing Logic for an Enterprise's IT Activities in the Digital Era--a Prognosis of Practice and a Call for Research," *Information Systems Research* (11:2), p 105.

Sanchez, R., and Mahoney, J.T. 1996. "Modularity, Flexibility, and Knowledge Management in Product and Organization Design," *Strategic Management Journal* (17), pp 63-76.

Sandberg, J. 2010. "Coping with Complexity: Exploring Modularity and Flexibility in IT Infrastructure Adaptation," in: *Industrial Informatics: Design, Use and Innovation,* J. Holmström, M. Wiberg and A. Lund (eds.). Philadelphia, USA.: IGI Publishing.

Sanderson, S., and Uzumeri, M. 1995. "Managing Product Families: The Case of the Sony Walkman," *Research Policy* (24:5), pp 761-782.

Sayer, A. 1992. *Method in Social Science: A Realist Approach*. Psychology Press.

Schatzki, T.R. 2005. "The Sites of Organizations," *Organization Studies* (26:3), pp 465-484.

Schilling, M. 2000. "Toward a General Modular Systems Theory and Its Application to Interfirm Product Modularity," *The Academy of Management Review* (25:2), pp 312-334.

Scholz, R.W., and Tietje, O. 2002. *Embedded Case Study Methods: Integrating Quantitative and Qualitative Knowledge*. Sage Publications, Inc.

Schumpeter, J., and Opie, R. 1934. *The Theory of Economic Development*. Springer.

Schumpeter, J.A. 1942. *Capitalism, Socialism and Democracy*. New York: Harper and Row.

Schön, D.A. 1979. "Generative Metaphor: A Perspective on Problem-Setting in Social Policy," *Metaphor and thought* (254), pp 254-283.

Scott, C.R., Quinn, L., Timmerman, C.E., and Garrett, D.M. 1998. "Ironic Uses of Group Communication Technology: Evidence from Meeting Transcripts and Interviews with Group Decision Support System Users," *Communication Quarterly* (46:3), pp 353-374.

Scott, S.V., and Wagner, E.L. 2003. "Networks, Negotiations, and New Times: The Implementation of Enterprise Resource Planning into an Academic Administration," *Information and Organization* (13:4), pp 285-313.

Selander, L., Henfridsson, O., and Svahn, F. 2010. "Transforming Ecosystem Relationships in Digital Innovation," *Int. Conf. on Information Systems*, St Louis, USA.

Selander, L., Henfridsson, O., and Svahn, F. in review. "Capability Search and Redeem across Digital Ecosystems," *submitted to international journal* ( ).

Shah, S. 2006. "Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development," *Management Science* (52:7), pp 1000-1014.

Shapiro, C., and Varian, H. 2000. *Information Rules*. Harvard business school press Boston, MA.

Simon, H. 1962. "The Architecture of Complexity," *Proceedings of the American Philosophical Society* (106:6), pp 467-482.

Simon, H.A. 1971. "Designing Organizations for an Information-Rich World," in: *In Computers, Communications, and the Public Interest (1971),* M. Greenberger (ed.). pp. 37-72.

Simon, H.A. 1973. "The Organization of Complex Systems," in: *Hierarchy Theory: The Challenge of Complex Systems,* H.H. Pattee (ed.). George Braziller, pp. 1-27.

Simon, H.A. 1996. *The Sciences of the Artificial*. The MIT Press.

Simon, H.A. 2002. "Near Decomposability and the Speed of Evolution," *Industrial and Corporate Change* (11:3), pp 587-599.

Simonds, C. 2003. "Software for the Next-Generation Automobile," *IT Professional* (05:6), pp 7-11.

Smith, M.L. 2006. "Overcoming Theory-Practice Inconsistencies: Critical Realism and Information Systems Research," *Information and Organization* (16:3), pp 191-211.

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

Steenson, M. 2009. "Problems before Patterns: A Different Look at Christopher Alexander and Pattern Languages," *Interactions* (16:2), pp 20-23.

Steiner, C.J. 2009. "Ontological Dance: A Dialogue between Heidegger and Pickering," in: *The Mangle in Practice,* A. Pickering and K. Guzik (eds.). Duke University Press.

Strauss, A., and Corbin, J. 1998. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, (2 ed.). Sage Newbury Park, CA.

Sturgeon, T. 2002. "Modular Production Networks: A New American Model of Industrial Organization," *Industrial and Corporate Change* (11:3), p 451.

