***Organizational arrangements in Commons-Based Peer Production.***

***The Case of Debian and Take the Bus Project.***

**Inna Lioubareva and Barbara Feledziak**

EconomiX, Paris X-Nanterre

inna.lioubareva@u-paris10.fr

***Abstract.***

This paper analyses the emergence of organizational arrangements in Commons-Based Peer Production (CBPP). It draws on the evolutionary and competence-based theories and substantiates how the specific arrangements − division of labour, authority structure and approach to conflict resolution − permit to keep the integrity of projects and to ensure efficient cooperation. The paper argues that whereas self-organization is the prevalent form in the CBPP, Free Licences play an important role in structuring interactions. It demonstrates that this institution explains the emergence of specific organizational arrangements within the CBPP. Thereto, two different organizational architectures are put forward: Take the Bus Project published under the Creative Commons licence (collective music creation) and Debian published under the GPL (collective software development). Finally, the paper substantiates that the choice of a licence for the projects is non-random: coordination introduced by the licences meets the inherent characteristics of the software and music contents.

*Keywords:* Evolutionary approach; Collective production; Self-organization; Licences; Production coherence; Decomposability.

***Acknowledgements.***

We gratefully acknowledge the helpful comments and suggestions from Eric Brousseau, Michel Delapierre, Frederic Gannon, Fabrice Rochelandet and Maria Alessandra Rossi, as well as the permission to use the study on the Open Source Software realized in July 2005 by Nicolas Auray and Michael Vicente, ENST, Paris. We also thank the leader of Take the Bus project, Philippe Chavaroche (Ana Boat), and the creator of the Musique-libre.org platform, Bituur Esztrem, for informative discussions. None of those who have helped can be held responsible for the defects that remain, or for the views expressed here.

***Introduction.***

The emergence and expansion of a specific mode of production called the Commons-based Peer Production[1] (CBPP) is a recent phenomenon. The specificity of the CBPP consists of two aspects: the way the products are created and its property philosophy [West and Gallagher, 05].

First of all, the membership, in the case of CBPP, is relatively open. As a result, participants with different skills and experiences, geographically[2] dispersed, can exchange their *works-in-progress* while gaining "extra sets of eyes to catch mistakes, identify problems, and improve quality" [Lee and Cole, 03]. Secondly, the coordination of the CBPP decentralized and distributed system is ensured by the scope of informal norms, without relying on methods of command and control. The formal rules are represented only by means of free licences, such as the GNU General Public Licence (GPL), the Creative Commons licence (CC), the BSD licence, etc. Despite some important differences in the structure of these licences, their essence consists of the idea that "authors do not renounce their rights but only the monopoly rent such rights would authorise in a copyright regime" [Jullien and Zimmermann, 06]. In practice it means that functional work (such as software, encyclopaedias or dictionaries), artwork or other creative contents are publicly available (for example, on the Internet) and can be shared among users [Ibid.].

On the one hand, "…connecting a large amount of individuals around a common project and without any closure… is a way to take advantage of a fantastic potential of

[1] The concept was firstly defined as "Commons-based peer production" by Y. Benckler in "Coase's Penguin, or Linux and the Nature of the Firm" [Benkler, 02].

[2] The fact that participants in the CBPP are usually geographically dispersed doesn't deny that depending on the task, in some situations, "physical" face-to-face contacts are indispensable. For example, in the Debian OSS project the role of collective meetings is crucial. However, by contrast to some firm-based model these "physical" contacts are of periodic nature.

distributed skills" [Ibid.]. This efficiency of the CBPP in terms of product quality (its adequacy to the users' needs, security and variety) is largely discussed in the literature (for example, among others [Lee and Cole, 03]; [Bonaccorsi and Rossi, 03]; [Moon and Sproull, 02]; [Benckler, 02]). On the other hand, by connecting heterogeneous individuals working together in such a decentralized and distributed manner, the CBPP could make it more difficult to ensure the coherent and synergic cooperation: to achieve effective cooperation, it takes complex processes of individual and collective learning facilitated within the shared organizational context ([Johnson, 92]; [Daft and Weick, 84]).

The purpose of our study is to examine the emergence of particular organizational arrangements which allow the eventual efficiency of collective innovation in the CBPP. This paper focuses on the analysis of the link between self-organization and institutions in the CBPP. It argues that whereas agents make their choices mainly on the bases of their informed interpretations of local information and self-organization is the prevalent form [van Wendel de Joode, 05], there is one institution, namely Free Licences, which precedes the interactions among participants in the CBPP. We show that this institution plays an important role in structuring interactions and explains, at least partially, the emergence of organizational arrangements within the CBPP.

