The (LA)TEX project: A case study of open source software

Alexandre Gaudeul
University of Toulouse
France
alexandre.gaudeul@univ-tlse1.fr

Abstract

The TEX typesetting software was developed by Donald E. Knuth in the late 1970s. It was released with an open source license and has become a reference in scientific publishing. TEX is now used to typeset and publish much of the world's scientific literature in physics and mathematics.

This case study serves as a critical examination of the stylized facts uncovered in previous studies of other open source software projects, such as GNU/Linux, an operating system, and Apache, a web server. It is sponsored by CNRS, a French research agency, and is supported by the University of Toulouse in France and the School of Information Management and Systems in Berkeley.

The comparison centers on the historical development of the project, the organization, both formal and informal, that supports it, the motivations of the developers, and the various dynamics that are at work and influence the project.

The case study explores the economic impact of the TEX software which is sold through TEX-based commercial applications and used in the typesetting industry and various institutions. It is an exploration of how the open source nature of the program made a difference relative to what would have happened had it been commercial software.

1 Motivation

I have been working for one year now on a case study of TEX as open source software. Since TEX branched out into many different projects, this case study is in fact a sum of case studies about those different projects, and a reflection on the dynamics of the whole project. This whole project will be called the TEX project or simply 'TEX'. My aim is to provide some elements to improve the way in which open source software projects ('OSSPs') are managed, and also help policy makers gain a better understanding of the open source ('OS') phenomenon.[1] This case study serves as a critical examination of the stylized facts uncovered in previous studies of other open source software projects. Some better known and studied OSSPs are GNU/Linux, Perl and Apache (an operating system, a programming language and a web server, respectively). The TEX project differs from those projects: While TEX did fulfill unmet software needs and was general-purpose software, its users' community was not necessarily technically sophisticated, and the software was not part of a computing infrastructure. It was indeed quite specialized (font design, typesetting) and what is more, had to face intense competition on all sides, from word processing software to industrial publishing software.

There are few case studies that deal with one open source software project and try to look at its functioning in economic terms. In the last few years, open source software economics has been the subject of much empirical and theoretical research. That research relied on an examination of the most well-known and successful OS projects, or on the study of limited aspects of open source software, based on some partial statistical measures like the number of contributors, lines of codes, bugs or release dates.

This case study tries to go beyond these well-trodden areas by studying a less well-known software project, which differs in many ways from those that have already been studied; it also aims at having a global vision of its history and functioning so as to generate new measures of the economic impact of open source. The conclusions from this study chal-

This research paper includes open-ended questions and projects for future research. This is very much a work in progress, and no statements here are definitive. I am very interested in feedback from all participants in the TEX community, and you are invited to point out my errors, false opinions and omissions.

[1] For simplicity, the difference between free and open source software will not be dealt with here, and the term 'open' will be used.

lenge the consensus built from previous case studies on open source software ('OSS') development. This case study goes deeper into the complexity of the internal working of the various TEX projects, and eliminates the 'survivor' bias present in previous case studies by going into the TEX project problems encountered along the way as much as the successes.

This case study is sponsored by GREMAQ, a CNRS research group in mathematical economics at the University of Toulouse in France, and IDEI, a research institute in industrial economics. A widely attended conference on the economics of the software and Internet industries is held in Toulouse every year, and open source software is one important research area for those two laboratories. This case study also benefited from the support of the School of Information Management Systems in Berkeley. I have worked with Jacques Crémer and Jean Tirole in France and Hal Varian in the USA, and I thank them for their advice and suggestions. I also thank the many TEX developers, maintainers and associations members who answered my questions with unflappable kindness.

In the first part of this paper, the theoretical background to this case study is presented; in the second part, the choice of TEX as a case study subject is motivated; and the third and main part presents some preliminary findings.

2 Research background

There are three main themes in the existing body of economic literature on open source software. Economists first tried to explain how people could collaborate freely and for free and produce in that way valuable information goods. Some principles were then expressed for the regulation of such economic activity, and finally, tools were devised to evaluate the welfare impact of OS production.

How do open source software projects work, and why do they work so? The literature on this topic builds upon the theory of incentives: the way somebody is motivated determines what he will do. Bessen (2002) defined the different categories of participants in an OSSP and their motivations. Core developers are those whose work determines the pace of the overall development, as other developers' work depends on what they do. Satellite developers are those who build upon the work of core developers to add features that are geared to special interests. Other developers make that work available to the general public by building interfaces to the program, maintaining distributions, or reporting problems with the software. There is generally an organization that coordinates the work of every developer and defines some goals for the project. That organization usually builds around an individual, usually the initiator of the project but, with time, coordination and development tasks are shared.

The existence of OSSPs can be explained with simple economics — OS software is cheaper than proprietary ones, developers want to work on it to develop their reputation and then trade on it in the job market or to develop an expertise in software they use professionally. It can also be explained with other reasons that Richard Stallman of the Free Software Foundation was instrumental in promoting — free expression of creativity, sense of belonging in a community, ideological motivations, wanting to reciprocate the gifts in software codes made by others, etc. From a technological point of view, the birth of OSSPs may have been inevitable: as people learned how to program and could customize software to their own needs, they developed a common body of work and shared it like general knowledge.

