RECONFIGURING TECHNOLOGY INTERDEPENDENCIES WITHIN BUSINESS ECOSYSTEMS TO FACE DIGITAL CONVERGENCE: THE ROLE OF PRODUCT ARCHITECTURE AND MODULARITY

Azoulay, Alexandre

Université Côte d'Azur, CNRS, GREDEG UMR7321 France

alexandre.azoulay@univ-cotedazur.fr

ABSTRACT :

The ecosystem literature points product modularity as a condition to efficiently manage technology interdependencies within business ecosystems. Indeed, modularity allows to reduce coordination needs and costs but also to accelerate innovation through the development of extensive loosely coupled innovation networks. But beyond issues related to actors coordination, the structuralist approach to ecosystems points that the structure of these interdependencies determines their individual opportunities for value capture through defining their "position" along innovation flows. However, the management literature remains relatively silent about the impact of modularity on the structure of interdependencies and ecosystem actors' position. This gap seems all the more critical as, when facing important technological changes such as those brought by digital convergence, ecosystems can have to reconfigure in order to adapt the structure of these interdependencies to new technical and strategic challenges. In such context: **How can modularity help a firm to reconfigure the structure of its ecosystem interdependencies?** To address this research question this study focuses on the case of one of the leading global car marker, which is currently leading unprecedented architectural changes to reconfigure its business ecosystem in response to the convergence of automotive and digital technologies. We conducted a longitudinal analysis of the evolutions of the electronic architecture of its vehicles over the period 1984-2020. We found that to do so the car maker did not only adjust the degree of modularity but rather shifted the nature of modularity, from a siloed modular logic into a layered modular logic in order to radically transform the structure of its ecosystem interdependencies.

Keywords :

Business Ecosystems; Reconfiguration; Modularity; Technology Interdependencies; Digital convergence.

INTRODUCTION

An efficient management of technology interdependencies is of critical importance for the success of innovations (Adner & Kapoor, 2010; Adner, 2017; Teece, 2018). Technologies can be defined as sets of technical knowledges and practices underlying the design and manufacturing of an innovation but also determining how it delivers its functions (Pavitt, 1998; Brusoni et al., 2001). In other words, technologies are intangible assets embodied in the innovation during its design phase (Dosi, 1982). Technology interdependencies thus occur "when the value of an innovation depends on altering the nature of one or more existing technologies and/or on creating new ones" (Teece, 2018). Thereby, managing such interdependencies mainly consists in ensuring the compatibility between the various components of a system through adapting their respective "core design concept" to each other (Henderson et Clark, 1990). Consequently, it seems all the more challenging as these interdependencies must be considered from the earliest phases of the design of innovations. In addition, it can be particularly complex when brand new technologies jostle established ones that tend to improve according to well-defined innovation trajectories derived from their usual core design concept. In such case, technology paradigm shifts (Dosi, 1982) can occur and lead to completely new innovation trajectories characterized by technology imbalances due to uneven rate of development and leading to hard to identify and unpredictable interdependencies (Brusoni et al., 2001).

Nowadays, such contexts tend to characterize a growing number of industries, especially because of the acceleration of the "digital convergence" phenomenon (Teece, 2018). Indeed, in the last three decades, digital technologies have led previously distant and siloed industries to converge around more complex multi-technology products (Iansiti et Levien, 2004). This phenomenon can be explained because digital technologies often stand as "enabling technologies", which are continuously improving and allow a large amount of niche applications within several industries (Teece, 2018). These technologies' pervasive potential for disruption often force incumbent industry players to adapt their products but also to collaborate with new partners that masters these digital technologies (Adner, 2012; Teece, 2007, 2018). Current trends in the automotive industry offers a typical example of this phenomenon. Indeed, the automotive industry is currently experiencing one of the most radical changes in its history with the emergence of Connected, Autonomous, Shared and Electric Vehicles (CASE). These four new attributes of vehicles share the common point to rely heavily on digital technologies coming from the computing, Internet and mobile industries which are

converging with those of the automotive industry. This rupture is mainly driven by new entrants to the industry - such as Google, Apple, or even Uber, to name a few - and is turning automotive innovation from trajectories focused on mechanical technologies into trajectories focused on digital technologies. This shift poses significant issues for automakers, especially because they must collaborate with these new players who master complementary technologies but also threaten to take a prominent role in the industry.

As David J. Teece acknowledged in 2018 such digital convergence context "now makes a discussion of ecosystems imperative" (Teece, 2018, p.1375). Since the first elaboration of this concept by James F. Moore in 1993, the growing literature on business ecosystems strives to enlighten the mechanisms through which complementary firms strain to manage their interdependencies in innovative context. Business ecosystems usually gather actors coming from different industries with heterogeneous specialization, profile, culture, and goals, and which have to collaborate because of a "shared fate" resulting from their interdependences (Moore, 1996). Thereby, business ecosystems can be understood as a kind of open meta-organizations supported by varying degrees of hierarchical control (Gulati et al., 2012) and specifically dedicated to the management of interdependencies between complementary innovative firms (Jacobides et al., 2018). The recent structuralist approach to ecosystems coined by Ron Adner (2017), suggests that the structure of these interdependencies is determined by the "alignment structure" of the ecosystem members, understood as the way in which innovation activities are distributed among them. A strong hypothesis flowing from this structuralist approach is that ecosystem actors can shape the structure of their technology interdependencies by carefully designing their alignment structure. The alignment structure must be designed to ensure both collective value creation, through an efficient management of interdependencies, and individual value capture, determined by the firms' position within the flows of activities (Adner, 2017). Also, Adner (2012) emphases that the emergence of new interdependencies – as in the case of digital convergence – can strain ecosystem to reconfigure in order to reshape technology interdependencies and integrate new contributions within the alignment structure. This hypothesis finds many echoes in the ecosystem literature and scholars almost systematically point modularity as a critical lever to shape ecosystem interdependencies and to run complementary innovation processes in parallel (Iansiti et Levien, 2004; Moore, 2007; Jacobides et al., 2006, 2018; Teece, 2018).

These assertions usually build on the findings of the modularity theories which consider modularity as a powerful design strategy to partition knowledge among firms, to accelerate

innovation through the constitution of extensive networks of highly specialized innovators but also to optimize their coordination and associated costs (Sanchez et Mahoney, 1996; Baldwin & Clark, 2000; Langlois, 2003, 2006). However, to date, the developments of these theories are somehow disjointed from those of the ecosystem literature. At first glance, these two literatures seem quite complementary since they both flourished from the 1990's, sharing some common emblematic cases, such as IBM, Intel or Microsoft, and getting both interested with the management of interdependencies in innovative contexts. But they also strongly differ on their lens and focal preoccupations. Indeed, modularity theories are mainly focused on outsourcing strategies, trying to understand how modularity can help to lower transaction costs, to increase innovation outcomes and to improve coordination between an architect firm and its suppliers (Sako, 2003, Frigant, 2005). As such, it focuses on the optimization of existing supply chain interdependencies within a given industry while the ecosystem literature mainly focuses on the management of emerging interdependencies between heterogeneous complementary actors. Moreover, many studies even show that modularity can prevent firms to identify and adapt to such emerging interdependencies but also to shift their innovation trajectory, because of excessive knowledge partitioning (Henderson & Clark, 1990; Brusoni et Prencipe, 2007; Fixson & Park, 2008; Chesbrough & Prencipe, 2008) referred by some authors as a "modularity trap" (Chesbrough et Kusunoki, 2001) or "mirroring trap" (Colfer & Baldwin, 2007). Also, while the ecosystem literature focuses on the articulation of value creation and value capture modularity theories exclusively investigate value creation, merely ignoring value capture concerns (Baldwin et Henkel, 2014). This could have simply been a gap. But some examples suggest that modularity can be detrimental in regard with a focal firm ability to capture value. It was the case with IBM which bet on modular computers with the IBM 360 and involuntarily let Microsoft and Intel, focused on rapid module innovation, take the reins for the development of the computer industry (Gueguen et Torrès, 2004; Jacobides et MacDuffie, 2013, Baldwin et Henkel, 2014). To date, similar settings characterize the automotive industry. Indeed, for more than four decades, the automakers have also bet on modular principles to design their vehicles allowing them to focus on system integration and to outsource the design and production of modules – including electronic and software ones – by constituting extensive networks of subcontractors (Frigant et Jullien, 2014). By doing so, they deprived themselves of control over an important part of the software of their vehicles. In light of the ongoing shifts in the automotive sector, regaining control over digital innovation trajectories has become a major challenge for them to keep their central position within their ecosystems.

Given the insights from both the ecosystem literature and the modularity theories, there is no doubt that product modularity plays an important role in shaping technology interdependencies within business ecosystems. But some critical gaps remain in our understanding of this role, especially in the context of digital convergence. First, regarding the role of modularity in the reconfiguration of the ecosystem, necessary to manage emerging technology interdependencies with new complementary actors. Second, with regards to its role in adapting value capture mechanisms, to the entry of these new actors. This study aims to fill these gaps by addressing the following research question: **How can modularity help a firm to reconfigure the structure of its ecosystem interdependencies?** To address this research question, this study focuses on the automotive sector which is an ongoing exemplar case of digital convergence. This contribution focuses on the case of one of the leading global automakers which is currently striving to reconfigure its business ecosystem through a radical transformation of its vehicles' Electric and Electronic (E/E) architectures. Thanks to an impressive variety of data collected within this company we built an in-depth longitudinal case study of the evolution of the E/E architectures designed by this automaker over four decades, from 1984 to 2020. We analyzed our data to enlighten how recent changes vehicles' E/E architecture helps the OEM to reshape the structure of technology interdependencies and to improve its position within its ecosystem. In the following section, we will present the theoretical framework we built to analyze the structure of technology interdependencies within this ecosystem. Then, we will present our research design and the case we studied in the second and third section. In the fourth section we will present our results and finally discuss them in the fifth section.

1. Theoretical framework: the role of modularity in managing ecosystem interdependencies.

In this section we propose a theoretical framework to assess the structure of technology interdependencies within ecosystem taking into account both value creation and value capture issues. To do so, we first structure a matrix that allow to characterize the type of technology interdependencies based on two criteria: directionality and intensity. Then, we discuss how modularity can influence interdependencies according to existing literature.

