Chapter 3. Consumer Electronics Software Ecosystems

This chapter is an extended version of the article which is published as:

Abstract
Due to the increasing size of software and because consumer electronics products are more and more connected to the internet, many consumer electronics firms are adopting an ecosystem centric approach for supporting 3rd party applications. In a mature market where successful platforms are available, a firm will mostly choose an existing platform. In an emerging market, a consumer electronics firm may need to create a new ecosystem or adopt a newly developed platform, both which has significant commercial and technical implications.

In this paper we identify three types of ecosystems that are used today: Vertically integrated hard-ware/software platforms, closed source software platforms and open source software platforms. We introduce a first step towards a multi-criteria decision support method. This method determines which of the three types of ecosystem is most likely to become successful for a specific product category. The method is based on the scope and technology design of the platform and takes a software engineering perspective.

We use this method to analyze a wide range of consumer electronics products. We show that the vertically integrated platform type is most suitable for product with a high degree of innovation and that open source software platforms are more suitable when a large amount of variants are needed to serve the market. The closed source software platform type is less suitable for consumer electronics, especially for products which require an optimal use of system resources and have a large degree of innovation.

Consumer electronics manufacturers can use this method to select the ecosystems type that they want to adopt or create. Firms that create software platforms can use this method to select the product types they want to target.

3.1 Introduction

Consumer electronics products have radically changed over the past two decades. Initially the functionality of these products, including products such as CD and DVD players, televisions and radios, was mainly implemented through hardware. Due to the reduced costs
for integrated circuits (ICs) and increased capabilities of these ICs more and more of the functionality is now implemented through software [Hartmann 2012], furthermore many products are now connected to the internet and support the downloading of applications in addition to the applications that are embedded in the devices.

Initially the products were entirely developed by individual firms, but nowadays 3rd parties are used for the development of software components. Specifically for downloadable applications, it is not feasible for an individual company to develop these applications themselves. As a result, many firms are adopting an ecosystem centric approach in which 3rd parties are supported to develop applications for their products.

In a mature market where there are existing ecosystems, a consumer electronics firm will usually choose an existing platform and this choice is largely determined by the commercial aspects, e.g. the amount of 3rd party applications and the license fees, and functional requirements, e.g. whether the platform supports the specific features that the a firm wants to offer. In an emerging market, i.e. when there are no existing or no dominant ecosystems, a consumer electronics firm may decide to create a new ecosystem and software vendors may decide to create software platforms to serve a this market. As an example consider the smart phone market. Nowadays that there are a few dominant ecosystems, and manufacturers of devices usually choose an existing platform, which is typically Android, Windows or Tizen. In the early years new ecosystems where created by consumer electronics firms such as Nokia, BlackBerry and Apple, and software platform vendors entered the market, e.g. Microsoft and Google. Some of these ecosystems became successful while other disappeared after a few years.

The adaption of ecosystem has significant implications, both commercially and technically. For instance, if a company decides to open its own platform for 3rd parties, it has to attract sufficient parties and it has to maintain its own ecosystem. When a company decides to adopt an existing software platform, changes are required to the architecture and a significant part of the revenues may go the platform owner. Furthermore it has to be confident that the adopted ecosystem will become successful. The success of an ecosystem relies on many aspects which includes the scope and design of the platform, how the platform owner co-operates with the complementors, how each of the participants is able to obtain revenues, how the sales channels are organized and how the platform is able to deal with changing requirements [Popp 2010, Gawer 2008, Axelsson 2014].

In this paper we classify the type of ecosystems that are used for consumer electronics and identify the strengths and challenges of each type, thus serving as a reference for emerging markets. Furthermore we analyze, giving the requirements from the market, which of the identified types is likely to become successful for a specific type of product from a software engineering perspective.

This paper answers the following research questions:

- What types of ecosystems are used in consumer electronics products to support 3rd party applications?
- Which of the ecosystems types is most suitable for a specific product category from a software engineering perspective?

To answer the first question a variety of consumer electronics domains are studied, resulting in a taxonomy. To answer the second research question, we introduce a multi-
 criteria decision support method [Gella 2006]. The decisions support method analyzes the scope and technology design for a software platform [Gawyer 2002] w.r.t. it’s abilities to meet the functional and non-functional requirements.

Our method finds its origin in methods for selecting software components and software vendors [Nazir 2014, Alpar 1998] and similarly uses multiple criteria which are based on an expert opinion and are using ordinal scales [Cooper 2011]. The criteria to evaluate types of ecosystems are general market requirements [Bolwijn 1990, Sheat 2010], requirements from the nature of consumer electronic devices [Henzinger 2006, Trew 2011], and requirements from the use of ecosystems [Messerschmitt 2004, Bosch 2010, Underseth 2007]. For each type of ecosystem we identify the strengths and challenges from a software engineering perspective. The decision support method is based on comparing the requirements for a product type with the strengths and challenges of the ecosystem types.

Since our decision support method does not cover commercial and organizational aspects we do not claim that our method is complete and this method is therefore a first step towards a full-fledged decision support method.

This paper is structured as follows. In Section 3.2 background is provided on consumer electronics. In Section 3.3 a classification of ecosystems is given. Section 3.4 describes the first steps towards a multi-criteria decision support method. Section 3.5 describes the case studies. This paper is concluded with a comparison with related art in Section 3.6, and our conclusions and a description of further research in Section 3.7.

3.2 Background on Consumer Electronics Products

3.2.1 Trends and challenges

Current consumer electronics products have characteristics of embedded devices as well as general purpose computing devices [Hartmann 2012]. As an embedded device they are aimed to perform dedicated functions, such as making telephone calls, playing games or movies, which have real-time performance requirements. There is a high pressure on cost price and therefore the integrated circuits (ICs) should operate as efficient as possible and using ICs with the smallest footprint achievable. Consumer electronics devices have constrained computing resources and should use as little energy as possible because they often have few options for heat dissipation. For mobile devices, which operate on batteries, this is even more important. As a consequence a consumer electronic product contains as little hardware and software components as possible.

These devices also have characteristics of general purpose computing devices because they are meant for a variety of tasks such as Internet browsing, reading and writing documents, accessing social media and playing games.

The combination of these two characteristics leads to numerous architectural and commercial challenges. As an example, consider Flash, an Internet browser plugin for watching videos. Apple has not allowed the use of Flash on their smart phones and tablets because it causes a significant shortened battery life [Jobs 2011], but has promoted the use of the H.264 video encoding standard for which their device is optimized.
Chapter 3

3.2.2 System architecture

In Figure 13 a high level system architecture of consumer electronics products is presented that support 3rd party applications, showing some of the actors for different product types.

The architecture consists of a hardware platform an OS-kernel and middleware that offers a framework for application developers [Halasz 2011, Trew 2011]. The hardware layer consists of a System on Chip (SoC) with peripheral ICs. A SoC contains several dedicated building blocks on a single IC such as audio decoding, memory and a configured CPU. A dedicated SoC, which is common for embedded systems, is better able to meet the performance requirements at lower power consumption than using a general purpose CPU and separate ICs, as in the PC industry. The design of the software and the SoC are tightly coupled, e.g. for controlling the power consumption of separate hardware building blocks.