We explore the competence-based and overlapping evolutionary approaches ([Nelson and Winter, 82]; [Dosi and al., 00]; [Hodgson, 98]; [Marengo and al., 00]) considering organizations as a "repositories of knowledge"[3] and analysing the relation between organizational architecture and organizational knowledge. According to this literature,

[3] Organization can be defined as "a *structure of elements* (subsidiaries, divisions, teams, individual people) that have resources and *repertoires of actions* (competences), with *decision rules* that govern *choice* from those repertoires, to achieve *goals*, in *co-ordination* (which include governance) between those elements." [Nooteboom, 04]

“organizational forms map into diverse a) problem representations, b) problem decompositions, c) task assignment, d) heuristics for and boundaries to exploration and learning, e) mechanisms for conflict resolution over interests but also over alternative cognitive frames and problem interpretations.” [Marengo and al., 00] With respect to these dimensions we will analyse the emergence of three types of organizational arrangements in the CBPP: authority structure characterizing the instance of decision making and underlying problem representations; labour division and distribution of roles characterizing decomposition rules, process of task assignment and boundaries to exploration; and mechanisms for conflict resolution. Our analysis relies on the comparative study of two distinct projects representing the CBPP – the Debian project (GPL) and the Take the Bus Project (CC licence) - which correspond respectively to collective software and music creation.

The first section (1) introduces the Debian and the Take the Bus projects as representative examples of the CBPP in their respective domains: it describes the basic principles of the projects’ functioning (1.1 and 1.2) and the licences these projects are published on (1.3). The second section (2) aims at the analysis of the structuring of particular organizational architectures in Debian and in Take the Bus and puts forward that these architectures emerge in response to the coordination principles introduced by the licences: it compares the projects following the above-mentioned aspects: authority structure (2.1), labour division (2.2) and conflict resolution (2.3). The third section (3) shows that the publication of the projects under particular licences is a non-random choice: it demonstrates that coordination form introduced by the licences meet the inherent characteristics of the software and music contents which are very different by nature.

## *1. CBPP and Free Licences: Debian and Take the Bus Project.*

The most well-known case where the CBPP is applied is the development of the open source software (OSS). However, its range of application has expanded quickly to other knowledge-intensive domains such as music and literature. Our analysis of the organizational arrangements within the CBPP relies on the comparative study of the cases of collective music (Take the Bus Project) and software creation (Debian). Both cases exemplify the CBPP with its specific features described above, but some important differences lie between these projects. One of the most important ones concerns the size of the projects and their life span: the Debian project has existed since 1993 and involves thousand of participants; the Take the Bus Project was initiated only in September 2005 and consists of thirty artists. However both projects are to a certain extent representative of their domains and, hence, permit us to reach some general conclusions about the CBPP functioning.

### *1.1. Debian.*

The Debian project, published under the General Public Licence (GPL), was initiated in 1993 by Ian Murdock, a student from Purdue University. It was the first attempt to create a complete operating system distribution[4] based on the Linux kernel. Debian operation system includes a large number of applications and 1400 different software components, called "packages"[5]. The development of Debian consists in the accomplishment of different tasks such as packaging, development of patches and bug fixes, support of the server infrastructure, translation of documents and web pages, development of specific management tools, etc. Packaging remains one of the most important activities within the project and most of the

[4] In the OSS, "distribution" is the term designating an official release of the operating system ready to be installed on the end-user computer.

[5] To create a distribution, it is necessary to implement different functionalities. Package is a sub-part of the operating system which permits to add a particular functionality. "Packaging" generally consists in making the source code ready to be installed.

developers work on it. Packaging does not necessarily involve writing of source code (sometimes the developers use the code developed by others). The source code itself is a type of prepared raw material, whereas packaging corresponds to the creation of a software ready-to-use for specific purposes. The integration of a package to the distribution is a complex operation which imposes many constraints: software should correspond to the GPL licence's terms, and to a large number of technical features to ensure the package's compatibility with other parts of the Debian distribution.

During the span of the project's existence, eight distributions (versions) were officially published and the ninth is expected soon. The fact that today Debian unifies nearly one thousand of developers from all over the globe and, over the years, remains the most popular free distribution of the Linux operating system, exhibits the consistency of the development process in Debian.

### *1.2. Take the Bus Project.*

The Take the Bus project is relatively new as well as the whole CBPP initiative in music (the oldest project, CC Mixter, has existed since 2002). Take the Bus was initiated in 2005 by Philippe Chavaroche and gathers today thirty musicians from different collectivities and countries. The project is published under the Creative Commons (CC) licence.