1.1. Assessing the structure of technology interdependencies within business ecosystems.

1.1.1. Ecosystem interdependencies determines value creation and capture within business ecosystems

The primary concern of ecosystem scholars lies in understanding how heterogenous complementary firms strive to manage their interdependencies (Moore, 2007; Iansiti & Levien, 2004; Adner, 2012,2017; Jacobides et al.,2018). In this vein, Adner (2012) underlines "co-innovation risk" as one of the more critical issue faced by ecosystem members. This risk refers to the need, for interdependent innovators, to coordinate carefully their activities in order, for their respective innovations, to reach markets simultaneously, so that they can offer a consistent value proposition for customers (Adner, 2012). The needs for coordination can differ depending on the localization of interdependencies within the flow of activities supported by the ecosystem. Scholars usually distinguish upstream interdependencies, with suppliers, from downstream interdependencies, with complementors (Brandenburger and Nalebuff, 1995; Adner and Kapoor, 2010). While suppliers are supposed to provide components that are integrated by the focal firm into its own products, complementors provide products or services that can be purchased independently from a focal firm's product and that can be integrated (or not) by the customer (Adner and Kapoor, 2010). Moreover, Jacobides et al. (2018), which frame upstream interdependencies as "complementarities", precise that they can unfold either in production, when some degree of coordination during the production or development of two products has beneficial effects, or in consumption, when using two products jointly has beneficial effects from a customer point of view. Both Upstream and downstream interdependencies can pose critical issues and even more since interdependencies in business ecosystems are recognized to be multilateral, which means that they can't be treated each in isolation from the others (Adner, 2017; Jacobides et al., 2018). But the specific case of downstream interdependencies arouses much interest from ecosystem scholars since they can be hard to identify until a focal innovation reaches its market and fail because of a lake of coordination with complementors (Kapoor, 2013). Successful ecosystems are thus usually characterized by specific mechanisms dedicated to the management of these interdependences, such as the sharing of common sets of resources, whether tangible or intangible (Gawer et Cusumano, 2014; Iansiti et Levien, 2004; Jacobides et al., 2006; Chesbrough, 2006; Dhanaraj and Parkhe, 2006; Clarysse et al., 2014).

Moreover, beyond determining the ecosystem members' collective ability to create value, these interdependencies also affect their individual ability to capture some share of this value (Adner et Kapoor, 2010). The mechanisms dedicated to the management of interdependencies thus

also must take into account an "adoption risk", referring to the need to ensure that each member is satisfied with its role in the ecosystem and its opportunities for value capture (Adner, 2012). According to the structuralist lens (Adner, 2017), ecosystem members respective opportunities for value capture are strongly determined by their "position" within it, understood as their location in the overall flows of activities. An actor can, for instance, position on bottleneck activities, which constrain the performance of its partners' activities and thus increase its power over them and its opportunities for value capture (Jacobides et al., 2006, 2018; Hannah et Eisenhardt, 2018). In order to apprehend actors position, Adner (2017) proposes to understand ecosystems as "alignment structures" determining specific patterns governing the repartition of activities among ecosystem actors and the flows between these activities, and since then, a specific equilibrium, at the ecosystem level, between collective value creation and individual value capture. An ecosystem strategy can then be understood as the way by which a focal actor envisions both how partners must align – determining the structure of their interdependencies, and their respective position – and mechanisms to secure its own role in the ecosystem (Adner, 2017).

1.1.2. Assessing the structure of ecosystem interdependencies

So far, the strategic management literature does not provide us with a consolidated template to assess the changes in the structure of these interdependencies nor to assess ecosystem members' position. Such a template would have to account both for value creation issues and for value capture issues. However, the literature offers interesting views on which to build one. To do so, we then retained two criteria: 1) the directionality of interdependencies; and 2) their intensity.

First, since Adner (2017) assumes positions to be relative to some "flows" of activities, we took the "directionality" of interdependencies as a first criterion to assess them. The typology brought by Thompson in 1967 looks particularly interesting to approach the directionality of interdependencies. Indeed, he identified three key "regimes" of interdependencies: 1) Pooled interdependencies occur when two tasks are interdependent but can be carried out in parallel; 2) Sequential interdependencies occur when a task must be carried out before another can be. For a given focal activity, we call such interdependency "outcoming", when this activity must be executed first, but "incoming", when another activity must be executed first; and 3) Reciprocal interdependencies occur when two tasks must be carried out simultaneously and require a high level of coordination. These four possible directionalities (pooled, outcoming,

incoming and reciprocal) give us a good overlook at the possible directions of technological interdependencies by accounting for how the design activities for each technology relate. In that sense, they account for the specific constraints brought by these interdependencies in terms of value creation.

The second criterion that we retained concern the "intensity" of the interdependencies. In order to account for this criterion, we adapted the recent typology built by Jacobides et al. (2018) to assess if a given business group can be characterized as an ecosystem or not. In their typology they identify three types of complementarities: 1) "Unique" complementarities occur when A needs B to function; 2) "Generic" complementarities occur when A needs B, but B is generic and easy to access; and 3) "Edgeworth" complementarities are characterized by a situation where the more A, the more B. However, in their typology, the Edgeworth complementarities undermine some specific assumptions regarding the directionality of interdependencies, since they assume interdependencies to be pooled. Thus, we prefer to use the notion of "continuous interdependencies" which appears less connoted but well illustrates the logic according to which two technologies are interdependent not only to work, but also to improve continuously. Moreover, for generic interdependencies, they seem to assume that a resource is generic or is not. As for us, we will consider in what follows that a resource can be more or less generic, thanks to varying degrees of standardization for instance. Beyond value creation, the intensity of interdependencies appears as a great criterion to account for value capture issues as we will discuss latter.

Figure 1: A typology of interdependencies

Directionality			Generic	Unique	Continuous
		Pool ($P = f[A, B]$)	The system require A and B but they can be developed in parallel and A and/or B is more or less generic	The system require A and B which can be developed in parallel	Improvments in A or B can occur in parallel and improve each other
	Sequential	Outcoming ($P = f[A \leftarrow B]$)	B must be developed to develop A, but A is more or less generic	B must be developed to develop A	Improvments in A require Improvments in B
	Sequential	Incoming ($P = f[A \rightarrow B]$)	A must be developed to develop B, but A is more or less generic	A must be developed to develop B	Improvments in B require improvments in A
		Reciprocal ($P = f[A \leftrightarrow B]$)	A and B must be developed simultaneously but one or both of them is more or less generic	A and B must be developed simultaneously	A and B must improve simultaneously
			Intensity		

Directionality and intensity appear as two complementary criteria that allow to assess technology interdependencies and their impacts on actors' position within ecosystems. The following matrix represent the different kind of technological interdependencies according to these two criteria. It also gives some simple-sentence examples to illustrate the intercepts between the different directionalities and intensities.

Ecosystem scholars usually recognize product modularity as a critical facilitator to manage technology interdependencies and distributed innovation within ecosystems (Iansiti et Levien, 2004; Moore, 2006). Jacobides et al. (2018) even stated that "modularity creates the conditions for an ecosystem to emerge" (Jacobides et al., 2018, p.2260). These statements echo deeper debates on the links between product architecture and the organization of innovation activities as discussed in the following section.

1.2. MODULARITY AS A LEVER TO MANAGE TECHNOLOGY INTERDEPENDENCIES.

1.2.1. Modularity as a pattern for product architecture design.

First of all, modularity stands as a specific pattern for product architecture design. Baldwin (2014) defines the architecture of a product by stating that it "defines its components, describes the interfaces between components and specifies ways of testing performance" (Baldwin, 2014, p.3). This definition mainly approaches product architecture through a material dimension: the organization of components and their interactions. However, beyond this material dimension, Ulrich (1995) points out that a product can also be described by its function, which can itself be divided into functional elements, according to different levels of abstraction. He then defines a product architecture as follows:

> *"The architecture of a product is the scheme by which the function of the product is allocated to its physical components. I define product architecture more precisely as: (1) the arrangement of functional elements; (2) the mapping from functional elements to physical components; (3) the specification of the interfaces among interacting physical components" (Ulrich, 1995, p.420)*

In the same vein, Ethiraj and Posen (2013) consider that "the architecture of a product includes both (1) the grouping of elements or functions in the components and (2) the mapping of the interdependencies between these components" (Ethiraj and Posen, 2013). These definitions approach the architecture of a product through a more functional dimension. These two dimensions of product architectures, material and functional, do not appear similar nor contradictory, but rather complementary and an architecture can therefore be characterized as

"the synthesis of the form in response to function" (Fjeldstad et al., 2012, p.735). In the design literature these two dimensions are also well identified, especially when authors distinguish the "functional domain" from the "physical domain" (Suh, 2001).

For a given system various architectures can be considered (Ulrich, 1995). However, authors generally distinguish between two major types of architectures, integral and modular, that must be understood as two extreme theoretical patterns for architecture design (Ulrich 1995; Sanchez and Mahoney, 1996; Baldwin and Clark, 2000). Integral architectures are characterized by strong interdependencies between the system's components due to the allocation of each functional elements on several components. Thereby, a change in one component affects the entire system and the system can't work with one or more missing components. Moreover, this pattern assumes strongly coupled interfaces between components that prevent them from being interoperable and interchangeable. Conversely, modularization consists in gathering highly interdependent components into subsystems, the modules, and to ensure that interdependencies between modules are the lowest possible, especially by allocating each functional element into unique modules. Moreover, modular architectures assume standardized and decoupled interfaces between modules that allows them to be interoperable and interchangeable. Thereby, changes in one module do not affect the others and some modules can be added to or removed from the system without perturbing its operations.

1.2.2. The impact of modularity over the directionality and intensity of interdependencies

Product modularization thus appears as a design strategy whose main objective is to better manage interdependencies between product components and many scholars investigated the links between modularity and organizational structures. Their works led to the formulation of the "mirroring hypothesis", stating that it tends to be a symmetry between the architecture of a product and the organization of the teams that develop it (Sanchez et Mahoney, 1996; Baldwin & Clark, 2000). Indeed, as it defines the interdependencies between the product components, the product architecture determines the flows of information that must be managed across the different teams working on the product. Thereby, it constitutes an "informational structure" (Sanchez et Mahoney, 1996) on which teams' boundaries and activities usually align in order to better manage their interactions (Baldwin & Clark, 2000).

Since modularity aims at reducing interdependences between modules, it allows teams to reduce the necessary transfers between them, and then decrease coordination needs (Baldwin & Clark, 2000) and their associated costs (Langlois, 2002; Baldwin, 2007). Nevertheless, these

organizational virtues of modularity require the interfaces between the modules to be standardized enough to clearly and unambiguously define the nature and the intensity of these transfers (Fixson & park, 2008). Scholars thus often state that in modular architectures interfaces between modules embed actors coordination, especially by publicly communicating some rules to guide the design of the complementary modules (Baldwin & Clark, 2000).Thereby, in inter-organizational innovation contexts, since the interfaces between modules are well defined and commonly known, each actor can specializes on a specific module and run its activities without caring about what happens in the others (Moore, 2006). Thus, modular architectures support a loose coupling of ecosystem's members, allowing them both to follow their own specific innovation trajectory and to stay consistent with the overall innovation trajectory of the ecosystem (Orton & Weick, 1990; Iansiti et Levien, 2004; Brusoni & Prenicpe, 2013). In such context, innovation processes at the module level can be achieved in parallel, which leads to faster innovation paths but also to better innovation outcomes. In other words, modularization can help firms to turn reciprocal or sequential interdependencies into pooled interdependencies following the Thompson (1967) typology. Moreover, as modularity support module interchangeability and interoperability, many actors can work on a same module concurrently, which can lead to the emergence of vast innovation networks (Chesbrough et Prencipe, 2008). Thus, modularity can be a good strategy to increase solution diversity and to solve technical issues faster since, for a same module, many actors can explore in parallel various technical solutions based on different technologies (Brusoni and Prencipe, 2007, 2013). Modularity thus appears as a way to better manage technology interdependencies by influencing their directionality.

