Abstract

This paper casts light on the question how coordination of distributed innovation processes is achieved by comparing the automotive and open source software industries. These industries represent very different settings of distributed innovation, one a traditional industries that has ‘opened up’ innovation, the other the extreme case of network innovation. They have not so far been object of a comparative study. We present a comparative literature review, and then draw on empirical research in the automotive and open source software industries to compare how coordination is achieved in each. We apply qualitative methods to the auto industry, where we draw from a 10-year case study of a major European auto firm. We apply quantitative methods to the open source software industry, where we do an econometric analysis of 5810 projects registered on SourceForge. We find a surprising similarity regarding how coordination is provided in both cases: the use of templates at early stages of the project, which foster coordination of joint search processes by distributed actors through acting on their representations of search trajectories and eliminating some of the potential trajectories.

Acknowledgements: this work benefited from the comments and advises received at the International User Innovation Workshop, Copenhagen Business School, June 26-28, 2007, and by Stefano Brusoni, Eugenia Cacciatori, Carliss Baldwin, Alberto Cammozzo, Toke Reichstein. We also thank Karim Lakhani and Inna Liubareva for having shared with us the ideas they are working on.
1. Introduction

The development of complex products, such as automobiles, aircraft or software, typically takes place in networks. In developing new products, firms that produce complex products draw on multiple actors such as suppliers, customers, or universities (von Hippel, 1988; Womack et al., 1990; Clark and Fujimoto, 1991; Wheelwright and Clark, 1992; Powell et al., 1996, Chesbrough, 2003). More recently, a growing literature has also shown that firms or networks of firms are not the only locus of organized innovation activities. As the case of open source software (henceforth OSS) shows, innovation might also emerge from a highly differentiated set of actors in which firms still play a minor role (David and Foray, 2003; von Hippel, 2001). Both in more traditional industries and the open software case, the key challenge in developing complex products is the same, however: how to coordinate the involvement of distributed sources of innovation in the new product development process so as to achieve the desired product and project performance. The coordination challenge becomes more difficult than in the case of innovation carried out inside firms because one of the main coordination mechanisms available is weakened and much less available, if it can be harnessed at all: direct supervision (Mintzberg, 1979). This restricts the set of coordination mechanisms that can be applied to provide coordination. In principle, new or different coordination mechanisms have to take the place of direct supervision, or at least to assume a larger weight in the bundle of coordination mechanisms used to coordinate distributed innovation. In fact, empirical research into industries characterized by distributed innovation and by networks more generally has identified the important role of factors such as artefacts, informal relations, and trust (Star & Griesemer, 1989; Carlile, 2002; Ouchi, 1980). In this article, we tackle the question how coordination of distributed innovation processes is provided.

We thus contribute to the understanding of the coordination mechanisms applicable to distributed innovation processes. We address this general question from the narrower perspective provided by the comparison of the differences and commonalities between how coordination is achieved in new product development projects in the OSS industry and in a more “traditional” context. The open source software case is paradigmatic: in this industry the innovation process is distributed almost by definition (von Hippel, 2001). The OSS industry is the exemplification of modes of coordination of innovation activities of multiple actors where the firm plays a very minor role. This makes the stream of literature on OSS of relevant for understanding distributed innovation processes. It is clear, however, that the OSS case shows organizational solutions that are very difficult to apply to different industrial contexts, despite the growing importance that distributed innovation has also for traditional industries (Chesbrough, 2003). Identifying which are the coordination mechanisms that provide coordination in the OSS industry and understand why they do so, can, however, contain important lessons also for other industries. Going beyond the descriptive part of the research question, hence, the comparative research design we employ enables us to also make a contribution to the general understanding of the coordination mechanisms applied in industries characterized by distributed innovation.

---

1 We define product complexity as a function of the number of technologies involved in the product, their relative heterogeneity and their pace of change. Our definition of complexity therefore includes not just the number of components in a product but also the interactions between them (Simon, 1962).

2 The other classical coordination mechanisms are mutual adjustment and standardization of inputs (such as skills), standardization of process, and standardization of output (Mintzberg, 1979).
This paper presents the first results of our comparative study. We arrive at two main findings: First, in both industries the means – namely system integration and modularity – that were supposed to achieve coordination of distributed innovation according to the literature display some limitations, both conceptually and practically. Neither systems integrators nor modular product architecture are easily implemented so they actually lead to good coordination of distributed innovation processes. Second, we identify a means of coordination that works in both industries. Interestingly, it seems to be based on the same underlying principle: integration problems are effectively dealt with when at the very early stages of the process in development projects, actors are able to share a representation (that we call template) of the set of possible trajectories the development activities could take. In what follows, we establish and explain this finding. The paper is structured as follows. We first provide an overview of previous research and present our research question in more detail. We present our methodology, and then compare the two industries drawing both on literature studies and through empirical studies, employing both qualitative and quantities method. The concluding section discusses the implications of the main findings.

2. Background and prior research

In many studies of distributed innovation processes, the problem is represented from the perspective of the business firm. This perspective explains why the expression “involvement of external sources of innovation” (e.g. Nishiguchi, 1994, Clark and Fujimoto, 1991) has emerged and is so dominant in an important part of the innovation management literature. A previously “autarchic” (Christensen, 2006) view of the business firm has rapidly become more concerned with distributed innovation processes, as a previously strong ‘inward-looking focus’ has given way to a focus on partners and networks even in traditional industries.

The major motivation for involving external sources of innovation in product development is that doing so can increase the performance of new product development projects (Clark, 1989; Womack et al., 1990; Clark & Fujimoto, 1991; Wheelwright & Clark, 1992), and hence, firm performance.

Previous literature has identified some of the main causes of problems in achieving high project performance in distributed innovation. The most important cause, especially when products are complex, is the management of technical interdependencies (Sosa et al., 2004). As it turns out, it is not trivial to develop a system that has high performance as a whole when the innovation process is broken down into smaller parts and such smaller development tasks are allocated to a number of different actors. The problem is that interdependencies require coordination of joint development efforts, as changes in one component or subsystem might be offset by negative performance implications induced in other components or subsystems due to technical interdependencies. The key issue here is the coordination of complex and distributed search processes. Prior research has identified four issues firms have to address to achieve high project performance: (1) how to divide the development task, (2) how many (and which) sources of external innovation to allocate the sub-tasks to, (3) how to coordinate the actors that develop components and subsystems, and (4) once the components and subsystems have been improved and developed, how to integrate the components and subsystems into a whole that has high product performance (Baldwin & Clark, 2000; Takeishi, 2001, 2002; Brusoni et al., 2001; Laursen and Salter, 2006).

The literature on innovation management in traditional industries identifies two ways of tackling the organizational challenges involved in network innovation. First, a focal firm acts
as system integrator (Hobday, Davies and Prencipe, 2005). This approach consists in orchestrating the other partners involved in the product development process through a top-down hierarchical approach. The second possibility is to rely on modular product architecture, i.e., an architecture where interdependences are bundled within modules, while modules are independent of each other, and have standardized interfaces (Baldwin & Clark, 2000). Such a modular product architecture supposedly allows using a modular organization structure (both within the firm and in the value chain) (Sanchez & Mahoney, 1996). It provides a powerful possibility: external sources of innovation can accomplish their development tasks independently and do not require coordination, as the standardized interfaces and independence between modules assure that modules will fit together even without coordination when integrated into the overall product (Baldwin & Clark, 2000).

