OPENING EMBEDDED SYSTEMS FOR OPEN INNOVATION

Constantin Soeldner, Chair of Information Systems I, University of Erlangen-Nuremberg, Nuremberg, Germany, constantin.soeldner@fau.de

Angela Roth, Chair of Information Systems I, University of Erlangen-Nuremberg, Nuremberg, Germany, angela.roth@fau.de

Kathrin Moeslein, Chair of Information Systems I, University of Erlangen-Nuremberg, Nuremberg, Germany, kathrin.moeslein@fau.de

Abstract

Openness for open innovation is becoming increasingly important in the field of embedded systems (ES). ES are traditionally designed for a particular purpose, however most ES firms are beginning to open up their systems. This allows third parties to enhance them with additional software, or hardware components. It is not enough to just provide access to a system though, it also requires that it be modularized accordingly. A specific challenge when modularizing for open innovation, is the consideration of ES characteristics such as safety, security and or real-time requirements. In this multiple case study, our goal is to explore how ES producers can technically open their systems for open innovation. To address this question, this paper first systematizes the different forms of openness in ES. Secondly, it uses modularity theory to analyze the implications of the specific requirements of ES on the encountered forms of openness. It turns out that, different types of openness require differing levels of effort, with regard to modularization. Overall, the study shows that, in order to guarantee the proper functioning of their systems, ES firms tend to partition their systems in accordance to the specific requirements of ES.

Keywords: Open Innovation, Modularity, Embedded Systems

1 INTRODUCTION

The number of apps available for a given smart phone platform, constitutes a major factor in the success of those said platforms. Smart phones are an example of embedded systems, which were traditionally closed, but are now increasingly opened up to enable open innovation. Open innovation denotes the trend to open up the innovation process for externals (Chesbrough 2003). The features smart phones offer, are to a large degree provided by external developers. However, open innovation is not confined to smart phone platforms. Due to increasing hardware capacity other types of embedded systems are increasingly opened up for open innovation, such as; car infotainment, the smart home or the consumer electronics sector. Opening embedded systems offers a wide array of new innovation opportunities by integrating external developers. Besides, it has the potential to greatly reduce the time to market for new features, whereas closed embedded systems often cannot be changed at all. Other advantages are the possibility to connect embedded systems to other devices, thus enabling the Internet of Things (Axelsson et al. 2014). By opening their systems, firms can also accomodate the increasing customer demand towards customization.

Embedded systems (ES) are applied computer systems, which are designated towards a particular function (Noergaard 2005). These applied systems are found in a wide range of devices; e.g. automotive electronics, aircraft electronics, trains, telecommunication, medical systems, military applications, consumer electronics, fabrication equipment and smart buildings (Marwedel 2010). In contrast to general-purpose computer systems, like personal computers, ES usually execute specific applications. In addition, ES are usually subject to specific requirements. In particular dependability requirements (reliability, maintainability, availability, safety and security), as well as real-time (RT) requirements (Marwedel 2010; Noergaard 2005). Due to the criticality of these requirements, they must be prioritized when opening a system, as they ensure the proper functioning of the system.

In contrast to a mere opening of innovation processes, conducting open innovation in ES requires opening the underlying ES as well. The notion of openness refers to the "easing of restrictions on the use, development and commercialization of a technology" (Shapiro & Varian 1999). However, openness does not necessarily imply that the whole technology is opened, but only parts of it (Boudreau 2008). Thus leading to partial openness (Balka et al. 2010; West 2003). In this paper, we draw on this notion of openness for ES. By opening parts of their system, firms can allow third-party innovation at specific parts of their system, while still being able to control the core of their system. One of the main challenges, when opening a system, is to reconcile adoption with appropriability. While opening a system leads to a higher adoption of the system, firms may appropriate fewer profits by opening (West 2003). However, in the case of ES, additional challenges arise, mainly to the dependability requirements of ES, such as safety or due to RT requirements. These requirements also differentiate ES from general-purpose platforms, like Windows, which are also opened for third-party innovation. Our assumption is that, ES providers need to modularize their systems in accordance with these requirements.

We therefore want to explore how ES producers can technically open their systems for open innovation. To achieve this goal, we will proceed as follows: In the next section (2.1), literature on openness and its applicability to ES, with its particular requirements, is discussed. To operationalize openness for ES, we draw on the layer model of embedded systems. This allows for a preliminary classification of the different degrees of ES openness. In addition, we consider the specific characteristics of ES that affect their openness. To explore how systems are technically opened under the influence of the ES characteristics, we draw on modularity theory. This allows to explore the implications of openness on the ES design. The next section therefore offers theory on modularity and discusses the implications of openness on modularity (2.2). The third foundational section then presents the principles, guiding a particular modularization (2.3).

Based on these theoretical foundations, we conducted a multiple case study with a number of ES firms. This covered industries where openness for third-party innovation is becoming increasingly important.

The findings put forward the clearly distinct forms of openness found in ES, from the perspective of the different layers of ES. Furthermore, the findings show how ES can be modularized to enable these forms of openness. After presenting these findings, the discussion frames results in the context of modularity theory. This is followed by the conclusion and future perspectives.

2 BACKGROUND AND FOUNDATION

2.1 Openness of Embedded Systems

Openness, as a strategy to allow the participation of external actors in a system, is already established in a variety of different contexts: research in open source, for example, addresses why firms are revealing source to the general public (Fosfuri et al. 2008; Henkel et al. 2014). Free revealing of open source code is a strategy. In the case of standards, openness promotes coordination between different parties and thus leads to a higher value of a particular standard (Simcoe 2006).