3.3 A Classification of Ecosystem Types

This section describes the move towards ecosystem and provides a classification of ecosystems that are used today.

3.3.1 Growing software size: move towards ecosystems

In an earlier paper we analyzed the transition in the consumer electronics industry from 1990 until 2010 [Hartmann 2012]. Initially these products were developed by vertically integrated companies that developed the hardware, firmware, applications as well as created the end products. Examples of companies that create traditional consumer electronics products such as Televisions, DVD players and Radios are: Philips, Sony and Matshushita. Examples of companies that created the first generation of mobile phones are: Nokia, Siemens, Ericsson and Motorola.

Over time, an increasing proportion of the functionality was implemented through software and this introduced the need for dedicated application processors and operating systems. The development investments, both for hardware and software, became significantly higher and therefore the companies needed to focus their activities. Separate
hardware platform suppliers were created as spin-offs of the vertically integrated companies, such as Qualcomm [Qualcomm 2014], NXP and Freescale. When hardware platforms and other components became commodities it was possible for newcomers to enter the market without large investments. Examples of these newcomers are TCL, HTC and Apple in the market of mobile and smart phones and Dell, Apple and Microsoft in the market of digital televisions and set-top boxes.

Several software vendors entered the market and created a common software platform in an attempt to replicate the Wintel model [Grove 1996, West 2011]. Similarly as the case with Microsoft Windows, a firm controlling the spanning layer can earn most of the revenues since it controls the interface towards the hardware as well as towards the applications [Messerschmitt 2004, Baldwin 1997]. The number of competing mobile platforms that entered the market was significant, e.g. WebOS, Android, LiMo, Symbian, Windows Mobile, MeeGo etc. Similarly in the domain of digital televisions and set-top boxes, multiple platforms entered the market, such as Windows, Horizon TV and OpenTV. Until 2009 most of these attempts didn’t gain industry wide adoption and the industry was highly fragmented.

In the mobile domain, starting in 2010, many of the vertically integrated firms lost market share and Android, an open source software platform, gained a share rapidly. Furthermore the dominant player in the PC market, Microsoft, increased their efforts to gain market share. For other consumer electronics products, such as digital televisions, and digital cameras a similar transition occurs in which vertically integrated companies start adopting open source software platforms [Engadget 2014]. As a result there is a fierce battle between different ecosystems in the market of many consumer electronics products.

### 3.3.2 Classification of ecosystems with their complementors

This paper discusses ecosystems for consumer electronics that are aimed to support 3rd party applications. For a comparison of these ecosystems we use a classification that is based on two properties: (1) is the software proprietary or open source, and (2) are the hardware and device included. Figure 14 shows this classification with examples from the domain of smartphones.

<table>
<thead>
<tr>
<th>Software Platform only</th>
<th>Hardware and device included</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3) Android, Tizen, Firefox OS</td>
<td>(4) Not present in the market</td>
</tr>
</tbody>
</table>

**Proprietary closed Source** | **Open Source**

*Figure 14 Classification of ecosystem types with examples from smart phones*
This classification results in four possible ecosystems:

1: Vertically integrated proprietary hardware/software platform. This platform consists of the hardware, proprietary closed source software and includes the device. The complementors are the application (App) developers.
   Examples of platforms and their leaders for smart phones: Apple with the IPhone, Rim with Blackberry and Nokia with Asha.
   Examples for digital televisions and set-top boxes: LG with WebOS, Apple with AppleTV, Samsung with Smart TV.
   Examples for personal computers: Apple with OS X.

2: Proprietary, closed source software platform. This platform consists of a proprietary closed source software platform. The complementors are the suppliers of hardware platforms, system integrators, device manufacturers and application developers.
   Examples of such platforms for smart phones: Windows Phone and WebOS (2010-2012).
   Examples for digital televisions and set-top boxes: Horizon TV, Microsoft TV.
   Examples for personal computers: Microsoft Windows.

3: Open source software platform. This platform consists of an open source software platform. The open source software platform is based on the concept that multiple parties can contribute to share development effort and since the source is open, the device manufacturers can change, add or remove functionality. The complementors are the suppliers of hardware platforms, system integrators, device manufacturers and application developers.
   Examples for smart phones are Android, Tizen and Firefox OS.
   Examples for digital televisions and set-top boxes: Android, Meego, Ubuntu TV.
   Examples for personal computers: Chrome OS, Linux, FreeBSD.

4: Open source software and hardware platform. This platform would consist on an open source software platform including the hardware and device. The users of such a platform, or better product, would be able to change the code of the software platform and add their own functionality. We have excluded this type from this paper as this type, to our knowledge, is not available in the market. Note that in the past some handset makers of smart phones used an open source platform of which they acted as the platform leader, e.g. Nokia that used Symbian when it was open source and Motorola of Google that used Android. In this situation the handset makers are in-house complementors of the open source software platform and therefore the combination is not a separate type of ecosystem.

3.4 Towards a Multi-Criteria Decision Support Method

This section describes the first step towards a multi-criteria decision support method for selecting the most suitable ecosystem type for a certain product category. This section starts with an overview of the method, followed by the background. Then each step is explained in detail.
3.4.1 Overview of the method

This method is based on comparing the strength and challenges of each ecosystem type with the market requirements of a product category. As shown in Figure 15, the method consists of 3 steps:

Step 1: For a product category the most important requirements are determined.
Step 2: The strengths and challenges of the three types of ecosystem type are identified for these requirements.
Step 3: A decision support matrix is created that shows a total score per ecosystem type, based on the importance of the requirements and the strengths and challenges of the ecosystem types.

Figure 15 Multi-Criteria Decision Support Method

This method uses similar steps and techniques as those that are used for the selection of software suppliers and software components [Nazir 2014]. In these methods similarly a multi-criteria analysis [Gella 2006] is used to compare multiple criteria, or requirements, with the ability of the supplier, or software component, to fulfill them [Alpar 2010]. In these methods a decision matrix is used to rank the options so that the decision can be made more objective [Pugh 1996], and ordinal scales [Cooper 2011] are used as part of these methods since quantitative information is mostly not available [Goulão 2007]. The decision support matrix is used in the same way as these method to support the selection of alternatives based on a multi-criteria analysis, and the importance and evaluation of the criteria are based on an expert opinion.

Since ordinal scales are used for the requirements and for the total score, the values do not represent absolute numbers but are solely used as a scale to express the differences between the types of ecosystems.

Section 3.5 presents case studies for different product categories. Step 2 of the decision support method is independent from the product category and therefore the strengths and challenges of the three identified ecosystems are described in Section 3.4.4 and then used in the case studies in Section 3.5.
Chapter 3

3.4.2 Background of the method

3.4.2.1 Platform scope and technology design

Gawer and Cusumano designed a framework for a company to become a platform leader. A platform leader is driving the innovation in the industry, ensures the integrity of the overall system and creates opportunities for firms that create complementary products [Gawer 2002]. This framework includes (1) how a company defines the scope of the platform, i.e. what part of the product is developed in-house and what it leaves for complementary firms, (2) the technology design, e.g. to what degree the platform should be open to the complementors and how the interfaces are defined (3) how external relations are managed (4) how the internal organization is structured [Gawer 2008].

