Chapter 2. The Changing Industry Structure of Software Development for Consumer Electronics

This chapter is published as:

Abstract
During the last decade the structure of the consumer electronics industry has been changing profoundly. Current consumer electronics products are built using components from a large variety of specialized firms, whereas previously each product was developed by a single, vertically-integrated company. Taking a software development perspective, we analyze the transition in the consumer electronics industry using case studies from digital televisions and mobile phones. We introduce a model consisting of five industry structure types and describe the forces that govern the transition between types and we describe the consequences for software architectures.

We conclude that, at this point in time, Software Supply Chains is the dominant industry structure for developing consumer electronics products. This is because the modularization of the architecture is limited, due to the lack of industry-wide standards and because resource constrained devices require variants of supplied software that are optimized for different hardware configurations. Due to these characteristics open ecosystems have not been widely adopted. The model and forces can serve the decision making process for individual companies that consider the transition to a different type of industry structure as well as provide a framework for researchers studying the software-intensive industries.

2.1 Introduction

The consumer electronics industry, i.e. TVs, set-top boxes (STBs), audio and video storage devices and mobile phones, has gone through substantial changes during the last 15 years. From its inception until around 2000, the industry was dominated by a small group of consumer electronics companies. Each company was responsible for the complete design of its products and developed many of the product components. For decades, the cost of designing a product was negligible, compared with the material and manufacturing costs. However, as the functionality increased as the capabilities of their integrated circuits followed Moore’s law; the relative design cost became a significant factor. Furthermore, as processors became fast enough to process video and audio in software, it became possible
to serve a wider customer base with more product variants from a single hardware platform. This contributed to a shift in the balance of development costs from hardware to software. In order to amortize these development costs more widely, independent software and IC companies emerged, either as spin-offs from consumer electronics firms or as newly created companies, and these became major players.

In this paper we analyze the transitions in the structure of the consumer electronics industry and we introduce a model that describes the different industry structures that we have identified. These structures are presented from the perspectives of software architecture and industry structure. Since the consumer electronics industry structure is still in flux, we draw analogies from the computer industry, which went through a similar evolution. Andy Grove described the changing structure that took place in the computer industry between 1980 and 1995 [Grove 1996]. In 1980 that industry was dominated by a small group of companies that behaved as vertical silos, meaning that they developed the entire system, from ICs to sales and distribution, see Figure 3, which is adapted from [Grove 1996].

By 1995 the vertical silos had been replaced by modular clusters, i.e. groups of companies that were dominant in certain parts, or layers, of the system. This transition was facilitated by the high degree of modularity that was created [Baldwin1997, Christensen2004]. This transition resulted in an ecosystem, which is formulated by Moore follows: “An intentional community of economic actors whose individual business shares in some large measure the fate of the whole community” [Moore 2006].

Some authors, e.g. [Genugten 2007] and [Suoranta 2006], predicted that the consumer electronics industry would follow an evolution similar to the computer industry. So far, however, this state has not been achieved and in this paper we will show why this is not the case. To explain the current industry structure and its evolution, we have developed a model consists of five industry structure types:

- Type 1: Vertically Integrated Firms
- Type 2: System Integrators and Specialized Suppliers
- Type 3: Supply Chains

![Figure 3 Changing structure in the computer industry](image)
The changing industry structure of software development for consumer electronics

Type 4: Closed Ecosystems
Type 5: Open Ecosystems

In this paper we show that the evolution in the consumer electronics industry followed a structured, logical pattern and, at the time of writing, a software supply chain is the dominant industry structure. The industry has not evolved to type 4 or 5 due to the characteristics of the products, i.e. the need to make optimal use of system resources, the high degree of variability and the high rate of innovation. The consequence of this is a lack of industry-wide standards and hence a limited degree of modularity.

The key contributions of this paper are twofold:

- We introduce a model consisting of five industry structure types which captures the evolution of software development for consumer electronics. This model is developed using a case study research method, with case studies from digital televisions and mobile phones.
- We describe the forces that govern the transition between types and the consequences for product line architectures. These forces can serve the decision making process for individual firms for selecting the most appropriate industry for their business.

The remainder of the paper is structured as follows. In Section 2.2 we present background and related work, followed by a section concerning our research method. Section 2.3 presents a high level overview of the case studies of digital televisions and mobile phones, from which we build our Industry Structures model that is presented in Section 2.4. Section 2.5 and 2.6 provided an elaboration of the case studies, focusing on the rationale behind the transitions and the consequences for software architectures. This paper concludes with a comparison of related work, our conclusions and the identification of areas for further research.

2.2 Background

In this section we describe the characteristics of embedded and consumer electronics products, and related work on industry structures and transitions.

2.2.1 Characteristics of consumer electronics products

A consumer electronic product consists of a combination of hardware and software that is designed to perform specific functions, often with real-time performance requirements and constrained computing resources. The products include DVD-players, televisions, cameras and mobile phones. Since different market segments usually have different requirements, a company may develop a range of variants to serve different groups of customers [Ommering 2004]. Each group of customers receives products that serve their specific needs. The concept of product line engineering is widely applied in consumer electronics [PHOF2010]. This allows for an efficient creation of variants of product using reusable development artifacts [Pohl 2005A].

There is a high pressure on the cost price, so the integrated circuits (ICs) should have the smallest possible footprint. Furthermore, for handheld products, these ICs should use as little
energy as possible, given the limited power sources and options for heat dissipation. In order to satisfy these requirements, the amount of software has to be minimized as much as possible and the software must operate as efficiently as possible. As a consequence, an IC usually only contains the software that is needed for the particular product, or a group of similar products, in which it is used.

2.2.2 Organizational networks and software supply chains

Michael Porter introduced the term Value Chain to describe the activities needed to make a product and the value that each of activities created for the end product [Porter 1980]. Later this method has been extended to analyze Value Chains in an inter-organizational context, denoted as the Value System [Porter 1980], including the activities of upstream and downstream participants in a supply chain. A supply chain is defined as: “A network that starts with raw material, transforms them into intermediate goods, and then into final product delivered to customers. Each participant consumes logical and/or physical products from one or more upstream suppliers, adds value, usually by incorporating them into more complex products, and supplies the results to downstream consumers” [Lee 1994]. In the software industry, companies have emerged that are specialized in a certain software product or service [Greenfield 2004]. As an example, consider the supply chain in Figure 4. There are four links, starting with an audio decoding software vendor and ending with a car manufacturer.

![Figure 4 Example of a software supply chain](image)

In a supply chain, each of the participants uses components containing variability, combines them with in-house developed components, and delivers specialized components containing variability to its customers. For example, a manufacturer of car infotainment systems uses multiple suppliers for media processing, navigation, and connectivity to create a product line for different car manufacturers.

Jansen et al. introduced the term Software Supply Networks (SSN) and identified that the software architecture is created based on products and services from other parties in the network [Jansen 2007]. Peppard and Rylander [Peppard2006] identified the strong cooperative behavior and relationships between the parties in such a network.
2.2.3 Industry de-verticalization and creation of software ecosystems

Moore formulated an ecosystem as follows. “An intentional community of economic actors whose individual business shares in some large measure the fate of the whole community.” [Moore 2006]. He described how an ecosystem is a network of interdependent niches that are occupied by organizations. These niches are more or less open, to the degree that they embrace alternative competitors. A general characteristic of an ecosystem is that most actors benefit from innovations that give value to the end-user. For instance Eclipse [Eclipse 2010], an open source multi-language software development environment, offers a user interface framework into which any company can add functionality though plug-ins, thus creating an overall product which serves a broad group of software developers [McGregor 2009]. For economic reasons, it is not feasible for one company to create all this functionality on its own. In earlier work [Bosch 2009], one of the co-authors identified that a company could use the notion of ecosystems to expand the variability of a product line by encouraging other companies or end-users to add functionality.