Suarez, F.F., and Utterback, J.M. 1995. "Dominant Designs and the Survival of Firms," *Strategic Management Journal* (16:6), pp 415-430.

Svahn, F. 2004. "In-Car Navigation Usage: An End-User Survey on Existing Systems," in: *Proceedings of the 27th Information Systems Research Seminar in Scandinavia*. Falkenberg, Sweden.

Svahn, F. 2009. "The Sociomateriality of Competing Technological Regimes in Digital Innovation," in: *Selected Papers of the 32nd Iris Seminar,* J. Molka-Danielsen (ed.). Tapir Academic Press, Trondheim, Norway.

Svahn, F., and Henfridsson, O. 2009. "Situated Knowledge in Context-Aware Computing: A Sequential Multimethod Study of in-Car Navigation," *International Journal of Advanced Pervasive and Ubiquitous Computing* (1:3).

Svahn, F., and Henfridsson, O. 2012. "The Dual Regimes of Digital Innovation Management," *HICSS-45*, Grand Wailea, Hawaii: IEEE Computer Society.

Svahn, F., Henfridsson, O., and Yoo, Y. 2009. "A Threesome Dance of Agency: Mangling the Sociomateriality of Technological Regimes in Digital Innovation," *Int. Conf. on Information Systems*, Phoenix, USA.

Takeuchi, H., and Nonaka, I. 1986. "The New New Product Development Game," *Harvard Business Review* (64:1), pp 137-146.

Tapscott, D., and Williams, A. 2006. *Wikinomics: How Mass Collaboration Changes Everything*. New York, NY: Penguin.

Teece, D., Pisano, G., and Shuen, A. 1997. "Dynamic Capabilities and Strategic Management," *Strategic Management Journal* (18:7), pp 509-533.

Teece, D.J. 1986. "Profiting from Technological Innovation: Implications for Integration, Collaboration, Licensing and Public Policy," *Research Policy* (15:6), December, pp 285-305.

Thiel, S., and Hein, A. 2002. "Modeling and Using Product Line Variability in Automotive Systems," *IEEE Software* (19:4), pp 66-72.

Thompson, J.D., and Bates, F.L. 1957. "Technology, Organization, and Administration," *Administrative Science Quarterly* (2:3), pp 325-343.

Tilson, D., Lyytinen, K., and Sørensen, C. 2010. "Research Commentary---Digital Infrastructures: The Missing IS Research Agenda," *Information Systems Research* (21:4), pp 748-759.

Tiwana, A., Konsynski, B., and Bush, A. 2010. "Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics," *Information Systems Research* (21:4), pp 675-687.

Trigeorgis, L. 1996. *Real Options: Managerial Flexibility and Strategy in Resource Allocation*. the MIT Press.

Tuomi, I. 2002. *Networks of Innovation: Change and Meaning in the Age of the Internet*. Oxford University Press.

Turing, A. 1937. "On Computable Numbers," *Proceedings of the London Mathematical Society* (2:42), pp 230-265.

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

Tushman, M., and Moore, W. 1982. *Readings in the Management of Innovation*. Ballinger.

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

Ulrich, K.T., and Eppinger, S.D. 2004. *Product Design and Development*. McGraw-Hill.

Utterback, J., and Abernathy, W. 1975. "A Dynamic Model of Process and Product Innovation," *Omega* (3:6), pp 639-656.

Utterback, J., and Suarez, F. 1993. "Innovation, Competition, and Industry Structure," *Research Policy* (22:1), pp 1-21.

Utterback, J.M., and O'Neill, R. 1994. *Mastering the Dynamics of Innovation: How Companies Can Seize Opportunities in the Face of Technological Change*. Harvard Business School Press.

Vaast, E., and Walsham, G. 2005. "Representations and Actions: The Transformation of Work Practices with IT Use," *Information and Organization* (15:1), pp 65-89.

Wade, J. 1995. "Dynamics of Organizational Communities and Technological Bandwagons: An Empirical Investigation of Community Evolution in the Microprocessor Market," *Strategic Management Journal* (16:S1), pp 111-133.

Walley, K. 2007. "Coopetition: An Introduction to the Subject and an Agenda for Research," *International Studies of Management and Organization* (37:2), pp 11-31.

Walsham, G. 1993. *Interpreting Information Systems in Organizations*. John Wiley & Sons, Inc. New York, NY, USA.

Walsham, G. 2006. "Doing Interpretive Research," *European Journal of Information Systems* (15:3), p 320.

Van de Ven, A. 1986. "Central Problems in the Management of Innovation," *Management Science* (32:5), pp 590-607.

