Toward Collaborative Open-Source Technology Transfer

Jean-Michel Dalle
IMRI-Dauphine, Univ. Paris 6 & Agoranov
jean-michel.dalle@upmc.fr

Guillaume Rousseau
INRIA
guillaume.rousseau@inria.fr

Abstract[1]

We analyze several occurrences of open-source technology transfer where research tools or prototypes developed in academic environments are transferred to private actors to be exploited economically. We enlight common characteristics which lead us to suggest that academic duality is a general consequence of the academic design of research tools and prototypes, and that the associated high transfer costs could be reduced first by implementing dual versioning using a dual licensing scheme, by associating a new academic public license with a traditional technology transfer one, and second through the transparent and shared maintenance of products built according to a dual architecture, as it would be precisely allowed by a dedicated collaborative development platform such as LibreSource.

1. Position of the problem

Software is an increasingly common output of academic research. Computer science theories and ideas are widely tested and implemented as software prototypes. Even more generally, software has become a common research tool in all disciplines, for calculus, simulation and many other tasks associated with academic life. To speak only of the most famous of the latter category, it is now well-know that academics at CERN in Geneva were largely responsible for the invention of the World Wide Web, first as a research tool. But then an important issue arises about how software developed in academic communities is or could be made accessible to a larger public, when of course relevant: and it would indeed be relevant for numerous software technologies that could be usefully exploited in the context of economic products and processes. To put it differently, the importance of the *technology transfer* issue for software research tools and prototypes is increasing rapidly.

Technology transfer issues have been studied for years in economics and other disciplines, and it is not our purpose here to assess the various results that this literature has brought, nor to discuss the hypotheses that have been suggested, and sometimes discarded, about how to improve the transfer of technologies from public-funded research to private actors and markets. It would certainly be fruitful to do so, but we would like to inquire here the issue of software technology transfer according to a different perspective: the reason for this is the rapid surge of a new and fashionable mode of technology transfer for software, namely, *open-source technology transfer*.

Building upon the success of open-source software, for which Linux has become a paradigm, software developed in academia is now increasingly made accessible in an open-source way: not only because what could be called "GPL-publishing", by researchers on their homepages, but also because of more developed attempts of open-sourcing software initiated by higher education and research institutions. Indeed, the initial idea of this study comes from the fact that, in the context of our work as researchers in the economics and management of software and innovation, we suddenly found ourselves coming across more and more numerous examples of academic software developed in open-source mode, many of which relatively recent, some of which older than we would have expected[2].

[1] We would like to acknowledge how much our work has benefited from various discussions with other participants in the 'LibreSource' RNTL grant, whose support is also most gratefully acknowledged: among them, we would notably like to thank Vincent Finet, Laure Muselli and Olivier Rhein. Other discussions with Paul A. David, Rishab A. Ghosh, Gérard Giraudon, Jesus Gonzales y Baharona, Nicolas Jullien and Laurent Kott have also been very helpful in shaping our views. This paper draws significantly from a previous one [3] which was presented at EPIP2 Conference in Maastricht, The Netherlands: most surprisingly, there was a slogan written there, on the wall of the room in which this conference was organized and the paper presented, that was said to date back to the Princes of Oranje, and which read "Je maintiendrai". This sentence in French translates into "I shall maintain": surprised as he was, it certainly helped one of us (JMD) understand that maintenance was key, and that organizing it via a collaborative platform could be crucial for open-source technology transfer.

[2] See [2, 17, 21] for early intuitions about this.

In any case, academia now appears as a *second major source of open-source software* after independent developer communities[3]: a source about which we do not know much yet, and about which we would certainly like to know more to properly assess the consequences of the current wave and perhaps tsunami of open-source academic software research tools and prototypes. This is the basic rationale of this work, which we hope will be followed by others as such studies appear to be rapidly needed by research policy makers, among whose preoccupations stands notably the current disarray of traditional technology transfer offices for which open-source software is still a new and puzzling phenomenon, apparently associated only to non-profit mechanisms – an erroneous conception which we will try to correct below – whereas these institutions have most of the time been created to foster the commercial exploitation of technologies developed in labs, be it by contractual research, licensing, or start-up creation. Open-source cases are on most of their desks now, but they do not know how to handle them according to this new mode of technology transfer[4].