In her analysis of the IBM System/360 mainframe computer, Baldwin observed that the de-verticalization of the industry and formation of an ecosystem was enabled by its high degree of modularity. Baldwin came to the conclusion that a company could increase its return on capital substantially by becoming a dominant leader in a cluster by setting industry standards and by using designs from other companies [Baldwin 1997]. An important prerequisite for an industry to be organized as modular clusters is that the interfaces between the layers must be standardized. This makes it possible for a system integrator to integrate software and hardware from different suppliers easily. Also, it allows an application developer to easily distribute a software solution to a large group of platform users. These characteristics explain the huge commercial success of computers based on the Intel and Microsoft platform.

Christensen et al. [Christensen 2002] analyzed several cases of de-verticalization, in their paper denoted as disintegration, and identified that a vertical industry structure was needed when the performance demands didn’t allow for a modular architecture. Once the performance delivered exceeded the customers’ expectations, modularization of the architecture was possible and a more horizontal industry structure emerged. Raskin and Mellquist identified the enablers for the de-verticalization of an industry, including standardization and modularity. They also identified stages of de-verticalization, which could start with outsourcing and identified that there are companies that recognize an opportunity to use a third-party supplier to gain an advantage over their competitors [Raskin 2005].

Messerschmitt and Szyperski [Messerschmitt 2004] identified the factors that govern ecosystems and the role of architecture since it defines the system, as well as the structure of the ecosystem. The typical layered architecture results in less efficient implementations, but in many cases Moore’s law allowed this form to emerge. Messerschmitt and Szyperski identified that the suppliers of middleware and spanning layers, i.e. layers that connect application layers to platforms, could become dominant players in the market.
Chapter 2

The concept of ecosystems has also emerged in E-business, here denoted Digital Business Ecosystems (DBE). The introduction of internet and modern ICT tools allows companies to interact more easily and create a network of companies that deliver shared products and/or services [Nachira 2007]. Similarly to the case of software ecosystems, technical standards are needed for a DBE to prosper. In the DBE domain the standards relate to data interchange between companies, shared databases [Dourmas 2009] and common reference architectures [Janjua 2009].

2.2.4 Research method

The research reported in this paper uses a case study research method as proposed by Yin [Yin 2003] and Eisenhardt [Eisenhardt 1989] and is based on two cases. The research questions for these case studies are based on the comparison with the transition in the computer industry [Grove 1996], the role that modularization and standardization plays in such a transition [Baldwin1997, Christensen 2004, Messerschmitt 2004, Raskin 2005] and the characteristics of embedded systems. The research questions are defined as follows:

- What transitions between industry structures have taken place? What where the forces and actors involved?
- How were the architectural interfaces between the actors defined, and what were the consequences for modularity, variability and product integration?

As the first case study we selected the transition of the development of televisions for the following reasons: (1) It describes the transition of a mainstream consumer electronics product with a significant amount of software; (2) The transition took place over a long period of time and involves multiple geographic regions; (3) We have access to detailed information on this case, being members of the involved organizations.

For this case study we used multiple sources: we acted as a participant-observer, we conducted interviews and we applied reviews on our findings within the involved organizations. We used information from internal presentations and technical descriptions as well as external literature. Much of this information is classified, and we therefore only make references to literature that was published outside the involved organizations.

As a second case study we selected the development of Mobile Phones for the following reasons: (1) mobile phones contain a significant amount of software and the market was initially dominated by a small group of vertically integrated firms (2) the product category is relatively young and has gone through a rapid evolution in an overlapping, but different time period, (3) the type of product contrasts that of a television, since mobile phones have a different feature set and business model (4) a significant amount of literature is available for this industry, covering multiple players. The data for the case study on mobile phones is exclusively derived from descriptions in literature.

By using significantly different cases and by using different data collection methods, our theory provides a strong basis for external validity [Eisenhardt 1989, Yin 2003].
2.3 Case Studies: Transition in the Development of Digital Televisions and Mobile Phones

This section presents an overview of the transitions in the development of software for digital televisions and that for mobile phones. In the Sections 2.5 and 2.6 we provide a more detailed description of each case study.

2.3.1 Transition in the development of digital televisions from 1990 until 2010

The examples relate to experiences at Philips Electronics and NXP Semiconductors, which were top-5 organizations, based on market share. At the time that the case study commenced in the early 1990s, Philips’ Product Divisions included Consumer Electronics and Semiconductors. Philips Semiconductors was spun out in 2006 to become an independent company, NXP Semiconductors. For reasons of clarity, these steps are described as discrete and sequential steps. In reality most of these steps have been gradual and sometimes overlapping.

1. **Analog products for specific regions.** Televisions were initially mainly analog devices and most functionality was determined by analog hardware components. Because of the different TV standards used in different regions, product development in, for instance, Europe and North America was independent. A modest portion of the functionality was implemented through software and a single team created the entire product.

2. **Increasing software content and global development.** As processing power became cheaper, TV software became more elaborate and the size of the software grew rapidly. It was no longer economic to support independent developments for different regions so an intra-organizational software product line approach was adopted. The systems were still developed by individual firms that created the entire product, but the development was distributed over multiple sites around the world and served products for each geographic region. This transition took place in the period 1995 to 1998.

3. **Integration of connectivity features.** COTS (commercial, off-the-shelf) components and software from independent software vendors were integrated in the system, e.g. for Wi-Fi connectivity and photo browsers photos in TV. The architecture was maintained by the integrator, i.e. the TV set maker, and was adapted to integrate the third party software. This transition took place around 2003.

4. **Specialized manufacturers of Systems on Chips.** The reception of digital broadcasts by TVs favored a move to processing video in software. This resulted in a transformation from a set of analog processing ICs to a single, digital System on Chip (SoC) solution. Around 2006 the television manufacturers spun out their IC manufacturing. As set makers have different requirements and software architectures, IC manufacturers created specialized software variants for each set maker. The overall software architecture was a combination of that of the set maker and the IC manufacturer.

5. **Integrators and suppliers of middleware.** Around 2008, independent software vendors (ISVs) and software integrators entered the TV market. These vendors
developed a substantial part of the software and determined a large part of the functionality of the product, e.g. middleware software. The overall architecture now had contributions from the IC manufacturer, the middleware supplier and the set maker, requiring a staged integration.

6. **Attempts to create industry standards.** To reduce the time needed for software integration, to serve more customers and to avoid dependency on specific ISVs, set makers attempted to create industry-wide standards. Due to commercial interest, the need for optimal resource utilization and fast innovation speed, none of these attempts has been successful.

### 2.3.2 Transitions in the development of mobile phones from 1990 until 2010

This case study relates to the development of mobile phones and encompasses that of the major players in this industry, including the new entrants. As in the case of digital televisions, these transitions have not taken place in discrete and sequential steps. Furthermore, the companies that initially acted as the vertically integrated companies executed these steps differently.

1. **Small group of vertically integrated companies.** Initially mobile phones were developed by a small group of vertically integrated companies that developed the entire product and sold these to their customers, usually through the Mobile Network Operators. The functionality was limited to that of voice calls and SMS.

2. **Introduction of functionality outside the traditional telephone area.** New functionality was introduced to mobile phones, such as cameras, downloadable ringtones and MP3 playback. The vertically integrated companies used specialized suppliers for these components, mostly originating from consumer electronics firms, and integrated these components into their products. This transition happened around 1996.