Another valuable literature stream, which analyses openness on a systems level, is the literature on platforms. In the context of platforms, Eisenmann et al. (2008) define platform openness as; the degree of restrictions placed on use, development and commercialization of a platform. Opening a platform faces the trade-off between adoption and appropriability. By giving up some control over the platform, the platform provider can increase adoption of said platform. However, at the same time they may capture a lesser share of the profits (West 2003). Boudreau (2010) proposed two different strategies to open a platform; granting access to a platform (to spur complementary innovations) and giving up control over the platform. The first approach allows us to 'use' the platform and build additional complements on top of it. Whereas giving up control, would also allow externals to change the platform itself. Giving up control is usually enacted to a certain degree, which allows the development of complementary components and ensures interoperability. Thus, both approaches are performed coincidently to some degree. Boudreau (2010) discovered that especially granting access leads to a considerable acceleration of third-part innovation (although both approaches can often coincide). There have also been studies in the context of mobile phone platforms, such as (Anvaari & Jansen 2010), which evaluated architectural openness of mobile phone platforms. They operationalize openness according to the different layers of the system. An approach also used in this paper, as an initial classification of openness. However, they mention the importance of modularity only briefly. Schlagwein et al. (2010) constructed a framework, showing different degrees of openness, of information resources in the context of mobile phone platforms. They also differentiated between access to a resource and control of a resource. In this paper they differentiate the following types of openness: of the hardware specification, of the system core, of the background services, of the application development platform and of the application distribution platform. Whereas these types of openness may be partly transferable to ES, this paper aims to consider ES in general.

To operationalize how openness occurs in the context of ES, we draw on the layer model of ES that shows the basic architecture of ES. ES are usually structured in different layers. In its most simple representation in; the hardware, the operating system and the application layer (Figure 1). The operating system layer provides an abstraction to the underlying hardware and offers the hardware features for applications via an Application Programming Interface (API) (Saltzer & Kaashoek 2009).

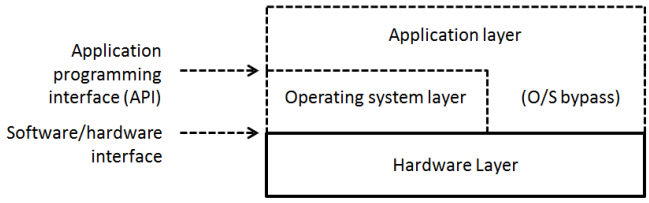

Figure 1. Layers of ES according to (Saltzer & Kaashoek 2009)

Opening an ES can be done at multiple levels, according to the objectives of openness. For instance, today's smart phones allow external developers to implement new 'apps', but deeper layers of the system cannot be altered. However, it is open to debate, whether smart phones still constitute ES, as they already have similar characteristics like general purpose systems. In contrast to ES, a general purpose computer system, like PCs or notebooks, exhibit openness on all three layers. Firms therefore can control the openness of their product by granting access to these different layers. The layer model provides a first approximation according to which, the phenomena of ES can be analysed.

However, opening a system is more than just the decision to open a particular layer. When opening a system, firms have to consider which parts they want to keep closed, as it constitutes valuable intellectual property (IP). Thus, openness usually takes place partially in alignment with the organizational goals (Balka et al. 2010; West 2003). Sometimes, the existing IP configuration would hinder firms in opening up their systems. In addition, opening systems such as ES needs to take into account the specific requirements of ES, which are not allowed to be violated. Therefore, opening ES for open innovation requires modularizing the system, in accordance with these different requirements. To analyse how ES can open their systems technically for open innovation, theory on modularity will be use. The next chapter will give an overview of modularity and how it is affected by openness.

2.2 Modularity and Openness

The notion of modularity applies to systems in general and refers to the degree in which, the components of a system can be partitioned into separate modules and be recombined (Schilling, 2000). Systems with a lower degree of modularity can be characterized as being more integral. With integral architectures, changes to one module require modifying other modules as well (Mikkola & Gassmann, 2003). Therefore, modularity is better described as a continuum than a discrete state. From an openness perspective, modular systems allow opening parts of the system, without affecting other parts (MacCormack et al. 2007; Mikkola & Gassmann 2003), thus enabling partial openness. The value of modularity has also been stated in open source literature, where modularity is held to be one of the key success factors of the Linux operating system. Linux itself possess a small kernel, which encompasses the basic operating system functions. The system can then be enhanced by independent modules (Bonaccorsi & Rossi 2003).

In contrast to open source software, ES firms would rather open their system partially. Partial Openness is in particular needed when firms want to protect the intellectual property (IP) of their systems. To still be able to open such systems, firms often re-modularize their system in accordance with their IP requirements. A strategy Henkel et al. (2012) coined as 'IP Modularity'. Following this approach, the system is modularized in such a way, that valuable IP would be put in a particular module, which remains closed. Hence, other modules with less valuable IP can then be opened.

This paper takes up the idea of modularizing a system in accordance with IP requirements. Further to this, it suggests that opening an ES technically requires taking into account additional factors, not yet considered in the literature. In contrast to regular computer systems, openness needs to take into account the specific requirements of ES, namely safety & security, and RT requirements. When opening an ES for third-party innovation, these requirements must not be violated. An opening of ES would thus, not only involve giving access to externals, but to modularize the system in accordance with IP requirements and technical characteristics of ES. The requirement to take into account these characteristics defines the main difference comparing the openness of existing platforms, with embedded systems. For instance, the Microsoft Windows platform for PCs, also offers openness for third-parties, but is not subject to RT or dependability requirements such as ES. Another reason ES would need to be modularized in accordance to openness, is that ES have been traditionally closed and have thus not been designed for openness. The next section shows how systems are modularized and lays out the principles leading to a particular modularization.

