

# Open-Source Technology Transfer

Jean-Michel Dalle

IMRI-Dauphine, University of Paris 6 & Agoranov  
jean-michel.dalle@upmc.fr

– DRAFT VERSION<sup>1</sup> –

## 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-known that academics at CERN in Geneva were responsible for the invention of the World Wide Web, first as a research tool. But then an important issue arises about how all software developed in academic communities is or could be made accessible to a larger public, when 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 now 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 by higher education and research institutions. Indeed, the initial idea of this study<sup>2</sup> comes from the fact that, in the

---

<sup>1</sup> Paper presented at EPIP2 Conference, Maastricht, The Netherlands, November 24-25<sup>th</sup>, 2003. This version is a draft version mainly circulated for comments. All errors are mine, but some of the ideas presented here have considerably benefited from discussions with other participants in ‘LibreSource’ RNTL grant, whose support is most gratefully acknowledged: among them, I would specially like to thank Guillaume Rousseau, Laure Muselli, Vincent Finet and Olivier Rhein. Other discussions about open-source software and technology transfer, with Gérard Giraudon and Laurent Kott, both from INRIA, and on open-source software with Rishab Ghosh, Nicolas Jullien and Jesus Gonzales y Baharona have also been very helpful. Finally, Paul A. David’s analyses about the economics of e-Science and open-source software have been instrumental in clarifying my own.

<sup>2</sup> We had suggested the idea of “open academic software” in the context of an NSF workshop organized in January 2002 at Arlington (VA) by Brian Kahin and Eric von Hippel. See Dalle (2002a) for a revised version of the presentation given to this workshop.

context of our work as researchers in the economics 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 – but this should not be a complete surprise if we remember that many of the origins of open-source software are to be found in academia (Jullien, 2001), and since there are similarities between the ethos of open-source software communities and the norms and institutions which govern open science (Dasgupta & David, 1994).

In any case, academia now appears as *a second major source of open-source software* after independent developer communities: 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<sup>3</sup>.

As a further consequence, we will not consider here whether, and under which conditions, open-source technology transfer would and could be more efficient than other modes for software, save to note that no other mode has actually reached a wide consensus according to the efficiency criterion. Considering instead how fast things are evolving, and that the open-source technology transfer wave would probably be especially hard to stop, 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 transfers, and from then the conditions under which open-source technology transfer could be made more efficient. As a matter of fact, answers to these questions are indeed 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 synthesizing some results of recent research on open-source software, so as to make subsequent analyses clearer, before we briefly present the 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.

However, before we begin, a final disclaimer has to be mentioned, which we believe is important enough for us to put it here rather than in the usual footnote: 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 has taught us (Dalle & David, 2003) –, is that we believe much can be gained by disclosing results, here on open-source technology transfer, early enough so that they can motivate further inquiries and further research.

---

<sup>3</sup> 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.

## A brief assessment of some recent results from open-source software research

Although more studies are still needed, notably more serious quantitative studies about market shares, open-source software is now widely recognized as a major driving force for the software economy and industry. We will therefore assume here that the reader knows the basics of its history and characteristic features<sup>4</sup>, and we will mostly present here several key elements of the economics of open-source software as they have been highlighted by recent literature. Needless to say, this list is far from being exhaustive in such a dynamic field. Another point worth of notice is that some of these elements have mainly to do with software developed by community of independent developers i.e. not with software born in academic contexts, but there are still relevant for us here as a benchmark.