Interestingly, research documents problems with how systems integration and modularity are implemented so that high product and project performance is achieved. In theory, the appeals of both are strong, and the working principles clear enough. Still, empirical research documents that firms have many problems in achieving high product and project performance based on modularity (Fine, 1998, Brusoni, 2005). Empirical evidence drawn from many industries (Chesbrough and Kusunoki, 2001, Brusoni, 2005, Mac Duffie, 2008, Zirpoli and Becker, 2009) shows that coordination does not simply follow the optimal decomposition of the development tasks and its allocation to actors3. Coordination and cooperation problems amongst different actors that participate in the innovation process, in fact, increase the time and cost of the development project, and lead to divergences with customer requirements. Moreover, coordination problems are increasing with higher number of participants, higher complexity, and stronger interdependencies between actors (Mintzberg, 1979). Finally, as actors involved in product development become more diverse (such as lead customers, universities, suppliers etc.), cooperation problems require stronger responses. Both are becoming more difficult with the involvement of more external sources of innovation in the development process.

The literature on open source software innovation has had a different, and yet parallel, development in the way it deals with the question. Being based on the voluntary contribution of a large number of developers spread throughout the world, the initial interest of the literature was mainly focused on the incentives individuals may have had to contribute to such a public good (von Hippel, 2001; von Hippel and von Krogh, 2003; Lerner and Tirole, 2002; Ghosh et al., 2002, David et al., 2003). A direct consequence of this peculiar feature of the OSS innovation model is that individuals self-select the tasks they perform (Langlois and Garzarelli, 2008), leading to the emergence of self-organized systems (David and Rullani, 2008). The initial interest in incentives, thus, was increasingly coupled with inquiries relative to the way in which individuals implement the self-organization of their collective work. The emergence of authoritative structures (Mateos-Garcia and Steinmueller, 2003) and of leaders (Lerner and Tirole, 2002) were among the first phenomena analyzed. Social processes leading to this emergence, such as criticism of the status quo (Lee and Cole, 2003) or creation of specific patterns of social ties facilitating the making of individuals into leaders (Dahlander and O’Mahony, 2008), became the center of many recent studies in the field. By the same token, attention was devoted to the limits of such processes, investigating how conflicts are resolved (Elliott and Scacchi, 2003) and how authoritative structures are challenged and changed (O’Mahony and Ferraro, 2007). This stream of literature grew usually keeping the question of incentives in the background (Dalle et al., 2009), and developed several different

3 Recent works, in fact, show that also system integration performed by a leading firm can be highly problematic when firms lose grasp of component specific knowledge (Takeishi, 2002, Zirpoli and Becker, 2009) and the boundary of the integrating firm difficult to draw due to persistent integrality (Mac Duffie, 2008).
conceptualization of governance mechanisms in the OSS setting (Markus, 2007; O’Mahony and Ferraro, 2007; Dahlander and O’Mahony, 2008).

More recently a new stream of literature has emerged, trying to merge the social side dominating the previous points view with the “materiality” (Orlikowski, 2007) of the production process itself. Developers, in fact, do not interact only exchanging opinions, but also acting on artifacts—the lines of source code composing the software—that they exchange and develop jointly (Lanzara and Morner, 2005). The structure of the code, and in particular its modularity, has been identified as a crucial issue. It has been shown that modularity is a crucial enabling factor, whose integration mechanisms solve the trade-off between each developer’s cost of participation (which calls for a partition of the innovation process into narrow and independent tasks) and the need to assure overall consistency of the product (MacCormack et al., 2006). The peculiar characteristics that modularity assumes when implemented in the OSS environment moved at the center of the debate. Narduzzo and Rossi (2005) show that a conceptualization of modularity as information hiding is not appropriate to describe how the structure of the code evolves over time. They show that modularized OSS projects keep information hidden only to the extent to which the original architecture assures effective coordination. As soon as the current structure of the software is challenged by unexpected interdependencies between modules, OSS code availability allows developers to access information specific to modules they were not working on before, and discuss a new structure of the product with the other participants. This study suggests that in OSS, modularity has to be redefined to cope with an environment where the self-organization typical of adaptive systems (Kogut and Metiu, 2001) does not act only at the level of the single module, but also at the level of the whole architecture.

From the review of these streams of literature it emerges that coordination, cooperation and integration in distributed innovation are still controversial issues. The presence of a complex governance structure, either as the result of the self-organizing system of individuals (O’Mahony and Ferraro, 2007) or the implementation of a network structure driven by the interests of firms with different power (Kogut, 2000), has been found in both industries. By the same token, the literature has also highlighted the limits of those structures. A similar argument can be put forward when discussing modularity, as scholars have not only shown the importance of modular interfaces in the growth of both industries, but also highlighted its limits and its peculiar problematization in the different contexts (Narduzzo and Rossi, 2005; Baldwin and Clark, 2006).

As our review of prior research has shown, coordination, cooperation and integration in distributed innovation are still controversial issues. Moreover, while a considerable body of literature has looked into the coordination of distributed innovation in several industries, the empirical evidence is contrasting both within and across industry domains. To our knowledge, however, there are very few systematic attempts to compare the OSS industry and traditional industries with regard to how coordination of the innovating actors is achieved. This is the gap our study addresses.

---

4 Maggioni (2004) analyzes the similarities between open source and industrial districts in terms of the gradual emergence of a hierarchical structure. A similar comparison is used by David and Rullani (2008) to infer some properties of the open source system dynamics on the basis of what has been observed for clusters of firms. Lee and Cole (2003) conceive the open source software as a possible alternative model to a firm-based one, and thus develop a comparison between the two production modes. To the best of our knowledge, among the very few studies undertaking such comparison, no one has focused on the coordination mechanisms applied in both realms.
3. The study

To address this gap, we engaged in a long term project that considers the differences and commonalities between how the coordination of distributed actors is achieved in OSS and in a traditional industry. We chose to compare the OSS case with the automotive industry for three reasons. First, the automotive industry is a paradigmatic case, too: the involvement of external sources in the automotive new product development process is probably the most studied and replicated in other industries (e.g. Womack et al., 1990). Second, twenty years after the pioneering empirical contribution on the issue (e.g. Clark, 1989) there are still many controversial points on how to effectively coordinate distributed innovation processes in the automotive industry. For example, although both system integration and modular solutions are powerful theoretical ideas that found many followers in theory and in practice, much recent empirical research shows that implementing these two solutions is not trivial, in particular when high project performance is the aim (Fine and Whitney, 1996, Takeishi, 2002, Mac Duffie, 2008). Finally, we selected the automotive industry because we had accumulated a ten year empirical record on coordination in distributed innovation processes in this particular industry when we started this study in 2006 (Zirpoli & Caputo, 2002; Becker & Zirpoli, 2003).

We start the study from a theoretical comparison, i.e. comparing previous studies on how the two industries have tried to cope with coordination problems. Subsequently, we compare the empirical data we gathered in the automotive and OSS cases. These data concern the coordination of new product development projects carried out at Fiat, a European car manufacturer, and the data from SourceForge, an OSS online development platform. Part of the data was developed during our long prior research record at both sites (Zirpoli & Caputo, 2002; Becker & Zirpoli, 2003; David and Rullani 2008; Giuri et al. 2008, 2007). To these data, we added the results of new data gathering campaign we undertook specifically for this research project from 2006 onwards.

The main problem we had to face was that of comparing data gathered from two completely different settings: a large multinational carrying out huge projects, on one side, and an online community of independent individuals that develops projects of a much smaller magnitude, on the other. The solution we employed followed a two-step process. We first defined the unit of our analysis, and then identified the most appropriate comparison method. Building on the insights from prior literature, we decided to focus our analysis on the development project, so that data were gathered at the single project level. This choice is motivated by the fact that it is in projects that the coordination processes we are particularly interested in can be observed. In the industries we analyze, the project is the organization form used for managing the development of new products. Projects define the task to be solved, and therefore the coordination challenge. In both empirical settings, we thus observed coordination within projects in action and analyzed data at the project level.