3. **Introduction of smartphones / creation of mobile platforms.** Functionality that was originally used in Personal Digital Assistants (PDAs), such as touch screens, Wi-Fi connection and GPS was incorporated into mobile phones. This introduced the need for dedicated application processors and operating systems. Separate hardware and software platform suppliers were created, often through spin-offs of the vertically integrated companies. The overall software architecture was now controlled by a combination of the mobile phone manufacturers, the operating system vendors and the platform suppliers. This transition happened around 2002 and still exists today. Nokia, the company with the largest market share worldwide, acted much longer than other mobile phone manufacturers as a company that owned full control over the software architecture. Two firms, i.e. RIM and Apple still control the overall architecture and use specialized suppliers for specific parts of their products.

4. **Connection to internet and third party application development.** The availability of third Generation (3G) technology made it possible for faster connection to the internet. An API was created that enabled third parties to develop applications that the end-user could download and purchase through the internet, thus forming an ecosystem. This situation emerged around 2008 and still exists today.
5. **Attempts at industry standardization.** Several attempts for industry standardization have been made since the year 2000 in order to create a modular architecture, allowing for easier product integration and multiple parties to contribute more easily. For instance several attempts have been made to define a common software platform. Most of these attempts failed to gain industry-wide attraction and were abandoned after a few years.

2.3.3 **Analysis of these case studies**

From these case studies we learn the following:

- The transitions took place in several small steps. The initial steps of de-verticalization started when functionality outside the traditional domain was introduced for which specialized suppliers where used.
- The fast growth of the amount of software started when functionality was shifted from being implemented in hardware towards software. Product manufacturers had to focus their investments and spun out their IC business. Also, software integrators, middleware suppliers and operating systems vendors emerged. The overall software architecture was now controlled by several parties. Consequently products are created through a supply chain where each customer receives specialized components from its suppliers.
- Standardization attempts for creating a modular architecture across the industry have failed due to different viewpoints of the involved parties and rapid changes of the technology.
- For the development of downloadable applications, ecosystems have been created, currently mainly for mobile phones.
- There is a great deal of similarity between the transitions in these two case studies, but the transition of mobile phones occurred at a much faster pace because the Mobile Network Operators enabled large investments. Furthermore some players act as more vertically integrated companies.

2.4 **Model of Industry Structures for Software Development in Consumer Electronics**

When analyzing the case studies presented in the previous section, we identified that various approaches are deployed by companies in the software development industry for consumer electronics. In this section we present a model that captures these types of industry structures and the rationale to move for an individual firm to move from one structure to another. We provide examples from the consumer electronics domain as well as examples from other software products to illustrate the different types.

Our model is based on 5 “types”. Figure 5 gives a simplified graphical representation of how software architecture might look like for the different types of clusters. Here the different colored segments represent the relative size of the stack that different companies contribute to the system.
Chapter 2

Figure 5 Model of industry structure types, visualized

Type 1: Vertically Integrated firms
The vertically integrated firm develops their entire product and delivers it to the market. An integrated firm fully owns the specification, architecture and develops the software based on its customers’ needs.
Characteristics: Complete control over the architecture; close contact with the final customer; all functionality has to be developed by one company.
Examples: Digital televisions and mobile phones during the early nineties; the computer software industry in the 1970s; products with small dedicated software such as coffee machines, shavers and pacemakers.

Type 2: System integrators and specialized suppliers
This structure consists of system integrators that, like the vertically integrated firms, are responsible for the end-solution. The system integrators use components from specialized suppliers to reduce their own development effort and/or when the knowledge is not available to develop that part of the product. The system integrator is responsible for the requirements of the product and the high-level software architecture. It delivers the product to the market.
Characteristics: The system integrator has complete control over the overall architecture and has close contact with the final customer; a part of the functionality is developed by third parties; one party, typically the product manufacturer, acts as integrator, using one or more suppliers.
Examples: Digital televisions between 2003 and 2006; the majority of mobile phones between 1996 and 2002 and Nokia until around 2008, manufacturers of PDAs or smart phones, such as Apple and RIM (see Section 2.6).

Type 3: Supply Chains
In this type the product is developed by a group of companies, each with their own specialty. The difference with type 2 is that the system integrator does not own the product architecture.
The changing industry structure of software development for consumer electronics

Instead, the product architecture is shared by the players and might not be defined through rigorous industry-wide standards. The software is developed using supply chains, where the supplied software has to be tailored for each customer and integration may be done in stages. The software integration accounts for a substantial proportion of the development effort.

Characteristics: The participants share control over the architecture; the functionality is developed by several parties; each party delivers specialized variants of products to the next party in the chain.

Examples: Digital televisions, mainstream mobile phones; hard disk recorders, ZigBee wireless devices [Hartmann 2009].

Type 4: Closed Ecosystem
In an ecosystem, the product is developed by a group of companies and the product architecture is defined through standards. In a closed ecosystem the ecosystem is formed around one specific vendor [Popp 2010]. This vendor opens up their platform, e.g. by offering an API or a domain specific language, so that other parties can add functionality for their own use. Other parties could be the customers, the end-users or third party software vendors. The advantage for the vendor is that it can offer a wider product line with less development effort and the customers or third parties can add functionality more easily [Bosch 2009].

Characteristics: Control over the architecture is owned by the party that develops the platform; third parties and customers can add functionality, based on close contact with the final customer.

Examples: Microsoft Windows, some of Intuit’s applications, Apple iPhone applications, eBay, SAP.

Type 5: Open Ecosystem
In an open ecosystem the architecture is rigorously defined through industry standards and the several components are interchangeable. The standards are open, meaning that they are available for anybody to implement, well documented and unencumbered by intellectual property mechanisms [Messerschmitt 2004]. Open standards allow competition at the subsystem level, since suppliers can create competitive components. Hence the likelihood of one firm becoming dominant is reduced.

The “openness” of an ecosystem is not an absolute because some of these properties can be relaxed making the standard less open but still not closed or proprietary [Messerschmitt2004].

Characteristics: The architecture is defined through open industry standards. The integrator has close contact with the final customer and can create a product by combining interchangeable components; all parts of the system can developed by alternative parties and are interchangeable; Tailor-made solutions can be made by several parties for specific customer needs.

Examples of open standards: The USB interface, JPEG, MPEG compression standards, HTTP and HTML web standards

Examples of open ecosystems: Eclipse Development Environment, Web browsers and servers
2.4.1 Rationale for transitions between industry structures

The types discussed categorize different approaches taken by companies, but over time the approach taken by a company or a company cluster evolves. Typically, an industry evolves from a vertically integrated to a more ecosystem-centric approach due to the increasing amount of functionality. However, there also are forces that restrain this evolution. In Figure 6 we show the forces that are driving companies to evolve over time.

![Figure 6 Forces of moving from one industry type to another](image)

The advantages of staying, or moving to a more vertically integrated approach can be summarized as follows:

- **Force (F1):** Modular and layering introduces inefficiency in the implementation [Christensen 2002]. In a vertically integrated approach, direct control of the hardware is possible, thus allowing for better resource utilization.

- **Force (F2):** In a supply chain or ecosystem centric approach there are fewer possibilities to change the system architecture and APIs since the industry structure depends on its stability. This may constrain innovation [Messerschmitt 2004] and impose the requirement for backwards compatibility. Two examples: the Eclipse development environment doesn’t support backwards compatibility, requiring that the whole ecosystem has to make changes for each major release to remain compatible; Android supports backward compatibility of the core API since that part of the implementation is controlled by Google [Halasz 2011]. One co-author, Bosch, identified that, in case of different evolving hardware configurations, compatibility of applications may easily be broken and could easily become a major source of inefficiency [Bosch 2009], which is a major concern for embedded...
The changing industry structure of software development for consumer electronics

devices. A company that uses a more vertically integrated approach has the freedom to innovate more freely.