1. Open-source software restores competition in markets that are otherwise subject to network effects and network externalities due to compatibility and technological interrelatedness issues<sup>5</sup>. As a consequence, open-source software appears a potential remedy to a market failure for which other remedies are missing (David, 1987), or whose actual strength, as for standardization committees, is still controversial. In this framework, open-source software basically proceeds by cloning existing proprietary software solutions, i.e. not by copying them but by re-developing them from scratch, and then by attracting the attention and efforts of communities of independent developers who organize through the Web and through other electronic means. This open-source mode production gives birth to products that essentially play the role of *software generics* (Dalle & Kott, 2002). Unlike generic drugs, these open-source software generics are not, at least for now, born from patents and property rights fallen into the public domain, and then can also become eventually superior to proprietary software they have cloned. But, like generic drugs, they first of all restore competition in previously quasi-monopolistic markets by allowing the existence of competitive alternatives to existing products, and secondly correct another common market failure which software markets share with markets for drugs, and which has to do with imperfect market segmentation in connexion with developing countries. Normal economic theory suggests that market should be segmented by producers between developed and developing economies as the latter are not able to pay a similar price as the former. It is often not so for drugs and in the case of software, a negligible cost of copying represents a very strong disincentive for producers, and software used in developing countries is thus most of the times pirate software which most of the time, as markets are small, does not attract any reaction from proprietary software producers, all the more so as it eventually contributes to promote their own standard. But pirate software does not come with its sources nor simply with manuals, which considerably limits its economic use. On the very contrary, as it is available for free with its sources and with tutorials, open-source software is suitable of a larger use by a larger number of small local service companies, as they appear in conjunction with open-source software (Coris, 2002). In fact, the open-source software mode of production allows for the de facto emergence of software generics without having to rely on private investments: these investments would indeed be considerably more important in the case of software than in the case of medicine as there is much more to replicate and clone than a formula and a process in the source code of a commercial application, which would probably prevent and their financing by markets, specially when there exists a proprietary

---

<sup>4</sup> If not, possible references, among others, are Dalle & Jullien (2003) and Feller & Fitzgerald (2002).

<sup>5</sup> Dalle & Jullien (2000, 2001) have developed a model of technological competition between proprietary and open-source software, and suggest explanations for the puzzling market power evidenced by open-source solutions in markets initially dominated by de facto proprietary standards.

standard with strong market power, and even if a de jure mechanism as it exists for drugs and their appropriation mechanisms was to be implemented. Somewhat paradoxically, the non-profit character of open-source production is a better candidate than traditional appropriation mechanism to restore competition on software markets and to foster the de facto emergence of truly open standards.

2. This being said, the issue which initially attracted attention from the community of economists about open-source software was however not competition but the so-called motivation and incentive issue, essentially because volunteer contributions were relatively puzzling for classical economic theory (Lerner & Tirole, 2002; Lakhani & von Hippel, 2000). We now have evidence from numerous developer surveys<sup>6</sup> that these motivations actually exist; at the same time, the ‘delayed reward’ theory still holds for at least part of the developer communities, as many developers appear to be working for private companies, some of which being voluntarily sponsored to contribute to open-source projects because the company which employs them does some business with open-source (see below, §3). This is still perfectly compatible with a more general explanation about reputation, as in open science communities where voluntary contributions rewarded by reputation have been the rule for a much more than a century. And if we turn back to competition issue, it is indeed puzzling that open-source motivations would play an indirect role to allow for a better functioning of markets, that is, of direct monetary incentives and rewards as they operate in private companies and as they dysfunction for software vendors in monopolistic situations!
3. More interesting perhaps are the various kinds of businesses that private companies have been able to build with open-source software. Indeed, open-source software vendors exploit various business models most of which exploit *bundles* and *complementarities* (Dalle, 2002b): since they cannot sell usually free, and anyways easily accessible open-source software, vendors sell it together with complementary assets. First among these are simply services to customize and adapt open-source software to the specific needs of each client, all the more so as customisation and specific developments benefit from openness of source codes. They are often packaged with another strongly complementary asset, namely liability and some insurance, which does not exist when software is simply downloaded on the Web. Generally speaking, the stronger the complementarity between open-source software and non-free assets, the more potential there is to be found in marketing that particular bundle. Other examples include documentation and tutorials – see what editor O’Reilly has done for instance –, or else bundles which package software with hardware, be it software embedded in various hardware platforms or software dedicated to usual computers but which is sold ‘bundled’ with them, as it now the case from most hardware vendors like IBM or HP. In this last case, open-source software plays a key role in the value chain of these companies, as it replaces an element which used to represent considerable value appropriated by external suppliers – as for OS’s for instance –, sometimes from monopolists. Value sold is then linked to assets that the firm produces in-house, like hardware and services at the two ends of the *layered* structure of computer-related products. Indeed, we have suggested further that the most valuable bundles were organized along these vertical layers<sup>7</sup>: *upper vertical bundles* which consist in selling products closer to the needs of the clients, like customisation services, together with open-source OS and middleware, and *two-sided vertical bundles* as we have just described them and which mostly benefit to *systems integrators*. Among the companies exploiting the former, *specialized integrators* seem of special interest, i.e. companies which develop