The scope and technology design highly determine the success of a platform. It should support the capabilities of the platform leader, allow for complementors to contribute and be able to respond to changing requirements. While all three ecosystem types provide a platform for 3rd party developers, they differ in the contributions the complementors can make to the final product and to what degree the complementors can alter in the platform.

For the vertically integrated platform type the scope is very wide and the complementors have no options to contribute or alter the platform. The closed source platform types provides the system integrators and device manufacturers the possibility to add distinguishing functionality but leaves little to no options to alter the platform, e.g. to make changes to the API (Application Programming Interface) and interfaces towards the hardware layer. The open source software platform provide the complementors the options to contribute to the platform and make changes to the platform thus allowing a large degree of freedom. The latter also leads to risks because the platform owner may lose control over the platform [Iansito 2004].

Since the scope and technology design constitute the largest differences between the three identified types of ecosystems, this is the basis of the decision support method.

3.4.2.2 Evaluation criteria

In order to evaluate the best suitable scope and technology design for a platform, it is needed to define the criteria that determine this. Here we draw an analogy between the selection of an ecosystem type and the selection of software components [Nazir 2014] and the selection of a software vendor [Alpar 2010].

A selection of software component is based on whether it fulfills the functional and architectural requirements as well it is able to address nonfunctional requirements. This set includes: Costs of the components, quality, whether it fits the architecture, is modifiable, maintainable, portable and so on [Kontio 2010]. A supplier selection is typically based on the abilities of a supplier to fulfill the functional and non-functional requirements [Alpar 2010] and includes the maturity of an organization, its customer base, and the way it cooperates with its customers. Since we use the requirements from the market we will further use the term requirements to describe the criteria.

The requirements that will be used in our decision support method, is a combination of those that are used for component and supplier selection. Similar as with a vendor selection,
the choice for an ecosystem is based on its abilities to fulfill the functional and non-functional requirements. Similar as with the selection of components, an ecosystem type has to be able to provide a platform which fulfills the architectural requirements.

For evaluating the non-functional requirements we use the competitive forces of Bolwijn and Kumpe [Bolwijn 1990] which determine the success of a consumer electronics firms: **Cost, Quality, Flexibility and Innovation.** In a later analysis Sheate [Sheat 2010] identified **Variability** as a requirement, which is strongly related to flexibility. In this context cost is interpreted as the software development cost since reproduction costs for software are negligible.

The key architectural requirements for embedded systems and consumer electronics products are: **Efficient use of system resources and hard real time requirements** [Henzinger 2006, Trew 2011].

Furthermore we use requirements that are specific for ecosystems, namely: **Stability of the interface** since complementors have to rely on a stable and, ideally, backwards compatible interface [Messerschmitt 2004, Jansen 2013], and **Effort for system integration**, since for the set-makers the integration of the hardware and software is a substantial and increasing proportion of the development effort [Underseth 2007].

This gives the following set of requirements:

**Development Costs, Quality, Variability, Speed of Innovation, System resources and hard real time requirements, Interface stability and Integration effort.**

In the following sections, these requirements are elaborated. Other criteria that determine the success of an ecosystem are not covered since this paper looks from a software engineering perspective.

### 3.4.3 Step 1: Identification of the important of each requirement for a product category

The requirements are evaluated based on the characteristics of the product and the market demands. For each requirement, it is determined whether it is “key”, meaning that it is an absolute necessity to support this, or of “high” importance.

- **Development cost:** This is based on the estimated size of the software, the effort that is required to maintain the platform and develop new functionality. This requirement will be evaluated as of high, or key, importance when the amount is large and continues enhancements are needed.

- **Quality:** This is based on the costs when a product or one of its functions fail which includes the usability of a product and the commercial impact of product failures.

- **Variability:** This is based on the amount of variants that are needed to satisfy the different market segments and customers. This typically includes different features, prize setting, screen size, the support of different regional standards etc..

- **Speed of Innovation:** This is based on the speed at which new products with new features are introduced and whether the new functionality is a reason for consumers to buy a (new) product. E.g. when a new product replaces an older version within one or two years, and customers replace their old product, the speed of innovation is determined as of high importance.
• **System resources and hard real time requirements**: This based on the pressure of the cost price of the ICs, the need to ensure a good system performance, and the requirements for ensuring a long battery lifetime for mobile devices. So for small products, such as wearables, this is a very important requirement and also for products for which there is a high pressure on the cost price. When real time aspects are important, e.g. to play music and games without audible and visual interruptions, this requirement is also evaluated as of high importance.

• **Interface stability**: This is based on whether the API has to be rigorously followed for an application to work correctly and whether a high amount of 3rd party applications is needed to gain sufficient traction in the market.

• **Integration effort**: This is based on the development and test effort that is required to create an end-product using its software and hardware components. This is a result of the amount of hardware and software components that are used and whether the interfaces between them are stable, i.e. whether a modular architecture has been created. For innovative products that contain components with new functionality and changing interfaces the integration and test effort is high.

Note1: These requirements are not completely independent. For instance a high degree of innovation will cause higher development costs and integration effort.

Note2: We do not further distinguish between requirements that are of “Medium” or “Low” importance, since for such a fine-grained distinction our method is not sufficiently accurate and complete at this moment in time.

### 3.4.4 Step 2: Evaluation of the requirements for each ecosystem type

In this section we analyze each of the three types of eco-system and each requirement is “scored” on whether it is challenging or easier to address, i.e. a strength of an ecosystem type. Here + means: easier to address, 0 means: neutral, and - means: challenging. This analysis is based on the experience of the authors, the information in literature and the case studies that are presented in our earlier work [Hartmann 2014].

**A: Development Cost.** The larger the scope of the platform, the more development and maintenance effort is required by the platform owner. When multiple parties are involved these costs can be shared between the participants [Genuchten 2007]. Because hardware and software development is costly, this is the main reason why ecosystems became widely adopted. Firms that aim for low costs specialize on a few products so that tasks become routine [Bolwijn 1990]. Consequently products that are developed in consort by specialized firms, especially when interfaces are pre-defined, can be made at lower costs.

For the vertically integrated type this requirement is challenging since they develop the entire platform including both hardware and software and can only amortize their investments over their own devices (hence score = -). The open source platform can more easily address this, especially when the complementors contribute to the platform and often individual developers develop code in their free time [Gawer 2008] (hence score = +). For the proprietary software platform we evaluate this as neutral (score = 0): Although this
Consumer Electronics Software Ecosystems

platform owner has to develop the platform on its own, the costs can be amortized over multiple products from different device manufacturers.