As scale and complexity of innovation projects carried out in the two contexts are substantially different, we employed two different research methods. After having considered the scale, number and complexity of each development project at Fiat, in the automotive case we opted for carrying out the observation of how coordination is carried out through a qualitative data gathering process. We analyzed company documents (more than 2000 pages) and carried out interviews at Fiat and its first tier suppliers, major contributors to new product development projects in the automotive industry. Table 1 represents an overview of the interviewees in the automotive industry.
<table>
<thead>
<tr>
<th>Company name</th>
<th>Product/role</th>
<th>People interviewed</th>
<th>Date of interview</th>
<th>Total length of interviews (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OEM Research Centre</td>
<td>Research Centre</td>
<td>CEO, Business Development Director Technologies Division</td>
<td>8_02_2006 22_03_2006</td>
<td>5</td>
</tr>
<tr>
<td>Company A</td>
<td>Sealing systems/supplier</td>
<td>Plant General Manager, Engineering &amp; Design Technical Director Assistant, Quality Manager</td>
<td>21_02_2006 07_06_2006</td>
<td>9</td>
</tr>
<tr>
<td>Company B</td>
<td>Brakes/supplier</td>
<td>Business Development Director, Product Engineering R&amp;D Manager Brake Systems</td>
<td>28_03_2006</td>
<td>3</td>
</tr>
<tr>
<td>Company C</td>
<td>Car design development, turnkey development projects/supplier</td>
<td>Business Strategy Development Manager, Project Manager, Technical Division Manager, Customer Manager, Project Leader</td>
<td>23_03_2006 15_05_2006</td>
<td>10</td>
</tr>
<tr>
<td>Company D</td>
<td>Chassis control (ABS – ESP, etc.), power train, car multimedia/supplier</td>
<td>Manager Automotive Technology Product Planning and Marketing, Cross Functional Project Manager</td>
<td>29_03_2006</td>
<td>4</td>
</tr>
<tr>
<td>Company E</td>
<td>Car interiors, seats/supplier</td>
<td>CEO, Senior Program Manager OEM Account Manager</td>
<td>9_02_2006 23_03_2006</td>
<td>5</td>
</tr>
<tr>
<td>Company F</td>
<td>Safety systems (airbags, seat belts, brakes, chassis control (ABS – traction control systems, etc.) /supplier</td>
<td>Account Director, Manager Programs &amp; Application Engineering Inflatable Restraint Systems</td>
<td>9_02_2006 23_03_2006</td>
<td>5</td>
</tr>
<tr>
<td>Company G</td>
<td>Stamped parts in metal/supplier</td>
<td>Plant manager</td>
<td>3_02_2006</td>
<td>5</td>
</tr>
<tr>
<td>Company H</td>
<td>Thermal systems/supplier</td>
<td>Sales &amp; Marketing General Manager, OEM sales manager, R&amp;D Thermal Systems Division Manager</td>
<td>3_04_2006</td>
<td>6</td>
</tr>
</tbody>
</table>
In order to assure comparability between the results of the previous analysis and the outcome of the exploration of the open source phenomenon, we need to establish the mapping between the focus of analysis defined in the previous section and the elements constituting the open source environment. FIAT is a subject promoting a series of different projects. It manages their evolution and involves some external actors in each. In other words, FIAT can be conceived as the incubator of the projects it develops. The correspondent function in the open source world would then be an online incubator of OSS projects. However, as seen in the literature review, the bottom-up structure of production typical of OSS distributes the authority and participation among different individuals (Mateos-Garcia and Steinmueller, 2003). In the OSS world many developers create their own initiatives in an independent manner. Focusing on an incubator with the same leading role that FIAT has in all the projects it launches would reduce the representativeness of our sample of OSS projects. The incubator then needs to be the host of an ecology of independent (at least in principle) projects, for example a provider of online services that has been able to create an ecology of projects around its activity. The online platform implemented by SourceForge, Inc. CA, USA (www.sourceforge.net), is a natural candidate for this purpose. With its 230,000 software projects and 2 million registered users (February 2009) it represents the largest incubator of OSS projects worldwide. We obtained access to the data relative to the population of projects hosted on the online platform from the moment of its foundation (November 1999) up to October 2008. In order to be able to investigate such a large sample of projects and account for the larger variety of coordination structures and project forms populating the OSS scenario, we opted for a statistic analysis able of capturing the mechanisms we want to investigate, beyond the many differences between the projects. Mirroring the nature of our study, the next section reports both the outcome of our literature-based comparison of the two contexts and the result of our comparative empirical study.

4. Theoretical insight: comparison of the two literatures

4.1. The coordination of distributed innovation processes in the automotive industry

Early research on the automotive industry offered many indications regarding how distributed innovation processes are coordinated, pointing towards early involvement of suppliers, supplier segmentation according to for instance the service they provide, and different supplier management practices and types of relationships. The insight of this first wave of research was that for coordinating the involvement of suppliers in the development process, it was essential to distinguish different types of suppliers, and to match them with different types of relationships that leveraged different bundles of supplier management and governance mechanisms (along the lines of contract-based ‘arm’s-length’ relationships and different types of closer relationships) (Clark & Fujimoto, 1991; Dyer et al., 1998; Liker et al., 1996, Lamming, 1993; Dyer, 1996).

More recent literature has brought new variables into the picture. Following the knowledge-focus of the management literature (Grant, 1996, Spender, 1996), studies of coordination in new product development projects have emphasized the importance of problems of knowledge integration and learning (Takeishi, 2001; 2002). At the same time, technological developments and other factors led to an increased need for ‘openness’ and involvement of suppliers in the development process of auto manufacturers. As mentioned, two ways of tackling the organizational challenge of distributed innovation had been

---

5 Part of the data were provided by the SourceForge Research Data Archive (SRDA). University of Notre Dame, http://zerlot.cse.nd.edu/; see Gao, et al. (2007) and Madey (2009). We combined those data with those already used in David and Rullani (2008) and Giuri et al. (2007, 2008). We refer the reader to those articles for a further discussion of the data.
discussed in the wider literature: system integration and modular product architecture. These two solutions were also applied in the auto industry (MacDuffie, 2008).

Studies in the auto industry indicate that these two ways of tackling the issue have turned out to be difficult to implement in the auto industry. They have, overall, not led to the great benefit in the coordination of joint development efforts that was hoped for. Studies of systems integration and the implementation of modularity have identified several reasons why systems integration and modularity have severe limits in the auto industry:

To start with, it has turned out that in the auto industry there are many limits to the full decomposability of the architecture of the automobile, one of the prerequisites for modularizing product architecture (which often is an important ingredient in a systems integrator strategy). As MacDuffie (2008) notes in his review of the state of the auto industry, automobiles have a persistently integral architecture, as some of the car makers that tried to modularize it more have had to learn the hard way. The problem is not just that very few industry standards regarding component and system interfaces exist, thus posing a serious limit to the use of modular product architecture in practice (Baldwin & Clark, 2000). More serious still is the fact that the product performance of a car as a whole is not fully decomposable – many components and systems contribute to important performance dimensions such as noise, vibration and harshness. These performances, however, are emergent phenomena that cannot be neatly decomposed into the performances of particular components or systems. Rather, the interaction between the different components and systems is also an important factor in generating the overall performance on the level of the car as a whole. The problem, however, is that in a modular architecture it is very difficult to split up the responsibility for assuring the car as a whole has a certain performance. Because the performance cannot be decomposed neatly into modules that make precise contributions to the performance on the level of the car as a whole, such a performance cannot be achieved if it is parcelled out in non-overlapping bits that different suppliers develop independently. Precisely that, however is the idea of modularity: the development of the different modules can be decoupled and does not require coordination. It is, thus, difficult to achieve high product performance in the automotive industry by leveraging product modularity (MacDuffie, 2008; Becker & Zirpoli, 2008).