- Force (F3): The overall product quality depends on the combination of software from different vendors and failures often occur because of component interaction, unclearly documented APIs or unknown use of the system [Trew 2004, Trew 2006, Oberhauser 2007]. A more vertically integrated company has control over the architecture and its constituent components and can therefore guarantee the product quality more easily.

The advantages of moving to a more ecosystem-centric approach can be summarized as follows:

- Force (F4): Specialized firms are used to develop part of the system. In this way the total development costs are reduced because an individual firm does not have to invest in developing the entire software stack [Genugten 2007].

- Force (F5): Variability can be offered more easily to the customers by using components from different suppliers or by enabling third parties and customers to develop the functionality they need [Christensen 2002, Moore 2006, Bosch 2009].

- The disadvantages, or consequences, of moving to a more ecosystem-centric approach can be summarized as follows:

- Force (F6): Increased uncertainty and time needed for system integration [Bosch 2006, Underseth 2007]. This is especially the case in the Supply Chain, in which heterogeneous architectures have to be combined [Vitharana 2003].

- Force (F7): Co-operating with suppliers leads to additional interaction costs [Raskin 2005].

- Force (F8): When the industry adopts an ecosystem centric approach, some firm may be able to take a dominant position and create a monopoly [Moore 2006]. This is often possible for a firm that develops the middleware or the spanning layer [Messerschmitt 2004].

2.5 Transitions in the Development of Digital Televisions

In this section we present the rationales and consequences of the transition of software development for digital televisions using the historical perspective, as presented in Section 2.3 and present it from the viewpoint of Philips and NXP Semiconductors. We end this section with a summary.

2.5.1 Analog products for specific regions (vertically integrated)

2.5.1.1 Forces and actors

Until the early 2000s, TVs were receiving analog TV signals and the processing also occurred in the analog domain. Televisions also supported broadcast data: closed captions in North America and Teletext in Europe. This was also primarily implemented in hardware. A small amount of software was required to provide a limited user interface and to set values
in the hardware registers in response to user inputs. The cost of developing the software was
ingsignificant compared with the manufacturing costs.

Because of the substantial differences between the standards used in different regions,
such as different standards for broadcasting signals, there was little commonality between
the hardware used in the sets. The TVs were therefore developed locally for each region.
The actors were the set makers that created the entire product, e.g. Philips, Sony and Zenith.

2.5.1.2 Software architectures

Most of the functionality and variability was determined by the selection of hardware
components. Variability was expressed through different analog components on printed
circuit boards.

As more powerful control processors became cost-effective, a market developed which
competed on software-intensive features. This led to an increase in code size, reaching
600KLOC by the end of the 1990s. Rapid prototyping was used, with code generation to
compose “reusable components” [Jonkers 1993]. The method provided both the means to
specify reusable components and the technology to bind them in different configurations.
However, it did not provide a modular architecture nor a product line, with clearly defined
commonality and variability. Individual development teams were located in different
regions to serve their local markets, with no reuse between teams.

2.5.2 Increasing software content and global development
(vertically integrated)

2.5.2.1 Forces and actors

By the mid 1990s the amount of software in consumer products roughly followed Moore’s
law [Ommering 2004] and, to contain development costs, a significantly higher level of
reuse was required. Independent product development in each region became uneconomical
and the, now shared, software architecture was expected to explicitly provision for the
variability between regions. This need to support variability was reinforced in the early
2000s with the industry transitioning from CRT to Plasma, LCoS and LCD displays. To
achieve optimal picture performance these displays require an optimized video processing
chain under software control (F1).

In addition, features from high-end products started to migrate down to the mid-range
TVs, requiring sets in different ranges to have sufficient architectural commonality to
facilitate the effective sharing of implementations. Finally, with a booming dot-com industry,
hiring sufficient developers in one location become impossible and support for multi-site
development became a priority. These trends led to more attention being paid to the
development process and software architectures [Rooijmans 1996].

2.5.2.2 Consequences for software architectures

While the previous TVs notionally had a layered architecture, the lack of dependency
manangement and, in particular, explicit required interfaces, had caused architectural
degradation [Linden 2000]. This led to two changes:
The changing industry structure of software development for consumer electronics

- Software product line engineering approaches were deployed [Pohl 2005A] using a new architecture, based on reusable, configurable sub-systems. Each sub-system could be allocated to a different location and could be developed with limited communication. The architecture supported variability through both variation and composition, allowing the large range of product line variations to be supported by exchanging components or sub-systems.
- A component model, Koala, [Ommering 2004] was introduced. This supported variation and composition with minimal run-time overhead (F1), allowing software component technology to be used down to the mid-range TV segment.

The general form of the TV architecture is shown in Figure 7. In this, “middleware” is the term used to denote the layer that provides services such as TV channel installation and selection and the electronic program guide.

![Figure 7 Reference software architecture of a TV developed in a single company](image)

Crucial to the success of this architecture was the definition of an Audio/Video API (A/V API), which was sufficiently abstracted from the hardware to be stable, despite the variation in the hardware. Interface stability was achieved by including all the logic for coordinating the hardware components in the platform. The result was a platform with a high degree of modularity with few constraints on its usage (F1).

### 2.5.3 Integration of connectivity features (transition to suppliers and integrators)

#### 2.5.3.1 Forces and actors

Around 2000 digital video broadcasting and connectivity features were maturing. Increasingly users started to watch TV via a PC. As a response to this threat, also known as a “battle for the living room” [Waters 2011], the TV manufactures incorporated new functionality in the TV, like USB connections, Wi-Fi and Internet on TV. Finally, the penetration of terrestrial digital video broadcasting had grown to the point that there was also consumer demand for integrated digital receivers in TVs. The competitive advantage moved from the quality and cost-effectiveness of the video processing to new interactive
applications and conditional access for pay TV, which provided additional revenues. This functionality resided in the middleware layer, for which implementations had already matured in set-top boxes. As certain digital broadcasting features differ by region, in-house development of all variants started to become prohibitively expensive. Instead, the required functionality was delivered with lower development costs by integrating software from external suppliers (F4, F5).

TVs were developed by the same small group of vertically integrated firms as in the previous stage, but specialized independent software vendors provided specific parts of the software. The TV manufacturers had to develop new skills, such as software vendor selection, while the engineers were faced with the technical challenges of integrating of 3rd party components (F6) [Trew 2006].

### 2.5.3.2 Consequences for software architectures

The components providing the new features were of a relatively small size and the supplied software was integrated into the in-house developed system by using alternative components from different suppliers (F5) [Hartmann 2009]. During the evaluation process of the software, the variability offered by the suppliers was taken into account, as well the fit of the purchased software into the existing architecture (F7) [Pohl 2005A]. Few of these features had standard APIs and the decision was made to wrap the supplied software to form Koala components [Ommering 2004], so that the product integration and configuration was unchanged (F6). To avoid vendor lock-in, these wrappers might have to perform significant transformations to the native API, allowing alternative suppliers to be substituted easily. No changes to the layered, modular architecture, presented in Section 2.5.1.2, were required. Given the relatively small size of the wrapper software in comparison to the overall software stack this was an economically feasible approach.