Moreover beyond influencing their directionality, modularity can also serve as a basis for strategies that aims at turning the intensity of technology interdependences. Those strategies mainly rely on the isolation of specific technologies within independent self-contained modules. For instance, if it succeeds in isolating a complementary technology within a module, a focal firm can push for its standardization in order to turn its unique incoming interdependencies into more generic ones (Baldwin, 2014; Baldwin and Henkel, 2015). Ulrich (1995) even considers that modularization is a critical facilitating condition for standardization. Through such a strategy the focal firm can improve dramatically its position since its complementary assets become more mobile thanks to standardization (Jacobides et al., 2006). Contrarily, this firm can also isolate its own technologies within a single module in order to protect them and to reinforce its unique outcoming interdependencies to increase its power and

influence over its partners (Baldwin and Clark, 2004; Jacobides et al., 2006). In the same vein, if it succeeds in gathering, in a single self-contained module, a consistent set of technologies that are necessary for a wide variety of niche applications, it can turn its unique interdependencies into continuous outcoming ones which can constitute a great "architectural advantage" as called by Jacobides et al. (2006). Such an advantage can increase significantly the share of value the firm is able to capture. This logic is the same as that behind platform strategies, which are known both to accelerate innovation dramatically and to confer to the platform leader unrivaled opportunities for value capture and control over its ecosystem (Iansiti et Levien, 2004; Gawer and Cusumano, 2014).

Thus, it seems that modularization can be a powerful lever to better manage technology interdependencies by both influencing their directionality and their intensity. Since then, according to Baldwin (2014) the design of product architecture must rely on a trade-off between technical, organizational and strategic matters. In the context of business ecosystems, this trade-off unveils through the necessary articulation of value creation and value capture mechanisms. However, so far, management scholars were mainly interested in either technical or organizational matters when studying product architectures. Moreover, their studies usually focus on outsourcing relationships, and investigate how modularity can optimize them from a transaction cost perspective (Sako, 2002). At time, only a few contributions address the strategic implications of product architecture design (Jacobides et al., 2006; Baldwin, 2014; Baldwin et Henkel, 2015). Also, Baldwin (2007) points that "while modularity is always an option, it is not always a good option" (Baldwin, 2007, p.177). Therefore, our question arises: **How can modularity help a firm to reconfigure the structure of its ecosystem interdependencies ?** To address this question we studied the evolutions of the electronic and software architectures of one of the leading global automotive OEM and analyzed their impact over the structure of its ecosystem technology interdependencies. The following section gives some detail about the data and methodology we used, but also introduce this empirical context.

2. Research design

This work is part of a research program started in December 2018 in collaboration with a subsidiary of one of the leading global car manufacturers. This subsidiary gathers about 400 software engineers (including an important part of software architects) which are responsible for the design of the vehicle's software for this OEM. Therefore, we will refer to it as "the software subsidiary". This subsidiary was created in 2017 through the acquisition of software

research centers from one of the leading firms in the sector of embedded electronics. The objective of this acquisition for the OEM lies in increasing its involvement in the development of its vehicles' software. The key goal of this research program is to understand the role played by product architectures in the reconfiguration of business ecosystems. This context allowed me to interact closely with these engineers and to access to an important variety of data.

This contribution builds on an longitudinal single case study relying on qualitative data. It focuses on the evolution of this OEM's Electric & Electronic (E/E) architectures from 1984 to 2020. This research approach finds strong rationales in the nature of our research question. Indeed we do not aim at testing a theory, but rather at extending a theory by exploring its implications and limits in a specific context. Thus, our goal is to offer new conceptual articulations and hypotheses in order to enrich our theoretical understanding of the role of modularity in the reconfiguration of ecosystem interdependencies. To this end, a case study seems entirely appropriate. The case study method is ideal for confronting contradictory, even paradoxical observations, in order to emphasize and go beyond the limits of existing theories and to propose new theories and / or hypotheses (Eisenhardt, 1989). On the other hand, this contribution focuses on the study of current transformations, observed without any control over the environment and thus meets all three criteria, which, according to Yin (2009), found the relevance of using a case study.

2.1.Data collection and analysis

In order to build our case study, we get access to an important variety of data. First we collected technical data used by the OEM's engineers to design the E/E architecture and the vehicle's software, including : technical documentation describing the vehicle E/E architectures; an application used by software architects that gathers all the information regarding the vehicle's ECUs, their interactions and their functional roles within the observed architectures; internal presentation usually focusing on specific issues related to architectural design and/or software design (including both technical, organizational and/or strategic issues). We also conducted 17 free interviews (including 11 recorded interviews) with software architects involved in the design of the new E/E architecture. These interviews allowed us to better understand the technical issues related to the E/E architecture and the changes brought by the new architecture. Finally, we participated to many working sessions with these engineers, spread over a period of 14 months, during which we adopted a participant-observation posture. In these working sessions have usually been discussed either technical and/or strategic issues related to the

coming new E/E architecture. They allowed us to constitute an important collection of field notes.

Consistent with our research question, the analysis of these data pursued two key objectives. First, understanding the changes in the vehicles' E/E architecture over the period, and second, understanding the implications of these changes on the structure of the OEM's ecosystem interdependencies. On the basis of the technical data we get access to, completed by the free interviews we conducted, we identified the key changes between the architectures over the period. We then compared these architectures one the basis of two key dimensions: their material dimension, materialized by the Electronic Control Units (ECUs) that compose the E/E architectures, and their functional dimension, which can be associated with their software architecture. Once we identified the key changes regarding the E/E architectures, we used our theoretical framework and the typology of interdependencies we proposed to analyze the consequences of these changes over the structure of technology interdependencies. To do so, we mobilized both the presentations dealing with strategic and organizational issues we get access to, our field notes and the records of the interviews we conducted. Altogether, they allowed me to identify the evolving structure of technology interdependencies between the ecosystem actors all along the period we observed.

All along this analysis process we paid particular attention to the triangulation of the data, comparing primary data (interviews and field notes) with technical or archival data. Moreover, we submitted a first version of this contribution to two engineers involved in the design of the E/E architectures to ensure our analysis to be consistent. The following section introduces our case by detailing the evolution of the E/E and software architectures that occurred over the period we observed.

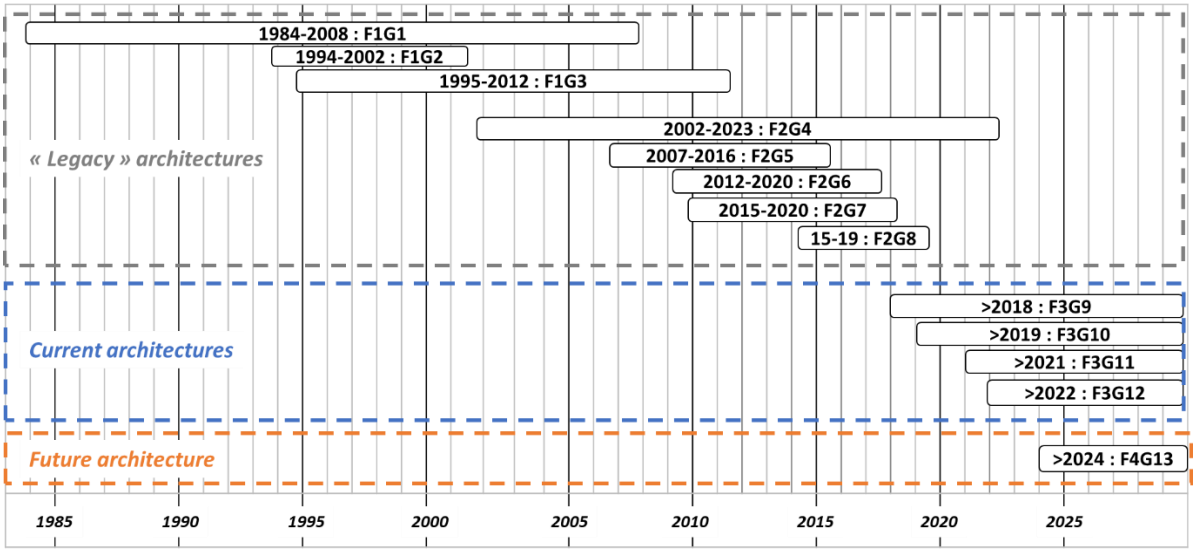
2.2. Case presentation: the evolution of the OEM's E/E and software architectures

A vehicle's E/E architecture gathers all the electrical and electronic components embedded in vehicles, including calculators (or "ECU" for "electronic control unit"), which are electronic modules that manage the various digital functions of vehicles. A vehicle's E/E architecture therefore describe how these modules interact, through flows of energy or data, to achieve the functions of a vehicle. ECUs can be approached through two dimensions: (1) their hardware dimension brings together all the material electronic components needed for ECUs to perform their functions; and (2) their software dimension determines how electronic components must

behave and interact to perform their functions within the system. Thus, software can be approached as the functional dimension of these ECUs.

Over the period we observed, the OEM's E/E architectures have improved along 13 main generations. Among these 13 generations, we can distinguish 4 families bringing together generations whose architectural design principles are similar. The figure 2 illustrate through a chronological view the evolution of these architectures. Within a family, the generations mainly differentiate by modular innovations, while across families they mainly differentiate by architectural innovation. The two first families gather the architecture designed before the creation the software subsidiary, between 1984 and 2015. In the following we will refer to these architectures as "legacy architectures". Since the software subsidiary was created, the evolution of the OEM's E/E architectures are guided by dedicated roadmaps so called the "SoftWare Electric and Electronic Technologies" (SWEET) roadmaps. These roadmaps define the third and fourth families we identified.

Figure 2: Chronological view of the successive generations of E/E architectures.[1]



The third family gathers the architectures developed since 2017 and that are currently used in the design of the OEM's new vehicles. These architectures stand as improved versions of the legacy architectures but introduce some foundational concepts for the future architecture. The fourth family correspond to a disruptive architecture that is being developed since 2015 and will be used in the OEM's vehicles by 2024. In the following, we introduce the legacy architectures (family 1 and 2), the current architectures (family 3) and the disruptive

[1] *In this figure, we named each architectures by referring to its family (F1, F2, F3, F4) and to its generation (G1, G2, G3, etc…). The periods we consider refer to the years during which each architecture was used for the design of vehicles. They do not take into account the development of the architectures (for instance F4G13 will be used in vehicles by 2024, but engineers are working with partners on its development since 2015).*

architecture (family 4). Finaly, we also introduce the evolution regarding the software architecture across these four families.