---

<sup>6</sup> See e.g. FLOSS (2002).

<sup>7</sup> Although business models obviously depend in some extent on each particular market.

specialized skills and services dedicated to a given open-source software project and specialized links, through employees notably with its associated community and project. These specialized integrators are sometimes also distributors, whereas pure distribution of pieces of open-source software bundled on a CD-ROM, together with an access on-line help, has not proven to exploit strong enough complementarities to allow pure-play distributors to attract sustainable businesses: on the contrary, when specialized integrators are also distributors, they can benefit from internal economies and at the same time increase their reputation in the market and in open-source software communities.

4. A related but neglected issue, which connects motivation issues with business model issues (see 2 and 3 above) in the context of governance mechanisms (Williamson, 1996), has to do with *credible commitment* mechanisms, and first of all with licensing schemes (Dalle & Jullien, 2003): voluntary contributions would not be attracted if there did not exist a sustainable long-term credible commitment mechanism, like the one implemented by the GPL. Indeed, the viral character of the GPL mostly prevents closure of previously open-source code, to the limit that it is feasible for private companies to develop a new proprietary and closed version from scratch – but with considerable help from previous developments, as it happened when VA Linux closed SourceForge, even changing its name to VA Software. We have also suggested (Dalle, 2002b) that a more sustainable credible commitment mechanism between open-source communities and private companies could be provided by the exchange of hostages (Williamson, 1996): here on the one hand workers who are hired by private companies exploiting economic opportunities associated with open-source software, and on the other hand the reputation of these companies which would largely depend upon their behaviour toward developer communities and upon their respect of their norms of openness. The main caveat has here to do with the bargaining power of these companies compared to communities of independent developers, and thus notably with their size. As a consequence, these credible commitment mechanisms are more efficient for start-ups and medium size companies than for major players: to put it differently, they work well, among bundle vendors, for medium-size specialized integrators than for major systems integrators.
5. There is at least another issue related to governance in open-source mode. In fact, contrary to early but yet common conceptions, open-source software communities obey to a mostly hierarchical mode of governance (Kogut & Metiu, 2001). This is typically the case for Linux's Linus Torvalds, often described as a benevolent dictator, but most other successful open-source software projects rely on a clear governance, either thanks to (rotating) leadership or to a voting committee. More, most projects are modular and local maintainers in charge of each module report to a central authority. These projects also develop by recruiting new contributors and by appointing some of them to various tasks according to their skills (von Krogh, Spaeth & Lakhani, 2003; Mateo-Garcia & Steinmuller, 2003). In all respects, open-source mode does not appear as close to a 'bazaar', as Raymond (1999) once suggested: on the very contrary, and as Raymond himself also suggests in the same book, governance of open-source projects relies on 'adverse possession' of modules and projects, i.e. leaders' decisions are *transparent and observable*, and inappropriate leadership results in it being challenged. Open-source leaders are somehow *disposable*, and we would like to suggest here that open-source has indeed invented an *open leadership* structure, where central authority is *monitored* and challenged and changed from time to time. Though in some open-source communities these changes are closer to coups than to democratic elections, this governance mechanism still guarantees control over central authorities and commitment from contributors, and therefore notably *prevents forking*.