### 2.5.4 Specialized manufacturers of system on chips (transition to supply chain)

#### 2.5.4.1 Forces and actors

The advent of the digital video broadcast reception in TVs favored a move to processing video, from both digital and analog broadcasts, from dedicated hardware to software executing on specialized digital signal processors. This change allowed the transition from a set of dedicated video processing ICs to a single system on chip (SoC) with peripheral ICs. The SoC contains several hardware building blocks on a common bus. These blocks are either general-purpose computing elements, such as DSPs or the control processor, or dedicated hardware functionality, such as MPEG decoding. Peripheral ICs are dedicated ICs such as tuners, analog video decoders and ICs for audio and video output.

These large SoCs are able to cater for a wide range of functionality and with huge variability between the several products and differences in price (F5). These variations, for instance, related to the display of video (high definition, screen size, picture in picture), audio capabilities (Dolby surround sound, equalizers), user interface, and connections to other devices (USB, Ethernet, HDMI). The underlying differences between TV standards...
The changing industry structure of software development for consumer electronics

between geographical regions, different set makers and market segments are now largely accommodated through variation points in the software. The choice of peripheral ICs vary depending on the exact TV set requirements.

In the new architecture, software now has time-critical requirements to be able to guarantee the smooth rendering of the processed video and resource management becomes more critical, requiring detailed insight into the intended use of SoCs (F1).

While the new hardware architecture allowed the total manufacturing costs to be substantially lower, the development costs for hardware and software became significantly higher. In order to amortize such investments most consumer electronics firms separated, their semiconductor divisions into independent companies that could serve multiple customers (F4) [Nagran 2007]. A few examples: Siemens spun-off Infineon, Mitsubishi and Hitachi together spun-off Renesas and Philips spun-off NXP Semiconductors. Note that this not only impacted DTVs but all kinds of consumer electronics products. As a result, three types of actors now constitute the supply chain of DTVs:

1. Set makers that created the middleware, application layer and offered the end product to the consumer market
2. Manufacturers of SoCs, i.e. the platform suppliers, who also developed the low-level software and are typically responsible for integrating it on the hardware.
3. Specialized suppliers of software components, as mentioned in the previous stage.

2.5.4.2 Consequences for software architectures

In the supply chain that has emerged, the products have a heterogeneous architecture consisting of sub-architectures from different parties. The typical split in responsibility between the platform provider and the set maker is shown in Figure 8. The layers above the API are the responsibility of the set maker and below the API, the responsibility lies with the platform provider.

![Software Architecture Diagram]

Figure 8 Reference software architecture of a TV developed in a supply chain

There are no widely agreed standards for the interfaces between the platform and the higher layers, so some form of glue layer is normally required (F6). In the remainder of this section
we will describe two particular challenges, namely the definition of an audio/video API and managing the variability across layers owned by different parties.

Definition of the audio/video API. The API that was defined earlier, see Section 2.5.2.2, allowed the platform provider to have full control of the system behavior (F1). This was achieved by forcing the customers to enumerate the audio/video stream graph configurations that they wished to use. The platform provider could determine how these stream graphs could be implemented most efficiently and could fully validate them, irrespective of the behavior of the higher-level software.

However, when selling to a broader market, this exposes the platform to much more variability, since different manufacturers have highly diverse requirements on how a TV behaves. For instance, a particular manufacturer will have specific requirements on how a tuner searches for a station, and not simply on the post-conditions of operations. The combination of these issues has resulted in the level of the platform API being lowered. This has two advantages. Firstly, most of the sources of customer-specific variability can be excluded, so that the platform software can remain stable and the customer-specific variation points are under the control of a single party (F2). Secondly, it is easier to bridge the gap between components at different levels than to bridge between provided and required interfaces that are at the same level but which use different concepts (F6). As an example, rather than providing search functionality for the TV’s tuner, only the capabilities of setting a specific frequency and detecting the presence of signals are offered, with the customer-specific searching functionality being implemented at a higher level in terms of these primitive operations.

The disadvantage of lowering the API is that the client can inadvertently configure the platform such that the performance requirements are not satisfied because of resource constraints. This requires greater cooperation between suppliers and customers (F6, F7). The issues encountered during an attempt to standardize the platform API are discussed further in Section 2.5.6.

Managing the variability across layers. The split in responsibility between multiple parties now has consequences for managing the variability:

- Because elements of features are implemented in each of the layers, a variation point in the application layer must be mapped down to platform and, in some cases, to the hardware configuration [Pohl 2005A]. Therefore, not only must the semantics of the diversity mechanisms be compatible across the layers, but also their implementation.
- It is usually necessary to apply staged configuration [Czarnecki2005]. In the DTV case the platform supplier makes the early configuration steps, including setting variation points that were imported from suppliers earlier in the supply chain, leaving other configuration decisions to the set maker.
- To protect its intellectual property, the platform supplier may be very reluctant to release source code. Consequently, the implementation of variation points through simple conditional compilation is rarely applicable. Since the Koala component model, which was used in the fully integrated company, requires that the final product integrator has access to all the source code this proved less appropriate in a software
supply chain. Therefore greater use is made of run time binding, using a component model similar to Microsoft COM, but adapted for resource-constrained products. Furthermore, the use of Koala also dictated that customers used this technology and, since it was not a widespread technology, this was unacceptable.

2.5.5 Integrators and suppliers of middleware (transition to supply chain with more parties involved)

2.5.5.1 Forces and actors

Over time, the functionality of the middleware layer continues to increase, e.g. with the ability to browse content, e.g. photographs, stored on a networked PC, and the management of video recordings on integrated hard disks. A group of specialized development and integration companies emerged, e.g. LogiWays, and Shanghai-BMC, offering a complete middleware layer, rather than just specific features. Therefore TV set makers no longer needed to invest effort in maintaining the middleware (F4) and, as a consequence, the supply chain described in Section 2.5.4.1 now also includes the middleware vendors.

2.5.5.2 Consequences for software architectures

In the newly created supply chain there are various parties that can be responsible for the software integration. The development of the different layers of the software stack can be allocated to suppliers in different ways (see Figure 8). Some set makers develop both the middleware and application layer, others only the application layer. Some SoC manufacturers develop both the platform and middleware layers, other only the platform layer. In principle, the integration of middleware and application layers could be performed by any of these players, including the middleware vendors. An advantage for a supplier, e.g. of the SoC or middleware, to do the integration is that they maintain customer intimacy, allowing this supplier to discuss future roadmaps with the set maker [Messerschmitt 2004] and to be able to respond to market demands promptly.

In Figure 9, the glue layer connects the Middleware to the platform API. There are economic advantages for the middleware supplier if it owns responsibility for the glue layer because parts of their revenues are obtained for creating the glue. Like the middleware itself, the glue code is proprietary software, whose source code is not normally provided to the other parties.
A platform provider will tend to prefer to work with a small number of middleware suppliers to reduce the number of architectural mismatches that must be identified and resolved (F5, F6, F7). Therefore the platform provider will prefer to promote existing integrators, but will not wish sales to new customers to be dependent on the availability of that integrator’s engineering effort.

While the platform provider will have a preferred selection of peripheral ICs, the set maker sometimes selects alternative ICs and the platform provider has to integrate these ICs and their corresponding software drivers into the platform. The peripheral ICs can be partitioned into Front End (FE), such as tuners or analog video decoders, and Back End (BE), such as audio and video output ICs. Therefore, the platform software should be modularized in the same way, as shown in Figure 10. However, to reduce the dependence on the middleware integrator, it is also necessary for the glue to be modularized, including the coordination logic, and for the source code of the FE and BE glue code to be available to the other parties.