Since the beginning of Take the Bus, fifteen music compositions were collectively created and all of them are available on several music platforms on the web. According to the statistics published on the Musique-libre.org website, the last composition of Take the Bus was downloaded and streamed 550 times in the span of one month since its creation. The composition posted one year ago was downloaded and streamed 980 times, which is more than the average number of times on this platform (the average is 840 times). At present the

Take the Bus project has already gained recognition within the free music movement and several new compositions are in the pipeline.

The development of music in this project proceeds through several stages. Firstly, the leader of the project provides some information about the tempo and the lyrics that serve as a common base. On this basis contributors create some pieces of music and send their folders to the leader, whose most important role consists in the selection and final arrangement of numerous folders in order to create a single music composition. If the contributors are satisfied by the result, then, they create collectively the cover for the final composition and provide it with the tags to signal the terms of the selected license for future users. The next step is to upload this "package" on the server and to diffuse it on the different platforms. At this stage the resulting creation becomes a topic discussed among the listeners (members of the project and others). Then, the composition evolves on the basis of the remarks and propositions received.

The main feature of the Take the Bus project which distinguishes it from other music projects based on the CBPP is that the sphere of its application is not limited to an exclusive musical style[6]: all the compositions created within Take the Bus belong to different musical styles. We think that this feature is of particular importance, because it indicates that the application of the CBPP in music can be manifold. From this viewpoint we propose to consider the functioning of the Take the Bus project as a representative example of the CBPP in this area.

[6] For instance, CC Mixer specialises only in the electro music, HiphopDomain creates rap music and Featuring Album produces French Music.

*1.3. Licences in Debian and Take the Bus Project.*

The development of all the projects within the CBPP starts with the choice of a licence. The Debian project is published under the GPL which was chosen by the project's initiator. This licence mobilises in a specific manner the similar principles as traditional copyright, but in order to protect four essential users' rights: freedom to run the program, to study how it works, to modify the program, and to redistribute it. It is based on the copyleft and the paternity principles. Copyleft corresponds to the restriction that all new pieces of code, "derived work", must be licensed under the GPL and, hence, the source code must be available to all users of the software. The paternity principle means that all the contributors participating to the software creation have to be named in the derived work. The last but not the least principle introduced by the GPL is the joint decision making and the norm of authority based on collective recognition: while being first of all a political initiative, the publication of the project under GPL signalises a high level of internal democracy.

The Take the Bus Project is published under the CC licence. The basic principle of this licence is that authors have the possibility to choose from an array of options, in particular to permit or not the commercial use or derivative works, as well as to require or not attribution and share alike. By contrast to the Debian project, in Take the Bus the licence, or more precisely the *terms* of the licence, become themselves the result of a negotiating process between the leader and potential contributors. Potential contributors, who decide to participate, express their opinion about the licence terms. Once the consensus among the potential participants and the leader is reached, the production starts. In the case of the Take the Bus Project, CC imposes the respect of the paternity right, does not permit the use of the product in a commercial way and forbids modifications (and hence does not contain a share-alike clause).

Thus, one can mention two important differences between the aforementioned licences. Firstly, the publication of Debian under the GPL represents an approach “to take or to leave”: potential participants either agree on the licence’s terms or they do not join the project. An opposite situation takes place in Take the Bus: potential participants express their opinion on the terms of the licence before the launch of the main production process. Secondly, GPL in Debian introduces an *ex post* coordinated system, where no decision rights are fixed before the development itself (the leader is not necessarily the initiator of the project). By contrast, CC in Take the Bus fixes *ex ante* the allocation of decision rights (non-modification clause) and the restrictions on the use (only non-commercial use). Next section will demonstrate that the licence choice influences the emergence of particular organizational arrangements in Debian and Take the Bus projects.

### *2. Emergence of organizational arrangements in CBPP: role of free licences.*

In this section we raise the question of how individual actors and groups dispersed across time and space co-ordinate their efforts to achieve coherent and synergic innovation processes. The existent literature on CBPP organization provides two distinct types of explanations: the first one is based on self-organization (e.g. [Axelrod and Cohen, 99]; [Kuwabara, 00]; [Lanzara and Morner, 03]); and the second one focuses on the role of institutions (e.g. [Bonaccorsi and Rossi, 03]; [McGowan, 01]; [O'Mahony, 03]). We propose to consider them as complementary: we will show that the licences, which represent the only type of institutions preceding the actors’ interaction, play an important role in orienting further self-organization and in determining the emergence of particular arrangements in CBPP.