6. Apart from *bundle vendors*, another class of business models is emerging, which has to do with *component vendors* i.e. companies which exploit modularity to sell software components to other software vendors and systems integrators (Dalle, 2002b). This appears to be feasible as soon as these companies typically adopt a dual licensing scheme, as it is the case for Qt or MySQL, as pure reliance on the GPL would preclude redistribution in at least partly closed form (Muselli, 2003). Indeed, the basic idea of coupling a viral and a commercial licence comes from the fact that the viral licence prohibits inclusion of the software module in closed developments, while allowing for testing and increasing diffusion and installed base: therefore, companies that need to use the module in closed developments to resell products and services to their clients, have then to turn back to the component vendor to be granted a commercial licence for which they will have to pay royalties. This emerging model works under the supplementary condition that the viral character of the first licence can be enforced, i.e. if it constitutes a *credible threat*; and meanwhile if the existence of a dual licensing scheme does not hinder the attractiveness of the project to new and external contributions i.e. if this licensing scheme does not appear to modify the credible commitment mechanism that exists between medium-size component vendors and open-source communities (see above, §4). On a pure economic basis, it is clear that the development of this model could increase the level of modularity in the software industry, and would thus bring associated benefits which the automobile industry has for instance been key in exploiting (Baldwin & Clark, 1997; Langlois, 2002). In the particular case of software, it would notably reduce the redundancy of development efforts, as software functionalities are currently being redeveloped ever and ever since no efficient and reliable component is available i.e. whose sources would be accessible so that it could be verified, adapted, and updated later. As a consequence, modularity is poorly exploited in software, and the software industry is still in search for ‘COTS’ – components off the shelf – , which open-source component vendors could easily provide.
7. Finally, the development of all these business models (§3 & 5), many of which are associated to start-up companies might also bring significant evolutions in the organization of the software industry and modify the current balance between its main actors. If open-source software continues to develop, the role of traditional software vendors could indeed really start challenged by systems integrators, while specialized integrators and module vendors would play an increasing role, either directly or precisely as sub-contractors of systems integrators (Dalle, 2002b).

There are clearly many other questions currently being studied, and still to be studied, about open-source software<sup>8</sup>. Among these, we believe that we really need to understand better how open-source as a production mode actually functions, and that this should also be done in connexion with researchers in software engineering. We have recently suggested in this respect an original simulation model, to be used as an integrative validation tool (Dalle & David, 2003). We have for instance suggested preliminary evidence which validates the importance of the ‘release early’ rule as it has been coined, again by Raymond (1999), the basic rationale for which being that early-released code attracts more new developers because enough work has still to be done for them to be valuably rewarded for doing so, i.e. to see their efforts acknowledged by the community of developers. To put it yet differently, developers are more attracted by preliminary work as their own contributions will be more visible.

---

<sup>8</sup> As a look at <http://opensource.mit.edu/> will easily prove.

## Case studies

To move back now to the subject of open-source technology transfer, we would first of all like to present briefly some of the case studies that are currently under way 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 the leaders of each of these groups<sup>9</sup>.

- 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 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.

---

<sup>9</sup> A more complete description will be included in the final LibreSource report (forthcoming in 2004).

- 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.
- Technology L is a collaborative development platform at a relatively early stage of development, being mostly rewritten and redesigned on the basis of a technology that originates in a public lab, and which has already been tested in other several contexts. This new version should allow for a more robust, more extensive, and potentially partly commercial use. A consortium of 3 members, 2 academic ones and a start-up, financed under a public R&D grant, is jointly responsible for developing technology L. Although its future licensing scheme is yet unclear, all partners have agreed that it will be open-source save perhaps for some specialized components that would be commercially distributed by the start-up in a proprietary scheme. On its part, the academic team that was instrumental in creating the technology is clearly willing to attract supplementary academic reputation and contracts with industrial partners.
- 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 by David & Spence (2003) and to which we refer the reader.