**B: Control over Quality.** The overall product quality depends on the combination of software from the different contributors and failures often occur because of component interaction, unclearly documented API or unknown use of the system [Trew 2006, Oberhauser 2007]. A firm that controls a wide scope of the architecture can guarantee the product quality more easily [Axelsson 2014]. In the situation where multiple firms are involved, and especially when the interfaces are not clearly defined, the quality can easily break down and externally developed code could access data in the system, causing malfunctioning or security problems. Furthermore, applications developed by complementors may not follow the UI standards, as set by the platform owner, thereby causing a reduced user experience [Bosch 2010, Jansen 2013].

The quality can be controlled easier by the vertically integrated platform type as they have full control over the end-product. Furthermore these firms are able to test the externally developed applications on their devices (hence score = +). For the proprietary platforms we evaluate this as neutral (score = 0): although they have no control over the hardware, they can control the API and, because of its closed nature, can easier avoid that code with defects is added to the platform. For the open source platform we also regard this as neutral (score = 0). Because of its open nature, faulty code can be added or code can be changed incorrectly by the complementors and applications may compromise the security. An advantage however, is that the software is developed and tested by a large variety of firms and open source developers.

**C: Variability:** Specialized firms can often add variability to a platform more easily [Moore 2006], e.g. because they have the required knowledge, can reuse existing hardware and software components and they can have a more intimate contact to the end-users [Bosch 2009].

Flexibility and variability is easier to achieve through the open source platform type, since complementors can add or remove functionality without the need to involve the platform owner (hence score = +). For the proprietary platform type we evaluate this as neutral (score = 0): The device manufacturers can create differentiating products with different hardware configuration; however this is limited to that which is supported by the software platform. As a comparison look at the PC industry where the different OEM suppliers of a Windows based PC can only compete on price, service and hardware quality, since the functionality is largely determined by the proprietary software platform. For the vertically integrated platform owner it is far more difficult to cover a wide range of products. In order for a firm to be flexible and respond quickly, it needs to focus on a number of core activities [Bolwijn 1990] and the development of all the required hardware and software components would simply be too costly. We therefore evaluate that this as challenging for this type (score = -).

**D: Freedom to innovate:** The optimal definition of the boundaries depends on where the major innovation steps in the architecture take place. When innovation takes place across
the boundaries of the platform the integrity of the platform is compromised and the complementors need to be involved thus slowing down the speed of innovation [Messerschmitt 2004]. Therefore, a wide scope allows for larger innovation steps more easily.

The introduction of multi touch screens in smart phones is such an example. Due to this innovation specialized hardware was needed; the interface towards the user has changed and a new API towards the application developers was required. Such a large innovation step couldn’t be done through small changes to an existing platform.

Large innovation steps are easier to establish by the vertically integrated platform type (score = +) because these firms control the entire architecture and complementors do not need to be involved. In the proprietary platform type the innovation is restricted because hardware is not part of the scope and the platform supplier has to involve hardware suppliers and device manufacturers for major steps (hence score = -). For the open source platform the hardware is also not part of the scope, however, the complementors may change the code and thus has the possibility for innovations, independently from the platform owner, for which the architecture does not have to be changed. Therefore we evaluate this as neutral (score = 0).

**E: Efficient use of system resources and hard real time requirements:** Due to the need for optimal resource utilization and low power consumption a direct control of the hardware is required. An embedded device usually contains a System on Chip (SoC) and each component is controlled separately. For instance, for video decoding in a digital television a separate building block of the SoC is used which can operate with low power consumption and is only active when needed. Furthermore, by controlling each part of the IC separately it can be avoided that processes interrupt each other. Therefore the design of the hard- and software are developed in parallel and require close co-operation [Halasz 2011].

As a comparison, look at the Wintel framework; the dominant ecosystem in the PC industry [Grove 1996]. This is a modular architecture with stable interfaces between the hardware and the software layer. Both Microsoft and Intel can independently innovate on their part of the architecture. Such a modular interface is possible because most demands of the end user can easily be met by existing hardware. For mobile phones such a modular interface is not yet possible since there are still large innovation steps that involve changes to hardware and software together and a modular architecture would lead to less efficient resource utilization [Christensen 2002].

Since the vertically integrated platform type has both control of the hardware and software this can controlled more easily (score = +). In the open source type the complementors can also change the code to accommodate the hardware and vice-versa and therefore this can also be controlled (hence score = +). For a proprietary platform type this is challenging since changes to hardware may require changes to the proprietary platform and vice versa (hence score = -).

**F: Stability of the Interface:** It is important for a platform to maintain a (sufficiently) stable API, thus avoiding that interoperability problems exist where applications do not work on the variety of devices based on different versions of a platform [Iansito 2004].
fragmentation is seen as the major challenge by the application developers [Gadhavi 2010, Gilson 2012].

Vertically integrated platforms can more easily avoid fragmentation since they have full control over the API (hence score = +). This also holds for the closed source proprietary platform, similarly as the case in the PC industry, which has proven to be the major advantages and success of this type of platform [Moore 2006] (hence score = +). For an open source platform this is challenging since fragmentation can easily occur because device manufacturers can change the API or the hardware (hence score = -).

**G: Effort for system integration:** The time needed to integrate the hardware and software components is taking an increasing amount of time especially when components from different parties are used, e.g. with different technologies, non-matching interfaces or heterogeneous architectures [Henzinger 2006]. When a modular architecture exists with stable interfaces, the integration time would be substantially less [Christensen 2004]. Furthermore, when multiple parties are involved, the time for interaction, definition of requirements etc., is forming a substantial part of the development effort.

For the vertically integrated platform type the integration is easier to manage since no complementors are involved and the integration is done with in-house development for which the interfaces can be defined (hence score = +). For the proprietary platform type this is challenging since this may require that code has to be altered or glue components have to be developed [Hartmann 2013] (hence score = -). In the open source platform type the device manufacturers or the system integrators can perform the hardware/software integration without (intensive) support of the platform supplier. Therefore we evaluate this as neutral (score = 0).

Table 2 gives and overview of the strengths and challenges per ecosystem type.

**Table 2 Overview of types of ecosystems and their strengths and challenges**

<table>
<thead>
<tr>
<th>Type / Requirement</th>
<th>A: Dev. cost</th>
<th>B: Quality</th>
<th>C: Variability</th>
<th>D: Innovation</th>
<th>E: System resources</th>
<th>F: Interface Stability</th>
<th>G: Integr. effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Vertically Integrated</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>2: Proprietary Software</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>3: Open Source Software</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

In this overview: + means: easier to address, i. a strength, 0 means: neutral, - means: challenging

**3.4.5 Step 3: Creation of the decision support matrix**

The determination, of which type of ecosystem is most suitable, is based on comparing the importance of each requirement with the strengths and challenges of the ecosystem types from Table 2. The decision support matrix of a product category shows the evaluation for each ecosystem type. An example is shown in Table 3. This matrix shows which of the requirements are of “high” importance and those that are “key”.
Table 3 Example of a decision support matrix

<table>
<thead>
<tr>
<th>Important requirements</th>
<th>A: Dev. cost</th>
<th>B: Quality</th>
<th>C: Variability</th>
<th>D: Innovation</th>
<th>E: System resources</th>
<th>F: Interface stability</th>
<th>G: Integr. effort</th>
<th>Total Score</th>
<th>Most used type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Vertically Integrated</td>
<td>-1</td>
<td>-1</td>
<td>+2</td>
<td>+1</td>
<td>+3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: Proprietary Software</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>-1</td>
<td>-3</td>
<td>n.a.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: Open Source Software</td>
<td>+1</td>
<td>+1</td>
<td>+2</td>
<td>0</td>
<td>+4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The total score is an addition of the individual scores for each requirement, i.e. ranging from -1 to +1. Here the scores of the key requirements are doubled, to express their relative higher importance, and are therefore ranging from -2 to 2.