As a further consequence, we will not consider here under which conditions open-source technology transfer could be more efficient than other modes. Considering how fast the open-source technology transfer wave is progressing, we have rather adopted here a more pragmatic approach, and we will try to determine using various case studies the major characteristics of open-source technology transfer, and from then on the conditions under which open-source technology transfer could be made more efficient. As a matter of fact, answers to these questions are relevant for future comparisons with other modes, and could also first of all provide a rationale to try to stop, or not to stop the current wave, or perhaps to try to reorient it in some particular way. In this context, we will start by briefly presenting case studies, and then our main findings for which we will try to provide interpretations, on the basis on which we will finally propose a more general framework for open-source technology transfer[5].

[3] Although general results about open-source software mostly apply: [10] provides a synthetic view. See also [11, 13, 14, 16, 20] about motivations and [6, 7] about competition issues.

[4] These issues are now crucial in France for INRIA, RNTL, CNRS, CEA, ANVAR and for several major universities, to name but a few major actors. We have had various occasions to verify that there were also emerging rapidly in several other countries, as in the UK for the EPSRC or in the US for the NSF.

[5] As should already be clear to the reader, the line of inquiry we have selected implies that our conclusions will for now stand only as conjectures, that is mainly as hypotheses for further and more quantitative research. This team is of course committed to contributing to this task, but a major motivation for this early release – applying to ourselves what other research on open-source development [5] has taught us –, is that we believe that much can to be gained by disclosing results, here on open-source technology transfer, early enough so that they can motivate further inquiries and further research.

2. Case Studies

We will now briefly present some of the cases studied in the context of the LibreSource project, and which all concern open-source software developed in academia. For now, and since it is an on-going work, we apologize for designating them only with initials: we would notably like to preserve their anonymity until complete conclusions have been reached and disclosed to all relevant parties[6].

- Technology A is a CAD tool for integrated circuit design at a very advanced stage of development. It has now reached wide recognition in its field, and is widely used in the academic community worldwide. However, it is not used by private companies, which prefer proprietary solutions and which have developed in-house plug-in libraries adapted to their own specific circuit elements. However, Technology A is modular and globally interoperable with these proprietary solutions. Technology is essentially a research tool, and is considered to have been a key element to grant the team that developed it with a high academic standard, and to allow it to receive around 1 M€ per year in contractual research from industrial partners, which was notably used to pay for an engineer who maintained Technology A and supplied user support. Technology A's licensing scheme was initially unclear, but it was later released under the GPL. 3 start-ups at least have been created in connexion with technology A, each of which has received VC funding. Two of them market circuits developed using technology A, i.e. research results obtained using A as a research tool, and the last one markets an improved closed and proprietary version of one of A's components, also compatible with proprietary solutions.
- Technology S is a tool for calculus and simulation at a very advanced stage of development. It is not completely modular, although dedicated toolboxes exist and are suited to specific needs. It is a direct competitor of a proprietary software solution that is in a monopoly situation, which benefits from a much greater number of specialized toolboxes, and with which it is not interoperable although S initially forked from the last open-source version of this other technology before it was closed and turned into a commercial product. S is released under a sui generis licence that grants the team which develops technology A with very strict control over contributions, and which does not guarantee explicit recognition for contributors. S is itself challenged by another open-source solution released under the GPL, and which is interoperable with their common proprietary competitor. A consortium of commercial users, mostly

[6] A more complete description will be included in the final LibreSource report (forthcoming in 2004).

big companies, has recently been set up, operated by one of the research institutions from which S originates, whose members pay an annual fee in exchange for premium services, and notably to see their suggestions for further development taken into proper account by the academic development team. In this context, technology S's licensing scheme is about to evolve, though the eventual choice is not completely determined yet, possibly toward a dual GPL – LGPL scheme with LGPL rights being granted to members of the consortium. At least one start-up is a member of this consortium and plays the role a specialized integrator for technology S: a similar company exists in Germany but does not seem to be part of the consortium at least for now, contrary to a non-specialized company dedicated to open-source solutions but which is willing to develop a similar business.