Our argumentation finds its root in the studies on the notion of coherence in the economics of organizations (c.f. [Dosi and al., 92]; [Cohendet & Llerena, 01]). The key idea of these studies is that to create collectively a product goes far beyond the sharing of distinct ideas, but necessitates complex processes of individual and collective learning to ensure the compatibility of emergent rules, norms and routines. In this view, "coherence… involves the creation of commonly shared bodies of knowledge… which are – at least partly – known to all the members of the organization involved in a given interaction" [Cohendet & Llerena, 01]. However, depending on the articulation between *ex ante – ex post* coordination in a system, coherence may have different attributes: "compatibility between individual plans" in *ex-post* coordinated systems *versus* "compatibility of the collective behaviour with the system objective" in *ex-ante* coordinated systems [Cayla, 06]. Consequently, distinct organizational arrangements determine the enhancement of coherence. In *ex-post* coordinated systems coherence increases with the emergence of commonly shared "abstract rules" [Hayek, 73] (e.g. routines) which result in actors' interactions. On the contrary, in *ex-ante* coordinated systems the degree of coherence depends on the extent to which the rules are clearly defined by central authority and respected by other members.

We will demonstrate that the organizational arrangements enabling coherent development processes in Debian and Take the Bus have different architectures, and that these architectural differences result, at least partially, from the articulation between *ex ante – ex post* coordination introduced by the licences.

*2.1. Authority structure.*

Since in the CBPP the volunteers participate in collective production, the authority structure can positively influence the venue of newcomers only if they find it appropriate and

effective. We suppose that this is one of the reasons why there can't be any other form of leadership than that based on collective recognition and joint decision making. However both projects represent different approaches to the implementation of this norm.

As we have already mentioned, GPL substantiates the norm of the leadership while being a political initiative: the publication of a project under the GPL implicitly introduces the leadership based on collective recognition and joint decision making. In this effect, in the Debian project, the leader is yearly elected by the developers[7]. The instance of decision making includes also so-called "admitted" or "official" members – participants whose contributions have been accepted for official distribution: the integration of individual contributions to the Debian distribution is a collective decision of the leader and the maintainers. Official members with the project's leader form the core of the project.

By contrast, in Take the Bus the project's leader is the initiator of the project providing the material (tempo and lyrics) to start. The norm of the leadership is introduced via the collective choice of the licence's terms: firstly, while choosing among the licence's terms, contributors participate in decision making; then, the establishment of the non-modification clause, signals the recognition of the leader's competencies by the contributors.

On the one hand, in both projects the leadership is assumed on the basis of collective recognition of the leader's competences and provides the ground for the cooperation development among the members: all the contributors have agreed on the respect of this basic norm. On the other hand, contrary to the Debian project, in Take the Bus the leadership is fixed explicitly "once and for all" by means of the licence: the leader who has the right to

[7] Collective leader elections in the Debian project mean that "anybody who is already admitted to the project is not only eligible to vote in this election but eligible to nominate themselves for the leadership position" – says the current leader of Debian, Branden Robinson [Auray and Vicente, 05]

make the final decision cannot be changed along the development process even if other participants do not agree to it[8]. So, authority in Debian is rather flexible and sensitive to the change in participants' individual preferences and objectives, thus stimulating their eventual compatibility through the ongoing negotiation. At the same time in Take the Bus the relatively rigid authority defines the final objective of collaboration (creation of music composition on the bases of particular tempo and lyrics); and coherence in this case is ensured via the establishment of the exclusive rights on the final components arrangement.

*2.2. Labour division.*

In the CBPP there are no explicit formal constraints for tasks accomplishment (like deadlines, task assignment or other duties). In Debian and in the Take the Bus Project the participation of contributors is based on so-called "emergent" distribution of roles: as opposed to being assigned to carry out a task, contributors choose any task depending on their own needs, interests and competences.

In Debian the efficient completion of a task (for example, to make a package) consists of software development, update and support along with the evolution of the main project (because the software components of the operating system are technically interdependent). In Take the Bus, artists propose some music compositions and, then, subsequently provide them with support and evolutions, like video or texts. Thus, in both cases in order to achieve efficient functioning of the projects, the author has to realize the task from the beginning to the end.

[8] For example, while making the arrangement of contributions the leader in principle can distribute it on the music platforms without an approval from other members.

To mobilise voluntary joint distribution of roles and to support the projects' evolution it is important to make the participants responsible for the performance of their contributions. The most obvious solution, which could enhance personal responsibility of the project's member, is that developers identify themselves and each others according to the tasks they have made. GPL puts in action this solution through the paternity principle which implies that all the contributions to Debian are not anonymous: for example, each package must contain the name and the email of its author. The CC in Take the Bus acts in a very similar manner as the GPL does: due to the attribution clause, all participants, the leader as well as the contributors, are collectively responsible for the final music compositions which will be diffused on different web platforms.