At this stage the longer integration time was noted [Raaijmakers 2009], causing delay in market introduction (F6). Had a standard platform API been established, this would have removed the need for glue code and enabled an open ecosystem. However, as will be discussed in Section 2.5.6, to date no standard has gained traction in the TV market.
2.5.6 Attempts at creating industry-wide standards

As the industry structure began to shift, the SoC suppliers and set makers would have the greatest opportunity to increase their market through the standardization of a platform API. This would enable a market for new middleware vendors, whose products could be readily hosted on any compliant platform, allowing set makers to select components from different suppliers and integrate them in a straightforward manner (counteracting F6). A group of semiconductor companies and consumer electronics companies started a forum to define an industry standard for consumer electronics devices, called the Universal Home API, or UHAPI [Uhapi 2010]. The objective was to define a hardware-independent industry standard API for audio/video streaming platforms. This API is orthogonal to most the kernel of most operating systems being used, such as VxWorks, pSOS and eCos, as illustrated in Figure 8.

When developing such an API, it is crucial to agree on its level, and therefore the split in responsibilities between the middleware and the platform. Previous architectures had exported separate drivers for each of the audio and video processing functions. This had exposed the difficulties of configuring the platform for the different combinations of active features required, which was compounded by the reluctance of the customer to reveal their planned usage of the platform. The semiconductor suppliers however, wished to raise the level of the API to ensure that it did not dictate a specific hardware architecture and that it provided guaranteed system performance (preserving F1), as described in Section 2.5.4.2. Consequently the UHAPI was defined [Uhapi 2010]. While optimal from the platform provider’s perspective, this approach had a number of disadvantages:

- Some set makers were reluctant to reveal their exact requirements; for fear that they would be disclosed to competitors (F8).
- For some middleware vendors, the development of the glue code was their primary source of revenue, and this work would no longer be required.
- The model of selecting an enumerated stream graph was a poor match with the streaming frameworks in the general-purpose operating systems, such as Linux and Windows Embedded CE, which were starting to be used in consumer products.

Although this API was ultimately standardized as ISO/IEC 23004, it failed to gain traction in the market.

Other industry-wide API standards in this application domain face other challenges, due to rapid changes in technology. For instance, the currently agreed version of “OpenMax” standard for A/V stream processing components [OpenMAX 2010], does not support multi-processor architectures or quality of service management. The functionality in consumer electronics products still isn’t stable and therefore no industry standard can be expected in the near future (F2). Another, more speculative, reason why attempts for standardizations of APIs have failed is that the several actors are afraid that one of the actors, e.g. a middleware supplier, might become the dominant firm and take the majority of the revenues (F8).

2.5.7 Summary of the case study of digital televisions

Initially, the amount of software for TVs was small. When software replaced functionality previously implemented though hardware, the amount of software grew quickly and
Chapter 2

layered architecture was introduced, as well as a formal approach to software product lines. Soon after this step, software was integrated from specialized suppliers to provide functionality that was already available in PCs and set-top boxes (F4, F5), thereby requiring small architectural changes.

When the IC manufactures where separated from the TV set makers, a supply chain was created and the control over the architecture was shared between the participants. Attempts at standardization of the audio/video API failed because more control over the system resources was demanded by the next participants in the chain (F1, F2). As a result, it became increasingly difficult for platform providers to guarantee failure-free system operations (F3).

Independent middleware vendors emerged, whose software was integrated into the TV. This again caused longer integration times (F6) due to the increased interaction between the participants (F7) and the need to combine heterogeneous architectures.

The use of supply chains is likely to continue for the foreseeable future. It will remain difficult to establish stable modular interfaces, given the increasing demand for additional processing power to support increasing requirements for video processing, e.g. 3D and QuadHD video and interactive applications, e.g. YouTube on TV.

2.6 Transition in the Mobile Phone Industry

In this section we present the rationales and consequences of the transitions in software development in the mobile phone industry, using the historical perspective presented in Section 2.3. In contrast to the case study of digital televisions we will look at all the major players and focus on the strategy of the players in dealing with the consequences. We will conclude the section with a summary.

2.6.1 First mobile phones (vertically integrated)

The early-generation mobile phones supported mainly voice processing. These phones consisted of a discrete single CISC-based microcontroller (MCU) core controlling a number of analog circuits. The move from analog to digital technology drove the need for a digital signal processor (DSP) core to be added to the architecture [Freescale 2008]. The first manufacturers of mobile phones, e.g. Nokia, Ericsson, Motorola and Siemens, were large, vertically integrated companies that owned the entire development, manufacturing and sales process [Bruhl 2005]. During the 1990s end-users and mobile network operators were willing to pay for enhanced functionality and usability. To meet these demands, products were made with customized components since standardized components could not meet the high performance requirements (F1, F3) [Anderson 2006]. Hence a vertically integrated firm was the most suited industry structure because only large companies could make these investments.

These companies also developed their own software, e.g. hardware drivers, real time operating systems and user interfaces. Different products were made to serve different standards and market segments. When the amount software grew, product line engineering
was applied to reduce development effort for creating variants of the products. An example of this is Nokia [PHOF2010].

2.6.2 Introduction of extra-domain functionality (transition to suppliers and integrators)

Over time, additional functionality, arising from other consumer products, was added to mobile phones, now called feature phones. To deliver this new functionality, such as cameras, and FM radios, to the markets quickly, components were used from specialized suppliers (F4). To name a few of these components: real time operating systems, Bluetooth, camera sensor and video playback, color displays, R.F. transceivers, power amplifiers [Anderson2006]. The specialized suppliers mostly originate from other consumer electronics manufacturers, e.g. Philips, Samsung, Hitachi, and software vendors, e.g. related to video and audio encoding.

The vertically integrated companies now used specialized suppliers and integrated these components, both hardware and software, into their existing architecture. The baseband processor and real time operating system that were already used were still capable of supporting this additional functionality [Infineon 2010] and the changes to the software architecture where limited (F3).

2.6.3 Smartphones (transition to supply chains/suppliers and integrators)

With the introduction of data and multimedia services, a mobile phone must interface with E-mail systems, the Internet and be able to handle a wide range of non-voice content, such as music, videos and games. As a result, several new technologies must be incorporated, such as GPS, e-mail clients, web browsers etc. [Kornby 2005]. This development was encouraged by the Mobile Network Operators, who offered phones at low price with a subscription, because the new functionality required large amounts of data transfer which was a major source of revenue [Rassweiler 2009, Strat 2008]. This functionality was originally used in personal digital assistants (PDAs) and personal computers and introduced the need for dedicated application processors, operating systems and software applications [Freescale 2008]. As an example, the first smart phone developed by Nokia was made as a combination of a PDA by Hewlett Packard and a phone by Nokia (F5) [NokiaHP 1996].

The development investments became significantly higher, both for hardware and software, and the vertically integrated companies needed to decide where they could excel and choose their own focus in this value chain (F4). Separate platform suppliers were created through spin-offs of the vertically integrated companies and specialized IC vendors [Kornby 2005]. For instance, Qualcomm focused on being a hardware and software platform supplier, Motorola spun off Freescale that focuses on delivering a hardware platform with software drivers, leaving integration to third parties. Ericsson Mobile Platform focuses on using hardware and software components from other suppliers (see Section 2.6.3.1) and so does MediaTek [Wang 2009].
As a result, the industry transitioned and the suppliers, who defined part of the product architecture, determined a significant part of the functionality (see Figure 11, [Suaranto2006]).