Moreover, it is also difficult to achieve high project performance by applying the modularity principle to the organization, and by loosely coupled organizational solutions (Sanchez and Mahoney, 1996) where decreasing the need to coordinate leads to higher project performance. One of the reasons is that under conditions of persistent integrality in the auto industry (MacDuffie, 2008) when modular interfaces are employed, inevitable and costly re-design activities have to be carried out to cope with technical interdependences that are unexpected or impossible to cope with ex ante.

A second major limit to systems integration with the help of modular product architectures that has emerged in studies of the auto industry derives from knowledge-related issues. Research has pointed out that when dividing labor in a modular way (e.g. when the development of modules is allocated to module suppliers), it is crucial that each supplier also has and maintains the knowledge required for competently developing the module. The problem, however, is that the knowledge required to develop particular modules is not modular – for instance, it also includes knowledge about how the focal module with interact with other modules (which sometimes requires some in-depth knowledge about the other modules). Moreover, it does not necessarily overlap precisely with the development task. Assuring that the division of labor is aligned with the division of knowledge, or the knowledge held by each supplier that is tasked with developing a module, has turned out very difficult to do in practice in the auto industry (Takeishi, 2001, 2002). In any case, it cannot be
assured simply by ‘decomposing’ the knowledge required for developing a car with a high performance at the level of the car as a whole into ‘modular’ knowledge domains.

These two points, persistent integrality of the product architecture of cars and the difficulties in assuring the alignment of development task and the knowledge required to carry the development task out competently by using product architecture, are strong indications that a systems integration approach that relies on a modular product architecture does not work very well in the auto industry, and is subject to strong limitations.

4.2. The coordination of distributed innovation processes in the OSS industry

For the purpose of the current analysis it is useful to distinguish two main streams of literature investigating two main coordination structures observable in open source projects:

1. Governance, where the focus of the analysis has been on the emergence of leaders or of authoritative structures and on the legitimation processes needed for their establishment and functioning.

2. Artifacts, where coordination is achieved thanks to the peculiar characteristics of the product created by the collective effort. The function of artifacts as objects that have peculiar functions in the organizational setting is in the focus (Cacciatore, 2008; D’Adderio, 2003; Orlikowski, 1992, 2000).

Markus (2007) provides a comprehensive survey of the studies on OSS governance. His distinction of the different definitions of governance structures allow us to provide a definition that fits the purpose of the present study. "In the operational coordination literature, OSS governance is understood as a solution to […] the problem of loss of operational control, and the solution is techniques for managing the process of OSS development work" (p. 156). In this conceptualization, coordination mechanisms in OSS are perceived as the solutions to the problem of control on the joint activities of the participants. Involving voluntary participation, OSS collaboration implies a very peculiar nature of control, which can be centralized into a leader (as happens for Linux, Raymond, 1998; Lerner and Tirole, 2002) but needs to be continually reproduced and legitimated by those constituting the “lower layers” of the pyramid. Raymond (1998) argues that legitimation as leader comes naturally as a consequence of project foundation: the founder sees recognized by other participants her right to take the final decision with respect to the development of the project. However, legitimation is not a static concept, and has to be created and renewed each time (O’Mahony and Ferraro, 2007). Muller (2006) shows through a simulation model how leadership emerges as the result of a dynamic legitimation process among peers clustering around ‘opinion leaders’. This effect, however, obtains a weaker support by O’Mahony and Ferraro (2007), who find that the antecedents of leadership acquisition are more related to face-to-face meetings and impact of members’ contributions rather than to online communication. Nevertheless, they admit this may be the result of the impossibility to account in their econometric analysis for the content of the online communication, as their ethnographic study reaches the opposite conclusion. The authors connect these results to the evolution of the OSS project they study (Debian) and identify the transformation the authoritative structure and the governance mechanisms go through when the project moves from one phase of development to a more complex one. Their description of the dynamic transformation of governance highlights the passages from an autocratic leadership towards a formalized authority structure that acquires legitimation through the construction of democratic regulatory processes. This echoes Lee and Cole’s (2003) identification of the community debate as the key mechanism through which the OSS community evolves. Following the same dynamic perspective, Mateos-Garcia and Steinmueller (2003) argue that “as the capabilities the integrator has for keeping up with a project’s development pace start diminishing, a structure with layers of trusted individuals emerge as a way of helping her or him cope with the increased complexity
and size of the project. These layers will be composed of proficient individuals with experience in the project. The vision they have of the project will also concur with that of the leader in some essential points” (p. 22). Thus, a pyramidal structure gradually emerges, but again based on legitimation mechanisms involving the leader’s vision as well as the technical capabilities of the developers coming to populate the intermediate layers.

Elliott and Scacchi (2003) apply a perspective that allows a more fine-grained definition of the legitimation process. Inspired by the studies that highlight the community-related aspects of OSS projects (e.g. Cohendet et al., 2001) the authors describe how developers resolve conflicts over the legitimacy of undertaking certain disputed actions (such as using software tools that are not open source to produce material for an open source project). The reference to common values and shared norms are the main rhetoric instruments used to solve the conflicts arising when some one’s actions are disputed. What is interesting to highlight here is that the authors recognize not only the importance of the norms themselves, but also the fact that they are embodied in the online discussion stored in easily accessible web repositories. “This fast access to archived information perpetuates the cultural beliefs that have been articulated and assists in resolution of conflicts” (Elliott and Scacchi, 2003; p. 9). This allows us to introduce together with to the discussion on the emergence of governance structures a second stream of literature, investigating the role of the artifacts, such as email messages, lines of code and IRC logs, in shaping the interaction context OSS developers face. The social ties and the hierarchical structure developers create at the level of their interaction, in fact, represent only one side of the coin. The other side is composed by the web of relations they create through the exchange of the artifacts they produce. David and Ghosh (2008) have shown that the structure of the technical dependencies between the different module composing Linux have a certain degree of correspondence to the pattern of social ties the authors of those modules have built through past collaboration. A similar perspective emerges from the analysis undertaken by Narduzzo and Rossi (2005) in their effort to define modularity in the OSS context. In OSS the architecture of the software is likely to be constantly changed over time, and so it is the degree of modularization it embodies. Narduzzo and Rossi study how a community of developers copes with the emergence of interdependencies, and notice that every time a new dependency connects two modules challenging the current architecture, the main principles of modularity –information hiding- is reversed: developers exchange module-specific information with the aim of discussing a common understanding of the new logic underpinning the product. Again the artifact’s structure and the social side are coupled in the process of product development. MacCormack et al. (2006) focus on a similar process but with an opposite perspective: they also study the transformation of a weakly-modular OSS product into a highly modularized software, but in the context of a transformation implemented top-down by a firm (Netscape) opening the source code of its software as a strategy to attract external developers. They observe that the project was initially stagnating because its weakly-modular structure increases the cost of external developers’ contribution, and that the subsequent increase in modularizing determined instead its success. The strict link between participation and modularity has been further developed by Baldwin and Clark (2006). Modularity is seen not only as the determinant of a lower cost of contributing. Moreover, a high level of modularity assures a higher option value for external developers in terms of the possible future configurations of the product, and thus increases their willingness to participate. Modularity is again seen has having a social effect: decreasing free-riding. The other social effect the artifact structure has is more relate to our topic: coordination. As Lanzara and Morner (2005) explain, in OSS the code is “exposed”, everyone can read it and evaluate it, run it and contribute to it. The code sends thus signals that the individuals interacting with it receive. These signals are relative to the parts of the program that need further development, to the functions that do not perform properly, and the like. For example Dalle et al., (2008) show that the level of complexity and
the level of modularity of different software packages are among the main determinants of the level of collaboration emerging around each specific package. But the code sends signals also about the different rewards one can expect to receive when contributing to a particular module instead of another. For example, modules with more collaborators or modules at the core of the program assure higher visibility, and thus can attract by developers moved by reputation concerns (Lerner and Tirole, 2002). In the contrary, developers with specific needs the software fails to fulfill may want to contribute to an obscure module if it is crucial for the function they care of (Dalle and Jullien, 2003). In this way different modules attract different developers. The allocation patterns of developers’ contribution emerging from this process will in turn change the structure of the code, and thus the rewards associated to each module in the next round. A new allocation of developers’ effort will emerge in response to this new pattern, and so on. The final code will be result of this co-evolving process. This last process as been described by Dalle et al. (2009), with the purpose of developing a stigmergic model of coordination in OSS, i.e. where the properties of the code are the means through which coordination is realized.