The final column show which of the ecosystem types are used most in the current market. In this example the open source platform type is used most, followed by the vertically integrated type, while the proprietary type is not available (n.a.).

3.5 Case Studies

The case studies in this section cover a wide range of consumer electronics products that support 3rd party applications: (1) Gaming Consoles, (2) Digital Cameras, (3) Digital Televisions and Set-top Boxes, (4) Smart Watches, (5) Smart Phones, (6) Tablets and (7) Personal Computers. This selection includes products that exist for a decade or longer as well as products that are recently introduced. Furthermore both large boxes and wearable devices are included. Each case study contains a brief description of the product, the firms that are active, a historical overview of the transitions, a description the ecosystems that are used and the ecosystems that are considered most suitable, using the multi-criteria decision support method, and those that are used most.

These case studies are based on publicly available information. The determination of the importance of each requirement is done by the authors based on the products that are currently available in the market and supported by references when available.

3.5.1 Gaming Consoles

Product description and historical overview

Gaming consoles have existed since the fifties and originally only contained games that were integrated in the consoles. Later it became possible to purchase games separately, initially as cartridges, later as CD-ROMS and today downloadable from the Internet. The development of games was initially done by in-house developers, the so-called 1st party developers. Later games were also developed by 2nd party developers, i.e. game developers that worked on a contract base for the console manufacturers and later 3rd parties who independently developed games for a specific console. Hence the use of ecosystems was
Consumer Electronics Software Ecosystems

introduced already early for these products. However, the sales of games is controlled by the publishers, i.e. the manufactures of the consoles.

Gaming consoles are replaced with new versions every three to four years and consists of radical new functionality compared to the previous version [worldfinance 2014]. Consumers are willing to buy these new products, e.g. to support faster games, better graphics, 3D, motion detection and so on. Furthermore, consumers spend a great deal of money on new games, which provides additional revenues for the console manufacturers. Since the nineties, only three or four console manufactures had a dominant position [Vgchartz 2014], which resulted in a customer base that was large enough to amortize the investments.

Manufactures do not create many variants of their products; only regional differences are supported. In this way the developed games can be played on all the devices, where backwards compatibility of older versions of consoles is not supported. The major manufacturers use a proprietary hardware/software platform and have done so since the beginning of this industry.

In 2013 gaming consoles where introduced based on the open source software platform Android, such as Shield, Gamestick and Ouya. However none of these initiatives gained a large market share because they did not create the gaming experience that the traditional manufacturers offered [Trew 2014, Kain 2014, Hollister 2014]. A proprietary closed source software platform is not used in this market.

Analysis of ecosystem type

This product category is characterized in which almost all the requirements are evaluated as being of high importance with speed of innovation and interface stability as key requirements. The development costs are high since consoles consist of a large amount of software that need to support a variety of input devices and to support different screens. Also the requirements for system resources and hard real time requirements are high because performance and graphics quality determine the attractiveness of the system. The speed of innovation is key because gamers switch between systems frequently and want to play the latest games using the latest technology. A stable interface an absolute necessity because it has to be guaranteed that a purchased games run on a given console.

| Table 4 Decision support matrix for gaming consoles |
|-----------------|---------|--------|---------|--------|---------|---------|--------|--------|
| A: Dev. cost    | B: Quality | C: Variability | D: Innovation | E: System resources | F: Interface Stability | G: Integr. effort | Total Score | Most used type |
| Important requirements | High | Key | High | Key | High |
| 1: Vertically Integrated | -1 | +2 | +1 | +2 | +1 | +5 | 1 |
| 2: Proprietary Software | 0 | -2 | -1 | +2 | -1 | -5 | - |
| 3: Open Source Software | +1 | 0 | +1 | -2 | 0 | 0 | 2 |

69
Chapter 3

Table 4 shows that the vertically integrated platform type scores substantially better than the other types. This is because the speed of innovation and hard real time requirements are supported the best in this ecosystem type while the variability is not an important requirement. Due to the speed of innovation it is not expected that a high degree of modularity can be obtained. The vertically integrated ecosystem type is by far used the most in this market and this is likely to remain the dominant type.

The development costs are a main challenge. As a consequence only a few, 2 to 3, manufacturers have been successful in the market and newcomers are rarely successful. This is in sharp contrast with most consumer electronics products, e.g. televisions, audio players, smart phones etcetera, in which many firms are active and have a profitably business, and for which the development investments are substantially lower.

3.5.2 Digital Photo Cameras

Product description and historical overview

Digital cameras have been introduced in the mid-nineties and rapidly replaced photographic films. Digital cameras contain dedicated hardware components and operating system that have hard real time requirements to ensure that the system response timely and store an image correctly. The size of this software has been relatively small and focuses on a few dedicated tasks. In later years software features where added such as facial recognition, however a general purpose operating system was not needed.

Around 2010 cameras became connected via Wi-Fi, at first mainly to upload pictures. Due to the growing popularity of smart phones, and the reduced costs for hardware components, manufacturers started using android as a platform to add functionality that exists in smartphones. At the same time smart phones were equipped with higher quality lenses and sensors and cross-over product, such as the Samsung Galaxy K zoom were created. This cross-over is not surprising since Android was initially intended as platform for digital cameras [Berenaum 2013]. Lythro has created a camera that is focused on 3D and allows for focusing after the picture has been taken. Lythro offers a software platform, along with their device that is able to share pictures with social media [Lytro 2014]. Until know Lythro has not yet made their platform available for 3rd parties.

In the early conception of digital cameras, i.e. until 2005 the speed of innovation was high resulting in consumer willing to purchase new cameras with better image quality and features created through software. In later years, e.g. since 2010, the speed of innovation is much lower and the innovation is limited. As a result the sales on new cameras dropped [Mintel 2014].

Analysis of ecosystem type

The variability requirements are high because digital cameras are offered in large amounts of variants, e.g. ranging from compact cameras till single reflex cameras and a variety of types in between. Furthermore, different sensors, (touch screen) displays, flash lights etcetera are used. The uses of system resources and hard real time requirements is challenging because ICs with as small as possible footprint should be used because of the high pressure on cost price and because the response time when taking a picture is crucial. In contrast with gaming consoles, innovation is not a requirement of high importance.
because digital photo cameras usually follow the functionality in smart phones. Also interface stability is less important because supporting 3rd party applications is not a key selling feature.