- Technology J is a J2EE compliant application server at a very advanced stage of development. It has for the last 3 years become part of a consortium supported by several major companies, and which regroups technology J together with a set of other components dedicated to related functionalities. The consortium is here also, as for S, a contract between users and the research institution from which the technology originates, according to which an annual fee is paid in exchange for premium services and notably inclusion of their expressed needs in upcoming versions. Most of the components regrouped in this consortium along with J are licensed under the GPL, including J, and the consortium plays a coordination role among them to try to foster interoperability and convergence. Numerous individuals, both academics and employees of various companies using J, and also several start-ups, are members of this consortium, together with a major systems integrator, which has partly embraced an open-source strategy, and for which technology J constitutes a key middleware component. The consortium's web site is basically a repository for all these components, and has recently started to host yet another well-known middleware component.
- Technology X is a desktop grid solution at a relatively early stage of development. It has been developed in the context of a successful PhD thesis and was from the beginning released under the GPL, although this licensing scheme does not seem to reflect a proper decision of the university under the auspices of which it has been developed. Technology X is in the process of rapidly achieving a wide recognition and installed academic worldwide. Several potential commercial users have already approached the team that maintains it and plans to continue to develop it with relatively pressing needs, and this team is therefore currently involved in several start-up projects. The business model of these start-ups is not completely determined yet, between a specialized integrator which would answer clients' needs in the context of contracts which could otherwise have been directly passed through the university as research contracts, and a more ambitious component strategy which would be more competitive worldwide with other desktop grid solutions.
- Finally, we would also like to add to this least another well-known example of open-source technology in the process of being transferred to commercial uses, namely the Globus technology for grid computing, denoted here as G for coherence reasons, as it has been studied recently in [9] and to which we would like to directly refer the reader.

3. Preliminary Findings

(i) *Maintenance and governance.* Academic labs and research institutions are generally willing to retain at least *partial control* over software maintenance and development using appropriate licensing schemes (S, G), or by creating consortia that they themselves more or less control (S, J), or simply by continuing to invest in the development and act as maintainers of the technology (all including A & X). The existence of consortia basically tends to balance pure academic leadership and to shift part of the authority to steering committees where both academics and users are represented. Consortia also seem to increase the number of non-academic users, sometimes although on a purely individual basis (J).

(ii) *Licensing.* Very different licensing strategies have been and are being explored, and no general solution has emerged yet. However, viral licensing is now present is all projects we have studied. In this context, and this is clearly related to maintenance and governance issues mentioned above, we have found several examples of legal control over contributions brought by a centralization of rights in the licensing schemes: rights on contributions are to be given to the maintainer's institution, contrary to pure GPL licensing which as a consequence does not prevent forking – although commitment strategies of academic research teams seem to play a counterbalancing role and are generally held to prevent it. The possible existence of software patents held by contributors can also be taken into account so that it is guaranteed that they will not be harmful to the maintainer and to the users of the technology. In addition, we have also found evidence of the desire of some contributors at least to be rewarded for their contributions and notably granted, although we have found no BSD-like clause. How licensing schemes could be designed both to allow for commercial exploitation and to correspond to the strategy of research teams is a major concern for several projects.

(iii) *Commercial partners and start-ups.* Major companies (A, S, J, G) and start-ups (all) are present in most cases. As for major companies, they can be either

users (A, S, J) or systems integrators (J, G), and will deal with academic teams through one-to-one contracting (A and probably G) or through consortium agreements (S, J). As for start-ups, it is unclear whether their prevalence is an artefact due typically to the fact that IT technologies have been especially prone to start-ups creation during the recent years. However, we would like to conjecture here that access to open-source technologies, which are also more visible from outside academic communities and acquire more rapidly a larger, at least academic, installed base, is by construction associated with lower barriers to entry: as a consequence, open-source technologies born in academia are more suitable to be marketed by newly created specialized integrators, i.e. are specially prone to start-up creation. More, labs can here also be more attracted by partners whose bargaining power is relatively limited, meaning that the deals will be more in their advantage and that they will retain more control.

(iv) *Maintenance costs.* In all cases, there is a more or less pressing need of academic players to find a way to finance maintenance costs, since higher education and research institutions experience difficulties in justifying long-term financial commitment of development work. These maintenance costs are sometimes paid for by contractual research attracted by the academic reputation that the team has earned notably due to its achievements in developing a powerful technology and obtaining good research results and academic leadership in its field (A specially and in some extent all others). Consortia partly play a similar role, but extend it by increasing the bargaining power of private actors i.e. by linking a fee paid by commercial entities to more influence on future developments.