## Preliminary findings and interpretations

- (i) *Governance.* Academic labs and research institutions are generally willing to retain at least partial control over software development using appropriate licensing schemes (S, G, L), or by creating consortia that they control (S, J), or generally by continuing to invest in the development and act as maintainers of the technology (all including A & X): pure GPL licensing does not prevent forking but academic commitment strategies of research teams seem to play a counterbalancing role and to mainly prevent it. The existence of consortia tends to balance pure academic leadership and to shift part of the authority in the governance structure to their steering committees where both academics and users are represented, while they seem to increase the number of non-academic users, sometimes on an individual basis.
- (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 in all projects we have studied. In this context, and related to maintenance and governance structures (see i above), we have found several examples of legal control over contributions and centralization of rights in the licensing schemes: rights on contributions are to be given to the maintainer's institution, while 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 (i, again) 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 their prevalence is an artefact due typically to the fact that IT technologies are especially prone to start-ups creation. However, we conjecture that access to open-source technologies, which are more visible from outside academic communities and acquire more rapidly a larger installed base, and have lower entry barriers and are by construction more easy to market by new specialized integrators, are even more 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 (see i above).
- (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 thanks to 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, L) 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, L 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 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, L) 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 (Baldwin & Clark, 1997; Langlois, 2002), 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 that it 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 indeed 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<sup>10</sup>. 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<sup>11</sup>.

---

<sup>10</sup> See e.g. Cowan & Foray (1995).

<sup>11</sup> 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.

## **Reducing and organizing academic duality: a modest proposal**

Academic design and academic duality tends to characterize scientific artefacts, both research tools and prototypes, and 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:

### *a) Reducing academic duality for research tools and prototypes*

It could first be possible to optimize the early design of prototypes, while 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 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). Technology transfer offices and 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.

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.

### *b) Organizing academic duality*

Complementarily to the former way of reducing academic duality, 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 and a commercial version, but to organize this coexistence in an efficient way: namely, versioning both in the economic sense (segmenting

markets) and in the software engineering sense (two different versions). A way to do this is to open-source technology transfer and to use a double licensing scheme similar to those which have been implemented by component vendors (see above)<sup>12</sup>.

The academic version of technology T should be licensed under a viral licence, which almost completely prevents any commercial use since it implies that any piece of software incorporating T should be under the same licence, 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 licence which would typically include royalties and similar clauses<sup>13</sup>. 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 to be centralized i.e. given to an institution which is able to a) re-licence them 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-licence any of these projects because it does not possess all relative rights. As a consequence, the viral licence will in this context, contrary to other dual licensing schemes, not be GPL-like but rather QPL-like (a licence created and used for technology Qt) or GTPL-like (a licence created and used for grid technology Globus) as both of these licences 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 licence, 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 licence* 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.
- ii. A clause that would clearly specify applicable law and courts. All projects to which the academic public licence 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.

---

<sup>12</sup> This point has benefited from discussions during the eScience workshop held in Oxford on May 2<sup>nd</sup> 2003, notably with Paul A. David, Rishab Ghosh and Tony Hey, after which Ghosh (2003) 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 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.

<sup>13</sup> 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.

The design of such an academic licence would also have the supplementary benefit of being in line with David and Spence (2003) astute suggestion that modularity and standardization should be developed for technology transfer contracting. Indeed, the rights of the academic public licence 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.

c) *Further suggestions*

Both reduced and organized duality should and could of course be combined, and duality could also be further reduced and organized using other complementary means:

- Modularity could reduce transfer costs, as long as some modules would be maintained by academics, while others would be maintained by commercial entities, while the connection between the two would be correctly organized. Ideally, contributions to the academic components could then play a role similar to citations for academics and these modules would in a sense become *software journals*.
- Organizing this improved division of innovative labour could obtain more easily in consortia and appropriate contractual environments, but also if supported by appropriate open governance structures i.e. which would combine authority and leadership with *transparency*. Transparency, and an easy monitoring of decisions, is here a way to democratically change maintainers when needed, and 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 this is to host development on a well-adapted collaborative development platform, which would make decisions both visible and challengeable.