2.3 Principles of Modularization

Usually, the modular structure of a system is mainly determined by functional considerations. Modularity in product design, relates to the allocation of functions to modules and the relationship among modules (Campagnolo & Camuffo 2009). The scheme, according to which the functions of a system are mapped to physical components, is specified by the product architecture. A function indicates what a system does in contrast to its underlying physical composition (Ulrich 1995). The product architecture comprises (1) the arrangements of functional elements, (2) the mapping of functions to physical components and (3) the interface specifications of the physical components. Modular architectures are characterized by a one-to-one relationship between physical components and the functions of a system (Sanchez 2000; Ulrich 1995). Modularity is especially beneficial when the emphasis is on flexibility and rapid innovation (Brusoni & Prencipe 2001; Ulrich & Eppinger 2000). By relying on specified and standardized interfaces, components can be more easily substituted (Mikkola & Gassmann 2003).

Modularization as a design principle has a long tradition, especially in the field of software engineering. Parnas (1972) already describes it as a mechanism to increase flexibility and comprehensibility of a system. One core principle, put forward by Parnas, is information hiding, which aims to hide the complexity of a particular module. This allows developing and changing modules without knowing the details of other modules which might depend on it. The other modules are rather accessed by well-defined interfaces. Developers just need to know the specification of the corresponding interfaces.

The principles guiding modularization have also been laid out by Baldwin & Clark (2000): abstraction, information hiding and partitioning as well as standardization of interfaces. The principle of abstraction allows hiding the complexity of certain parts of the system and providing access to these modules via standardized interfaces. A particular modularization can be described by the notion of design rules. These constitute principles defining the composition of an artefact, the way it works and the way it is manufactured (Brusoni & Prencipe 2006). A complete set of design rules describes the following categories of design information: architecture, interfaces, integration protocols and testing standards.

In our multiple-case study, later in this paper, we will draw on these principles and design rules to explore the impact of openness on modularity. The next section gives an overview of ES and its specific characteristics which need to be considered when modularizing them. Out of these considerations, the following research question results: “How can ES be modularized to enable open innovation?”

3 RESEARCH DESIGN

To answer the research question, ‘how ES producers can technically open their systems for open innovation’, a multiple-case design was chosen. According to Yin (2008), case studies are used to explore a phenomenon in depth, in the context in which it occurs and where the boundaries between phenomenon and context cannot be clearly determined. To account for the complexity in the context of ES, multiple case studies have been conducted. This allowed us to cover a broad range of different ES. The following chapter describes the data collection and the data analysis process conducted for this paper.

3.1 Data Collection

Our study relies on multiple cases, in order to follow a replication logic, which allows us to extract generalizable patterns, but also to find contradictory evidence in the other cases (Eisenhardt 1989; Yin 2008). The cases themselves, have been selected in accordance to the guidelines laid out by (Eisenhardt & Graebner 2007). These state that the sample selection should be performed by choosing

cases, which are particularly suitable for uncovering and extending relationships and logic among constructs. Therefore, both cases representing 'traditional' ES, with a high level of dependability and RT requirements, have been selected. As well as cases which are not, to the same degree, subject to these requirements. One particularly important case selection criteria, was to reflect systems with different degrees of openness. The selected cases range from ES, where open innovation only contributes to a minor degree, to cases where open innovation determines the majority of the system functions. We also covered different industries. This allows us to consider the factors constraining openness and to explore the phenomena of openness more holistically. In total we selected 10 cases, eight being commercial industry examples, the other two being based on research projects focused on openness to third-party enhancements. This allows us not only to include current developments in industry, but also to include developments not yet implemented in practice. Although the cases span different industries, we believe that this strengthens our finding, as it allows covering ES subject to different degrees of safety & security requirements and RT-constraints. Data acquisition took place between July 2013 and January 2014.

After identification of the cases, we contacted the organizations in order to conduct interviews with either; the corresponding project manager, or an employee with in-depth knowledge (with regard to technical aspects of the strategy concerning openness of their ES). The interview was conducted by using a semi-structured interview guideline, which was developed based on theoretical findings in the literature. The main categories of questions concerned the motivation for opening the ES, its degree and scope, the factors by which openness was influenced and the modularization for openness. In total we conducted interviews with 9 out of 10 of the organizations participating in the study. Although one case does not include an interview, there was circumstantial documentation available, which allowed us to include this case. In addition we collected different kinds of written material, which was available publicly from the companies' websites and from other external sources (e.g. press articles, professional journals as well as research papers). Thus, more insights regarding the motivations of openness, the factors influencing design decisions as well as more details on the modularity of the opened ES, were gained. Table 1 gives an overview of the cases, as well as the respective industries, sources used and interview partners.