(v) *Markets*. Various market conditions characterize each of the technology we have studied: it can be characterized by a de facto standard (S), by a limited number of proprietary solutions (A) probably on the way to de facto standardization (J), or by a rapidly evolving competition (X) where the technology can be dominant (G). In the case of S, a previous technology transfer had been attempted, in traditional closed and proprietary mode, which did not meet success. Actually, these market conditions, and the characteristics of software markets, have most probably played a role in driving most of the technologies to an open-source technology transfer strategy, compared to a direct transfer of the technology in a non open-source mode (A, S, J at least). Due to market dynamics provoked by network effects and externalities, the real chances of a technology, specially when it is not among the early entrants but a latecomer and when it is not supported by considerable investments, to gain a fair share in a direct proprietary competition against other proprietary software producers are relatively low and hazardous. It is particularly so as soon as a de facto standard has already emerged from a former standard race (S, probably soon J), or an equivalent market structure with very high barriers to entry (A). In this respect, a technology born in academia can be a late comer for at least two different reasons: first, because it was developed in a pure academic context for years before the opportunity of commercial exploitation was really explored (J), or because a first and attempt has failed and a new one is launched several years later in completely different market conditions (S); and second, because this technology has often been created as a research tool, i.e. for the research team which develops it and the academic community to have access to an open-source, easy-to-use, and quasi-free research tool, and therefore to avoid having to pay for expensive proprietary solutions whose sources are in most cases unavailable, even on a monetary basis, which renders academic work considerably more difficult, if not unfeasible (A). Whatever the reason – sometimes both could apply – proprietary technology transfer strategies would appear rather quixotic when it would be so, and the technology might experience a better diffusion trajectory and reach a larger installed base and a better global outcome (considering also the reputation reward for the research team), according to an open-source and even a software generics strategy, i.e. in playing the role of a software generic in its particular market. In parallel, supplementary indirect technology transfer strategies can be implemented and include selling components interoperable with the solutions that dominate the market, or exploiting of research results obtained by using the technology as a research tool (A). In this context, more recent technologies (X) are wondering whether they should immediately adopt a software generics strategy or if they have entered the market early enough to have other opportunities, i.e. if the tipping point of the standard race in their market is not too close they could for instance adopt strategies like the one that G has selected i.e. trying to play the role of an open-source de facto standard which most proprietary software producers now implement.

(vi) *Academic installed base.* In most cases, these technologies are widely used in the academic world for teaching and research, and their academic installed base is much bigger than their commercial installed base. This is clearly due to the openness of their sources and to their being accessible for free. There is at least one major limit for this (S notably), which concerns interoperability with dominant proprietary solutions: the basic issue faced by higher education institution when they use an open-source alternative technology for teaching, is that students would have to learn later different procedures if it is not interoperable enough with the solutions which dominate the market, which significantly reduces their employability when it is so.

(vii) *Modularity and design.* Most technologies have some degree of modularity [1, 15], but still relatively imperfect. They are often said to be modular, but we have not been able to really obtain evidence of the extent in which modularity had concretely been achieved in their design, while in some cases it is even not completely clear whether modularity is implemented in the current version of the code or if plans exist to improve it in future versions. Other puzzling design characteristics include for instance the non-interoperability of S for instance, which harms its competitive power both against the existing proprietary standard and against its open-source competitor, even in the academic institutions in this last case. Clearly, imperfectly modular design has to do with the fact that modularity was less of a rule when the older technologies were designed. But we would also like to conjecture that design by academic teams also reflects the context and objectives according to which these technologies were initially conceived, namely, to serve as research tools or as research prototypes, without a proper taking into account of other and notably commercial considerations which could have pleaded for a more modular design and also for other kinds of specifications such as interoperability. *Academic design*, as we suggest to denote this phenomenon, is indeed a good candidate to explain some of the difficulties in technology transfer attempts, as the technology is not well adapted ex ante to commercial users' needs, and the necessary investments to modify its design are thus all the more difficult to finance that initial design decisions are path-dependent. What we are basically suggesting is that technologies created in academia, often as research tools and sometimes as research prototypes, are simply initially designed in an academic way without taking into account other considerations, and that this design can render technology transfer more difficult i.e. more costly for any commercial entity which would be willing to attempt it.