Van de Ven, A.H. 2005. "Running in Packs to Develop Knowledge-Intensive Technologies," *MIS Quarterly* (29:2), pp 368-378.

Vanhaverbeke, W., Van de Vrande, V., and Chesbrough, H. 2008. "Understanding the Advantages of Open Innovation Practices in Corporate Venturing in Terms of Real Options," *Creativity and Innovation Management* (17:4), pp 251-258.

Weber, M. 1949. *The Methodology of the Social Sciences*. Glencoe, IL: Free Press

Weick, K.E. 1976. "Educational Organizations as Loosely Coupled Systems," *Administrative Science Quarterly* (21:1), pp 1-19.

Wenger, E. 1999. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge university press.

West, J.W., and Gallagher, S. 2006. "Challenges of Open Innovation: The Paradox of Firm Investment in Open-Source Software," *R&D Management* (36:3), June, pp 319-331.

Westergren, U. 2011. "Disentangling Sociomateriality: An Exploration of Remote Monitoring Systems in Interorganizational Networks," in: *Informatics*. Umeå: Umeå University.

Widrow, B., Glover, J., McCool, J., Kaunitz, J., Williams, C., Hearn, R., Zeidler, J., Dong Jr, E., and Goodlin, R. 1975. "Adaptive Noise Cancellation: Principles and Applications," *proc. IEEE* (63:12), pp 1692-1716.

Williamson, O. 1973. "Markets and Hierarchies: Some Elementary Considerations," *The American Economic Review* (63:2), pp 316-325.

Williamson, O.E. 1971. "The Vertical Integration of Production: Market Failure Considerations," *The American Economic Review* (61:2), pp 112-123.

Wimelius, H. 2011. "Duplicate Systems: Investigating Unintended Consequences of Information Technology in Organizations," in: *Informatics*. Umeå: Umeå University.

Wirfs-Brock, R. 2009. "Creating Sustainable Designs," *IEEE Software* (26:3), pp 5-7.

von Hippel, E. 1988. *The Sources of Innovation*. Oxford, New York: Oxford University Press.

Von Hippel, E. 1990. "Task Partitioning: An Innovation Process Variable," *Research Policy* (19:5), pp 407-418.

von Hippel, E. 2007. "Horizontal Innovation Networks - by and for Users," *Industrial and Corporate Change* (16:2), pp 293-315.

Yin, R.K. 2003. *Case Study Research: Design and Methods*, (4 ed.). Sage Publications.

Yoo, Y. 2010. "Computing in Everyday Life: A Call for Research on Experiential Computing," *MIS Quarterly* (34:2), pp 213-231.

Yoo, Y., Boland, R., Lyytinen, K., and Majchrzak, A. 2009. "Call for Papers–Special Issue: Organizing for Innovation in the Digitized World," *Organization Science* (20:1), pp 278-279.

Yoo, Y., Henfridsson, O., and Lyytinen, K. 2010a. "The New Organizing Logics of Digital Innovation: An Agenda for Information Systems Research," *Information Systems Research* (21:4), pp 724-735.

Yoo, Y., Henfridsson, O., and Lyytinen, K. 2010b. "Research Commentary: The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research," *Information Systems Research* (21:4), pp 724-735.

Yoo, Y., Lyytinen, K., Boland, R., Berente, N., Gaskin, J., Schutz, D., and Srinivasan, N. 2010c. "The Next Wave of Digital Innovation: Opportunities and Challenges: A Report on the Research Workshop'digital Challenges in Innovation Research," Fox School of Business & Management, Temple University, Philadelphia, PA, USA.

Yoo, Y., Lyytinen, K., and Jr., R.J.B. 2008. "Distributed Innovation in Classes of Networks," in: *HICSS*. IEEE Computer Society, p. 58.

Yoo, Y., Lyytinen, K., Thummadi, B., and Weiss, A. 2010d. "Unbounded Innovation with Digitalization: A Case of Digital Camera," in: *Academy of Management*. Montreal, Canada.

Yoo, Y., Lyytinen, K., and Yang, H. 2005. "The Role of Standards in Innovation and Diffusion of Broadband Mobile Services: The Case of South Korea," *Journal of Strategic Information Systems* (14:3), pp 323-353.

Zittrain, J. 2006. "The Generative Internet," *Harvard Law Review* (119:7), pp 1974-2040.

Zittrain, J. 2008. *The Future of the Internet: And How to Stop It*. Yale Univ Pr.