2.2.1. The legacy E/E architectures: Distributed architectures.

Since 1980's, to face the rapid increase of vehicles' electronic features requested by both customers and regulators, the OEM bet on the design of modular E/E architectures where each ECUs is considered as a self-contained integrated part. Following this logic each ECU is designed as an independent module dedicated to the implementation of a specific feature and within which hardware and software are strongly coupled. Thus, ECUs are designed to perform their specific features as independently as possible. This means that they must be able to collect data coming from vehicle's sensors, to compute these data, take decisions and then to act on other vehicle mechanic and/or electronic parts – such as the engine, brakes, steering wheel, screens or lights for instance – through electronic actuators. Each of them is thus wired to a dedicated set of sensors and actuators managed by coupled interfaces which ensure the good circulation of data and energy flows. Each of the ECUs thus concentrates a part of the vehicle intelligence, bounded in its software, and the dominant logic of ECUs design thus lies on optimizing each ECU local performance in running their own dedicated set of functions. Such architectures can thus be considered as distributed modular architectures.

However, while each ECU is supposed to collect and process its own data, some exchanges between them can be necessary, especially when it comes to run complex features that imply to compute many data or act on several parts of the vehicle. Ensuring the good circulation of data, energy and signals between ECUs implies them to be connected. The key evolution between the first and second family of legacy architectures lies in the ECUs connection and interfaces. In the first family ECUs are wired on a one-to-one basis and interact through coupled interfaces. While in the first generation of architectures in this family, the wiring is kept to a minimum, almost every ECUs are connected in the second generation in order to better account for possible interactions between them as well as the increasing complexity of functions. In the second family of legacy architectures, every ECUs are connected through a common "bus network". In electronics, a "bus" is a scalable wiring network that connect every electronic component and thanks to which they can communicate altogether following some rules and protocols. In our case, the ECUs interact through this common network using the CAN protocol, which is a standardized protocol allowing the ECUs to communicate in turn. The key change brought by this second family of architectures thus lies in the standardization of

communication and interfaces between the modules. Within this family, changes across the five generations mainly consist in the addition of new ECUs.

2.2.2. The current E/E architectures: Centralization by domain

The third family we identified gathers E/E architectures that are currently used for the design of the OEM's new vehicles. Within this family, the architectures largely build on the legacy architectures but with more than twice as much ECUs. To deal with this unprecedented increase in the number of ECUs, the OEM has had to introduce some new architectural design concepts, whose goal is to centralize the vehicle's functions by domain. These evolutions were operated in two times corresponding to the first and second generation of architectures in this family.

The first generation had to deal with the introduction of 51 new ECUs within the architecture, which came along with an impressive increase regarding the flows of data exchanged between the ECUs. In this architecture, these flows are managed through different bus networks connecting the ECUs by "domains". A domain gathers ECUs whose functions are linked and who interact a lot. Five domains were defined by the OEM: 1) Powertrain; 2) chassis; 3) ADAS (Advanced Driver Assistance Systems); 4) Body; and 5) Multimedia/Connectivity. In this architecture, one or more bus networks are dedicated to each of these domains. Moreover, as some ECUs wired on a specific network could sometimes need data coming from ECUs or components wired on another network, a "central gateway" ensure the connection between them. The central gateway is an ECU that appeared in this architecture and whose role is to manage the flows of data and to redirect them across the networks to ensure each flow find its target. Thus, almost every domain network is connected to the central gateway which centralize data flows among them.

The second step towards domain centralization correspond to the introduction of "domain masters" (also called Domain Control Unit) in the second generation of this family. These domain masters are ECUs dedicated to the management and synchronization of the overall functions corresponding to a specific domain. This architecture embeds five domain masters (one per domain), which manage the flows of data within their domain networks but also the activities of the ECUs wired within these networks. They are responsible for the consolidation of features within the domain and centralize their implementation. To do so, they benefit from better technical characteristics than other ECUs and embed specific software architectures as we will describe latter. The introduction of these domain master thus led to a more hierarchical

architecture where the legacy ECUs are subordinate to the domain masters, which centralize data flows and the vehicles' feature by domain.

2.2.3. The future E/E architecture: Centralization at the vehicle level

The fourth family we identified correspond to an architecture that is being developed since 2015 and will be used to design the OEM's vehicle by 2024. This new architecture relies on more radical changes combining both architectural and module innovation. This architecture is being designed around two new types of module.

First, this architecture is built around a central calculator, so called "Physical Computing Unit" (PCU) whose role is to centralize the implementation of the whole set of vehicle's functions. Thus, the PCU has to provide the vehicle with both the whole processing capabilities, computing memory and storage memory needed to implement the entire set of the vehicle's features. Therefore, it must be able to run many complex operations in parallel at the same time. To do so, the PCU is designed as a high performing computing unit that concentrates, a large part of computing capabilities previously distributed among the different ECUs. Its inner hardware architecture relies on a highly optimized integrated design based on high performing "System on a Chip". A system on a chip (SoC) is an integrated circuit that brings together the main components of a computer – such as processors, graphical processors, computing and storage memory, communication modules, sensors, (etc…) – on a single electronic board. This technology allows to gather computing resources within a single integrated component whose performances are optimized.

Second, this architecture includes calculators whose role consists in ensuring the interface between the physical world and the digital world. These so called "physical Interface Units" (PIU) are directly connected with the vehicle's sensors and actuators and are mainly dedicated to the management of the data and signals flowing between the sensors/actuators and the PCU. Sensors turn the measures of a physical phenomenon, such as speed or temperature, into exploitable signals such as electric pulses. Conversely, actuators trigger physical phenomenon, such as the rotation of a fan or the lighting of a headlight, on the basis of signals such as electric pulses. PIUs, meanwhile, both ensure the translation of signals into digital data, to provide the PCU with computable information, and the translation of digital data into physical signals, to trigger vehicle behavior based on PCU decisions. As for those of the computing unit, the PIUs capabilities are, in the legacy architectures, distributed among the ECUs. In the new architecture, three to five interface units, depending on vehicle model, are each responsible for

a specific subset of sensors and actuators and are distributed over different areas of the vehicle so that each one is as close as possible to the sensors and actuators it is responsible for. In this sense these PIUs are said “zonal”. Moreover, for the first generation of the new architecture, the choice to keep some legacy ECUs has been made. The role of these legacy ECUs mainly consists in the management of some specific sensors and actuators. PIUs are thus also responsible for the management of these legacy ECUs and for the circulation of the data coming from them.

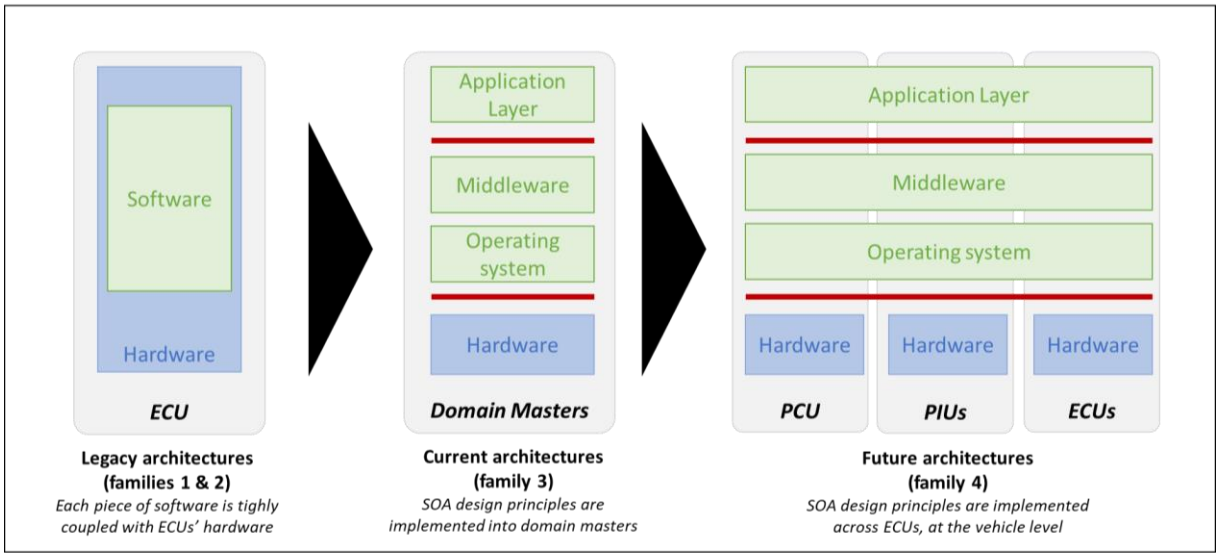
This architecture thus stands as a highly hierarchical architecture where the PCU centralize decision making and vehicle control while PIUs are responsible for the circulation of data and legacy ECUs to the management of sensors and actuators. Finally, to ensure their connection to be good, fast and secure enough, the PCU and the PIUs communicate using an ethernet protocol. The use of ethernet protocol thus constitute a third critical hardware change coming up with this new E/E architecture, since it supports the circulation of more sophisticated data (such as real time data, videos, etc…) between the PCU and the PIUs.

2.2.4. The software architecture: Towards a Service Oriented Architecture.

A key objective of the SWEET roadmaps lies in revising completely the relationship between hardware and software. In the legacy architectures every ECU embeds its own, self-contained, piece of software which is tightly coupled with its hardware components. Even if these pieces of software can interact and exchange data, they are not designed as a whole at the vehicle level. SWEET roadmaps aim to remedy this by organizing the progressive consolidation of a consistent and scalable software architecture at the vehicle level, based on the application of “Service Oriented Architecture” design principles. SOA is a specific type of software architectures – coming from the mobile industry – whose design logic relies on the decomposition of the whole software into discrete modular software components called “software services” which must be independent, self-contained and lead to specific outcomes. The application of such principles to automotive software led the OEM to consider its vehicles’ software architecture through 3 different software “layers”. Moreover, these principles were introduced progressively in the third and fourth family of architecture we identified. In the third family, they were used to design the software embedded in two domain masters ECUs (Multimedia & Connectivity and ADAS) in order to consolidate software architectures by domain. In the fourth family they are used across the entire set of ECUs to build a unique

software architecture for the vehicles. The figure 3 offers a schematic representation of these changes.