4.3. Discussion of the review

The parallel literature review has highlighted the important role that system integration and emergent authority structures on the one hand, and of modularity on the other, have played in the literatures on the automotive and OSS industries. It has also shown the limitations of both. In both industries, the existence and emergence of unexpected interdependencies, resulting in the need for redesign, remained, and so do the associated costs. In the automotive industry, modularity has strong limits in providing coordination of innovation by a distributed network of agents. This is because it relies on the architecture designed by the system integrator leading the network. More importantly still, it is also because of the limits to decomposability posed by a product architecture of cars that has by now been recognized to be persistently integral (MacDuffie, 2008), and because of problems with finding a division of knowledge that matches the division of labor indicated by the modular product architecture (Takeishi, 2002). Moreover, in both industries system integrators were seen to face limits imposed by the impossibility for bounded rational agents to forecast all the possible interdependencies between modules that could emerge at the level of the product performance. In OSS, the authoritative structure in which few developers hold the “final word” on the architecture (as in Linux) and the management of the modular architecture itself are based on much more flexible principles, such as legitimation through democracy and information disclosure and renegotiation of the tasks by the contributors. However, the obvious downside of this arrangement is the cost of the self-organization of the innovation process. Self-organizing collectives need to dissipate resources to guarantee the stability of the processes they are undertaking. As the previous literature review shows, in OSS this means mobilizing many more individuals than those that will actually be able and willing to contribute, spending energies in resolving conflicts and in building and maintaining legitimation structures and procedures able to sustain the authoritative structure supporting the organization (David and Rullani, 2008; Lanzara and Morner, 2005). The ‘stigmergic’ construction of the code (Dalle et al., 2009) also results in the multiplication of redundancies and unforeseen interdependencies, and when redesign becomes a necessity, developers need to gather information on other modules and discuss the architectural changes to be done. All these costs diminish the effectiveness of the coordination mechanisms the literature has found.

In the next section, we turn to studying empirically both the Fiat case and the population of projects hosted on SourceForge.
5. Empirical investigation

5.1. Evidence from the Fiat case

Fiat Auto is an almost perfect object of study for the research question, and for observing how a firm that used to carry out much of its innovation internally gradually ‘opened up’ towards a stronger integration of suppliers in development efforts. More important still, Fiat represents an extreme case of distributed innovation in the automotive industry. From 1996 onwards, Fiat massively stepped up outsourcing of product development (rather than just manufacturing) tasks to suppliers. At the same time, Fiat also gave suppliers an ever increasing role in the development process: they were included earlier in the process, including the pre-development phase in which concepts were developed; they were given larger responsibilities, including the responsibility to develop entire modules; and they were considered strategic partners. Fiat’s target for the end of 1990s was to outsource complete systems of components, and finally, complete pre-assembled modules. The aim was to deal with just two types of suppliers during the New Product Development (NPD) phase: systems and module suppliers. What makes Fiat particularly interesting in the context of the research question we pursue here is that it did indeed reach an exceptional level of design outsourcing: up to 85% of the total value of a car was engineered by suppliers at the beginning of the 1990s. Between 1996 and 2001, Fiat was one of the firms with the highest degree of outsourcing of design engineering in the automotive industry. Most likely, it is the firm that has pushed design outsourcing further than any other auto manufacturer.

One of the topics that require attention when stepping up outsourcing of design tasks and the involvement of suppliers in the development process is to employ an appropriate set of supplier governance mechanisms to manage the relationship with suppliers so that high project and product performance is attained. Although Fiat implemented the supplier management practices that were state of the art, such as for instance co-locating Fiat staff at suppliers’ and vice versa (Zirpoli & Caputo, 2002), the result fell short of expectations: increasingly involving suppliers in the product development project led to substantial problems. While there were other problems as well, without doubt the most important one was that by 2001, Fiat had lost its competence to design dashboards, suspensions, and occupant safety systems (Becker & Zirpoli, 2003). For the purposes of the present article, it suffices to take note of this very strong sign of problems with how Fiat worked together with suppliers to whom important parts of the development process were outsourced. What were the causes of these highly significant problems?

Our analysis of Fiat’s problems underlying the erosion of its design competences shows that what was crucial for this competence was to be able to decide performance trade-offs that showed up when the performance of the individual systems was integrated into a complete car with a particular performance. The first cause was that in implementing outsourcing of design tasks to suppliers, the tasks of Fiat’s engineers increasingly shifted to assigning and monitoring systems/components performances to suppliers, to be achieved in a specified time schedule at a given cost. Fiat’s engineers were setting and monitoring targets, consistent with a ‘light-weight’ design and engineering division, and their involvement in carrying out design tasks kept decreasing. However, knowledge about the interaction of components and systems to generate system performance has an important tacit knowledge component that can only be acquired through learning by doing (Zirpoli & Becker, 2008). Databases such as the product data management system (PDM) provide access to information and codified knowledge, but not to learning by doing. Co-location of Fiat and supplier
engineers – widely practiced at Fiat – did not prevent the erosion of Fiat’s design competences, due to the engineers’ decrease in absorptive capacity (Cohen & Levinthal, 1990). Such decay in design competences (and in knowledge regarding component technology) had a consequence of great importance for the research question: because Fiat engineers had increasingly decaying competences and knowledge, it was more difficult for them to guide the involvement of supplier in the development process. This showed up in the form that Fiat left ever more decisions in the development process to suppliers, also in the early phases that had a huge impact on the possible trajectories as the development project progressed. The mechanism was simple: where Fiat’s engineers were less knowledgeable than the suppliers’ engineers, they lost their capability to enter technical discussions and carry through a position there. Thereby, Fiat lost its grip on the technological trajectory of the development projects, a grave matter in a system integrator approach where the system integrator is to guide the efforts of external sources of innovation and assure that the results of their (somewhat independent) development efforts (perhaps even of modules) will be integrated to a car with high product performance.

By 2005, Fiat found itself in a massive crisis and had to react. Part of the reaction was the appointment of a new Chief Technology Officer who implemented a major reorganization of the development process in 2005, which also impacted how coordination amongst the actors in the product development process was provided. Before, the overall design task was decomposed along the same lines of the components and systems the car was decomposed in. Some tasks were then outsourced, allocating the tasks of designing the safety system, for instance, to a safety system supplier. This decomposition scheme was then applied to all vehicles. For instance, dashboards were always outsourced and for no car models did Fiat design dashboards in-house.

Fiat’s Chief Technology Officer decided to make this division of labor of development tasks the main focus of his intervention in re-organizing how Fiat managed the involvement of suppliers in the product development process (and thus, which levers it could pull to coordinate them). Fiat continued to involve system suppliers, but decided to become fully responsible for the engineering and the application of all of the most relevant systems in the vehicle. It assumed the responsibility for designing all key systems in selected projects, fully developing, i.e. designing and engineering, all the most important systems of a model that used then as a template to design and engineer all the other derivative products. These derivative development projects can then either be led by Fiat or by engineering suppliers, and the detailed design and engineering of components and systems of the derivative models can be allocated both to suppliers and to Fiat itself. Having formulated templates, Fiat realized it was possible to outsource the complete development of entire vehicles for derivative projects. Our interviews and data indicate that the move towards the new logic held the key to economize on the overall amount of resources invested in the NPD process and, at the same time, achieve unprecedented performances in all NPD projects (for details see Zirpoli & Becker, 2008).