Åkesson, M. 2009. "Digital Innovation in the Value Networks of Newspapers." Gothenburg: Gothenburg University.

Published Reports in Publication Series

Research Reports in Information Processing and Computer Science. UMADP-RRIPCS with ISSN 0282-0579 (series terminated Spring 1996) and

Research Reports, Department of Informatics. RR with ISSN 1401-4572 (series started Autumn 1996)

UMADP-RRIPCS

1. Bergman, B., Brodén, B. & Granlund, J.: Från projektering till produktion – Erfarenheter från två installationer av datorbaserade tillverkningssystem. Report UMADP-RRIPCS l.84, 1984.
2. Ivanov, K.: Expert-Support Systems: The New Technology and the Old Knowledge. Report UMADP-RRIPCS 2.86, 1986.
3. Forsgren, O.: Samskapande datortillämpningar – en systemteoretisk ansats för lösning av vissa förändringsproblem vid administrativ datoranvänd¬ning. Report UMADP-RRIPCS 3.87, 1987. (Doctoral Thesis).
4. Zellini, P. & Ivanov, K.: Humanistic and Ethical Aspects of Mathematics. Report UMADP-RRIPCS 4.88, 1988.
5. Nilsson, K.: Project Description – Design of Interactive Information Systems. Report UMADP-RRIPCS 5.87, 1987.
6. Nilsson, K.: Some Problems on Data Modelling and Interactive Database Applications. Report UMADP-RRIPCS 6.88, 1988.
7. Nilsson, K.: A Model of Relational Algebra. Report UMADP-RRIPCS 7.89, 1989.
8. Nilsson, K.: Designing for Creativity – Toward a Theoretical Basis for the Design of Interactive Information Systems. Report UMADP-RRIPCS 8.89, 1989.
9. Forsgren, O. & Ivanov, K.: From Hypertext to Hypersystem. Report UMADP-RRIPCS 9.90, 1990.
10. Ivanov, K.: Learning to Design Learning Systems – The Metaphor of Future Generations and Computer Technology. Report UMADP-RRIPCS 10.90, 1990.
11. Ivanov, K.: Critical Systems Thinking and Information Technology – Some Summary Reflections, Doubts, and Hopes Through Critical Thinking Criti-cally Considered, and Through Hypersystems. Report UMADP-RRIPCS 11.90, 1990.
12. Ivanov, K.: Information Systems Design Through Creativity – A tutorial route towards aesthetics and ethics. Report UMADP-RRIPCS 12.90, 1990.

13. Ivanov, K.: Hypersystems – A Base for Specification of Computer-Supported Self-Learning Social Systems. Report UMADP-RRIPCS 13.91, 1991.
14. Stolterman, E.: Designarbetets dolda rationalitet – En studie av metodik och praktik inom systemutveckling. Report UMADP-RRIPCS 14.91, 1991. (Doctoral Thesis).
15. Whitaker, R.: Venues for Contexture – A Critical Analysis and Enactive Reformulation of Group Decision Support System. Report UMADP-RRIPCS 15.92, 1992. (Doctoral Thesis).
16. Ivanov, K.: Proceedings of the 14th IRIS – Revised papers of the 14th Information Systems Research Seminar in Scandinavia, Umeå - Lövånger, 11-14 August, 1991. Report UMADP-RRIPCS 16.92, 1992.
17. Forsgren, O. m.fl.: Idealorienterad Design – Om konsten att hålla idealen levande i systemutveckling. Report UMADP-RRIPCS 17.94, 1994.
18. Grönlund, Å.: Public Computer Systems, The Client-Organization Encounter, and the Societal Dialogue. Report UMADP-RRIPCS 18.94, 1994. (Doctoral Thesis).
19. Levén, P.: Från användning till handling – Om kvalitet i ett marknads-orienterat informationssystem. Report UMADP-RRIPCS 19.95, 1995. (Licentiate Thesis).

Research Reports

1997

97.01 Kaptelinin, V & Nardi, B A: The Activity Checklist: A Tool for Representing the "Space" of Context. Report RR-97.01, 1997.

97.02 Levén, P.: Kontextuell IT-förståelse. Report RR-97.02, 1997. (Doctoral Thesis).

1998

98.01 Ågren, P-O.: Att förstå virtualisering. Report RR-98.01, 1998. (Licentiate Thesis).