As a result, in both projects the "emergent" distribution of roles coupled with the principle of paternity has important consequences for the coherence of the development process. In Debian it stimulates interactions among participants: to provide better performance of their contributions, the projects' members have to follow the evolution of the project and, thus, to co-ordinate their contributions with the contributions of other participants. In the Take the Bus Project the attribution clause encourages contributors not only to propose their ideas, but to contribute to the project's maintenance and support while improving the project's performance.

At the same time, there are some differences in the logic of functioning between Debian and Take the Bus. Despite the fact that contributors can choose any task to perform according to their own preferences, in Debian there is a sharp distinction between the "core" and the "periphery". The periphery consists of non-admitted members who contribute to the project by realizing some simpler tasks: for example, patch making, debugging or translation. Whereas any participation is possible at the periphery, the integration of contributions of non-

admitted developers to Debian distribution is rather a long process. Firstly, in order to become an official member, the newcomer (contributing at the periphery) has to establish contacts with the "core" developers and ask him/her to "sponsor" him/her. Then, the sponsor checks the technical features of the package proposed by the newcomer, submits the newcomer to specific tests to be sure both if the newcomer is competent to enter the project, and if his/her ideas correspond to the ideology of Debian. Finally, the sponsor expresses his/her opinion concerning the possibility of integration of the newcomer's contributions to the Debian distribution and his/her admittance as an official member [Auray and Vicente, 05].

In contrast, in Take the Bus the leader publicly provides the material which serves as the common base for potential contributors, thereby newcomers may choose between starting to learn at the periphery or sending their contributions directly to the leader and becoming at once official members if their contributions are selected.

We think that the separation between the core and the periphery as the specific arrangement in Debian results from its orientation to the *ex post* coordination via the negotiation process. On one hand, GPL introduces a very 'open' authority structure based on collective decision making all along the development process described above (contrary to the CC licence). On the other hand, the development of an operating system still necessitates a certain co-ordination among the contributions, due to high interdependencies between the components. So, the most important problem which could entail discordance and discontinuity is the problem of crowding: it is easier to co-ordinate the contributions if the fluctuation of participants and contributions is not very intensive. Moreover, to resolve some complex tasks it usually necessitates repeated and durable interactions of the key contributors who are in closer association with each other, than with other contributors. That's why there is a certain barrier to entry to the core which serves as an implicit coordination mechanism,

necessary to support *ex post* negotiations among developers, provides an additional protection of relative intensity and longevity of interpersonal relations in the core and has important advantages for learning in context by means of "Legitimate Peripheral Participation"[9] [Lave and Wenger, 91].

*2.3. Conflict resolution.*

Connecting a large number of participants with different skills and experiences within a common project could result also in a large number of conflicts. In both projects the conflicts are resolved mainly by means of negotiations among the core participants. Some implicit mechanisms are mobilised to support the negotiation process: for example, mailing lists and Concurrent Versioning System in Debian; or forums in Take the Bus (see [Egyedi and van Wendel de Joode, 04] for a detailed discussion). There is however a difference between the two projects in approaches to the situations when the consensus cannot be reached among developers.

In contrast to the Take the Bus Project, in Debian, developers have the right to leave the project and to initiate their own project in parallel (the *fork*) [van Wendel de Joode, 04]. The *fork* is "what occurs when two (or more) versions of a software package's source code are being developed in parallel with once shared a common code base, and these multiple versions of the source code have irreconcilable differences between them."[10] To enable this option, the GPL licence allows to modify the program and to redistribute it, and in such a way this licence permits the "exit option" [Hirshman, 1970] as an approach to conflict resolution.

[9] "Legitimate peripheral participation provides a way to speak about the relations between newcomers and old-timers… A person's intentions to learn are engaged and the meaning of learning is configured through the process of becoming a full participant in a socio-cultural practice." [Lave and Wenger, 91]
[10] http://jargon.watson-net.com/jargon.asp?w=fork (last visited on January 5th, 2007) [cited in: van Wendel de Joode : 2004]

On the one hand, this option can negatively influence the continuity of the project and entail the value destruction [van Wendel de Joode, 04]. On the other hand, the exit option can be viewed also as a very important additional coordination arrangement in Debian. Due to the high interdependence among the project's components, suspense in one of the parts can entail important negative consequences for the development of the whole project. GPL introduces *ex post* negotiations as the key coordination mechanism along the project's development. So, when some important discrepancies take place and when the negotiation processes do not manage to solve the conflict, the option to exit permits to reduce the number of conflicts and to go on with the work in progress[11]. There might be even some advantages at the level of the project[12], and at the level of the variety of the products in OSS. The exit option can only have important negative consequences for the initial project development, if the new project does not belong to the OSS (if the development process is closed): in this case the collective work of many developers will be appropriated by one (or several) participants who decided to leave the initial project. To protect the rights of contributors and to permit the exit option, the copyleft principle in GPL guaranties that any fork initiated on the bases of the Debian project will also be open.