Figure 3: Schematic view of the evolutions of software across the four families



The first layer, so called "operating system" builds on top of the hardware components to define some sets of standardized rules and specifications for designing and running the software services. These rules include the definition of some basic software modules that support the development of software services by defining the interfaces between them, standard communication protocols and some methodologies to develop them. This operating system is built around three software standards developed for the automotive industry : Autosar Classic , appeared in 2003 and used in the majority of automotive software applications; Autosar Adaptive, appeared in 2017 and targeting highly complex, dynamic and scalable applications; and the Linux standards, used to develop "infotainment" applications. All together, these different standards constitute a common infrastructure that allows the vehicle software services to access the relevant physical resources they need in order to run. In other words, they define how the software services can leverage physical resources from hardware parts (such as memory, computing power, connectivity, etc…). To this end, the operating system exposes a set of common Application Programming Interfaces (API) that standardize the access to the physical resources for the upper software layers. The key interest of such an operating system therefore lies on the decoupling of the upper software layers from the hardware components used. Moreover, this operating system, also embed a configuration interface that allows it to adapt to different hardware components.

The second layer, so called "middleware" brings together all the functional "services" the vehicle can implement. These “services” are discrete modular software components that manage more or less simple standardized actions of the vehicle. Within this layer, two types of services are distinguished. First, the “basic services” correspond to the basic actions of the electronic and mechanical components of the vehicle. Taking the example of a headlight, a basic service could be a software component controlling the ignition of the right front headlight bulb. Each of these services therefore only concerns specific and localized components of the vehicle. Second, “composed services” correspond to more complex actions which may require acting on different parts of the vehicle. To do so, they leverage different basic services at the same time. In that sense they can be understood as vehicle-level functions. Taking the example of distress signals, a composed service would both have to flash all the headlight bulb (basic services), and to beep in the passenger compartment (basic service). In order to ensure the proper functioning of these services, the middleware also brings together a set of "platform management services", dedicated to the coordination of these different services between them.

Finally, the third layer, called "Application layer", brings together all the vehicle's applications. The applications correspond to the operational characteristics of the vehicle, which consist in a specific behavior in response to a context or to a trigger. Applications are therefore the software components which add value to the experience of using the vehicle and which are directly appreciated by the user. Their objective is to answer a need related to the use of the vehicle. In this architecture, an application is designed as the combination of different basic and/or composed services. By mobilizing these different services, the application acts simultaneously on different parts of the vehicle. To ensure applications can access to the services they need to mobilize, the applicative layer and the middleware are linked by a standardized interface called “Software Development Kit” (SDK) and which exposes the interfaces of the services and provides a set of tools, methods, protocols and documentation to develop to combine them and develop applications.

Through the implementation of SOA principles, the OEM objectives are no longer to associate a specific and unique functionality with each ECU. Rather, the goal is to allow the vehicle software to run as a coherent whole abstracted from its hardware while ensuring a proper allocation of the vehicle's electronic resources according to the needs of each software module.

3. Results: changes in the structure of technology interdependencies.

In the following we present our results by detailing the structure of technology interdependencies implied by the legacy, the current and the future architectures.

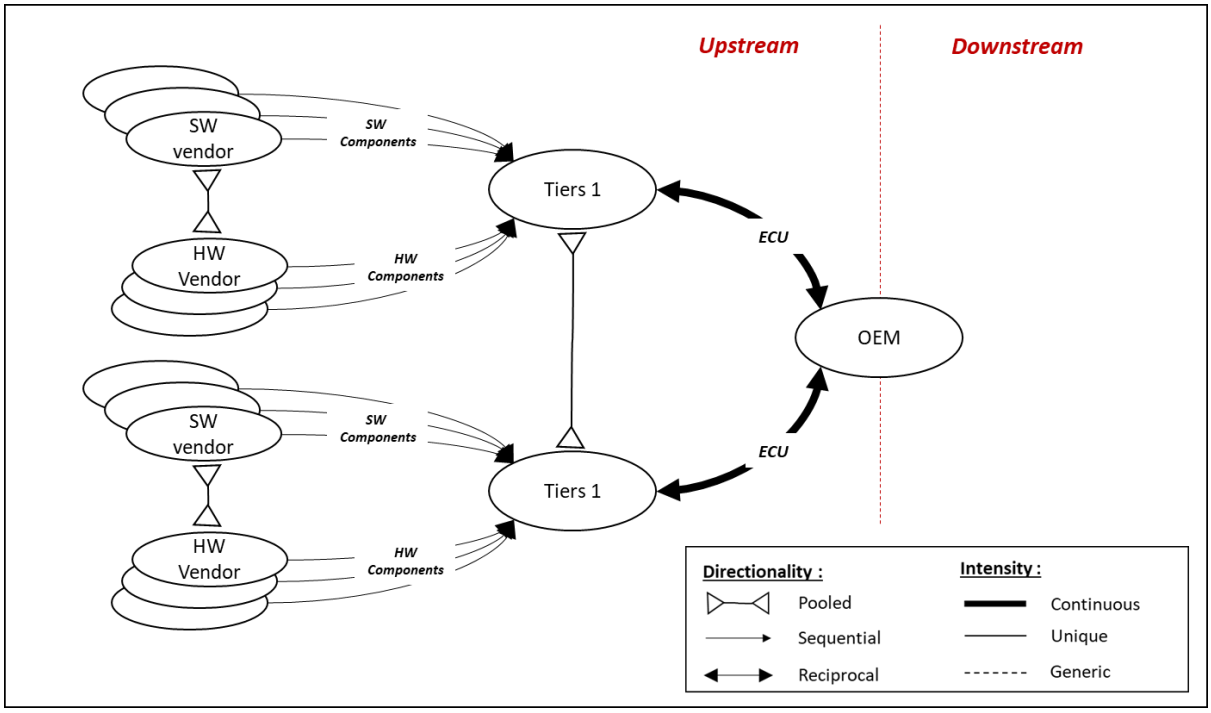
3.1. The structure of interdependencies with the legacy architectures.

For many years the modular design of the legacy architecture has allowed the OEM to externalize the design, development and production of every ECUs to direct suppliers, also called Tiers 1, while keeping architectural design activities in-house. This strategy led to growing reciprocal interdependencies between the OEM and its Tiers 1 suppliers. Indeed, since it has kept no competences in module design and development in-house, the OEM must call for its suppliers each time he needs to implement a new feature in its vehicles or to improve an existing one. But each OEM has its own E/E architectures and usually sets its own specific requirements for a given feature. Tiers 1 thus cannot design standard solution for a given ECU or features. Consequently the development of a new feature almost every time consist in developing custom solutions where the OEM specifies the role and interaction of the module within the architecture and lets the Tiers 1 design the solution as a "black box", or in other words, without caring about the inner functioning of the solution. Moreover, as the OEM usually asks for optimized module performance and does not care about the inner architecture of the solution, the tiers 1 suppliers are used to design highly integrated solutions that struggle to absorb any inner change. In this context, the implementation of a new feature often implies either to develop a new module or to redesign entirely an existing on. In other words, interdependencies between the OEM and its Tiers 1 are continuous since they have to collaborate for any digital improvement within the modules. Nevertheless, since the OEM endorses the system level design and integration activities, the different Tiers 1 do not have to care about the interdependencies between the modules they develop respectively. So, they profit from pooled unique interdependencies between them. Finally, the OEM's Tiers 1 suppliers, meanwhile, usually buy both the module's hardware and software components to specialized suppliers, also called Tiers 2. They thus depend from these Tiers 2 to develop the architecture of modules. Nevertheless, this dependence is not reciprocal but rather sequential, since Tiers 1 usually source from Tier 2 their usual products. The figure 3 synthesizes interdependencies between these actors. The figure 4 illustrate the structure of interdependencies implied by the legacy architectures.

In light of the current changes in its ecosystem, this structure of interdependencies has significant disadvantages for the OEM. First it implies to contract with a dedicated Tiers 1 for the integration of each ECU, leading the OEM to deal with more than 120 different ECU

suppliers today. Moreover, since it strongly depends from Tiers 1 suppliers to integrate the different technologies within the ECUs, the OEM cannot deal directly with the developers of these technologies, the Tiers 2 suppliers or even complementors, without involving Tiers 1 suppliers. This bottleneck status of Tiers 1 suppliers largely impedes the power of the OEM to influence innovation trajectories in component technologies or complementary technologies. In the same way, both the central role of Tiers 1 and the high integration of software and hardware within ECUs, prevent the OEM to evolve them incrementally. Indeed, for every new feature it is necessary to re-design entirely one ECU's software and hardware or to design a new one in the architecture. This is also true when it comes to develop new interactions with external systems. Therefore, in this "legacy" structure of interdependencies, its position prevents the OEM from effectively managing the new interdependencies between automotive and digital technologies.

Figure 4: The structure of interdependencies with the legacy architectures



3.2. The structure of interdependencies with the current architectures

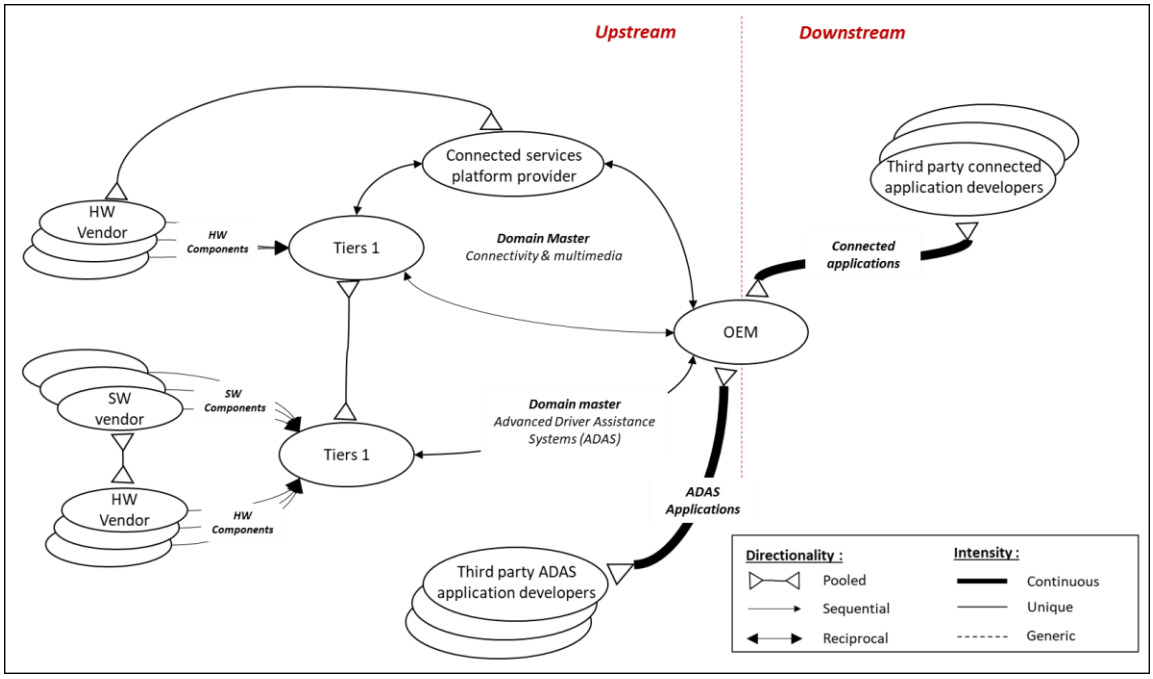
The current architecture tends to solve these issues, partly at least. Indeed, the design of service-oriented architectures for the Domain Masters' software allows the OEM to implement some new digital feature incrementally, without redesigning existing modules or designing new ones. This is particularly true for connected applications (Multimedia & Connectivity domain) and autonomous driving applications (ADAS domain). Nevertheless, the structure of interdependencies differ between this two domains.