Software Platforms and Operating Systems
A mobile software platform typically consists of an operating system, middleware and base applications [Limo2010, Maemo 2011, Halasz 2011]. The PC industry showed that the OS vendor could obtain most of the revenue, but in current days not only through licenses but also through the sale of applications and advertisements, such as with Android [Pang 2011]. The choice of consumers has very much shifted from a choosing a specific brand or network operator to choosing a specific software platform since the platform determines the applications that are available for that phone, see Section 2.6.4. For the development of a these platform both new entrants and existing players tried to gain a dominant position (F8). Examples are Google Android, Palm, and Windows. Also market leaders such as Nokia supported the development of open source mobile platforms, e.g. Symbian and MeeGo, to reduce their development effort (F4) [Bosch 2009].

Comparable to the case of digital televisions for middleware, a mobile platform supplier or handset maker has to integrate different software platforms to serve different customers (F5). Some examples (March2011): Samsung supports: Bada, Android and Windows [Samsung 2011], Nokia supports Windows, Symbian, Maemo, MeeGo [Nokia2011]. This leads to additional challenges for integration as shown in the following case study (F6).

2.6.3.1 Challenges for integration, case study from Ericsson Mobile Platform
Ericsson Mobile Platform (EMP) was transformed from an in-house development group for Ericsson into a supplier for different handset manufacturers. EMP develops a complete...
The changing industry structure of software development for consumer electronics

platform of ICs and software components, licensed from third parties, by integrating these components and customizes them for each customer [Kornby 2005]. A platform API is offered to the customers for developing applications on top of that, see Figure 12 ([Kornby 2005] © 2005 Ericsson Review, used with permission).

Figure 12 Software architecture EMP

EMP takes the requirements of network operators into account, e.g. that the phones they sell contain specific applications to increase their revenues (F7). The variability that EMP offers to its customers is therefore related to the supplied HW and SW components (F5) and the customization that is done on the reference platforms. Three different product lines are offered, for Entry Phones, Feature Phones and Smart Phones, and different software platforms may be integrated to cover a wide range of customers. To obtain the best performance while minimizing the resources required, system use cases are obtained from the handset maker and network operators and these are analyzed, e.g. imaging, multi-player games etc. (F1) [Kovacevic 2007]. Also simultaneous usage scenarios are taken into account, such as simultaneous streaming and voice call, since these scenarios often require use of the same hardware resources, such as the DSP and the CPU, for different features. The NoTA Platform [Kronlof 2006] follows a similar approach.

EMP face the same challenges as did the platform supplier NXP in Section 2.5.4.2 and 2.5.5.2: How to incorporate components from different suppliers (F5) to serve a wide customer range and maintain a platform API that enables third parties to develop applications, while assuring optimal system utilization (F1). The longer time required for integration and testing became visible in this industry (F6) [Underseth 2007].

2.6.4 Internet connectivity (transition to ecosystems for third party applications)

Third Generation (3G) communications standards enabled faster connections to the Internet. By giving third parties the possibility to develop applications that the end-user could directly purchase and download, an ecosystem was cultivated. In this way, the handset makers can
Handset manufactures such as RIM and Apple use this business model with their proprietary platforms, as did the open-source platforms, such as Android and Symbian. The rapid commercial success of the Apple iPhone can be explained because it made use of two existing supply chains, namely that of the handset, described in the previous section, and the internet supply chain through their existing iTunes store [Wehlage 2008, West 2010]. The industry structure for the majority of mobile phone industry is now a combination of two different structures. The structure for third party applications and the industry structure for the mobile devices. Here a decoupling point exists between that part of the functionality that has hard performance requirements (F1), e.g. speech processing and music playback, and the applications with only soft performance requirements, which do not require the tight balance between system performance and resource utilization.

2.6.5 Attempts at creating industry-wide standards

For wireless telecommunication standards, e.g. Open Mobile Alliance [OMA 2010], there is a broad adoption since that enables a mobile device to function in a network, but does not dictate software architectures. For the standardization of the hardware and software there are a number of different strategies of the different players. For instance the MIPI alliance aims to establish standards for interfaces [MIPI] and provides a standard interface to the OS. However these standards do not cover software or system architecture. Another standardization body, Khronos, includes the OpenMAX standard [OpenMAX 2010]. This standard, which addresses the API for multimedia streaming, gained only limited industry adoption, due to the rapid change of technology, e.g. for quality of service and power management.

Several attempts have been made to define a common software platform in an attempt to replicate the Wintel model and create an ecosystem, by separating the supply of key components from hardware sales [West 2010]. The number of competing mobile platforms that have entered the market is huge, e.g. Palm OS, Google Android, LiMo, Symbian, Windows Mobile, Maemo, MeeGo etc., resulting in increasing fragmentation of the market. Often these platforms are supported by major contributors to create a dominant position in the market (F8), e.g. Symbian by Nokia, MeeGo by Nokia and Intel, Android by Google. Companies such as Apple and RIM challenged the need for standardization by offering more integrated and closed systems for the iPhone and BlackBerry, including OS, hardware, built-applications, and online services [West 2010].

Most of these platforms offer a stable interface for applications, since that allows the applications to be used by different phones (F5). If the standardization of the interfaces is strictly controlled, this constrains the changes that a handset maker might wish to introduce to create unique selling points (F2). However, these platforms usually do not dictate a standardized interface towards the hardware since an embedded system developer requires a high degree of flexibility over the software to make use for optimal utilization of the hardware (F1) [Halasz 2011]. Over time some of these platforms gained a significant market share but where abandoned later because new functionality entered the market, e.g. multi-touch screens, which could not easily be incorporated into existing platforms (F2).
The changing industry structure of software development for consumer electronics

There are several reasons for the lack of standardization and this is why we believe that, except for the end-user applications, the forces in the mobile phone industry do not currently favor an ecosystem centric approach:

- The fear that a dominant firm takes most of the revenue (F8).
- The convergence of additional functionality from other consumer products can be expected (F4), e.g. from gaming consoles [Digitaltrends 2010] and therefore the speed of innovation will hinder the definition of stable interfaces (F2).
- A high degree of modularity cannot be achieved because this would cause inefficiency and optimal use of resources remains an important design objective (F1).

### 2.6.6 Optimal industry structure for different market segments

Other authors have discussed whether a more vertically integrated or a more horizontal structure would be most suitable for the mobile phone industry [Anderson2006, Constantinou2010]. Using our model of industry structures and the forces, these strategic choices can easily be verified:

- Anderson identified that for the entry-level devices, which contain little variability, vertically integrated firms offers better possibilities to obtain the lowest silicon costs because increased resource efficiency (F1).
- Anderson and Constantinou argued that a more integrated approach might be more suited for high-end and new-to-market products [Anderson2006, Constantinou2010]. For this product range, bringing novel functionality to the market is more important than the higher development costs since the sales price is much higher than the manufacturing costs. As a small case study, consider Apple’s mobile products. Initially, Apple and Motorola co-developed the ROKR, a mobile phone with MP3 playback and iTunes connectivity [Vogelstein 2008]. The resulting device lacked sufficient innovative features to attract many customers (F2). Then Apple developed the iPhone, now keeping control of the hardware architecture, although based on a combination of existing hardware components from various suppliers. Moreover, Apple providing the software, derived from the Macintosh OS X. This degree of control enabled Apple to create a very innovative product and immediately attracted many customers. The development costs could be amortized because of the large market share that Apple has obtained and a huge margin on the manufacturing costs [Rassweiler 2009]. Furthermore additional revenues are generated by selling additional applications developed by third parties. Note that for the handset, Apple’s range contains very little variability, with the amount of internal memory being the principal variation point (F5). Although Apple has become an Integrator that uses specialized suppliers (F4), they continue to rely heavily on the closed ecosystem approach for downloadable applications.