How does the template system contribute to such performance improvements? A template project is a project that, besides developing a car model, has the additional goal to develop a bundle of archetypical solutions to be leveraged on derivative models. For template models, the integration of components and systems that affect the whole product performances is managed completely by Fiat. This creates the possibility of outsourcing ‘derivative models’ in which archetypical solutions are applied as standards that characterize products in a given market segment. Some examples of design archetypes are the architecture of a suspension for small cars, the layout of the panel instruments for sports cars, or the design of the sealing system for luxury cars.
In other words, this set of design archetypes defines a model that becomes the ‘ancestor’ which then gives rise to a family of derivative models. Following a template means that every time a derivative project is started within a segment, engineers will have to apply the template solution from the same segment (the segment’s ancestor). This is done either by carrying over the same components or, in the case of a physical misfit, by designing the new components by scaling the archetypical solution up or down.

This organizational innovation had a powerful impact on the coordination of suppliers involved in the development process: It enabled Fiat to regain its grip on the reins that allow it to align the trajectories along which the different suppliers develop the systems they are tasked with. (Keep in mind that being able to influence trajectories along which the suppliers develop the systems they are tasked with is particularly important as parallel development is standard in the auto industry, and a development project takes around 3 years. Therefore, any problem of aligning the efforts of the suppliers in developing the different systems has major consequences for product and project performance). The template enabled this in three ways: it provided the opportunity and mechanism for learning to regain lost competences, thus enabling Fiat to take more influence in technical discussions and leave less influence to suppliers in decisions that had important consequences regarding the technological trajectory. Second, the template system created a two-step structure in the development process (if we consider the development of a series of cars in one segment). Fiat first develops the ‘big outlines’ of all the most important systems, deciding for instance on the type of suspension system, the type of electronic systems and so on. Then it passes this set of archetypical engineering solutions on to the suppliers for the derivative models. The suppliers then work out and decide the details within the framework of the archetypical solutions indicated to them. The possibility space that they can use has therefore been bounded by Fiat. With the template, Fiat puts the suppliers not only to a particular starting point in the possibility space, but also sets them on a particular trajectory (a car with such and such a type of engine, suspension system, etc.). With the template, Fiat has eliminated at an early stage of the development process a large number of directions in which suppliers could go. Furthermore, the template system also provides Fiat with a means to communicate the ‘vision’ of the derivative car it wants suppliers to develop systems for, and the directions in which suppliers should go.

5.2. Evidence from SourceForge

Our analysis of the OSS context draws on data relative to all the projects populating the SourceForge platform from November 1999 to October 2008. Out of this population we selected 5810 projects registered onto the platform from January 15th, 2005, to April, 15th 2005. We tried to select the most recent projects we could, provided that we had enough data to analyze subsequent developments of the projects. Moreover, as the data consist of monthly snapshots of the situation observed on the platform, and as the way data were stored and managed changed over time, we needed to carefully select the period of analysis in order to preserve data consistency between the different snapshots.

We divided the period in three parts: a first period, called $t_0$, spans the first months of the projects’ life up to July 8th, 2005. A second period ($t_1$) starts from this date and reaches March 20th, 2006. A third time window ($t_2$) is opened from that day to September 20th, 2006.

The described data are used to investigate if the provision of an initial template at the moment of a project’s foundation leads to a reduction of the number of possible trajectories explored by the members of the team during the development of the project, and if this effect in turn generates a gain in terms of project performance.
As a first step, we need to be sure we capture these three phenomena through our variables. In the OSS environment an initial (at \( t_0 \)) version of the software to be developed can be conceived as a template. A piece of software, even if in its early stage, usually consists of a structured system of commands. However, and most importantly, even in this form it already contains solutions and mechanisms that define the space of future development of the product, that pose boundaries to the possible development paths the team can take. For example, if two functions are divided and a different module is provided for each one of them, development trajectories that imply a product design with strict interferences between these modules is likely to be abandoned – even implicitly - by all the project’s contributors. The initial code embodies its author’s vision (also implicitly and unconsciously) and bounds all the subsequent development to a specific set of possible trajectories. Thus, we can identify the template as the initial code provided by the OSS project founder. If the project has no initial code, for example because the founder has just an idea and wants to discuss it with possible collaborators before creating the initial code, then the project has no template (or the template will come later in the history of the project). This conceptualization of the role of the initial code is very close to that applied by Baldwin and Clark (2006) and Dalle et al., (2009)\(^6\).

Our idea is that without an initial template the projects will not be able to coordinate the trajectories each developer is implicitly following. This will result in a lack of coordination not simply for the ongoing processes, but also relative to the future structure of the product, as every participant will be heading towards a different direction. In order to capture the possible variety of direction the project could take as a result of these different points of view, we can rely on a specific set of data found in the dataset. As projects are self-categorized by their members into different classes aimed at specifying the intended audience of the project, the programming languages it employs, the operating systems it runs on, and so on, we can capture the “dispersion” in the contributors’ trajectories measuring the number of changes that occurred in this classification in the 8-month period spanned by \( t_1 \). Projects that remained stable in terms of their classification are projects whose developers experienced a certain level of implicit agreement since the beginning on the future developments of the product. On the contrary, if a project has undergone several changes of categorization, this can be considered the consequence of the many adjustments needed to reconcile different visions of the final (i.e. future) product among the participants.

We also want to investigate if the restriction of the number of possible development trajectories due to the presence of the template has a positive effect on the project performance. We are able to detect if the project has produced and released any file over the 6 months composing period \( t_2 \), so that we can discriminate productive projects from the others, thus accounting for performance. We also introduced some controls in the analysis, reported in the following table together with the main variables described above (see table 4 in the appendix for the summary statistics and the correlations).

---

\(^6\) In order to set a minimum threshold to the definition of “initial code”, we restricted out analysis to the provision of “running code”, i.e. to the presence of files released by the project members as official releases.
Table 2 – Variables used in the regression analysis

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATEGORIES_CHANGES(_{t1,t2})</td>
<td>number of categories the project has &quot;changed&quot;, i.e. acquired or lost, between July 2005 and March 2006</td>
</tr>
<tr>
<td>INITIAL_CODE(_{t0})</td>
<td>dummy variable equal to 1 if there was at least one file posted by the project in the first months of its activity on SF.net, i.e. between its foundation and July 8th, 2005.</td>
</tr>
<tr>
<td>CODE_RELEASED(_{t1})</td>
<td>dummy equal 1 if the project has posted at least one file between July 8th, 2005 and March 20th, 2006</td>
</tr>
<tr>
<td>CODE_RELEASED(_{t2})</td>
<td>dummy equal 1 if the project has posted at least one file in the 6 months from April 2006 to September 2006 included</td>
</tr>
<tr>
<td>MEMBERS_TENURE(_{t1})</td>
<td>registered date of those who were project members at July 2005 (average)</td>
</tr>
<tr>
<td>MEMBERS_TENURE(_{t2})</td>
<td>registered date of those who were project members at March 2006 (average)</td>
</tr>
<tr>
<td>NUM_MEMBERS(_{t2})</td>
<td>number of project members at March 2006 (average)</td>
</tr>
<tr>
<td>NUM_MEMBERS(_{t1})</td>
<td>number of project members at July 2005 (average)</td>
</tr>
<tr>
<td>REGISTRATION_DATE(_{t0})</td>
<td>registration date of the project on the platform</td>
</tr>
<tr>
<td>USE_CVS_TOOL(_{t0})</td>
<td>dummy equal 1 if the project uses the CVS, concurrent versioning system, a tool to manage distributed software development (May 2005)</td>
</tr>
<tr>
<td>USE_FORUM(_{t0})</td>
<td>dummy equal 1 if the project uses forums (May 2005)</td>
</tr>
<tr>
<td>DUMMY_CATEGORIES(_{t0})</td>
<td>dummies for development status, language, programming language, license, operating system, topic, retrieved July 2005, i.e. end of period 10</td>
</tr>
</tbody>
</table>

*Notice: dates are measured in UNIX time, i.e. in number of seconds from midnight of January 1, 1970, a standard measure in computer science.