On the contrary, in Take the Bus via the collective choice of the CC licence's terms all the members *ex ante* cede the right of the final word in conflict situations to the project's leader. Even in the case of serious conflicts within the project, the Take the Bus Project's

[11] One of the Debian former leaders said: "It is essential that you can decide to leave if you do not agree. There is a rule, that you should not do it if you do not think that it is strictly necessary. But sometimes it is necessary!" [Cited in: van Wendel de Joode, 04].

[12] There may be an opportunity to improve the quality of the product by integrating some relevant solutions created in the framework of the fork. For instance, the Python project is a good example where in the OSS the exit option positively influences the continuity and advancement of the project. One of the active members of Python, Armin Rigo, had an idea on how to make the Python programming language faster, but his idea contradicted with some common viewpoints from other members. In order to bypass the resistance, he has decided to leave the project, to initiate his own and to specify his ideas in the framework of this new project, Psycho. In such a way, he proved the appropriateness of his solution. Finally, the Phyton's project leader recognized that this solution should be included in the initial project.

members can use only the “voice option”: a contributor or a group of contributors do not have the right to initiate in parallel their own project on the basis of music compositions developed in the framework of Take the Bus. Thus, the development of this project is only possible in its initial form. This approach permits to keep the integrity and continuity of the project, but the excessive use of the voice option might lead to new conflicts [Hirshman, 1970].

## *3. Licence choice and product architecture.*

In previous section we have described the organizational arrangements in Debian and in Take the Bus Project. We have shown that the differences in the projects’ organization are conditioned, at least partially, by the licences under which the projects are published. This section raises the question about the choice of a licence for a particular content: Why the CC licence which introduces the elements of *ex-ante* coordination is applied for the collective music creation and not for the software development? Or *vice versa* why the GPL introducing *ex-post* coordinated system is mainly used for the collective software development and all the existent music projects are published under the CC? We argue that one of such factors rely in the nature of the product: since music and software products are very different by nature, the development of these products can imply different coordination requirements.

### *3.1. Software and music products’ architectures.*

The literature on the product’s architecture (e.g. [Ulrich and Eppinger, 00]; [Ulrich, 95]) states that the manufacturer has “substantial latitude in choosing a product architecture” [Ulrich, 95]: in other words, the producer can adopt different schemes of allocation of product’s functions to physical components from completely modular to integral solutions. In accordance with the existent typology [Ibid.] the following “ideal” types of mapping between functional elements and physical components can be distinguished: “one-to-one mapping”

where one function is allocated to one component; “many-to-one” where many functions are allocated to one component; and “one-to-many” where one function is performed by many components. “A modular architecture includes a one-to-one mapping from functional elements in the function structure to the physical components of the product, and specifies de-coupled interfaces between components. An integral architecture includes a complex (non one-to-one) mapping from functional elements to physical components and/or coupled interfaces between components.” [Ibid.]

However one can suppose that some products can *a priori* have very different structures. For example, software and music contents are very different by nature. Software represents an example of the product, the role of which is to enable the subsequent creation of value. Music is usually in itself the “final product”, destined for the satisfaction of the users’ tastes. From this viewpoint, the component-function mapping is naturally different in these two examples. The structure of operating system is not modular in the strict sense, because the interface (Application Programming Interface, API) cannot be completely decoupled and a change in one component often induces some change in other components. However, its structure is closer to the case of one-to-one mapping: each software application represents a particular function in the operating system. By contrast, in the case of music composition the interface (tempo and lyrics) can be easily decoupled, but the independently created components do not constitute the single whole (as in the case of operating system). Music composition is closer to the case of one-to-many mapping: the aesthetic function of the music composition is enabled by the components *and* by the final arrangement of these components. So, one can say that the music content is inherently more integrated than the operating system: changes in one of the components and/or new arrangement directly influence the

major aesthetical function of music composition and give birth to a new product (evidently it is not the case for an operating system).

*3.2. Products' architectures and coordination requirements.*

The product's architecture relates to the product's performance [Ulrich, 95; Clark and Fujimoto, 91] or "how well the product implements its functional elements" [Ulrich, 95]. In particular, in the case of more integrated products the performance targets can be achieved only via the optimisation of "global performance characteristics" (in contrast to the optimisation of "local performance characteristics" in more modular products) [Ibid.]. Consequently, different products' architectures may require specific coordination approaches along the products' development.

As it was presented in the previous section in the CBPP authority is based on collective recognition, there are no formal constraints for task accomplishment, and membership is relatively open. However, the adoption of this mode of production for the development of a product which is inherently integrated may impose some supplementary coordination requirements.