Case	**Industry**	**Sources used for the cases**	**Position of Interviewee**
AutoPNP: SW architecture for automation systems	Research Institution / Automation	Interview; Documentation	Project Leader
Inca: Open Camera Platform by Fraunhofer Institute	Research Institution / Camera Industry	Interview; Documentation	Project Leader
Infotainment Platform: Automotive Infotainment platform	Automotive	Interview	Department Manager, Project Leader
John Deere: agricultural vehicles	Commercial vehicles	Interview; Documentation	Advanced Engineer
Kuka Youbot: Prototyping robotics systems for research and education	Robotics	Interview; Documentation	Product Manager
Prosyst E-Health Middleware for e-health platforms	ES Engineering; E-Health	Interview; Documentation; Presentations	Technology Evangelist

Qivicon: smart-home platform	Telecommunications	Interview; Documentation	Product Owner
OpenXC: open-source platform to develop vehicle-data based applications	Automotive / Infotainment	Documentation; Website	
RACE Project: decentralized ICT architecture for cars	Research Project by Firm Consortium / Automotive	Interview; Documentation; Research Paper; Press Releases	Project Leader
Commercial vehicle platform: platform for data-based SW applications for commercial vehicles	ES Engineering / Commercial Vehicles	Interview; Documentation	Business Unit Leader

Table 1. Cases Overview

3.2 Analysis

Each of the recorded interviews has been transcribed verbatim. In the next step, the coding of the data was achieved by applying template analysis (King 1998). The codes were based both on the literature as well as on the underlying research question (Coffey & Atkinson 1996). The main hierarchical categories were based on, the literature on modularity, openness, open innovation and embedded systems. This also provided us with a starting set of subcategories. In addition, the initial set of codes was complemented during the systematic analysis of the quotes. This helped to derive appropriate codes describing the essence of the quote at hand. The newly identified codes were then subsequently grouped by comparing their meaning. For instance, regarding the main category of ES, it was started with an initial set of codes covering the main components and characteristics of ES based on the literature. Subsequently during the coding of the interviews, additional subcategories were added which were not reflected by the literature. Furthermore, codes were refined in case they were too narrowly or broadly defined (King 1998).

Coding of data has been accomplished independently, by two parties, with the help of the qualitative research software MaxQDA. After initial coding, we compared the data with the existing material by following an analyst triangulation process (Yin 2008). For triangulation, we were relying on different sources as depicted in Table 1. Some of the additional documentation was supplied by the interview partner themselves and gave us more insight into the specific ES, its architecture and openness. In other cases, we relied for instance on publicly available technical documents, which detailed certain technical standards or specifications. This also included material from standardization organizations, relevant for the cases. Company websites have been another source of documentation. Often the documentation available on the websites targeted external developers and therefore provided technical details as well as insights in the openness of the ES. Similarly to the interview transcripts, the additional documentation was also coded with MaxQDA. Coding was done by two independent researchers. If data collected from the various sources were inconsistent or contradictory and when the coding by the two coders led to different conclusions, we went back to the interviewee to clarify issues.

Theoretically, data analysis was driven by modularity theory, (according to Baldwin & Clark (2000)) with a focus on the design rules concept. Principles of abstraction, information hiding and partitioning, as well as standardization of interfaces, were used to analyse the impact of openness on the modularity of ES. These principles were used to describe how ES firms implement openness in their systems and the impact of the openness on the modular structure of their systems.

Broadly, the focus in the analysis was along the following themes: In the first step the degree in which different firms opened their systems to third-party innovation, was examined. During this process the

aims and use cases, firms pursue by opening their systems, were considered. Subsequently, the implications of the pursued openness, on the modular structure of the ES itself, were analysed. The focus here was on which factors affect the openness of a system and how the modularization of a system can help to facilitate open innovation (while minimizing technical and organizational risks). The data from the different cases has been compared by a cross-case analysis, which not only enhances the generalizability of case results, but also helps to gain a better understanding of a phenomenon (Miles & Huberman 1994; Yin 2008). We were thus able to draw general conclusions, moving from specific cases of ES openness, to a comprehensive view of ES openness and its required forms of modularization.

4 RESULTS

4.1 Openness of ES and the implications on modularity

The cases reveal that, openness for third-party innovation manifests itself quite differently in each of these cases. Classification according to the different layers of an embedded system, which we used as a first approximation, holds validity in the light of the cases, but does not fully describe the different nuances of openness of ES. Each of the cases shows that openness can be attributed to a particular layer. However, classification along the layer model does not sufficiently describe all kinds of openness encountered. The cases showed that openness can also occur by granting access to a system's data, but without allowing access to any system functions (which would allow controlling the device behaviour to some extent). Although data openness also takes place on the application layer, there are differences in scope and in the required modularization for data openness. In contrast to applications which are allowed to change the system's behaviour, modularizing for data openness leaves the system to a large degree untouched. The following paragraphs describe the different forms of openness, for third-party innovation, identified in the cases. They are ordered according to the level of access to a system and according to the layer model:

Data openness: Users can access certain interfaces of the system, in order to collect system or process data, enabling them to build data-driven applications. This form of openness has, for example, been observed in the OpenXC case, the commercial vehicle platform as well as in the Prosyst E-Health Platform and the Qivicon case. The OpenXC case represents an environment with high safety requirements and RT constraints which is not jeopardized by data openness. Similar motivations can be found in the commercial vehicle case. Our interview partner from the commercial vehicle case stated: "Essentially, this utility vehicle is opened to allow for data access". In this example, data openness is implemented to allow additional services, like fleet management, which merely requires access to vehicle data. In the John Deere case, which features agricultural machinery, one of the primary use cases for data openness, is to analyse data for crop management. This helps make crop planning more efficient. Many of these applications can also be based on historical data and do not necessarily have to reside on the ES itself. For instance, data-based services in the OpenXC case, can be run on an external device, such as a tablet PC. Thus, for data openness, ES providers often do not need to provide the same amount of resources.