Regarding connected applications, the OEM decided to build its software architecture on the basis of a platform provided by a complementor, which is currently the leader in the market of connected applications platforms. This platform covers the operating system layer, a part of the middleware and also provides some generic connected applications. But this platform has had to be adapted to the OEM's vehicles and to be integrated within the domain master. To do so, the design of this modules has been achieved through an intensive collaboration between the OEM, a Tiers 1 and this complementor, which were reciprocally interdependent for the design of the domain master. But once developed the features supported by this module can evolve incrementally. These interdependencies are thus unique, rather than continuous. Moreover, this third-party platform has determined the hardware used within the module, through design rules embedded in its operating system's standardized interfaces, leading to some pooled interdependencies between the platform owner and the hardware vendors (Tiers 2). Thanks to the SOA principles that guided the design of this Domain Master, connected applications can now be developed without redesigning this module. More precisely, the OEM can develop connected applications in-house and/or contract with third party connected application developers to release new connected applications on its vehicles. To develop them, third parties only have to comply with the standardized interfaces and use the Software Development Kit provided by the OEM. Moreover, the more complementors develop vehicles connected applications, the more the OEM's vehicle offers connected feature, the more their value increase for the customers. This SDK thus support pooled continuous interdependencies between the OEM and these complementor.

Regarding the Autonomous Driving domain, the same logic applies but with some key differences. First, the OEM decided to develop an in-house proprietary solution for the design of the module's software, without building on a third-party existing platform. It thus depends from Tiers 1 for module integration activities. But as for the connectivity Domain Master, once the module has been developed, its application can evolve incrementally, thanks to SOA principles. Interdependencies regarding the integration of this module are thus unique and the OEM can rely directly on third party ADAS applications developers to implement new features. Here lies the second key difference with the Connectivity domain. In the case of connected applications, the final customers are allowed to subscribe autonomously to new applications for their vehicle, directly through the platform embedded within the vehicle. Connected application developers are thus positioned as downstream complementors. In the case of Autonomous Driving Applications third party software developers can develop compatible applications independently, by complying with the standardized interfaces and

using the Software Development Kit. But their integration within the platform is managed by the OEM, which keep the power to decide what feature will be implemented within its vehicles. Here, as in the case of the connectivity domain, the interdependencies between the OEM and the third party application developers are pooled and continuous but the latter are positioned as upstream complementors. The following Figure 5 represent the structure of interdependencies implied by the current architectures.

Figure 5: The structure of interdependencies with the current architectures



To sum up, the current architectures allow the OEM to improve and release new connected and ADAS features in its vehicles with more agility and through shorter development cycles involving loose coupled third party application developer. But this does not apply to the Powertrain, Body and Chassis domains. Moreover, this architecture struggle to absorb features linking the ADAS and Connectivity domain since these two domains rely on different software platforms.

3.3. The structure of interdependencies with the future architectures.

While the legacy architecture constrains the OEM to redesign entirely its modules for every electronic or digital improvement, the future layered architecture is being designed to allow the OEM to decouple, at the vehicle level, applicative digital improvements from electronic ones and to manage them across different, loosely coupled innovation streams in its ecosystem. These new innovation streams correspond to the layers of the architecture and reshape entirely the structure of interdependencies in the ecosystem as described in the following.

The first innovation stream corresponds to innovations regarding hardware technologies, including both calculators and their basic software. Since the OEM stays responsible for the design and integration of the architecture and Tiers 1 responsible for the hardware integration, their interdependencies remain reciprocal. Nevertheless, as in this architecture, hardware and software are decoupled, these interdependencies turn into unique ones. In other words, the hardware layer has to be developed to enable digital features to run, of course, but improvement in vehicles' digital features do not imply systematically changes in the hardware layer, and conversely. This layer is designed to remain stable in the middle term at least and across vehicle ranges and models. Also, to ensure its scalability, the OEM engineers oversized the capabilities of this layer in order to ensure it would be able to absorb the rapid expansion of vehicles' features. In regard to Tiers 1 developing the different computing and interface units, their interdependencies remains unique and pooled, since the OEM manage the interfaces between modules during the architecture design and integration. Finally, the interdependencies between Tiers 1 and their suppliers remain the same as with the legacy architecture. The only difference in this relationship is that Tiers 1 only have to source basic software in this stream, since a large part of software innovation is managed in the other streams.

The second stream corresponds to the development of the software environment and operating system layer and the third corresponds to the development of the middleware. As for the hardware layer, the OEM remains dependent from Tiers 1 for the integration of these two layers which have to comply with custom requirements defined by the OEM. The OEM and the Tiers 1 thus remain reciprocally interdependent. But as for the hardware layer, since the software environment and the operating system are bounded by standardized interfaces their development are decoupled from those of the other layers. So, in both of these streams the interdependencies between the OEM and the Tiers 1 turn into unique instead of continuous. Nevertheless, a key change in both of these streams is that software components have to comply with standards such as Autosar classic or adaptive. The Tiers 1 thus benefit from relatively generic interdependencies with their Tiers 2 suppliers, while these Tiers 2 suffer from incoming continuous interdependencies from the consortium responsible for the development of these standards. Indeed, the improvements of their software components are constrained by the roadmaps of these consortiums, which are mainly driven by OEMs and Tiers 1 suppliers.

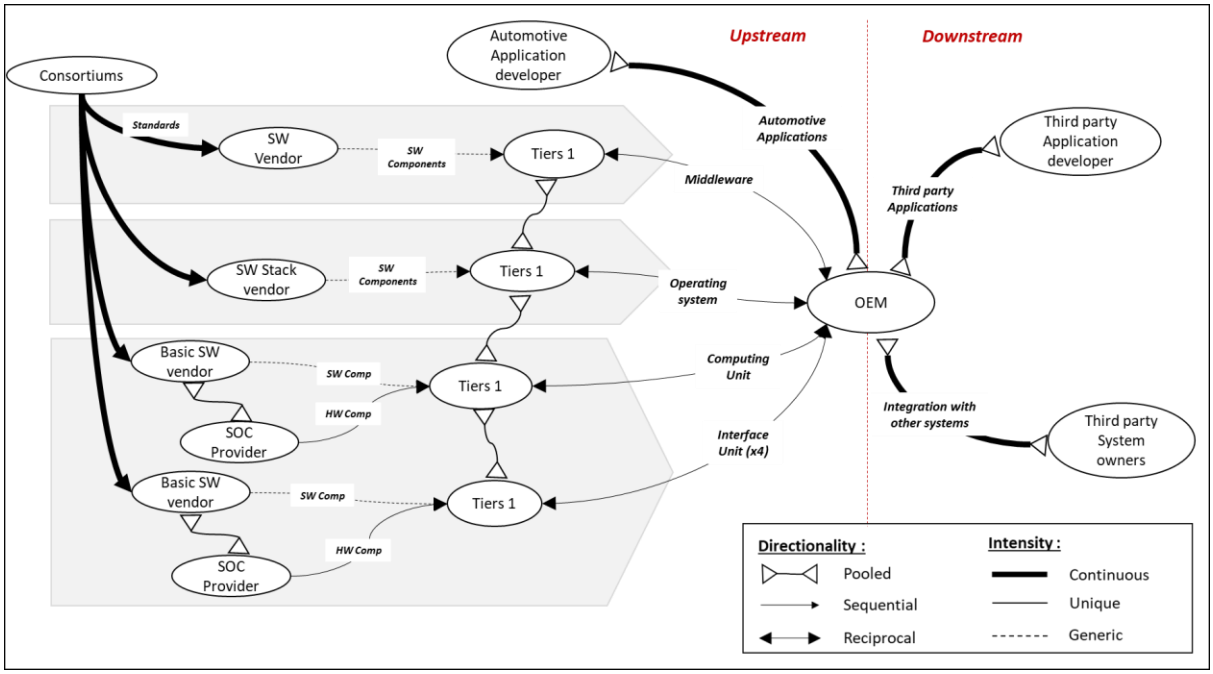
The last stream corresponds to the applicative layer of the architecture. Within this layer, application developers build on top of the OEM's Software Development Kit which provides them with the whole set of basic and composed services they can leverage in their applications.

Thanks to this standardized access to vehicle digital services, application developers can work independently from activities running in the other innovation streams, leading to pooled interdependencies between them and the OEM. Nevertheless, at least three sub-streams must be distinguished regarding innovation in this layer. The first concerns OEM-branded applications. They usually correspond to applications that are specific to the automotive industry and necessary to ensure the essential features of the vehicles, such as acceleration, braking, doors lock and unlock or headlights control, among many others. To develop them, the OEM either relies on suppliers specialized in automotive applications or on their own internal capabilities. In both cases these applications are owned by the OEM. But this new architecture also opens easier ways to collaborate with new third-party application developers. This constitute a second sub-stream of innovation regarding vehicle's applications. Taking the example of infotainment applications, this architecture allows software developer coming from the mobile or internet industry to easily adapt their applications – mainly through complying with the OEM's software APIs – so that they can run on vehicle screens for instance. Finally, the third sub-stream does not concern directly in-vehicle applications but rather the interactions between the vehicle's software and third-party digital systems. Smart cities monitoring systems, mobility operators' fleet management systems, or even any digital system dedicated to the management of road or city infrastructures – such as parking, toll stations or traffic lights for instance – can benefit from interacting with vehicles, and conversely. This architecture allows to establish easier connections with these systems by both providing a standardized interface to support these interactions but also by centralizing vehicle data and control. In these two last sub-streams both third party application developers and third-party system owner are positioned as complementor, not as suppliers. Whatever the sub-stream, the interdependencies between the OEM and application developers are pooled. Moreover, improvement in applications increase the value and/or performance of the vehicle and, conversely, improvement in lower layers of the architecture can open new opportunities to improve the applications. The OEM and the application developers thus benefit from continuous between them. The following figure 6 illustrates the structure of interdependencies implied by the future architectures.

This new structure of interdependencies allows the OEM to consider its innovation streams with different temporalities. The first three streams we described concern layers in the architecture that are supposed to remain relatively stable in time but also across vehicle models. While each of these layers would certainly know some incremental innovations, they will not

change radically before many years. Moreover, they will be able to improve independently from the others. Contrarily, the application layers will be able to evolve with more flexibility and on the basis of shorter innovation cycles. This will enable great and fast scalability of vehicles features across years and models. Thus, this new structure of interdependences will allows the OEM to better manage the technological trajectory of the ecosystem, including innovation in features and applicative field.