The second theme in economic research on OSS deals with the economic principles that must inform their regulation and legal environment. It uses the theory of organization and public economics to determine how OSSP should be regulated to produce maximum welfare. There is for example an important debate over what license terms are best in what setting. License terms balance the need for control over the development of the software versus the possibility of change under the influence of others, and balance private incentives versus group incentives: Proprietary license terms give individual developers more control over their work, while GPL-type licenses reduce individual economic incentives — the economic surplus generated by software cannot be appropriated — but may generate higher overall welfare. BSD-type license terms stand in between. The legal environment also influences the level of innovativeness in software design — people may not want to contribute their best ideas to OSSPs — but a wider pool of developers who are not concerned about the acceptability of their ideas to the wider users' community may end up generating more original ideas. License terms also influence how the welfare will be distributed, as they may favor developers vs. end-users. Finally, proprietary software favors efficient coordination in a closed environment at the expense of keeping development secret to most people.

The third theme in OSS economics is the interaction between not-for-profit and commercial software. Industrial economics and game theory explain how both types of development methods compete and complete, and how commercial firms use

OSS and draw on the OS developers' community. The various strategies for making money on OSS are studied — selling CDs, manuals, developing proprietary software based on OS and using it for professional purposes or selling it to the public, selling advice to OSS users, etc. The efficiency with which both types of software are developed are compared, as well as the end-product's quality and how they compete on the software market. Because we already have tools to evaluate the welfare effects of proprietary software, the comparison between OSS and proprietary ones gives some leads for the appraisal of their welfare effects.

3 A reference point: GNU/Linux

Before presenting of some preliminary findings, the main differences between TEX and Linux are outlined; this gives a reference point to people who learned about open source from the example of Linux. I also motivate the choice of TEX as the subject of this case study. The TEX project differs in many important ways from the Linux project. They were not developed in the same period, TEX has a much longer history, and they are distributed under different license terms. The TEX project's size, evaluated by the number of people who develop TEX and LATEX, is considerably smaller than the size of GNU/Linux project; as a matter of fact, GNU/Linux distributions generally include TEX and LATEX. Finally, the goals of the projects were different.

Donald E. Knuth developed the TEX system in the late 70s, before the Internet came to be the tool it is today for organizing open source communities. The community surrounding the software went through many changes over many years, accompanying the evolution in the standards used for publishing and in the way software developer communities work. Linux, on the other hand, was started in the 90s, relied on the existing open source community, and used tools already developed for the GNU project and others. The TEX project provides a long-term view of the history of an open source software project. Its relatively self-contained developer community went through several stages in its development: this study may thus help in predicting the future of other more recent open source software projects.

TEX is a medium size software project; it is not an operating system like Linux, but is still a complete typesetting system with many interdependencies. TEX provides a sufficient level of complexity to be the subject of a self-contained case study, but small enough to be studied as a whole. The project can be understood without relying on catch phrases and slogans, unlike many studies of Linux.

The communities that grew up around the two systems were different. TEX was developed by academics as part of their research programs, publishers who used it for typesetting books and journals, and developers who provided commercial versions of the software. Development of TEX was pragmatic, funded by government research programs and universities, by its release under proprietary license terms, or from the revenues of selling CDs and manuals. Linux on the other hand drew a community that was motivated by more abstract, ideological goals — building an alternative to Microsoft — or by the programming challenge — getting to work on an operating system. Of course, the contrast should not be pushed too far; independent, 'amateur' developers who were not motivated by profit also contributed to the development of TEX.

The license under which TEX is distributed is essentially a BSD type license, while Linux was released under the GPL. Their license terms made a difference in the way both software developed; BSD licensed software must compete with proprietary systems based on the same source code. Because of that higher level of competitive pressure — and maybe for other reasons too — BSD projects are usually more disciplined than GPL ones; all OS development efforts bear onto the same, coherent distribution. This guarantees in principle that no development effort is wasted and that the OS software doesn't split into many incompatible projects.

The LATEX Project Public License thus promoted the creation of a single common TEX distribution; all changes to it must be distributed with the original distribution. The TEX system was thus very stable, but it was difficult for newcomers to integrate and influence the team that decided what that distribution was going to consist of. There were times when many competing versions of the same package existed until one became dominant and a part of the standard distribution. Therefore, no one person asserted him/herself as a leader for the TEX project; its development was the product of the competition between packages, and each package in TEX remained under the control of one person or of a stable and limited set of developers.

Modules in Linux drew a more diverse set of contributions and there was thus the need for a leader who would coordinate and integrate contributions. D.E. Knuth implemented changes in TEX's core (`tex.web` and the kernel) after consultation with other developers but essentially alone, as he took sole responsibility for the TEX core. Linus Torvalds, on the other hand, had to integrate changes in

the code that were proposed by others, because anyone could take the kernel and make his own changes in it. This is how D.E. Knuth's authority was built into the system while Linus Torvalds had to assert his authority based on his charisma as a leader.