In this effect, the case of collective music creation, which is inherently more integrated, raises some supplementary problems. In particular, the system-level design (development of the product architecture) requires the definition of a "system assembler" in addition to the development of communication interface: in the integrated products "whether the components meet their performance targets depends on their interaction and not on whether they meet some pre-specified criteria" [Ulrich, 95]. Thus, to test the product's performance, it is necessary to assemble the components and consider them as a whole. Presented in this way, the fact that the collective music creation induces the development of the common interface

and the *ex-ante* allocation of the decision right on the components arrangement seems to be an obvious coordination solution.

In the case of operating system, which is inherently more modular, the focus of the system-level design is to define an appropriate component interface, specifying communication protocols between the components and encouraging innovation: in view of high interdependencies between the components and uncertainty about the evolution of the product's structure, the *ex ante* specification of the common interface can limit the search space at the level of individual contributions and impede the whole dynamics of innovation [Zabel and Zeitlin, 04]. In the OSS the problem of interface specification is solved in a specific way: no interface specification is defined as a common base, but it emerges and evolves along with the evolution of the project and comes as a result of a negotiation process among participants. Contributors either choose to create independently any solution they want while respecting the common API; or, if a new solution calls for the interface redesign, they submit this question to other participants. This "open and emergent" interface allows at the same time the unlimited search space at the components level and certain coherence of production process by means of structuring the shared norm of API.

To summarize, we would like to underlie two interesting points:

Firstly, the CBPP allows important advantages for the content development. For example, the absence of formal constraints and authority based on collective recognition influence the conditions of work[13] and as a result the participants' efforts and the quality of the product [Mayo, 49; Akerlof, 82]. Open membership in the CBPP is an effective solution

[13] In CBPP the leader does not decide the tasks to accomplish or the deadlines. Consequently, the content development corresponds with the *creation* rather than *production* process, where the creation is not alien to the creator and where the participants can realize themselves through their work.

because it permits to cope with the “identifying and assigning of human capital” [Benckler, 02] and to take advantages from distributed knowledge.

Secondly, in these conditions the choice of a licence seems to be non-random. Since the range of application of CBPP includes the products which have different inherent internal structures, the adoption of the CBPP for their development might demand different coordination requirements. We have seen that in Debian and Take the Bus Project, licences explain, at least partially, the emergence of the projects-specific organizational arrangements. Now we can say that these arrangements play the important role in enabling modular architecture for the development of products with different types of coupling among the components.

***Conclusion.***

We have analysed two examples where the CBPP is applied for software and music collective creation: The Debian and the Take the Bus projects which can be considered as representative of their respective domains. Our analysis is just the first step towards the study of the CBPP functioning and a far more elaborated study is necessary to describe the organizational forms mobilised in the CBPP.

However, the comparative study of Debian and Take the Bus permits us to reach some general assumptions about the CBPP.

First of all, we can say that authority, as well as the “emergent” distribution of roles and the resolution of conflicts by negotiation play an important role for the dynamic and coherent development of the projects in the CBPP.

Secondly, despite the fact that there are common features in organizational arrangements of the projects; rather than talking about *the* CBPP organization, the scope of production specific architectures should be studied: even the processes of software development could differ depending on the concrete product, all the more are the divergences between different domains of CBPP (like between software and music).

Finally, we suppose that the role of the free licences can be much larger than it is usually presented in the literature: free licences not only allow knowledge sharing, but they can also serve as coordination mechanisms stimulating individual and collective learning through the introduction of organizational arrangement effective for the concrete developments process.

The future work should be based on the comparison of a larger number of projects from different domains. This will permit to provide a sort of typology of organizational arrangements of the CBPP and, in such a way, to explain the efficiencies and failures of this mode of production for the development of various contents.

REFERENCES

Akerlof, G.A., 1982. Labor Contracts as Partial Gift Exchange. *Quarterly Journal of Economics*, 97, 543-569.

Auray, N., M. Vicente, 2005. Empirical study on the Open Source Software realized in July 2005 by Nicolas Auray and Michael Vicente. ENST, Paris.

Axelrod, R. and M.D. Cohen, 1999. Harnessing Complexity: Organizational Implications of a Scientific Frontier. New York: Free Press

Benkler, Y., 2002. Coase's Penguin, or, Linux and The Nature of the Firm. Yale Law Journal, 112, pp. 369-446.

Bonaccorsi, A. and C. Rossi, 2003. Why Open Source Software can succeed. Research Policy 32(7), pp. 243-58

Cayla, D., 2006. Ex Post and Ex Ante Coordination: Principles of Coherence in Organizations and Markets. Journal of Economic Issues, Vol. 40, N°2, pp. 325-32.