Figure 6: The structure of interdependencies with the future architectures



4. DISCUSSION

The case we explored shows that architectural innovation can be a powerful lever to influence the structure of technology interdependencies within ecosystems. Thereby, it helps to discuss both modularity theories and the ecosystem literature through two key contributions.

4.1. SILOED MODULARITY, LAYERED MODULARITY AND THE FUNCTIONAL TRANSPOSITION OPERATOR

First, our results extend our understanding of modularity. Assessing the modularity of a system is a tricky task. Authors already pointed that the path from integrality to modularity must be understood as a continuum on which a given architecture positions, rather than a gap which can be fully crossed (Simon, 1962; Ulrich, 1995; Baldwin et Clark, 2000). Thereby both

scholars, managers and engineers have to approach modularity in terms of degree rather than strongly opposing full modularity and full integrality as two reachable goals. Management scholars then pointed different criteria to assess the degree of modularity of a given architecture, such as the way functional elements are allocated to physical components (Ulrich, 1995), the interdependencies between modules (Baldwin and Clark, 2000) or the characteristics of interfaces between modules (Fixson and Park, 2008). An other way to assess the modularity of a system lies in the "purpose" of modularity. Indeed modularity can serves different objectives which can be framed as "modularity in design", when modularity aims at optimizing the design processes for a system, as "modularity in production", when modularity aims at optimizing the production processes for a given system, or as "modularity in use", when modularity aims at improving the evolvability, declinability and flexibility of a system for the customer (Baldwin et Clark, 2000; MacDuffie, 2013). These purposes are not necessarily mutually exclusive, but often leads to different design logic. The degree of modularity and the purpose of modularity can thus serve as two way to assess one system's modularity. But they struggle to characterize and explain the changes between the legacy, the current and the new architecture in our case. Indeed, both of the architecture we discussed looks highly modular, in terms of design and production at least.

Rather than on the degree or purpose of modularity, the two architectures we presented differ on the nature of their modularity. In light of our results we define the nature of modularity as the logic governing the way in which the functional elements of the architecture are gathered within its modules. Our results illustrate two different logics, that we frame as "Siloed modularity" and "Layered modularity". Siloed modularity characterizes an architecture where functional elements (the software components in our case) are allocated to physical components (the hardware parts in our case) in a way that ensure each module embeds all the functional elements needed to run a specific set of functions with (relative) independence from other modules. Conversely, with layered modularity each module gathers some related functional elements that, altogether, constitute a common set of resources for the whole functions covered by the architecture. These modules represent the different layers of the architecture and since they can each evolves independently from the others, this layered architecture can be understood as modular, in design at least.

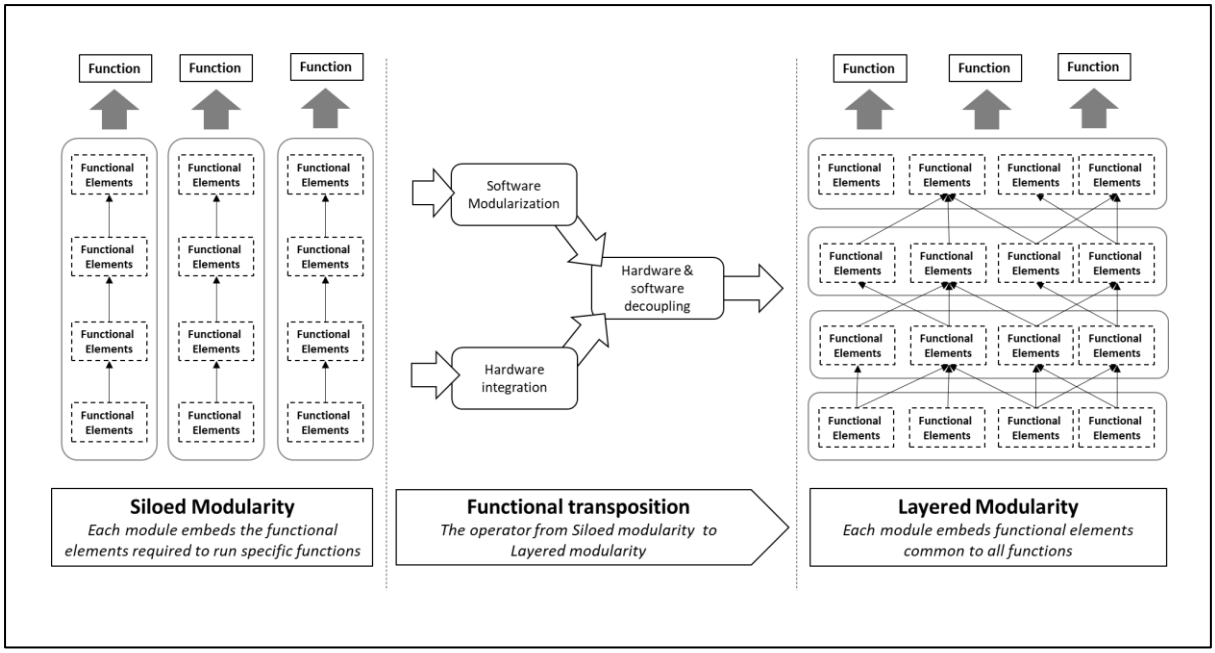
As illustrated by our results, siloed and layered modularity serve different goals and answer different issues. While siloed modularity looks consistent when functions are very different and leverage distinct resources, layered modularity looks consistent when different functions

can be built on the basis of a common set of resources. Neither siloed nor layered modularity seems to be new patterns of modularity, but previous studies does not distinguish them, and discuss modularity the same way for both. Nevertheless, discussions about external platforms somehow echoes what we call layered modularity, since platforms can be defined as common sets of technical resources that must be complemented or customized to deliver specific niche applications (Gawer et Cusumano, 2014). The distinction between the siloed and layered nature of modularity can thus serves as a bridge between modularity and platform theories.

Finally, by detailing how the OEM is moving from a siloed architecture to a layered one, this study offers precise insights about how a firm can leverage product architecture to reconfigure its ecosystem. While most studies in management analyses and interprets architectural changes by pointing increasing or decreasing degree of modularity our results illustrate the use of a more complex, composed architectural change. Indeed, while the legacy architectures were siloed modular architectures relying on a high degree of integration between hardware and software within modules, the current and new architectures stand as more layered modular ones based on a loose coupling between the hardware and the software. The architectural changes we documented enlighten 3 key operators that altogether constitute a path from siloed to layered architectures: 1) first, the integration of hardware parts (materialized in the case by the path from legacy ECUs to Domain Master and then to PCU); 2) second, the modularization of the software architecture, through a design based on SOA principles; 3) and third the decoupling of the hardware and the software architectures allowed by the appearance of the Operating system and its standardized interfaces. These three changes were necessary and would have not allowed this transformation if they were each implemented in isolation from the others. Altogether, they thus constitute a single consistent architectural change that can be understood as a "functional transposition" operator. Consistent with the definition we gave for the "nature" of modularity, this operator allows to turn a siloed modular architecture into a layered modular architecture by changing the logic supporting the gathering of functional elements within modules. These insights about the "functional transposition" composed operator, thus complement our understanding of how architectural changes can influence ecosystem structure by going beyond the observation of simple operators (as opposed to composed operators), such as integration or modularization. Moreover, since technical platforms are usually based on layered architectures, this functional transposition operator could also help scholars to explain and understand how platform born, especially when they

born from initially non-platform-based architectures. The Figure 7 illustrates both siloed modularity, layered modularity and the functional transposition operator.

Figure 7: Functional transposition as a path from siloed to layered modularity



4.2. ECOSYSTEM RECONFIGURATION.

Second, this study contributes to the development of the structuralist approach to ecosystems (Adner, 2017) by showing how a focal firm can reshape the structure of ecosystem interdependencies to improve its position. While ecosystem scholars commonly agree that actor's interdependencies play a structuring role in ecosystems (Moore, 1996; Iansiti et Levien, 2004; Jacobides et al., 2006, 2018; Adner, 2017), we know very little about them. In this study we offer both a consolidated theoretical framework to assess these interdependencies and strong empirical evidences about how a firm can transform their overall structure, both in terms of directionality and intensity, through architectural changes. Our results show that the new architecture designed by the OEM does not aim at improving the management of a specific subset of interdependencies, but rather at reshaping entirely the structure of ecosystem interdependencies to keep a central position within it.

The need for this radical reconfiguration finds its rationales in the unprecedented technology paradigm shift faced by the automotive industry, from innovation trajectories centered on mechanical technologies to innovation trajectories centered on digital technologies. This shift, which opened in the last decade new paths for fast and radical innovation regarding vehicles' features also led to the entry of many new actors in the OEM's ecosystem. These new actors –

often digital natives coming from the mobile, internet or computer industries, such as Google, Microsoft, or Huawei among many others – entered the ecosystem with both strong innovation capabilities, market power, and their own ecosystems of partners and customers, but also with ambiguous ambitions and positioning regarding the existing ecosystem structure. These entries led to the emergence of new interdependencies between the OEM and these actors that could have only been poorly managed by the OEM within the existing ecosystem structure. These settings led the OEM to turn the logic according to which it manages its interdependencies, from one mainly driven by the need to decrease coordination costs and efforts with its suppliers on software activities, into one mainly driven by its need for high scalability, flexibility and control over its new complementors and vehicles' software. Thereby the reconfiguration of its ecosystem can be understood as a path from a system integrator position answering supply chain optimization issues, to an hybrid position relying on a system integrator role for the three lower layers of the architecture, but on a keystone, or platform leader role, regarding the development of its vehicles' applications. We then showed that product architecture does not only appear as a way to manage ecosystem interdependencies, but also to (re)structure them.

This hybrid ecosystem structure seems original in regard with the existing strategic management literature which tends to strongly differentiate system integration activities (Brusoni et al., 2001) from platform ecosystems (Gawer et Cusumano, 2002). Our case shows that these two logics coexist and even are strongly dependent from each other in the new ecosystem structure. Indeed the OEM ability to provide its complementors with a common set of standardized resources depends on its ability to design and integrate the lower layers of the architecture. This situation constitutes, by the way, a great illustration of the multilateral nature of ecosystem interdependencies: here, the ability of the ecosystem leader to manage its interdependencies with complementors depends on the structure of its interdependencies with its suppliers. Moreover, this hybrid structure can be explained because of the context of digital convergence which characterize our case. Indeed, system integration strategies are mainly observed in industries developing "Complex Products Systems" (CoPS), such as automotive or aircraft for instance (Brusoni et Prencipe, 2001), while external platform strategies are linked to electronics and digital products (Gawer et Cusumano, 2002; Iansiti et Levien, 2004). Therefore, we could reasonably think that this hybrid ecosystem structure finds some rationales in the convergence of the automotive technologies, which require some degree of system integration, and digital technologies, which enable and/or require some degree of platforming.