Modularizing for Data Openness: Data openness has a rather small footprint in the overall architecture of a system. Basically, a firm needs to make sure that external users can only access certain data, without giving access to more confidential data. In addition the data access procedures themselves, should not compromise the system's proper functioning. The data-oriented applications themselves often run on external devices, for instance on mobile devices, like smart phone and tablet PCs. However, ES firms can also allow integration in the cloud. To realize this partitioning of data, access to it can be allowed via an additional data access layer. A common way to implement this is to add a module to the system. This module acts as a proxy for externals to control access to the data. This approach has, for instance, been chosen in the OpenXC platform, where the rest of the car's architecture remains unchanged. The device acting as the proxy, in the OpenXC platform, is the so-called "vehicle interface". According to the OpenXC specification, "it passively listens for a subset of

CAN messages, performs required unit conversion or factoring and outputs a generic version to the USB interface." The vehicle interface only provides this particular function of translating the CAN messages, but additional applications are running outside this particular module. The separation of applications developed by users and the message definitions also increases IP protection and security, as e.g. reverse engineering is more difficult to accomplish. Similarly, the commercial vehicle case offers a central component of access for vehicle data. In contrast, John Deere realized the separation of data by implementing a secondary bus system, the so-called ISOBUS, a standardized bus system which can be used for communication between different system components. Both the OpenXC as well as the approach chosen in John Deere help to protect IP as well as to guarantee the systems.

A modularization strategy to minimize costs and to minimize the impact on the original system is to outsource new applications outside the boundaries of the original system, e.g. by relying on mobile devices or a cloud infrastructure for additional applications. The OpenXC case uses this approach, where mobile devices act as the host for additional applications and at the same time provides the user interface for the applications. This strategy is especially applicable for data openness, as besides being able to access data, applications do not have strong dependencies to the system. In the infotainment case, we observed a similar strategy where new applications are streamed from the web and use the on-board display as a user interface.

Application Openness: In contrast to data openness, application openness allows external applications not only to read data from the system, but also to change the behaviour of the system. We found this form of openness in the cases of Inca, Kuka, RACE, Qivicon and Prosyst case. In contrast to data openness, application openness allows not only data access, but also to change the behaviour of the ES. Application openness can be implemented to different degrees, with some ES already providing a fixed application core, which can be expanded at certain endpoints whereas others, like Qivicon, leave the most part of application development to externals. Prosyst's e-health middleware explicitly supports different degrees of application openness as the degree of openness can be determined by the systems provider building on the Prosyst platform: "The firm who develops such a gateway on this base, of course controls, to what degree it should be opened" (Interview Partner Prosyst Case).

For ES whose use cases have not been fully developed, or are subject to change, a high degree of application openness can account for uncertain customer demands. Where the opening firm does not take the risk of investing in additional applications, without having detailed knowledge about customer preferences ("the interesting question is, in which direction the market will move […] the same with the partners who will develop their first applications and become smarter in this process", (Interview Qivicon)). This statement also demonstrates the value of application openness as an open innovation strategy.

Modularizing for Application Openness: In contrast to data openness, the footprint of application openness on the system is more pronounced, as ES must possess the capacity for additional applications. Commonly ES are designed in accordance with pre-defined functions, which are not subject to change. To cater for additional applications, ES providers may need to remodularize their systems. An example where this is particularly apparent, is the INCA camera. Whereas the traditional camera architecture consists of a Field-Programmable Gate Array (FPGA), the INCA camera architecture is built on a System on a Chip. This enables greater abstraction between hardware and software and therefore requires less knowledge of hardware from developers. According to the project leader of the INCA camera: "we screened the market and realized, that we can use the core parts of smart-phones, so-called System-on-a-Chip's [….] they are cheaper, but also the development effort is smaller and at the same time, they offer more functionality."

Running applications on the ES often constitutes a challenge. Many ES, which have been designated for a specific purpose, need to accommodate both for the core functions of an ES and for third-party applications. To achieve this, ES firms often partition their systems in separate domains for base functions and for third-party innovation. The main reason for separation, is to ensure safety and RT

requirements of the core functions. In the infotainment platform case, this allows "to bring innovation cycles known from the consumer industry to the infotainment domain". This is especially done, to protect safety-critical functions from being affected by additional applications. The separation of a system, in different parts, may also affect core interfaces of ES. As the John Deere case showed, the addition of a separate bus system (ISOBUS) provides a higher degree of freedom for additional third-party applications. On the one hand, the ISOBUS is less safety-critical than the CANBUS, which is used by the base vehicle functions. On the other hand, it allows a finer graduated control of openness by regulating access to the core parts of the system.

The isolation of critical parts can also be conducted gradually. The RACE project outlines two different paths towards achieving open architecture in the automotive domain, a revolutionary and evolutionary approach. Whereas both approaches would lead to the same architecture in the end, the evolutionary approach suggests a gradual isolation of critical parts. Thus, more and more critical parts would be isolated without taking too many risks.

Operating System Openness: Openness on the operating system layer, means that users are allowed to make changes to the operating system, including switching to another operating system itself. This can be motivated, because the OS does not provide certain functionalities. Qivicon, for example, allows external partners to integrate certain device standards, to broaden the range of supported devices on the platform. Thus, users can extend the platform with additional interfaces.

In the case of the KUKA Youbot, users are also allowed to install their own OS. In practice this case is however more of a theoretical option, as most users stick to the preinstalled Ubuntu Linux. Offering externals the possibility to install their own operating system also entails Application Openness, as the application layer directly builds on the operating system layer. This decision implies that the firms offering the ES, refrain from determining how a device is used. In the other cases, Operating System Openness was not encountered, partly because of missing use cases but also due to RT and safety requirements.