Another way to put it is to say that academic design significantly increases *transfer costs* and therefore the investments private actors have to make, including sustained contractual relations with the initial designers of the technology so as to benefit as much as possible from their skills and from the tacit and uncodified knowledge they possess, to create marketable products and processes. This phenomenon actually has a name in economics, although it does not usually apply to academic but to military technologies: it is called duality and characterizes the fact that military specifications imply characteristics – or design decisions in a more modern terminology – which do not correspond to civilian uses, hence creating a high transfer cost for so-called dual technologies . Here, transfer costs between academic and commercial use are probably be more limited but still exist and stand as a logical explanation for weak commercial use.

Academic duality is a consequence of academic design: most technologies born in labs are created as research tools without any commercial strategy and are designed according to academic standards, and this applies also in a large extent to others research prototypes as the main motivation of scientists is to design them so as to earn reputation from their peers, therefore in line with specifications chosen independently by researchers according to the preferences of the academic community. A further consequence now is that contract research between academic and commercial partners can then be interpreted, at least in some cases, as reflecting an *opportunity cost*: the opportunity cost for the lab of adopting this or that design characteristic instead of the 'academic one'. Interested commercial partners can pay for these opportunity costs on a direct contractual basis, but this rationale is even more clearly reflected by their willingness to engage financially into consortia and to receive in exchange extra bargaining power on design decisions. However, they can most probably not afford these costs when it comes to modifying older design decisions[7].

4. Reducing and organizing academic duality

Academic design and academic duality tends to characterize software, and probably most scientific artefacts[8], both research tools and prototypes, which creates structural technology transfer costs. Clearly then, ignoring academic duality in the design of technology transfer mechanisms would probably lead to repeatedly unsuccessful strategies. In this respect, we would like to suggest here a modest and tentative proposal for open-source technology transfers: an open-source proposal indeed, which we hope will attract criticism and further contributions. Central in this proposal is the notion of organized and reduced duality, and it aims at structurally reducing technology transfer costs. The basic rationale is the following:

4.1. Reducing academic duality for research tools and prototypes

It could first be possible to optimize the *early design* of prototypes, while completely respecting academic motivations. As soon as some of the characteristics of spontaneous or emerging academic design are not specified by the academic context, they should be

[7] At least not in a consortium where others would also benefit i.e. if they would not be able to appropriate returns sufficiently: a traditional public good dilemma, which exclusive rights can solve.

[8] This corresponds indeed completely to other personal experiences that both of us have acquired in various technology transfer offices and science-based incubators.

specified so as to reduce the future cost of technology transfer if such an opportunity would occur. This is especially important when choices made implicitly by researchers are particularly relevant for commercial exploitation, such as interoperability and standardization issues (see above).

We believe that technology transfer offices within higher education and research institutions should have a policy about how to contribute in this respect: in any case, the earlier the better, to the limit that academic technologies are initially developed voluntarily by researchers in the context of their research projects without basically reporting to their host institutions. However, as soon as funding is sought by and allowed to research projects that involve prototyping or development of research tool, it could be easier to ask typically for an explicit design[9].

It would be in the interest of researchers as it could allow them to attract more interesting and rewarding research contracts in the future. As a matter of fact, former research contracts and contacts with private actors could play a significant role in providing researchers with information that they do not possess about relevant characteristics for commercial design, and could sometimes even result in early research contracts for which industrial partners would pay to cover the opportunity costs which correspond to design changes compared to spontaneous academic design, without influencing otherwise academic strategies.

Such a method for reducing academic duality is crucial for software, because design decisions such as interoperability are of special importance, and because software technologies need less capital investment to be developed and are often necessary and expected by peers to validate research work, which implies that software prototypes are often developed by researchers in academic contexts. However, it would clearly apply in other disciplines too, as soon as they engage in creating research tools, be they software or not, or in developing various prototypes.

4.2. Organizing academic duality: dual versioning and dual licensing

Complementarily to the former way of reducing academic duality, *dual versioning* could be a further method of organizing it and of reducing technology transfer costs. The basic idea is to allow for the coexistence of both an academic version and a commercial version, but to organize this coexistence in an efficient way: namely, reconciling versioning both in the economic sense (i.e. segmenting between markets with different products) and in the software engineering sense (allowing for different versions properly speaking). A way to do this is to open-source technology transfer and to use a double licensing scheme similar to those that are now been implemented by various open-source vendors[10].