CONCLUSION

In this study we pointed both the structuring role of product architecture in regard with ecosystem actors' interdependencies and positions but also the need to approach product architecture design as a trade-off between technical, organizational and strategic issues. Thus, this study contributes to the ecosystem literature in two main ways. First, it offers a consolidated theoretical framework to assess ecosystem actors' interdependencies that builds on two criteria: their directionality and their intensity. Second it offers precise insights about how a firm can reconfigure its ecosystem thanks to architectural changes. Moreover, it also contributes to a better understanding of modularity by identifying siloed and layered modularity as two different patterns for architecture design, and the "functional transposition" operator as a way to move from siloed to layered modularity. In light of this, and consistent with existing literature, we then propose to assess the modularity of a given system through 3 criteria: the nature of modularity (siloed or layered), the degree of modularity, and the purpose of modularity.

This study also contributes to practice by pointing how firms can face digital convergence thanks to architectural changes. Indeed, digital convergence is a phenomenon that is currently reshaping many industries, both bringing new opportunities to seize and leading to the emergence of new critical interdependencies to manage. In such context strategists play a critical role which can be hard to assume since radical changes can both lead to high uncertainty and complexity. We pointed that in such context, ecosystem strategy and product architecture must be designed together around the central issue of managing technology interdependencies. This undermines that in a given firm, teams responsible for strategy building and teams responsible for products' architectural design must conduct their thinking in a tightly coupled way to ensure the consistence between this firm's expectation regarding the structure of its ecosystem and the architecture of its products.

BIBLIOGRAPHY:

Adner, R., 2017. Ecosystem as Structure: An Actionable Construct for Strategy. Journal of Management 43, 39–58. https://doi.org/10.1177/0149206316678451

Adner, R., 2012. The wide lens : What successful innovators see that others miss. Penguin group, New York.

Adner, R., Kapoor, R., 2010. Value creation in innovation ecosystems: how the structure of technological interdependence affects firm performance in new technology generations. Strategic Management Journal 31, 306–333. https://doi.org/10.1002/smj.821

Baldwin, C.Y., 2014. Bottlenecks, Modules and Dynamic Architectural Capabilities. Harvard Business School Working Paper 53.

Baldwin, C.Y., 2007. Where do transactions come from? Modularity, transactions, and the boundaries of firms. Ind Corp Change 17, 155–195. https://doi.org/10.1093/icc/dtm036

Baldwin, C.Y., Clark, K.B., 2004. Architectural Innovation and Dynamic Competition: The Smaller "Footprint" Strategy 54.
Baldwin, C.Y., Clark, K.B., 2000. Design Rules : The power of modularity, The MIT Press. ed.
Baldwin, C.Y., Henkel, J., 2015. Modularity and intellectual property protection: Modularity and Intellectual Property Protection. Strategic Management Journal 36, 1637–1655. https://doi.org/10.1002/smj.2303
Brandenburger, A., Nalebuff, B., 1995. The Right Game: Use Game Theory to Shape Strategy. Harvard Business Review.
Brusoni, S., Marengo, L., Prencipe, A., Valente, M., 2007. The value and costs of modularity: a problem-solving perspective. European Management Review 4, 121–132. https://doi.org/10.1057/palgrave.emr.1500079
Brusoni, S., Prencipe, A., 2013. The Organization of Innovation in Ecosystems: Problem Framing, Problem Solving, and Patterns of Coupling, in: Collaboration and Competition in Business Ecosystems, Advances in Strategic Management. Emerald Group Publishing Limited, pp. 167–194. https://doi.org/10.1108/S0742-3322(2013)0000030009
Brusoni, S., Prencipe, A., Pavitt, K., 2001. Knowledge Specialization, Organizational Coupling, and the Boundaries of the Firm: Why Do Firms Know More Than They Make? Administrative Science Quarterly 46, 597. https://doi.org/10.2307/3094825
Chesbrough, H., Prencipe, A., 2008. Networks of innovation and modularity: a dynamic perspective. International Journal of Technology Management 42, 414. https://doi.org/10.1504/IJTM.2008.019383
Chesbrough, H.W., 2006. Open innovation : The new imperative for creating and profiting from technology. Harvard Business Review Press, Boston.
Chesbrough, H.W., Kusunoki, K., 2001. The Modularity Trap : Innovation, Technology phae shifts and the resulting limits of virtual organizations, in: Managing Industrial Knowledge: Creation, Transfer and Utilization. Ikujiro Nonaka, David J Teece.
Clarysse, B., Wright, M., Bruneel, J., Mahajan, A., 2014. Creating value in ecosystems: Crossing the chasm between knowledge and business ecosystems. Research Policy 43, 1164–1176. https://doi.org/10.1016/j.respol.2014.04.014
Colfer, L.J., Baldwin, C.Y., 2016. The mirroring hypothesis: theory, evidence, and exceptions. Industrial and Corporate Change 25, 709–738. https://doi.org/10.1093/icc/dtw027
Dhanaraj, C., Parkhe, A., 2006. Orchestrating Innovation Networks. Academy of Management Review 31, 659–669. https://doi.org/10.5465/amr.2006.21318923
Dosi, G., 1982. Technological paradigms and technological trajectories. Research Policy 11, 16.
Eisenhardt, K.M., 1989. Building Theories from Case Study Research. The Academy of Management Review, Vol. 14, No. 4. pp. 532-550. 24.
Fine, C.H., 1998. Clockspeed: winning industry control in the age of temporary advantage. Perseus Books.
Fixson, S.K., Park, J.-K., 2008. The power of integrality: Linkages between product architecture, innovation, and industry structure. Research Policy 37, 1296–1316. https://doi.org/10.1016/j.respol.2008.04.026
Fjeldstad, Ø.D., Snow, C.C., Miles, R.E., Lettl, C., 2012. The architecture of collaboration. Strategic Management Journal 33, 734–750. https://doi.org/10.1002/smj.1968
Gawer, A., Cusumano, M.A., 2014. Industry Platforms and Ecosystem Innovation. Journal of Product Innovation Management 31, 417–433. https://doi.org/10.1111/jpim.12105
Gulati, R., Puranam, P., Tushman, M., 2012. Meta-organization design: Rethinking design in interorganizational and community contexts. Strategic Management Journal 33, 571–586. https://doi.org/10.1002/smj.1975

Hannah, D.P., Eisenhardt, K.M., 2018. How firms navigate cooperation and competition in nascent ecosystems. Strategic Management Journal 39, 3163–3192. https://doi.org/10.1002/smj.2750
Henderson, R.M., Clark, K.B., 1990. Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. Administrative Science Quarterly 35, 9. https://doi.org/10.2307/2393549
Iansiti, M., Levien, R., 2004. The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability. Harvard Business Press.
Jacobides, M.G., Cennamo, C., Gawer, A., 2018. Towards a theory of ecosystems. Strategic Management Journal 39, 2255–2276. https://doi.org/10.1002/smj.2904
Jacobides, M.G., Knudsen, T., Augier, M., 2006. Benefiting from innovation: Value creation, value appropriation and the role of industry architectures. Research Policy 35, 1200–1221. https://doi.org/10.1016/j.respol.2006.09.005
Jacobides, M.G., MacDuffie, J.P., Tae, C.J., 2016. Agency, structure, and the dominance of OEMs: Change and stability in the automotive sector. Strategic Management Journal 37, 1942–1967. https://doi.org/10.1002/smj.2426
Kapoor, R., 2013. Collaborating with Complementors: What Do Firms Do?, in: Collaboration and Competition in Business Ecosystems, Advances in Strategic Management. Emerald Group Publishing Limited, pp. 3–25. https://doi.org/10.1108/S0742-3322(2013)0000030004
Langlois, R.N., 2002. Modularity in technology and organization. Journal of Economic Behavior & Organization 49, 19–37. https://doi.org/10.1016/S0167-2681(02)00056-2
MacDuffie, J.P., 2013. Modularity-as-Property, Modularization-as-Process, and 'Modularity'-as-Frame: Lessons from Product Architecture Initiatives in the Global Automotive Industry. Global Strategy Journal 3, 8–40. https://doi.org/10.1111/j.2042-5805.2012.01048.x
Moore, J.F., 2006. Business Ecosystems and the View from the Firm. The Antitrust Bulletin 51, 31–75. https://doi.org/10.1177/0003603X0605100103
Moore, J.F., 1996. The death of competition : Leadership & strategy in the age of business ecosystems. Harper Collins, New York.
Moore, J.F., 1993. Predators-and-Prey-A-New-Ecology-of-Competition.pdf. Harvard business review.
Nonaka, I., Teece, D.J. (Eds.), 2001. Managing industrial knowledge: creation, transfer and utilization. SAGE, London ; Thousand Oaks, Calif.
Orton, J.D., Weick, K.E., 1990. Loosely Coupled Systems: A Reconceptualization. The Academy of Management Review 22.
Pavitt, K., 1998. Technologies, Products and Organization in the Innovating Firm: What Adam Smith Tells Us and Joseph Schumpeter Doesn't. Idustrial and corporate change 20.
Posen, H.E., Ethiraj, S.K., 2013. Do Product Architectures Affect Innovation Productivity in Complex Product Ecosystems?, in: Collaboration and Competition in Business Ecosystems, Advances in Strategic Management. Emerald Group Publishing Limited, pp. 127–166. https://doi.org/10.1108/S0742-3322(2013)0000030008
Sanchez, R., Mahoney, J.T., 1996. Modularity, flexibility, and knowledge management in product and organization design. Strategic Management Journal 17, 63–76. https://doi.org/10.1002/smj.4250171107
Simon, H., 1962. The Architecture Of Complexity. Proceedings of the American Philosophical Society 106.
Teece, D.J., 2018. Profiting from innovation in the digital economy: Enabling technologies, standards, and licensing models in the wireless world. Research Policy 47, 1367–1387. https://doi.org/10.1016/j.respol.2017.01.015

Teece, D.J., 2007. Explicating dynamic capabilities: the nature and microfoundations of (sustainable) enterprise performance. Strategic Management Journal 28, 1319–1350. https://doi.org/10.1002/smj.640
Teece, J., 1986. Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy 21.
Thompson, J.D., 1967. Organizations in Action Social Science Bases of Administrative Theory, ed. Mc Graw-Hill Book Company.
Ulrich, K., 1995. The role of product architecture in the manufacturing firm. Research Policy 24, 419–440. https://doi.org/10.1016/0048-7333(94)00775-3
Ulrich, K.T., Seering, W.P., 1990. Function sharing in mechanical design. DESIGN STUDIES 11, 12.
Yin, R.K., 2009. Case Study Research: Design and Methods, ed. SAGE.