Modularizing for Operating System Openness: Providing users with the possibility to add additional drivers, requires the users to have limited access to the OS. Quivion realizes it by providing 'sandboxes' for additional OS modules, with limited permissions, changing certain OS functions, thus allowing extensions to the OS. Therefore, the OS itself has to be sufficiently modular to allow the 'plug-in' of new OS modules.

Hardware Openness: Hardware openness allows extending the existing HW base of an ES. Thus, firms typically allow the integration of complementary hardware modules at the periphery of the system. An example of this is the John Deere case, where other firms can integrate additional implements to the tractors of John Deere. The hardware base, in this case the tractor, with its integrated ES, has not been opened for externals. As John Deere cannot supply the complete range of possible agricultural implements, they rely on specialized firms providing complementary additions to their tractors. "The tractor constitutes the system base, whereas the actual function is delivered by the device connected to the tractors […] which are supplied by other firms. John Deere also manufactures their own implements, but they cannot cover every niche." The software integration takes place via the ISOBUS, already described in the application openness section. To a broader extent, this motivation is visible in the Qivicon case, where the integration of a broad spectrum of different devices on the Qivicon platform is rather essential: "as an example, we also work on the integration of protocols, in particular radio-based protocols, to allow partners the integration of their devices."

Hardware openness usually encompasses application openness as well, as it often entails allowing additional applications, which make use of the new hardware modules. Another insight gained by the cases, is that hardware openness often also implies Application Openness (as external parties would also develop additional applications, which make use of new hardware modules).

Modularizing for Hardware Openness: From a modularity perspective, enabling hardware openness for ES puts these systems closer to general-purpose systems (supporting heterogeneous hardware

devices). This requires an additional middleware, or hardware abstraction layer, with the purpose of abstracting hardware details to developers. For instance, e-health platforms require interoperability with a large number of connectivity standards. Prosyst offers a specialized middleware for e-health platforms, which abstracts the details of various connection standards like Zigbee, Z-Wave etc. An additional advantage of using a middleware, is that software applications can be developed and tested independently of the target hardware, thus facilitating the development for externals (Fortiss 2011).

Platform providers integrating a diverse set of heterogeneous devices, such as Qivicon or Prosyst, offer a broad variety of different interface standards and protocols. To allow for the integration of devices, not conforming to existing standards, Qivicon leaves room for external developers to enhance the system with additional interface standards. The platform supports this by providing a "system sandbox", with enhanced permissions to implement these changes. Whereas systems like Prosyst and Qivicon need to conform to a broad variety of different connection standards. Other ES, like John Deere, follow a different approach concerning interfaces. The interface strategy of John Deere, with its ISOBUS, has already been discussed in the section "Modularizing for Application Openness". As the CANBUS constitutes a critical interface, the ISOBUS as a second bus system is used by external hardware providers to interact with other system components.

Table 2 classifies the cases according to the forms of openness of ES.

Case	**Forms of Openness**	**Openness description**
OpenXC	Data Openness	Access to vehicle data to build additional applications on third-party devices
Commercial vehicle platform	Data Openness	Access to vehicle data for fleet management, operations management, repair and maintenance
Infotainment platform	Data Openness, Application Openness	use of vehicle data in combination with external data for additional applications; control certain functions via external devices
RACE Project	Application Openness, HW Openness	Allow upgradability of hardware and software applications; add additional software functions
Qivicon	Data Openness, Application Openness, HW Openness	Integration of new hardware modules to the system; Development of SW applications on top
Prosyst E-Health Middleware	Data Openness, Application Openness, HW Openness	Integration of new hardware modules to the system; Development of SW applications using the existing hardware base
John Deere	Data Openness, Application Openness, HW Openness	SW integration of implements provided by third parties
INCA	Application Openness	Application development on open camera platform
KUKA Youbot	Data Openness, Application Openness Operating System Openness, HW Openness	Additional hardware modules for robotics platform; SW development for robotics platform
AutoPNP	HW Openness	Facilitated integration of new HW components

Table 2. Forms of Openness found in the cases

In Figure 2, the implications of the forms of openness on modularization are depicted. As it can be seen, operating system openness has the least implications regarding the modularity of the system.