Combined together, these suggestions integrate into a proposal which we believe would reduce and organize academic duality, and which fits with the various observations we have made and reported above, about governance of development by academic teams, licensing schemes, contracting with commercial partners including start-ups and their role in supporting maintenance costs, the dynamics of software markets, the existence of a large academic installed base, and, last but not least, modularity and design issues for both research tools and research prototypes.

## **Conclusion**

We have tried to suggest here that, if some academic duality was largely inevitable in technology transfer situations due to academic design, and thus increased technology transfer costs, it could be reduced and organized better. As they have been presented here, these ideas are mainly, though not completely, specific to software, but we actually believe that similar suggestions could be made in other fields, as for instance for biotechnologies. In any case, we feel that reducing duality implies not only the commitment of technology transfer offices and higher education and research institutions, but also of economists who could have an important role to play in this respect.

## **Bibliography (to be completed)**

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

Coris M. (2002), *Les Sociétés de Services en Logiciels Libres : l'émergence d'un système de production alternatif au sein de l'industrie du logiciel?*, WP IFREDE-E3i, accessible at <http://beagle.u-bordeaux4.fr/ifrede/e3i/publications/2002/2002-5.pdf>.

Cowan R., Foray D. (1995), *Quandaries in the Economics of Dual Technologies and Spillovers from Military to Civilian Research and Development*, Research Policy 24:851-868.

Dalle J.-M. (2002a), *Open code : the sources of open-source innovation*, paper presented at the DRUID Summer Conference on "Industrial Dynamics of the New and Old Economy - who is embracing whom?", Copenhagen-Elsinore 6-8 June, mimeo.

Dalle J.-M. (2002b), *Quid novi ? Ouverture des sources et organisation industrielle de l'informatique*, paper adapted from the conclusion of the final report of RNTL grant NME-NEL, mimeo.

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>.

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.

Dalle J.-M., Jullien N. (2001), *Open-Source v Proprietary Software*, paper presented at ESSID Summer School, Cargèse, accessible at <http://opensource.mit.edu/papers/dalle2.pdf>.

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

Dalle J.-M., Kott L. (2002), *Plaidoyer pour des logiciels génériques*, La Recherche, January, pp. 70-73.

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

David P.A. (1987), *Some new standards for the economics of standardization in the information age*, in Economic Policy and Technological Performance, Partha Dasgupta and Paul Stoneman (eds.), Cambridge University Press.

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'.

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

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>.

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

Jullien N. (2001), *Impact du Logiciel Libre sur l'Industrie Informatique*, PhD Thesis, Ecole Nationale Supérieure des Télécommunications de Bretagne.

- Kogut B., Metiu A. (2001), Open-Source Software Development and Distributed Innovation, *Oxford Review of Economic Policy* 17:248-264.
- von Krogh G., Spaeth S. & Lakhani K.R., Community, Joining, and Specialization in Open Source Software Innovation, paper presented at HBS-MIT Sloan Free/Open Source Software Conference, June.
- Lakhani K., von Hippel E. (2000), How Open Source software works : ‘free’ user-to-user assistance, MIT Sloan School of Management WP #4117.
- Langlois R.N. (2002), Modularity in Technology and Organization, *Journal of Economic Behavior and Organization* 49:19-37.
- Lerner J., Tirole J. (2002), Some simple economics of open source, *Journal of Industrial Economics* 50:197-234.
- Mateos Garcia J., Steinmueller W.E. (2003), The Open Source Way of Working: A New Paradigm for the Division of Labour in Software Development?, SPRU-INK Research WP n°1, accessible at [http://siepr.stanford.edu/programs/OpenSoftware\\_David/oswp1.pdf](http://siepr.stanford.edu/programs/OpenSoftware_David/oswp1.pdf)
- Muselli L. (2003), Les strategies de licences libres, paper presented at Autour du Libre conference, Paris, May.
- Raymond E.S. (1999), *The Cathedral and the Bazaar*, O’Reilly.
- Williamson O.E. (1996), *The mechanisms of governance*, Oxford UP.