Table 5 Decision support matrix for digital photo cameras

<table>
<thead>
<tr>
<th>Important requirements</th>
<th>A: Dev. cost</th>
<th>B: Quality</th>
<th>C: Variability</th>
<th>D: Innovation</th>
<th>E: System resources</th>
<th>F: Interface Stability</th>
<th>G: Integr. effort</th>
<th>Total Score</th>
<th>Most used type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Vertically Integrated</td>
<td>-1</td>
<td>+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>n.a.</td>
</tr>
<tr>
<td>2: Proprietary Software</td>
<td>0</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>n.a.</td>
</tr>
<tr>
<td>3: Open Source Software</td>
<td>+1</td>
<td>+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+2</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5 shows that the open source software platform has the highest total score, because the variability is high while the speed of innovation is very important for this type of product. A high degree of modularity is not likely to be obtained because of the need to support a large range of products while using ICs with a low cost price is essential. Therefore a proprietary software platform is not a suitable ecosystem type. In the current market only an open source platform is used.

3.5.3 Digital televisions and set-top boxes

Product description and historical overview

Televisions, and later digital televisions and set-top boxes, were developed by specialized firms that developed and manufactured the entire product. Over time more and more software and hardware components from specialized suppliers were used [Hartmann 2012]. The last decade years, i.e. since 2007, the possibility to add 3rd party applications is introduced so that consumers can use applications that they are familiar with from their personal computers and smartphones. Examples are YouTube, weather forecasts and sharing of pictures. For these types of TVs, the term “Smart TV” is used.

Many firms have created an ecosystem based on a vertically integrated platform, such as Samsung with Smart TV [Samsungforum 2014], LG with WebOS [Onlinereport 2014] and Apple with AppleTV [Apple 2014], a set-top box.

A large number of open source platforms have entered the market, including Android TV [Android 2014], Meego [Meego 2014], Ubuntu TV [UbuntuTV 2014] and XBMC [XMBC 2014]. Also vendors of proprietary platforms have entered the market, such as Horizon TV [Horizon 2014], Microsoft TV [TheVerge 2014A], and Yahoo! TV [YahooTV 2014]. Many of these platforms aim both digital televisions as well as set-top boxes since these devices largely contain the same hardware and software components. A variety of platforms is used, where none has gained a dominant position. Recently a number
of firms, with a relatively small market share, have adopted the open source software platform Android, such as Philips, Sony and Sharp [Engadget 2014].

**Analysis of ecosystem type**

The development cost are high because the software in these systems have to support a large range of broadcasting standards and require a high amount of software to efficiently create a good quality of pictures with high frame rate and high pixel density. The variability is high because digital televisions are developed with different screen size, price setting, should be able to support different regional standards and should be able to operate together with many other products. System resources and hard real time requirements are evaluated as main requirement because there is a large pressure on cost price and because power consumption has to remain low because cooling fans are not accepted in these products. Furthermore showing video and playing audio needs to be done without visual or audible interruptions. Finally, the integration effort is high since a large number of components from different suppliers need to be integrated and a high degree of modularity has not been obtained.

**Table 6 Decision support matrix for digital televisions and set-top boxes**

<table>
<thead>
<tr>
<th>Important requirements</th>
<th>A: Dev. cost</th>
<th>B: Quality</th>
<th>C: Variability</th>
<th>D: Innovation</th>
<th>E: System resources</th>
<th>F: Interface Stability</th>
<th>G: Integr. effort</th>
<th>Total Score</th>
<th>Most used type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Vertically Integrated</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: Proprietary Software</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: Open Source Software</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
<td>+3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6 shows that the open source software score best on most aspects, directly followed by the vertically integrated platform type. This analysis is in contrast with current practice in which the vertically integrated platform type is most used, where the two major players, Samsung and LG own 40% of the market share [Strategy 2014]. We believe that this is caused by commercial reasons, i.e. the manufacturers are reluctant to use an open source or proprietary platform with the fear that a dominant firm may take most of the revenues. Furthermore the investments to switch from an existing platform to a new one are significant and the major players have a large market share thus causing little commercial pressure.

The main player in this market, i.e. Samsung [Strategy 2014], uses a proprietary platform which is based on the open source platform Tizen [Samsungforum 2014]. Consequently, Tizen may become an important candidate to dominate this market and a battle for dominance with Android may be expected [Information 2014A].
3.5.4 Smart Watches

Product description and historical overview
Smart watches have been introduced in recent years but already exists in a wide variety. The functionality of most watches is, in addition to displaying the time, focused on showing notifications of incoming calls and messages [Theguardian 2014], while some watches are able to play music, take pictures and can be used for making telephone conversations, such as the Samsung Gear [Androidwatches 2015]. A selection of watches is focused on fitness and health by monitoring heart rate and physical activity, such as the Pebble and the Basis Peak [Pebble 2015, Mybasis 2015]. Apple introduced the Apple Watch that can be used for contactless payment [AppleWatch 2015], which was earlier introduced for smart phones [Trustedreviews 2014].

While the functions of smart watches are currently limited, in comparison with smart phones, a major architectural challenge is size and battery lifetime. As a consequence special operating systems have been introduced to accommodate this. At this moment in time there is no dominant platform used: Google has introduced Android Wear, which was first used by firms such Samsung, LG and Motorola [Androidwatches 2014]. Samsung also started using the open source platform Tizen [TheVerge 2014B] By using this platform, for which Samsung act as main contributor, it has more control over the platform and is not dependent on Google. Other firms, such as Apple, Sony and Pebble use a proprietary platform [Pebble 2014, AppleWatch 2014, Connectedly 2014].

Analysis of ecosystem type
The variability is evaluated as key because with different functionality, size, price setting etcetera are needed to satisfy the different customer needs, and because watches are often a personal statement, which is why Android wear is not considered sufficiently modifiable by some vendor [Johnson 2015]. The speed of innovation is high because these products are new and it is uncertain which functionality will become successful in the market. The use of system resources and hard real time requirements are evaluated as key requirement, given the size of these devices and the increasing amount of ICs that are used in these products. Currently most smart watches need to be charged daily even when they are not used continuously.
Chapter 3

Table 7 Decision support matrix for smart watches

<table>
<thead>
<tr>
<th>Important requirements</th>
<th>A: Dev. Cost</th>
<th>B: Quality</th>
<th>C: Variability</th>
<th>D: Innovation</th>
<th>E: System resources</th>
<th>F: Interface Stability</th>
<th>G: Integr. effort</th>
<th>Total Score</th>
<th>Most used type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Vertically Integrated</td>
<td>-2</td>
<td>+1</td>
<td>+2</td>
<td>+1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2: Proprietary Software</td>
<td>0</td>
<td>-</td>
<td>-2</td>
<td>-3</td>
<td></td>
<td></td>
<td></td>
<td>n.a.</td>
<td></td>
</tr>
<tr>
<td>3: Open Source Software</td>
<td>+2</td>
<td>0</td>
<td>+2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+4</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7 shows that the open source platform type is most suitable since this offers the consumer electronics and watch manufactures the best option to create distinguishable products. In the current market multiple open source platforms as well as multiple vertically integrated platforms are used, however little is known about the market shares. A proprietary software platform type is not available and based on this analysis this type is unlikely to become successful in this market especially because it is difficult for this type to support the requirements for optimal use of system resources.