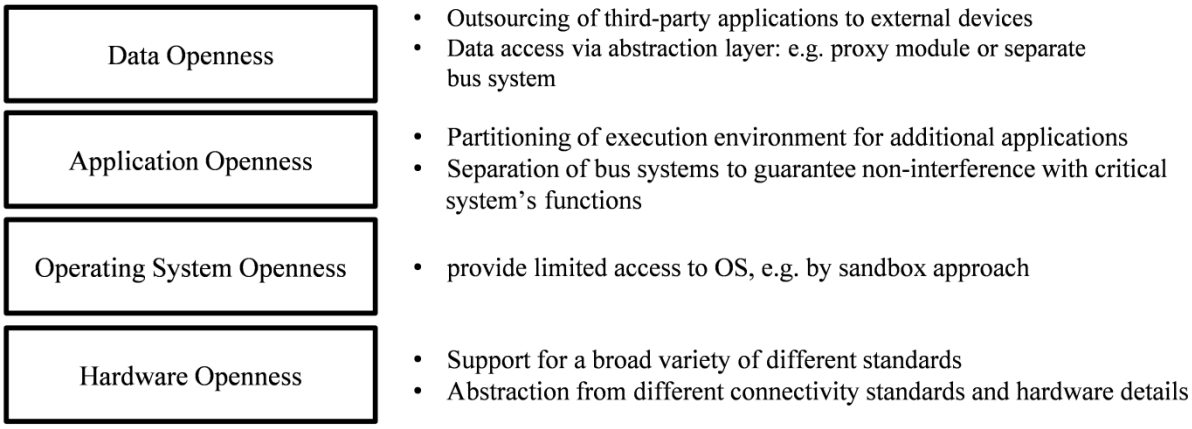

Figure 2. Implications of Openness of ES on Modularity

5 DISCUSSION OF KEY FINDINGS

The first part of our findings presented different forms of ES openness and can partly be operationalized according to the layer model of embedded systems. However, in the analysis an additional form of openness, namely data openness, has been identified. Although similar to application openness, it allows us to build additional applications on top of the system. There are a couple of reasons to differentiate between these two forms of openness. First of all, applications which build on data openness do not change the behaviour of the system and can thus be executed independently of the ES in question. In addition, for ES providers, the effort regarding modularization to implement data openness is rather low. To grant externals access to certain data, often an additional abstraction layer is introduced. Otherwise, the system largely remains untouched. Data openness can thus be pursued as a first step to enable 'read-only' applications, without having to invest too much effort and resources. The other forms of openness, based on the layer model, were; application openness, operating system openness and hardware openness. These forms of openness are not mutually exclusive, but can also occur concurrently. Operating system openness played the least important role in the cases.

Regarding the modularization of ES for open innovation, the findings show the following: A common theme, which can be observed in the cases is a move towards a more clearly layered architecture, similar to general-purpose computer systems. Moving in this direction is however hindered by the technical characteristics of ES, in particular safety, security and RT requirements. One approach to nonetheless allow open innovation encountered in the cases was a corresponding partition of the system. This allows them to ensure these requirements. We observed two kinds of partitioning: a partitioning of the execution environment, for additional application, as well as a partitioning of the key bus systems (providing interfaces for the different modules). Relying on partitioning, ES providers can still keep their original architectures to a large extent. However at the same time, the opened parts would exhibit general-purpose systems' characteristics. Partitioning can, in some cases, also be achieved by relying on external infrastructure (like tablet PCs), which provide the necessary capabilities to run third-party applications. This approach is particularly applicable for data openness, but can also be pursued for some cases of application openness. On the same line, additional applications can also be partitioned from the rest of the system by using the cloud. ES providers can make use of these different ways of partitioning their system according to their technical and strategic requirements and constraints. In contrast, there were also cases which did not need to ensure safety and RT requirement to the same degree. As an example Prosyst or the Qivicon platform. The modularization of these systems therefore did not exhibit such a partitioning.

In general, the cases show that openness also leads to a more modular architecture. Especially the design principles of abstraction and interface standardization are more pronounced, when opening the ES. Abstraction helps external partners to be able to develop additional applications, for a system, without needing to know too much about the system's details. The need for interface standardization arises mainly to achieve compatibility and attract externals to contribute. In this regard, open ES are becoming more similar to general-purpose systems. However, the most important change regarding the modularity of a system is the increased need for partitioning a system to guarantee both critical RT and dependability requirements as well as accommodate for openness. This differentiates ES from general-purpose systems which are not subject to these requirements.

The required changes for openness, show that firms pursuing openness need to adapt a more complex system architecture, in order to cater for different sets of requirements. This can, in many cases, lead to higher costs, as the systems not only need to offer enough capabilities to handle the additional workload, but also to accommodate for different workload types. In terms of costs, data openness requires the fewest changes to the modularity of the system, thus causing fewer costs. Whereas application openness, in many cases, requires major changes in the modularity of ES.

6 CONCLUSION AND FUTURE PERSPECTIVES

This paper contributes to research in the field of open innovation, by showing how ES producers can technically open their systems for open innovation. First of all, the cases revealed different forms of openness occurring in the context of ES. The encountered forms of openness enhance the forms of openness derived from the layer model of ES by adding data openness as a fourth form of openness. To analyse how these forms of openness are realized, theory on modularity has been applied. Modularity theory was used to show how to technically open ES by using the main principles of modularity such as partitioning, abstraction and interfaces. Partitioning in accordance to RT and dependability requirements turned out to be a crucial factor to open ES. Partitioning can be achieved in different ways, e.g. by partitioning the execution environment or key interfaces of the system. In addition, partitioning can also be accomplished by making use of external computing resources, ranging from tablet or smart phones to cloud resources. Furthermore, the paper offers an operationalization of the notion of openness in the context of ES. It can be seen, that pursuing openness needs to take into account the technical structure of systems, and is also limited by it. In particular, it was shown that openness is not only an organizational decision, but involves considering technical constraints, such as safety or real-time requirements. Considering these requirements also influences the modularization of ES, when opening them. The results also hold value for the platform literature, as opening ES for open innovation basically moves these systems in the direction of platforms. There is already some literature on platform openness, which this paper would further expand by taking a more technical view on the requirements of ES. This paper in particular also contributes to research in open innovation by showing how an open innovation policy can be implemented on a technical level and how open innovation is informed by technical factors. Further research regarding open innovation in embedded systems should also investigate the organizational impact of these forms of ES openness.