The academic version of technology T should be licensed under a viral license, which almost completely prevents any commercial use since it implies that any piece of software incorporating T should be under the same license, i.e. it would necessarily be open and could not be integrated with closed proprietary components: this licensing scheme therefore creates a credible threat, as soon as it is enforceable, which forces any private entity willing to sell closed software incorporating T to ask for another license which would typically include royalties and similar clauses[11]. There would therefore coexist an academic version which could typically be used as a research tool by any interested researcher in the world, and commercial versions while private actors would develop by integrating T with complementary proprietary module which would typically make T more efficient for commercial users, a typical such module of course having for instance to do with efficient user interface.

A corollary is then that to successfully implement this scheme, rights have necessarily to be centralized i.e. given to an institution which is able to a) re-license them, notably to interested commercial partners b) represent a really credible threat for counterfeiters – remember that T is widely accessible in its open-source academic version – : the latter is actually done by the FSF for numerous free software projects as it has been spontaneously granted rights by many contributors, but the FSF could on the contrary not involve in the former i.e. re-license any of these projects because it does not possess all the rights. As a consequence, the viral license will in this context, contrary to other dual licensing schemes, not be GPL-like but rather QPL-like (a license created and used for technology Qt) or GTPL-like (a license created and used

[9] A very interesting policy in this respect has been implemented by INRIA, which has a dedicated crew of developers who are assigned to academic projects on a fixed-term basis but who also remain under the supervision and depend upon the authority of the technology transfer office.

[10] About dual licensing, see e.g. [19, 22]. In shaping our views, we have also benefited from several discussions during the eScience workshop held in Oxford on May 2nd 2003, notably with Paul A. David, Rishab A. Ghosh and Tony Hey, after which [12] has suggested a different dual licensing scheme for publicly funded research, which would combine the GPL and a commercial licence, but which we fear would however unfortunately not be appropriate to academic technology transfers, as it would not technically speaking comply with the necessary centralization of rights and would not correspond either to the motivations of academic scientists for maintenance and for citations.

[11] If relevant, exclusive rights could be granted, notably to start-ups, to allow them to raise private funding and to invest more in the development of an efficient commercial version, provided of course a regular assessment is made of the extent of the exploitation they make of the technology with a return clause if exploitation falls below a given threshold, as is usually the case now for technology transfer.

for grid technology Globus) as both of these licenses have implemented clauses that guarantee the centralization of rights, even when rights owners would have filed patents.

However, as soon as there is a crucial and pressing need to adopt a non-standard license, it might be useful to account also for several other features of open-source technology transfers as they have been presented above. The specifications of a new *academic public license* (APL) could then include: (i) A BSD-like clause that would guarantees that contributors would be explicitly credited, and would therefore account for the motivations of scientists in a similar way as the GPL corresponds to the motivations of independent developers: in the academic world, these motivations specially have indeed to do with citations, and committing to credit contributions would attract more academic contributors; and (ii) a clause that would clearly specify applicable law and courts: all projects to which the academic public license would apply would then point to only one court, which would help this court to develop appropriate skills in such new, specialized and delicate matters.

In this last respect, we would like to suggest that French law could have, in some extent, an interesting and distinctive feature here, as it does not rely on *copyright* but on *'droits d'auteur'*: on 'author rights' (*authoright*?). Among those rights that authors possess are "moral rights", which are specifically instituted by law as completely inalienable, meaning that they cannot be sold or abandoned, whatever way. As a consequence, granting all of their 'other' rights to a third party which would centralize them to re-license them according to a dual licensing scheme could then be counterbalanced for developers according to French Law by their retaining of their moral rights, and therefore by their being able to forbid exploitation under certain circumstances which would be strictly contrary to their motivations. This idea would certainly need further legal inquiries, but if it was so it would allow to re-implement simply in a dual licensing context the exact kind of 'credible commitment' mechanism [24] that we had suggested crucial to explain the success and sustainability of GPL'd 'Libre' software [7].