3.5.5 Smart phones

Product description and historical overview
A description of the transition in the industry is already given in section 3.3.1. This subsection is limited to the trends in functionality: Initially the functionality of these products was limited to making telephone calls and sending messages via SMS. Around 1996 the so called feature phones were introduced that supported downloadable ringtones, playing music and games. A major shift occurred when functionality was incorporated that was previously available in personal digital assistants (PDA’s), such as a Wi-fi connection, GPS and at a later stage, the introduction of touch screens. As a result, a mobile phone, now called smart phone, attracted a large group of users, which are using their devices for a variety of tasks such as making pictures, using social media and communication with friends and others using WhatsApp and Twitter.

Analysis of ecosystem type
The development costs and innovation are evaluated as of high importance given continues effort that is needed to introduce new functionality. The use of system resources and hard real time requirements: Evaluated as high, given the size of these devices and the need to have a long battery lifetime. The interface stability is regarded as an important requirement because application developers should be able to rely that their product functions on the different devices, to amortize their investments. The variability is evaluated as key because it is needed to support phones with different functionality, size, price setting, quality of the camera, design etcetera and users typically buy a product that exactly serves their needs.
Table 8 Decision support matrix for smart phones

<table>
<thead>
<tr>
<th>Important requirements</th>
<th>A: Dev. cost</th>
<th>B: Quality</th>
<th>C: Variability</th>
<th>D: Innovation</th>
<th>E: System resources</th>
<th>F: Interface Stability</th>
<th>G: Integr. effort</th>
<th>Total Score</th>
<th>Most used type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Vertically Integrated</td>
<td>-1</td>
<td>-2</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td></td>
<td></td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2: Proprietary Software</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td></td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td>3: Open Source Software</td>
<td>+1</td>
<td>+2</td>
<td>0</td>
<td>+1</td>
<td>-1</td>
<td>+3</td>
<td></td>
<td>+3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 8 shows that for smartphones, the open source platform type is evaluated as most suitable. This analysis reflects the market share of 2013 [Gartner 2014] since the open source software has 79% of the market and the vertically integrated platforms 18% and 3% is of a proprietary platform, i.e. Microsoft.

The overview also shows the challenges for each ecosystem type:

- For the vertically integrated type, Development costs & variability are the main challenges. Apple, a key player, addresses this by limiting the variability and development costs by focusing on a particular market segments and by only supporting the latest versions of the handsets [Grobart 2013].

- For the proprietary software type these challenges are: innovation & system resources. Windows addresses this by closely cooperating with a small group of HW platform suppliers [Duryee 2009] and acquired Nokia, a previously vertically integrated firm.

- For the open source software type, interface stability & fragmentation are the main challenges. Google, the main open source supplier, addresses this by offering a compatibility test suite offered [AndroidComp 2014] and by making agreements with handset makers to use standard Google apps and user interface [Gannes 2014].

3.5.6 Tablets

Product description and historical overview

Tablets have been introduced two decades ago by traditional computer manufacturers but did not gain much traction from the market in the first decade [Zdnet 2014]. With the introduction of the iPad by Apple, this product category become widely used. Soon other manufacturers followed that, similarly as Apple, used the same platform for tablets as for smart phones.

Tablets, especially with larger screens, are used for different purposes than smart phones. Smart phones are primarily used, besides making telephone calls, for taking and sharing pictures, sending messages and so forth. Tablets are more used for answering mail, reading books and magazines, and watching and editing pictures. Therefore, tablets are increasingly used in schools and in business environments and often replace books and personal computers [Washingtonpost 2014, Information 2014B].
Chapter 3

Analysis of ecosystem type

Development costs are evaluated as high given the continuous effort to introduce new functionality. The interface stability is also evaluated as high because these products are starting to replace laptops and these products are used by schools and business applications. Variability is also an important requirement, since tablets are offered with different price settings, screen sizes, and so on.

Table 9 Decision support matrix for tablets

<table>
<thead>
<tr>
<th>A: Dev. cost</th>
<th>B: Quality</th>
<th>C: Variability</th>
<th>D: Innovation</th>
<th>E: System resources</th>
<th>F: Interface Stability</th>
<th>G: Integr. effort</th>
<th>Total Score</th>
<th>Most used type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Important requirements</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: Vertically Integrated</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: Proprietary Software</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>+1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: Open Source Software</td>
<td>+1</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9 shows that there is no difference in the total score between the three ecosystem types. In comparison with smart phones, the interface stability is regarded as also an important requirement, while system resources are less important because tablets have more options for heat dissipation and larger batteries.

In the current market there is only one vertically integrated player with a sizeable market share, i.e. Apple. For this type the interface stability is a big advantage over the open source platform type which is visible because schools use the iPad rather than an Android device [Washingtonpost 2014]. As an example: To accommodate the use of iPad in schools and business environments, Microsoft has ported Microsoft Office for the iPad, rather than for Android which has a larger market share, because of the fragmentation of the Android platform [Appleinsider 2014].

The market share of tablets, in comparison with smart phones reflects the differences. In comparison with smart phones, both the vertically integrated platform, i.e. Apple and the proprietary software platform have a larger market share [Thenextweb 2014, Theregister 2014]. However, the share of the proprietary software platform is (still) very small.

For the vendor of the proprietary platform, i.e. Windows, this can be a very viable market, much more than smart phones, since the interface stability is regarded as important, while the use of optimal system resources is less important because of the size of these products. Therefore targeting this market with a proprietary platform is expected to be more successful than targeting the market of smart phones as well, which is the current strategy of Microsoft [BBCNews 2014].
3.5.7 Personal Computers

Product description and historical overview

Personal computers were originally developed by vertically integrated firms, but with the introduction of a Microsoft’s MS-DOS and a modularization of the PCs hardware an ecosystem was created, the so called Wintel platform [Baldwin 1997].

The innovation of PCs has been small since more than a decade and consumers only switch to a new product when the old system is becoming too slow or does not support the software that they need to use. This often leads to version of new operating systems that are not adopted by the business and only limited by consumers. For instance Windows Vista was hardly adopted in a business environment. In this market one proprietary software platform is present, i.e. Microsoft Windows, and only one vertically integrated platform type, i.e. Apple. Also some open source software vendors are present, e.g. Linux, Chrome OS and FreeBSD.

Analysis of ecosystem type

Development cost: Development costs are evaluated as high because PC’s should be able to support a wide range of products, require frequent updates for security reasons and so on. Furthermore, users only switch to a new version when it is really needed, thus making it difficult to amortize the investments. Quality is evaluated as high, since these products have to support business critical applications and interface stability: This is evaluated as an absolute must because a user or company has to rely that an application he purchases will function correctly, and that it keeps on functioning during a long period of time.