98.02 Waterworth, J.: Virtual Reality in Medicine: A Survey of the State of the Art. Report RR-98.02, 1998.

1999

99.01 Henfridsson, O.: IT-adaptation as Sensemaking. Report RR-99.01, 1999. (Doctoral Thesis).

2000

00.01 Holmström, J.: Information System and Organization as Multipurpose Network. Report RR-00.01, 2000. (Doctoral Thesis).

2001

01.01 Lindh-Waterworth, E.: Perceptually-Seductive Technology – designing computer support for everyday creativity. Report RR-01.01, 2001. (Doctoral Thesis).

01.02 Wiberg, M.: In between Mobile Meetings: Exploring Seamless Ongoing Interaction Support for Mobile CSCW. Report RR-01.02, 2001. (Doctoral Thesis).

2003

03.01 Lund, A.: Massification of the Intangible. An Investigation into Embodied Meaning and Information Visualization. Report RR-03.01, 2003. (Doctoral Thesis).

03.02 Nordström, T.: Information Systems Stewardship – advancing utilisation of information technology in organisations. Report RR-03-02, 2003. (Doctoral Thesis).

03.03 Wiberg, C.: A Measure of Fun. Extending the scope of web usability. Report RR-03.03, 2003. (Doctoral Thesis).

03.04 Fällman, D.: In Romance with the Materials of Mobile Interaction: A Phenomenological Approach to the Design of Mobile Information Technology. Report RR-03.04, 2003. (Doctoral Thesis).

2005

05.01 Lindblad-Gidlund, K.: Techno therapy – A relation with information technology. Report RR-05.01, 2005. (Doctoral Thesis).

05.02 Raoufi, M.: How can I help you? The delivery of e-government services by means of a digital assistant. Report RR-05.02, 2005. (Doctoral Thesis).

2006

06.01 Amcoff-Nyström, C.: Designing Intranets for Viability – Approaching organizational empowerment and participation. Report RR-0601, 2006 (Doctoral Thesis).

06.02 Jakobsson, M.: Virtual Worlds and Social Interaction Design. Report RR-06.02, 2006 (Doctoral Thesis).

06.03 Croon Fors, A.: Being-with Information Technology: Critical explorations beyond use and design. Report RR-06.03 (Doctoral Thesis).

2008

08.01 Nyberg, A. F.: Att studera digitala artefakter I människors vardagsliv. Report RR-08.01, 2008 (Doctoral Thesis).

08.02 Waterworth, J.A., L-Waterworth, E, Riva, G and Mantovani, F: Form, Content and Consciousness – An Evolutionary Account of the Sense of Presence in Real and Mediated Environments. Report RR-08.02, 2008.

2009

09.01 Jegers, K.: Pervasive Gameflow – Exploring and identifying the mechanisms of player enjoyment in pervasive games. Report RR-09.01, 2009 (Doctoral Thesis).

09.02 Orre, C.J.: Using Technology with Care – Notes on Technology Assimilation Processes in Home Care. Report RR-09-02, 2009 (Doctoral Thesis).

09.03 Harr, R.: Striking a Balance – Managing Collaborative Multitasking in Computer-Supported Cooperative Work. Report RR-09-03, 2009 (Doctoral Thesis).

2010

10.01 Skog, D..: Mjukvarumiljöer för gemenskap – en studie av nätgemenskap, teknik och kultur. Report RR-10.01, 2010 (Doctoral Thesis).

10.02 Jonsson, K.: Digitalized industrial equipment: An investigation of remote diagnostics services. Report RR-10.02, 2010 (Doctoral Thesis).

10.03 Danielsson-Öberg , K.: Att främja medverkan: utmaningar och möjligheter för barns och ungdomars delaktighet vid design av edutainmentspel. Report RR-10.03, 2010 (Doctoral Thesis).

10.04 Björkman, C, Elovaara, P, Sefyrin, J and Öhman, M: Travelling thoughtfulness – feminist technoscience stories. Report RR-10.04, 2010 (Antologi).

2011

11.01 Wimelius, H.: Duplicate systems: Investigating unintended consequences of information technology in organizations. . Report RR-11.01, 2011 (Doctoral Thesis).

11.02 H. Westergren, U.: Disentangling Sociomateriality: An Exploration of Remote Monitoring Systems in Interorganizational Networks. Report RR-11.02, 2011 (Doctoral Thesis).

2012

12.01 Hoshi, K.: Here and Now: Foundations and Practice of Human Experiential Design. Report RR-12.01, 2012 (Doctoral Thesis).

12.02 Svahn, F.: Digital Product Innovation: Building Generative Capability through Architectural Frames. Report RR-12.02, 2012 (Doctoral Thesis).