Clark, K., and T. Fujimoto, 1991. Product Development Performance. Boston, MA: Harvard Business School Press.

Daft, R., K. Weick, 1984. Toward a model of organizations as interpretation systems. Academy of Management Review 9(2), pp. 284-295.

Dosi G., M. Hobday and L. Marengo, 2000. Problem-Solving Behaviours, Organisational Forms and the Complexity of Tasks. LEM working paper, St. Anna School of Advanced Studies, Pisa.

Dosi, G., D. J. Teece and S.G. Winter, 1992. Towards the theory of corporate coherence: Preliminary remarks. in: Dosi, G., Gianetti, R., Toninelli, P.A. (Eds), Technology and Enterprise in a Historical Perspective. Oxford University Press. Oxford.

Egyedi, T. M., R. van Wendel de Joode, 2004. Standardization and Other Coordination Mechanisms in Open Source Software. International Journal of IT Standards and Standardization Research, Vol. 2, No. 2, pp.1-17, 2004.

Hayek, F.A., 1973. Law, Legislation, and Liberty. Vol. 1: Rules and Order. Chicago: The University of Chicago Press.

Hirschman, A. O., 1970. Exit, Voice, and Loyalty: responses to decline in firms, organizations, and states. Cambridge, Massachusetts and London: Harvard University Press.

Johnson, B., 1992. Institutional learning. in: Lundvall, B.-A. (Ed.), National Systems of Innovation: Towards a Theory of Innovation and Interactive Learning. Pinter, London, pp. 23-44.

Jullien, N., J.-B. Zimmermann, 2006. New approaches to Intellectual Property : from Open Software to Knowledge-based Industrial Activities. DIME Working Papers on Intellectual Property Rights, N°5.

Kuwabara, K., 2000. Linux: A Bazaar at the Edge of Chaos. First Monday. Peer reviewed journal on the Internet, 5.

Lanzara, G.F. and M. Morner, 2003. The Knowledge Ecology of Open-Source Software Projects. Presented at 19th EGOS Colloquium, Copenhagen.

Lave, J., E. Wenger, 1991. Situated Learning: Legitimate Peripheral Participation. Cambridge University Press. Cambridge.

Lee, G.K., R.E. Cole, 2003. From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development. Organization Science, Vol. 14, pp. 633-649.

Marengo, L., G. Dosi, P. Legrenzi and C. Pasquali, 2000. The structure of problem-solving knowledge and the structure of organizations. Industrial and Corporate Change, Vol. 9 Nº 4: 757-788.

Mayo, E., 1949. The Social Problems of an Industrial Civilization. Routledge and Kegan Paul, London.

McGowan, D., 2001. Legal Implications of Open-Source Software. University of Illinois Review 241(1), pp. 241-304.

Moon, J.Y., L. Sproull, 2002. Essence of Distributed Work: The Case of the Linux Kernel. in: Hinds, P., Kiesler, S. (Eds.). Distributed Work, Cambridge, MA: MIT Press, pp. 381-404.

Nelson, R. and S. Winter, 1982. An Evolutionary Theory of Economic Change, Harvard University Press, Cambridge, MA.

Nooteboom, B., 2004. Inter-firm Collaboration, Learning and Networks. An integrating approach. Routledge, London and New York.

O'Mahony, S.C, 2003. Guarding the Commons: How Community Managed Software Projects Protect Their Work. Research Policy 32(7), pp. 1179-98.

Sabel, C. F. and J. Zeitlin, 2004. Neither modularity nor relational contracting: inter-firm collaboration in the new economy. Enterprise and Society, 5(3), pp. 388-403.

Ulrich, K., 1995. The role of product architecture in the manufacturing firm. Research Policy, Vol. 24, N°3, pp. 419-440(22).

Ulrich, K.T. and S.D. Eppinger, 2000. Product Design and Development (2nd ed.). New York: McGraw-Hill.

Van Wendel de Joode, R., 2004. Explaining the organization of open source communities with the CPR framework. The 10th Conference of the International Association for the Study of Common Property: "The Commons in an Age of Global Transition: Challenges, Risks and Opportunities", Oaxaca, Mexico, August 9-13.

Van Wendel de Joode, R., 2005. Understanding open source communities. An organizational perspective. Ph.d thesis, Delft University of Technology, the Netherlands

West, J., S. Gallagher, 2006. Patterns of Open Innovation in Open Source Software. Submitted for Chesbrough, H., W. Vanhaverbeke and J. West, eds., “Open Innovation: Researching a New Paradigm”, chapter 5, pp. 82-106, Oxford University Press, Oxford.