Table 10 Decision support matrix for personal computers

<table>
<thead>
<tr>
<th>Important requirements</th>
<th>A: Dev. cost</th>
<th>B: Quality</th>
<th>C: Variability</th>
<th>D: Innovation</th>
<th>E: System resources</th>
<th>F: Interface Stability</th>
<th>G: Integr. effort</th>
<th>Total Score</th>
<th>Most used type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Vertically Integrated</td>
<td>-1</td>
<td>+1</td>
<td></td>
<td>+2</td>
<td></td>
<td></td>
<td>+2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2: Proprietary Software</td>
<td>0</td>
<td>0</td>
<td></td>
<td>+2</td>
<td></td>
<td></td>
<td>+2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3: Open Source Software</td>
<td>+1</td>
<td>0</td>
<td></td>
<td>-2</td>
<td></td>
<td></td>
<td>-1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Table 10 shows that the proprietary software type scores higher than the vertically integrated platform and the proprietary software type. In the market Microsoft has the largest market of around 90%, with personal computers of Apple’s just below 10% [Theregister 2014]. Open source software platforms are marginally used. Although there are benefits of the open source ecosystem type, the disadvantage, i.e. the interface stability, is a key requirements and prevents that this type likely to gain a significant market share. The only open source software platforms that have some success is the Chromebook, which is completely focused
on using a personal computer as client for web usage [Chrome 2015], and therefore relies on HTML as a stable interface. A native API is not offered.

3.6 Comparison with Related Art

In our previous work [Hartmann 2012] we analyzed the transition of mobile devices from 1990 until 2010, where the current paper analyses the situation of today. In [Hartmann 2012] we introduced a model consisting of 5 industry structure types, ranging from vertically integrated to closed and open ecosystems. The different types of ecosystems in the current paper is a refinement of the two ecosystem types described in that paper, where the vertically integrated hardware software platform consists is based on a vertically integrated platform, but has created a standardized interface for 3rd parties to contribute. In that earlier work the difference between a proprietary and open source platform types were not identified.

Another previous work of the co-author [Bosch 2010] describes architectural challenges. Some of these challenges have been used in the current paper as criteria that determine the best suitable scope. However in that work the criteria have not been identified for mobile devices or embedded systems.

In the work of Knauss and Hammouda [Knaus 2014] an assessment method is introduced that determines the suitability of a software architecture for an eco-system centric approach. In their work they addressed the tradeoff between the degree of openness of a software platform and reasons to limit the freedom. Their work did not include a comparison between different types of ecosystems for consumer electronics products.

The work of Gawer and Cusumano [Gawer 2008] discusses the scope and technology design of a platform but does not discuss implications for software engineering nor does this work identify or compare different types of ecosystems for consumer electronics products. Other related work identified different types of ecosystems [Baldwin 1997, Jansen 2012, Bosch 2009] but did not compare ecosystems in a particular domain nor did it identify the challenges from a software engineering perspective.

To our knowledge there is no related art that analyzes different types of ecosystem for consumer electronics and related work that contains a decision support model for different types of ecosystems; and therefore this paper is one of the first of its kind.

3.7 Conclusions and Further Research

In this paper we identified three different types of ecosystems for consumer electronics to support 3rd party applications and identified the criteria that determine the best suitable scope and technology design form a software engineering perspective. Based on this analysis we introduced a first step towards multi-criteria decision support method to determine the most suitable ecosystem type for consumer electronics products and used this method to analyze a wide range of products. Although our decision support method is not complete at this moment in time, and further research is needed, the analysis using this method already revealed essential difference between the types of ecosystems and their suitability for different product types.
This analysis shows that when the speed of innovation is a key requirement while variability is less important, the vertically integrate hardware/software type is more suitable. This is the case for gaming consoles.

For product types that require a high degree of variability and when it is needed to share the development investment of the platform between multiple firms, the open source software type is more suitable. Examples of such products are digital photo cameras, digital televisions, smart watches and smart phones.

For product types for which a stable interface is a key requirement, and development costs and quality are also important requirements, the proprietary software platform type is most suitable. This has been analyzed to be the case for personal computers. The proprietary software platform is less suitable for traditional consumer electronics products given the high pressure on cost price, which requires ICs with a small footprint and optimal use of system resources. On the other hand, open source software platforms are unlikely to gain a dominant position in the market of PC’s, since the requirements for interface stability are key, which is difficult to achieve for this ecosystem type.

For tablets, all three ecosystem types are considered equally suitable since these products have characteristics of both smart phones and personal computers.

While the choice of an ecosystem type is also, and possibly even more, determined by commercial and organizational aspect, in most cases the type that was considered most suitable using our decision support method, turned out to be the type that is most used in the market. This provides confidence that the introduced model is a sound first step.

The case studies also showed how the individual requirements influence the market and the choices of consumer electronics firms and software vendors. For instance, the challenge of development cost for the gaming consoles results that just a few players are active. In the smart phone and tablet domain, Apple focusses on a limited variety of products, Microsoft took over an existing hardware firm to be able to act as a vertically integrated HW/SW ecosystem type, and Google is offering a compatibility test suite and makes agreements with the handset makers to use standard Android in order to avoid fragmentation.

A primary topic for further research is to include business and organizational aspects in a decision support model. Further research may also reveal more relevant criteria. Furthermore an evaluation that is based on the expert opinion of more experts, and a more fine-grained analysis of the importance of the criteria will provide higher confidence in the results of the analysis. Due to these limitations, we consider the method introduced in this paper a first step towards a full-fledged decision support method.

In the categorization and comparison of ecosystem types, a strict separation was made between open source and proprietary closed source. However, software vendors may use a mixed approach, in which part of the software is proprietary, e.g. to create a stable interface towards the 3rd party application developers, while other parts can be used without restrictions. Similarly a vendor may decide that a certain part of the closed source platform may be modified, e.g. to easier support different hardware configurations. These mixed approaches will lead to a combination of both types of ecosystems. More research is needed to identify what the consequences for each of these ecosystem types are.

The key contributions of this paper are threefold:
(1) Three types of ecosystems are identified that are used in consumer electronics to support 3rd party applications.

(2) A first step towards multi-criteria decision support method is introduced to select the most suitable ecosystem type for a product categories, from a software engineering perspective.

(3) Case studies of seven consumer electronics products are presented and a comparison is made between the suitability of the types that are used in traditional consumer electronics products with those that are used in mobile devices and personal computers.

Firms that create consumer products and are considering to adopt an ecosystems centric approach can use the method presented in this paper to choose the type of ecosystem that will suit the market demands in the best way and address the challenges that arise as a result of this choice. Especially in an emerging market where no dominant ecosystems are available, such a choice is non-trivial and of crucial strategic importance.

Firms that create software platforms can use this method to target the product types for which this is most suitable and pro-actively address the identified challenges to make their platform successful in the market.
Part II: Variability Management in Software Supply Chains

“It is not necessary to change. Survival is not mandatory.”

W. Edwards Deming