The design of such an academic license would also have the supplementary benefit of being in line with the astute suggestion [9] that modularity and standardization should be developed for technology transfer contracting. Indeed, the rights of the academic public license itself could be given to a relevant international institution, which would have the duty to maintain it in an open-source way so as to adapt it to new and unexpected events and situations as they would inevitably occur, and which would make this institution a good candidate to suggest generic clauses for the other commercial part of the licensing scheme we are describing here. And it could also be granted all the rights of some projects at least, but in this case several academic and semi-academic institutions would also have direct incentives to play a centralizing role for this or that project. By the way, a commitment from these institutions to use a fair share of whatever revenues they would get from open-source technology transfers to finance new and further developments, and also new research projects, would certainly play a major role in attracting more projects and could also be instrumental in implementing a credible commitment mechanism between academic communities, their representative institutions, and private actors. These representative institutions could then easily be foundations, though higher education institutions could also directly play this role: on the contrary, it is highly probable that commercial entities or even consortia including commercial partners would not be able to guarantee an appropriate governance structure and a credible enough commitment to such an organized versioning – the risk being that both versions could converge to a single one to the detriment of the academic version.

4.3. Organizing academic duality with a dedicated collaborative platform

Still, this is most probably not enough, as we have seen that maintenance and governance were specially relevant issues for open-source software technology transfer. We would like to suggest that the use of a dedicated and well-adapted collaborative development platform could considerably help here, in providing a joint framework in which all kinds of consortia could work and be implemented, while at the same time basically reducing maintenance costs.

First, different partners could maintain different modules separately. To put it differently, software could be designed to fit as best as possible with a *dual architecture* which could in itself reduce and organize academic duality: some modules would be academically maintained, while others would be commercially maintained. Organizing duality according to such a division of innovative labor, and *shared maintenance*, would indeed correspond well to two different incentive and motivation systems: one associated with reputation and peer-review [8], which would focus more on algorithmic aspects and lower layer issues, and the other one with profit and the size of the market with a special emphasis on graphic user interfaces and similar functionalities [23]. Indeed, contributions to the academic components could even play a role similar to citations for academics and these modules would in a sense truly become *software journals*.

Such an architecture would obviously reduce technology transfer costs, and would fit also well with dual licensing, but organizing correctly such an improved

division of innovative labor needs an appropriate environment: all the more so when there are numerous partners, and when, as is always the case, perfect dual architectures do not exist, and there are interdependencies between modules which imply that teams still have interactions with each other's work, that shared maintenance extends within modules, therefore generating transfer costs and potential conflicts, which only an appropriate governance structure which would combine authority and leadership with *transparency* could help to solve efficiently. Indeed, transparency, i.e. an easy monitoring of decisions by all relevant parties, is all the more necessary here as the centralization of rights reduces the risk of forking, and hence puts less pressure on the quality of decisions taken by authorities.

A simple way to implement all of this is to host development on a well-adapted collaborative development platform, which would also make decisions both visible and challengeable. Such a platform should be more distributed that CVS not only to simply make development more efficient, but also to accommodate concurrent development by several non-profit and for-profit entities with different time horizons and workloads. More, it should be able to implement efficient user right management, including intellectual property right management, via an efficient tracking of author contributions. And it should be able to provide efficient conflict resolution procedures, notably for the file system.

LibreSource is precisely being designed and developed as a distributed collaborative development platform to manage large collaborative academic and industrial software development projects involving academic and industrial resources. LibreSource possesses all the standard tools of a collaborative development platform, such as a project and user administration module, a Web interface, a bug tracking system, a forum, a wikki, etc. LibreSource also allows a high level of integration within network infrastructures (full compatibility with main firewalls), user environments (client side applications run through JavaWebStart), hosting services (Windows and Linux are supported, and LibreSource is fully based on J2EE-compliant technologies). In addition, LibreSource incorporates functionalities of flexible synchronization "DataFlow" using synchronization networks, a workflow engine "Bonita" allowing the implementation of complex specialised processes, and high-level synchronization and conflict resolution procedures [18].

5. Conclusion

We have tried to suggest here that, if academic duality is largely inevitable in technology transfer situations, due to academic design, therefore contributing to increase technology transfer costs, it could be reduced and organized. Combined together, we believe that the suggestions presented above could integrate into a new framework which would significantly reduce and organize academic duality, and which would fit well with the various observations we have made about open-source technology transfer situations: about maintenance; licensing schemes; contracting with commercial partners, notably to support maintenance costs; market conditions; academic installed bases; and, last but not least, modularity and design issues for research tools and research prototypes. Needless to say, this framework does not solve all issues, notably those which have to do with its practical implementation in various market conditions, and thus with the need for different commercial partners and for suitable contractual arrangements: all of which will have to be practically dealt with in every specific situation by technology transfer office professionals. Finally, we would like to stress that, as they have been presented here, these ideas are mainly specific to software: however, we believe that similar suggestions could be made in other technology transfer situations, as for instance for biotech databases and for educational resources [4].