The estimation we run is aimed at showing the relationship between the three main variables defined above. In order to do this we apply a Two-Stage-Least-Square-like technique. We run a first estimation, predict the values of the dependant variable and use it as the main regressor in the second equation. The obtained coefficients should capture the sign and significance of the relationship between the provision of initial code (INITIAL\_CODE\(_{t0}\)) and the project performance (CODE\_RELEASED\(_{t2}\)) exclusively through the effect of the former (and of the other included controls) on the number of changed categories (CATEGORIES\_CHANGES\(_{t1,t2}\)), and of this on the latter.

The first equation we need to estimate relates INITIAL\_CODE\(_{t0}\) and CATEGORIES\_CHANGES\(_{t1,t2}\). Measuring the independent variables at time \(t_0\) and the number of changes in the categorization as the delta between the beginning of \(t_1\) and its end (and also controlling for the specific categories projects listed at the beginning of \(t_1\)) should diminish the possible endogeneity problem. As CATEGORIES\_CHANGES\(_{t1,t2}\) is a count variable we could use a Poisson specification. However, the presence of overdispersion pushes us to prefer a Negative Binomial. By the same token, the presence of many zeros implies the use of an additional equation preceding the estimation stage and predicting the probability of being a project which has structurally 0 changes in the categories it lists. The results of this “zero-inflation” process are used in the main estimation equation. The final model for the first stage is then a Zero-Inflated Negative Binomial, ZINB (in the table the results relative to a Zero-Inflated Negative Poisson, or ZIP, are also reported as a robustness check). Notice that the sample of projects is also reduced to 5709 due to the fact that we excluded about 100 project that were deleted from SourceForge before March 2006.

Once the estimation of this first stage has been carried out, the predicted values of CATEGORIES\_CHANGES\(_{t1,t2}\) are used as independent variable in the second equation, which includes CODE\_RELEASED\(_{t2}\) as dependant variable. Being CODE\_RELEASED\(_{t2}\) a dummy variable, it seems appropriate to use a Logistic Regression Model to estimate the second stage of our Two-Stage-Least-Square-like model.
As said, some controls have been included in the two equations to make sure no confounding factors are at work. In particular, $\text{CODE\_RELEASE}_t$ and $\text{INITIAL\_CODE}_0$ are also included in the second stage, in order to control not only for the most recent performance of the project (that in period t1) but also for the direct effect of $\text{INITIAL\_CODE}_0$ on $\text{CODE\_RELEASE}_t$. This last passage assures that the observed effect is due to the indirect effect of $\text{INITIAL\_CODE}_0$ on $\text{CODE\_RELEASE}_t$ exclusively through $\text{CATEGORIES\_CHANGES}_{t,2}$. Another control is worth to be discussed. In order to be sure that the aforementioned indirect effect is “cleaned” by all possible influences of other factors, we also computed the residuals of the first stage (i.e. $\text{RES\_CATEGORIES\_CHANGES}_{t,2,p}$, the difference between the predicted values and the observed values of $\text{CATEGORIES\_CHANGES}_{t,2}$) and used them in the second stage as a control. The idea is that should $\text{CATEGORIES\_CHANGES}_{t,2}$ have a direct effect on $\text{CODE\_RELEASE}_t$ not captured by the prediction we have computed, this should be controlled for through the residual. The following table reports the results of the estimates.

<table>
<thead>
<tr>
<th>Stage 1</th>
<th>$\text{CATEGORIES_CHANGES}_{t,2}$</th>
<th>ZINB</th>
<th>ZIP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$b/se$</td>
<td>$b/se$</td>
<td></td>
</tr>
<tr>
<td>$\text{INITIAL_CODE}_0$</td>
<td>-0.312***</td>
<td>-0.251***</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0.094]</td>
<td>[0.083]</td>
<td></td>
</tr>
<tr>
<td>$\text{REGISTRATION_DATE}_0$</td>
<td>0.000</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0.000]</td>
<td>[0.000]</td>
<td></td>
</tr>
<tr>
<td>$\text{NUM_MEMBERS}_t$</td>
<td>0.042*</td>
<td>0.044***</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0.024]</td>
<td>[0.016]</td>
<td></td>
</tr>
<tr>
<td>$\text{MEMBERS_TENURE}_t$</td>
<td>0.000</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0.000]</td>
<td>[0.000]</td>
<td></td>
</tr>
<tr>
<td>$\text{USE_FORUM}_0$</td>
<td>-0.053</td>
<td>-0.047</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0.170]</td>
<td>[0.146]</td>
<td></td>
</tr>
<tr>
<td>$\text{USE_CVS_TOOL}_0$</td>
<td>-0.466**</td>
<td>-0.443***</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0.186]</td>
<td>[0.167]</td>
<td></td>
</tr>
<tr>
<td>$\text{DUMMY_CATEGORIES}_t$</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>$\text{Constant}$</td>
<td>-16.584</td>
<td>-19.879</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[22.675]</td>
<td>[20.744]</td>
<td></td>
</tr>
<tr>
<td>$\text{lnalpha}$</td>
<td>-0.786***</td>
<td>-0.181</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0.167]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{alpha}$</td>
<td>0.456</td>
<td>0.2820</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0.076]</td>
<td>[0.240]</td>
<td></td>
</tr>
</tbody>
</table>
As it is possible to see in the first stage the coefficient of $INITIAL\_CODE_{t0}$ is negative and highly significant. This means that the provision of an initial piece of code, which we conceived as the template, assures a decrease in the number of trajectory changes experience by the project. By the same token, in the second stage the coefficient of the predicted values of $CATEGORIES\_CHANGES_{t1,t2}$ is also highly significant, and also negative. This means that the smaller the number of changes in the categories a project lists thanks to the provision of the template, the more likely is that it has produced and released some files in the latest period considered. Moreover, as in the second stage we have used the predicted values of $CATEGORIES\_CHANGES_{t1,t2}$, and we have controlled for $INITIAL\_CODE_{t0}$ and for the residuals of the prediction, we can also state that the observed effect is due exclusively to the effect $INITIAL\_CODE_{t0}$ has on $CODE\_RELEASED_{t2}$ through $CATEGORIES\_CHANGES_{t1,t2}$.