The paper also holds managerial implications: It provides an overview of ES openness, which helps practitioners in pursuing ES openness and assessing the technical impact of openness, due to its required changes in the modularity of the system. For practitioners, the categorization shows different approaches to implement an openness strategy for their systems which considers common constraints and challenges in the field of ES. A limitation can be seen in the number of cases: to provide more details regarding the different industries, a higher amount of cases would need to be elicited. Further on, a more detailed exploration of the influence of other factors on the modularity of ES, such as legal or costs aspects, would be interesting. Another interesting research path would be to show how ES firms can gradually remodularize their system, to more openness. The RACE case, with its focus on an evolutionary approach towards modularity for openness, already constitutes an example.

References

Anvaari, M., & Jansen, S. (2010). Evaluating architectural openness in mobile software platforms. *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, *ACM*.

Axelsson, J., Papatheocharous, E., & Andersson, J. (2014). Characteristics of software ecosystems for Federated Embedded Systems: A case study. *Information and Software Technology*, *56*(11), 1457–1475.

Baldwin, C., & Clark, K. B. (2000). *Design Rules: The Power of Modularity*. Malden, MA: Blackwell.

Balka, K., Raasch, C., & Herstatt, C. (2010). How Open is Open Source? - Software and Beyond. *Creativity and Innovation Management*, *19*(3), 248–256.

Bonaccorsi, A., & Rossi, C. (2003). Why Open Source software can succeed. *Research Policy*, *32*(7), 1243–1258.

Boudreau, K. (2008). Opening the platform vs. opening the complementary good? The effect on product innovation in handheld computing. *HEC Working Paper*.

Boudreau, K. (2010). Open Platform Strategies and Innovation: Granting Access vs. Devolving Control. *Management Science*, *56*(10), 1849–1872.

Brusoni, S., & Prencipe, A. (2001). Unpacking the black box of modularity: technologies, products and organizations. *Industrial and Corporate Change*, 179–205.

Brusoni, S., & Prencipe, A. (2006). Making Design Rules: A Multidomain Perspective. *Organization Science*, *17*(2), 179–189.

Campagnolo, D., & Camuffo, A. (2009). The concept of modularity in management studies: A literature review. *International Journal of Management Reviews*.

Chesbrough, H. (2003). *Open innovation: The new imperative for creating and profiting from technology*. Boston, MA: Harvard Business School Press.

Coffey, A., & Atkinson, P. (1996). *Making sense of qualitative data: Complementary research strategies.* Thousand Oaks: SAGE Publications.

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *The Academy of Management Review*, *14*(4), 532.

Eisenhardt, K. M., & Graebner, M. E. (2007). Theory Building From Cases: Opportunities and Challenges. *Academy of Management Journal*, *50*(1), 25–32.

Eisenmann, T., Parker, G., & Van Alstyne, M. (2008). Opening Platforms : How , When and Why ? *Harvard Business School Working Paper 09-030*.

Fortiss. (2011). Mehr Software (im) Wagen : Informations- und Kommunikations- technik (IKT) als Motor der Elektro mobilität der Zukunft. *eCar-IKT-Systemarchitektur Für Elektromobilität*, (im).

Fosfuri, a., Giarratana, M. S., & Luzzi, a. (2008). The Penguin Has Entered the Building: The Commercialization of Open Source Software Products. *Organization Science*, *19*(2), 292–305.

Henkel, J., Baldwin, C., & Shih, W. (2012). IP Modularity: Profiting from Innovation by Aligning Product Architecture with Intellectual Property. *Harvard Business School Finance Working Paper*.

Henkel, J., Schöberl, S., & Alexy, O. (2014). The emergence of openness: How and why firms adopt selective revealing in open innovation. *Research Policy*, *43*(5), 879–890.

King, N. (1998). Template analysis. In *Qualitative Methods and Analysis in Organizational Research*. London: Sage.

MacCormack, A., Rusnak, J., & Baldwin, C. (2007). The Impact of Component Modularity on Design Evolution: Evidence from the Software Industry. *SSRN Electronic Journal*.

Marwedel, P. (2010). *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*. Springer.

Mikkola, J. H., & Gassmann, O. (2003). Managing modularity of product architectures: toward an integrated theory. *IEEE Transactions on Engineering Management*, *50*(2), 204–218.

Miles, M., & Huberman, A. (1994). Qualitative data analysis: An expanded sourcebook, 352.

Noergaard, T. (2005). *Embedded systems architecture: a comprehensive guide for engineers and programmers* (p. 640). Newnes.
Parnas, D. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*.
Saltzer, J., & Kaashoek, M. (2009). *Principles of computer system design: an introduction*. Morgan Kaufmann.
Sanchez, R. (2000). Modular architectures, knowledge assets and organisational learning: new management processes for product creation. *International Journal of Technology Management*.
Schilling, M. (2000). Toward a general modular systems theory and its application to interfirm product modularity. *Academy of Management Review*, *25*(2), 312–334.
Schlagwein, D., Schoder, D., & Fischbach, K. (2010). Openness of Information Resources–A Framework-based Comparison of Mobile Platforms. *Information Systems*.
Shapiro, C., & Varian, H. R. (1999). *Information Rules: A Strategic Guide to the Network Economy* (p. 352). Boston, MA: Harvard Business School Press.
Ulrich, K. (1995). The role of product architecture in the manufacturing firm. *Research Policy*, *24*(3), 419–440.
Ulrich, K., & Eppinger, S. (2000). *Product design and development*. New York: McGraw-Hill.
West, J. (2003). How open is open enough? Melding proprietary and open source platform strategies. *Research Policy*, *32*(7), 1259–1285.
Yin, R. K. (2008). *Case Study Research: Design and Methods*. Thousand Oaks, CA: Sage Publications, Inc.