6. References

[1] Baldwin C.Y., Clark K.B. (1997), Managing in an Age of Modularity, Harvard Business Review, Sept.-Oct., pp. 84-93.

[2] Dalle J.-M. (2002), Open code : the sources of open-source innovation, presentation at the NSF Workshop on « Advancing the Agenda on Open-Source » held in Arlington, VA, January 28th, and to the DRUID Summer Conference on "Industrial Dynamics of the New and Old Economy – who is embracing whom?", Copenhagen-Elsinore, June 6-8.

[3] Dalle J.-M. (2003), Open-Source Technology Transfer, presented at EPIP2 Conference, Maastricht, The Netherlands, November 24-25th

[4] Dalle J.-M. (2004), Vers un modèle collaboratif en matière d'éducation, forthcoming in a book edited by Fondation pour un Internet Nouvelle Génération (FING), (In French : Englgh translation available from the author).

[5] Dalle J.-M, David P.A. (2003), The Allocation of Software Development Resources in 'Open Source' Production Mode, SIEPR Policy paper No. 02-027, accessible at http://siepr.stanford.edu/papers/pdf/02-27.html.

[6] Dalle J.-M., Jullien N. (2000), NT vs. Linux, or some explorations into the economics of Free Software, in Ballot G., Weisbuch G. (eds), Application of simulation to social sciences, Hermès, Paris, pp. 399-416.

[7] Dalle J.-M., Jullien N. (2003), 'Libre' Software: turning fads into institutions, Research Policy 32:1-11.

[8] Dasgupta P., David P.A. (1994), Towards a new economics of science, Research Policy 23: 487-521.

[9] David P.A., Spence M. (2003), Towards institutional infrastructures for e-Science: the scope of the challenge, Final Report of the Oxford Internet Institute project on 'The Institutional Infrastructure of e-Science: The Scope of the Issues'.

[10] Feller J., Fitzgerald B. (2002), Understanding Open Source Software Development, Addison Wesley, Boston, MA, USA.

[11] FLOSS (2002), Final Report, Part IV: Survey of Developers, International Institute of Infonomics, University of Maastricht & Berlecon Research GmbH, accessible at http://www.infonomics.nl/FLOSS/report/Final4.htm.

[12] Ghosh R.A. (2003), Copyleft and dual licensing for publicly funded software development, Draft version (1.0), MERIT – Institute of Infonomics, University of Maastricht.

[13] von Krogh G., Spaeth S. & Lakhani K.R. (2003), Community, Joining, and Specialization in Open Source Software Innovation, presented at HBS-MIT Sloan Free/Open Source Software Conference, June.

[14] Lakhani K., von Hippel E. (2000), How Open Source software works : 'free' user-to-user assistance, MIT Sloan School of Management WP #4117.

[15] Langlois R.N. (2002), Modularity in Technology and Organization, Journal of Economic Behavior and Organization 49:19-37.

[16] Lerner J., Tirole J. (2002), Some simple economics of open source, Journal of Industrial Economics 50:197-234.

[17] Marzouki M., Greiner A. (1999), Du rôle des financements publics de recherche dans le développement du Libre. Étude de cas : la chaîne de CAO Alliance, presentation to the 1rst 'Autour du Libre' Conference, Brest, January 25-27.

[18] Molli P., Oster G., Skaf-Molli H., Imine A. (2003), Using the Transformation Approach to Build a Safe and Generic Data Synchronizer, mimeo.

[19] Muselli L. (2003), Les stratégies de licences libres, presented at conference 'Autour du Libre', Paris, May.

[20] Raymond E.S. (1999), The Cathedral and the Bazaar, O'Reilly.

[21] Rousseau G. (2002), Valorisation des ressources humaines et logicielles en relation avec les logiciels libres, presented at the 2nd NME-NEL Workshop, INRIA, Rocquencourt.

[22] Välimäki M. (2003), Dual Licensing in Open Source Software Industry, Systèmes d´Information et Management, 8.1: 63-75, 2003.

[23] Varian H.R. (1993), Economic Incentives in Software Design, mimeo.

[24] Williamson O.E. (1996), The mechanisms of governance, Oxford UP.