These results allow us to conclude that the presence of the template in the initial phases of a OSS project in the form of an initial artifact, an initial piece of code, reduces the number of possible trajectories developers may implicitly follow, and that this restriction results in a gain in terms of project performance.\footnote{Please note that the econometric analysis is preliminary.}

\begin{table}[h]
\centering
\begin{tabular}{l c c}
\hline
\textbf{Stage 2} & \multicolumn{2}{c}{\textbf{LOGIT for ZINB}} \\
 & \textbf{LOGIT for ZIP} & \\
\hline
\textbf{CODE\_RELEASED}_{t2} & \textbf{b/se} & \textbf{b/se} \\
\hline
$CATEGORIES\_CHANGES_{t1,t2}$ & -0.750** & -0.739* \\
[0.342] & [0.412] \\
$RES\_CATEGORIES\_CHANGES_{t1,t2}$ & 0.107*** & 0.109*** \\
[0.023] & [0.023] \\
$NUM\_MEMBERS_{t2}$ & 0.153*** & 0.120*** \\
[0.027] & [0.022] \\
$MEMBERS\_TENURE_{t2}$ & 0 & 0 \\
[0.000] & [0.000] \\
$CODE\_RELEASED_{t0}$ & 2.376*** & 2.382*** \\
[0.115] & [0.115] \\
$INITIAL\_CODE_{t0}$ & 0.922*** & 0.932*** \\
[0.134] & [0.134] \\
$DUMMIES_{t0}$ & YES & YES \\
\textbf{Constant} & -2.136* & -2.187* \\
[1.312] & [1.312] \\
\hline
\textbf{N} & 5703 & 5703 \\
\textbf{ll} & -1277.426 & -1279.1849 \\
\textbf{Pseudo R-squared} & 0.297 & 0.296 \\
\textbf{chi2 (df)} & 1080.89 & 1077.38 \\
\textbf{Prob > chi2} & 0.000 & 0.000 \\
\end{tabular}
\caption{Results of the regression analysis – stage 2}
\end{table}
6. Discussion and conclusion

Despite many substantial differences between how coordination is achieved in OSS and automotive distributed innovation processes, we have also found similarities. In particular, we have seen that, when the discussion focuses on coordination mechanisms as applied, the two systems seem to react to the same challenges in a very similar way.

In both industries, the objective is to attain high product and project performance, and in both industries, there are similar problems in attaining these two objectives. To put it briefly, the first challenge is to create some room for involving many distributed agents in development in parallel (i.e., some form of division of labor). The second is to provide coordination that is appropriate for the division of labor that has been adopted.

It is interesting to note that in both cases, the problems with each are quite similar. As regards the first, in both cases, the implemented organizational structure or the modular architecture itself are very likely to be challenged by the emergence of complex and unforeseen interdependencies between the single components. It seems difficult to pick a division of labor, decomposition of product (product architecture), and organization structure (including a structure of the network or design value chain) that avoids the emergence of unforeseen interdependencies. As a consequence, both product and project performance suffer (in particular project performance is important here, as product performance can be recovered by costly re-design processes; the propagation costs (MacCormack et al., 2006) of these changes increase, and can create disruptive results).

With regard to providing coordination, it emerges from both cases that someone needs to guide the involvement of the distributed sources of innovation, be it independent software developers or supplier firms. This role was filled out by Fiat and the project leader in OSS, respectively, who guided the other actors to assure that the parallel developments efforts led to outcomes that could integrated to a product with high product performance. In both cases, however, there were problems with this approach. To a large extent, they are due to interdependencies that are little-understood at the outset and emerge as the development project unfolds. The consequence is that coordination requirements increase, but in the degree that interdependencies increase, the actor that is supposed to provide such coordination has less and less understanding of those and can thus provide less and less guidance. Coordination becomes more difficult and cumbersome, with negative effects on project performance.

The principal contribution of this paper is to have identified a specific mechanism through which problems with integrating inputs developed by distributed actors can be attenuated, by influencing the coordination of the joint development efforts those actors. This mechanism acts on the trajectories that actors involved in joint search efforts are inclined to take. Both the template in the Fiat case, and the initial code release in the OSS case provide a vision of future development trajectories of the product. The consequence is to exclude some trajectories (such as using alternative technologies to provide a function, for instance), and to guide distributed actors in their joint search effort. The effect is very powerful as this happens at the very beginning, where each decision about trajectories has a huge impact due to path dependencies. The template thus is a means for leveraging coordination by influencing cognitive representations about what it is that actors have to develop. The template helps to making the lead actor’s vision about the trajectories (more) explicit; to improve the communication about these trajectories (here, the physical quality of the template seems to make an important difference); and to provide visibility of those trajectories, which means that also actors other than the lead actor can independently discuss them and relate to them (such as different developers or suppliers amongst themselves).
The template appears to be a means of coordination that has emerged in situations of distributed innovation. The fact that we have identified it in two very different industries supports the idea that it might be a mechanism that fits the coordination requirements of coordination joint search. In directing search efforts of distributed actors, it appears to be an alternative to direct supervision that is less available and less strong in contexts of distributed innovation. In any case, it has beneficial effects on product and project performance and therefore merits a closer look as a means for coordination of distributed innovation.

References


Clark K.B., and Baldwin C.Y., (2001), Modularity after the Crash, Harvard NOM Research Paper No. 01-05


### Table 4 – Summary statistics and correlation table of the main variables used in the regression

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>N</strong></td>
<td>5810</td>
<td>5810</td>
<td>5810</td>
<td>5810</td>
<td>5810</td>
<td>5810</td>
<td>5810</td>
<td>5810</td>
<td>5810</td>
<td>5797</td>
<td>5797</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td>0.357315</td>
<td>0.4373494</td>
<td>0.1874355</td>
<td>0.095525</td>
<td>1.11E+09</td>
<td>1.616007</td>
<td>1.566471</td>
<td>1.07E+09</td>
<td>1.720718</td>
<td>1.07E+09</td>
<td>1.720718</td>
</tr>
<tr>
<td><strong>Std.</strong></td>
<td>1.627478</td>
<td>0.4961021</td>
<td>0.3902943</td>
<td>0.293964</td>
<td>2225876</td>
<td>1.566471</td>
<td>4.35E+07</td>
<td>4.33E+07</td>
<td>1.880526</td>
<td>4.33E+07</td>
<td>1.880526</td>
</tr>
<tr>
<td><strong>Median</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.11E+09</td>
<td>1</td>
<td>1.09E+09</td>
<td>1</td>
<td>1</td>
<td>1.09E+09</td>
<td>1</td>
</tr>
<tr>
<td><strong>Min</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.11E+09</td>
<td>1</td>
<td>1.09E+09</td>
<td>1</td>
<td>1</td>
<td>1.09E+09</td>
<td>1</td>
</tr>
<tr>
<td><strong>Max</strong></td>
<td>21</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.11E+09</td>
<td>39</td>
<td>9.42E+08</td>
<td>9.42E+08</td>
<td>41</td>
<td>9.42E+08</td>
<td>41</td>
</tr>
</tbody>
</table>

**1 CATEGORIES_CHANGES**

- **INITIAL_CODE**:
  - 0.016
  - 1

- **CODE_RELEASEDT**: 0.2241*
- **CODE_RELEASEDT**: 0.1467*
- **REGISTRATION_DATE**: 0.0195
- **NUM_MEMBERS**: 0.0469*
- **MEMBERS_TENURE**: -0.0001
- **USE_FORUM0**: -0.0096
- **USE_CVS_TOOL**: -0.0490*
- **NUM_MEMBERS**: 0.0773*
- **MEMBERS_TENURE**: 0.0111

**2 INITIAL_CODE**:

- **CODE_RELEASEDT**: 0.2399*
- **CODE_RELEASEDT**: 0.4606*
- **REGISTRATION_DATE**: 0.0299
- **NUM_MEMBERS**: 0.015
- **MEMBERS_TENURE**: -0.0259*
- **USE_FORUM0**: -0.1080*
- **USE_CVS_TOOL**: -0.1087*
- **NUM_MEMBERS**: 0.0269*
- **MEMBERS_TENURE**: -0.0231

**3 CODE_RELEASEDT**: 0.3429*

**4 CODE_RELEASEDT**: 0.4606*

**5 REGISTRATION_DATE**: 0.0117

**6 NUM_MEMBERS**: 0.1032*

**7 MEMBERS_TENURE**: 0.0359*

**8 USE_FORUM0**: 0.0393*

**9 USE_CVS_TOOL**: 0.03159*

**10 NUM_MEMBERS**: 0.0596*

**11 MEMBERS_TENURE**: 0.0801*