### 2.6.7 Summary of the case study of mobile phones

Initially the size of the software was limited, so it could be developed by vertically integrated companies, and enabled them to use the system resources most efficiently (F1). With the
introduction of feature phones, the players used components from other consumer electronics firms (F4, F5) but no major architectural changes were required. A major shift occurred when functionality from PDAs was incorporated, thereby requiring general purpose operating systems, which required large development investments. Several software platforms entered the market with the aim to create an ecosystem for the mobile phone handset. However none of these attempts have gained industry wide adoption and no high degree of modularization was attained. This led to the challenge to use different suppliers while assuring optimal resource utilization and, furthermore, an increase in the time needed for software integration was noted (F3, F6, F7).

The supply chain is the dominant industry structure at this moment in time, although some players use a more vertically integrated approach (F1, F2). The need to ensure optimal resource utilization (F1), the ability to serve multiple customers and the lack of industry-wide standards prevents the development of handsets from becoming more ecosystem-centric. The supply chain industry structure is likely to remain the dominant structure since no industry wide standardization may be expected due to the speed of innovation (F2) and the fear that a dominant player may take most of the revenues (F8). For the development of downloadable applications, closed ecosystems have been created because this functionality has no hard performance requirements nor does optimal resource utilization need to be guaranteed.

There is a great deal of similarity with the transition of Digital Televisions, but the transition of mobile phones occurred at a much faster pace. In the mobile phone industry the Mobile Network Operators play an important role in the definition of the functionality and encouraged large investments. A similar situation is present in the development of set-top boxes where the service providers determine a significant part of the functionality. For digital televisions, however, the functionality is mainly decided by the set makers. Another difference is that, for digital televisions, no ecosystems have been created for downloadable applications, since this type of applications are not (yet) common for Digital Televisions. However, initiatives in this field have been started, e.g. by Google and Yahoo.

2.7 Related Work

Although there is some related work on ecosystems and supply chains in the computer industry and ICT, as well as documented case studies, little work exists in the domain of consumer electronics with resource-constrained devices.

In the general management literature, software platforms are studied from a business strategy perspective. The work on Value Chains [Porter 1980, Porter 1980], Value Networks [Peppard2006], Value Chain Modularity [Sturgeon 2003] and platform leadership [Gawer 2002], focus on the strategy for an individual firm to become successful in an existing industry structure. This literature does not present a model of industry structures nor the forces that govern these transitions.

One co-author, Bosch, provides a taxonomy for different forms of (closed) ecosystems [Bosch 2009] and described the successful examples in the domain of desktop and online software. Jansen et al. [Jansen 2009] discuss different types of software ecosystems and discuss the difference with a supply network, but do not address embedded systems, or
industry transitions. Popp and Meyer also discuss closed ecosystems in the ICT domain and their work focuses on business models. Messerschmitt and Szyperski [Messerschmitt 2004] focus on open ecosystems. Although they describe the consequences of transitions, also these authors do not address the embedded systems domain. Also, they neither classify different types of ecosystems, nor do they discuss the differences between ecosystems and software supply chains.

In the literature of digital business ecosystems [Nachira 2007, Dourmas 2009, Janjua 2009] the emphasis is on creating the ICT technology that enable an easier communication between companies which is needed for an ecosystem to prosper. In this domain the challenges relate to data interchange and interoperability, different governmental regulation, different business processes and so forth. The literature in this field does not address the de-verticalization of industries nor models that describe industry structures.

Bruhl and Stiegllitz [Bruhl 2005] characterize industries that undergo a de-verticalization and identify the role that an individual company can play and also use examples from the mobile phone industry. Christensen et al. [Christensen 2002] discussed the drivers behind de-verticalization and the role of modularization. Raskin and Mellquist [Raskin 2005] recognize stages of de-verticalization and the enablers for this de-verticalization. None of these authors identified the transition steps nor identified different industry structures.

Krafft [Krafft 2003] identified different degrees of vertical specialization in the telecommunications industry, identifying a layered industry structure for equipment provision, network operators, up to content provision. His work looks at this industry as a whole and is not directed at the mobile devices. Suoranta [Suoranta 2006] envisioned a disruption of the mobile phone industry with opportunities for open innovation, based on a modular architecture at different levels of abstractions. Although the disruption has taken place, this has not led to the creation of a modular architecture, as we showed in Section 2.6. Anderson and Jonsson describe the transition of the mobile industry [Anderson2006] until 2005 and draw analogies with the PC industry and the role of modularity. They recognized that different companies and for different market segments a more vertically of horizontal industry structure is appropriate. Since their analysis included the period until 2005 it does not include the emergence of ecosystems for downloadable applications. Constantinou analyzed the mobile phone industry and recognized the move between vertical and horizontal structures [Constantinou2010]. None of these authors provides a model for different industry structures and describe the consequences for software architectures.

2.8 Conclusions and Future Research

The software development of consumer electronics industry has gone through a substantial transition in the last 15 years, passing through a number of structural types. In this paper we captured these structural types in a model of industry structures that we developed, based on case studies from digital televisions and mobile phones. While related art has claimed that a high degree of modularization and standardization is a driver and pre-requisite for de-verticalization, we showed that that the de-verticalization of the consumer electronics
industry started when functionality outside the traditional domain was introduced, despite a low degree of modularity and standardization.

We demonstrated that the challenges in developing software for consumer electronics lie in providing the right balance between delivering the required variability in a timely fashion, while retaining sufficient control over the systems architecture to obtain optimal resource utilization. These characteristics originate from the nature of embedded systems that are designed to perform specific tasks, in contrast with general-purpose computers that are designed to be flexible and meet a wide range of needs. Furthermore, efforts for standardization have failed because of the rapid changes in functionality and the fear that a dominant firm could emerge. As a consequence the transition of the software development for consumer electronics has not followed a similar path to that of the computer industry, as envisioned by other authors.

In this paper we described the forces driving towards a more ecosystem-centric approach, a vertically integrated approach as well as those forces that restrain changes towards a more ecosystem-centric approach. We showed how these forces drove the individual players in the industry of digital televisions and mobile phones to change their position. Furthermore we described the consequences for the software architectures during the transition steps. The model of industry structures we presented can serve the decision making process for an individual company to enter or change into a different type of industry structure. The forces and consequences we described provide insight into the benefits and expected problems when moving from one type to another. We showed that our model and forces provides a better decision support than related art does. Since the choices that individual companies make in choosing the industry structure has a large impact on their profitability, we believe that the current research is important for practitioners as well as provide a framework for researchers studying the software intensive industries.

Preliminary research showed that our model and the governing forces are also applicable to the wider domain of embedded systems, given the commonality in characteristics between these products. More speculatively, our model might also be applicable for software intensive system in general, and we provided some examples of those in Section 2.4. However, the forces might be different from the ones presented in this paper and therefore further research is needed.

Another important topic for future research is that an industry can adopt a hierarchical model where at different places in the architecture a different industry structures are used, thus leading to a more fine-grained model of that industry’s transitions. Furthermore, additional case studies are required to enhance the model into a full-fledged decision framework for companies transitioning into a different type of industry structure.

The research presented in this paper is one of the first of its kind. While earlier research has described de-verticalization of industries, little work exists that analyzed the consumer electronics or embedded systems industry. Furthermore, none of the earlier work neither provided a comprehensive overview of the forces governing the transitions nor identified different forms of industry structures for software development of embedded systems.
Acknowledgements

The Authors would like to thank Rob van Twist, Jack Goossen and Clara Otero for their contributions to the case study of digital televisions and the reviewers of the Journal for their comments, which helped us to improve the paper.