TEX was user-oriented from the beginning; it was meant to provide an interface between authors and publishers. People without any programming background were to be able to learn how to use it. This is in contrast with Linux or Apache, which were meant for people with a programming background. This difference allows one to test whether OSS can be popular beyond the programming community. While Linux versions were released very frequently, the users' orientation of TEX led its developer to release new versions of their packages only after consultations with the user base, and only after having made sure they would maintain compatibility with older versions of the software and that they did not contain serious bugs. The development of Linux was made in the open while TEX packages were mostly created in small developers' groups and released only after full completion. In both cases, though, the interface between developers and users was taken care of by the people who managed the distributions of the software — those who organize and classify others' independent work, make their code work together, and choose which packages to include in a standard installation of the software.

4 Some preliminary findings

This part shows you have to be very careful when writing the initial code of a software project, as it will influence all future development. Any choice at this stage should be carefully evaluated using the lessons from the past.

This part is organized into three main sections. The first deals with the output from the TEX OSSP — the software code. Its initial quality influenced its later development. The software's quality is evaluated by comparing it to equivalent proprietary software. The second section examines the software development process and its dynamics, and will focus on its leadership: OSSPs need independent minded leaders who begin by implementing their ideas and only then share the result with others. The third section is a study of the framework in which the development of the software took place — it is concerned with the governance and institutional design of OSSPs. TEX provides a rare example of an OSSP where users organized to influence the development of the software.

4.1 TEX code: Characteristics and quality

4.1.1 Importance of the initial code

There is a conflict between the perfection of the coding of the initial software and the ease with which it can be changed afterwards. Knuth wanted to produce compact software that would run fast and be devoid of any bugs. He thought a stable system was preferable to an evolving one. This was justified for TEX, as it was to become a system used by non-specialists. The OS development model — 'release early and often' — would have led to much confusion in the user community, and to compatibility problems for those using different versions of TEX.

Knuth's code was originally organized in modules but, as it got optimized, the code became very tightly integrated. Each part became dependent on each other and the whole began to look monolithic. The language that was chosen at the beginning (Pascal) soon went out of fashion, and the software's restrictive license terms made it difficult to change, as changes couldn't gain official status.

On the other hand, while the software remained monolithic, TEX82 was a complete reworking of TEX78 that made many settings parametric instead of automatic, making powerful macros from TEX's primitives possible. This satisfied TEX developers for a long while, during which the core code remained firmly under Knuth's control.

This is why it is only quite late in TEX's development that the core's limitations became apparent, and it became necessary to make it easier to change, for example by organizing its modules into libraries. TEX's license terms are such that the name "TEX" is reserved, so that Knuth was able to freeze TEX's core. This would not have been a problem — developers always could take the TEX program and rename it — but since Knuth did not designate a successor, there was no focal point on which developers could synchronize. Developers were not able to change the core, or more to the point, couldn't initiate a group dynamic to adopt the changes they made. This would have required a long-term commitment, perfect knowledge of the program and close coordination since any change by one would affect all the others. It soon became clear it was not possible to lead such a project with people linked only through electronic means; the core of TEX had to be reworked by a devoted team so as to make it modular.

This task was taken up by the NTS team, but it took too long to deliver a finished product. When NTS was finally delivered, it was not used except for experimental purposes. This shows the importance

of getting things right in the first place; by the time the program had been rewritten, most TEX users and developers had preferred to base their future use of TEX on other, less ambitious, alternatives, such as pdfTEX.

In summary, independent development of the program was delayed because TEX evolved into a monolithic program that was intended to become a standard in publishing and was developed in a closed academic setting. While Knuth's objectives were realized, subsequent development was difficult, in a new setting where TEX users and developers had to coordinate through electronic means and the OSS developer community was established around concepts and tools different from those of the first TEX implementors. The TEX program had to be translated into the now-ubiqitious C programming language, and when the rewriting of the core into a modular structure proved impossible, the efforts had to be directed towards helping TEX users manage the TEX legacy by making it compatible with the new typesetting standards.

On the other hand, it is not clear that TEX could have been developed from the beginning in an OS fashion. While the core did not get changed in an OS way, the program did attract a lot of independent development, notably on LATEX. That TEX's core was not developed strictly according to OS paradigms could be evidence that OSS development methods are only appropriate when a base product has already been completed but are difficult to put into practice for the base product. It is also possible this was due to the nonexistence of an organized OSS developer community at the time.

4.1.2 Impact of TEX on innovation and welfare, and evaluations of quality

In this section, the quality of the software from various points of views — users, developers, and computer science researcher — is compared with competing proprietary software.

There was no software even remotely up to the standards of TEX when it was developed.[2] The general program used at that time for typesetting was called ROFF, a text formatting language/interpreter associated with Unix, and for a long time there was some competition between the partisans of ROFF and TEX. The main competing software for the casual user is now Microsoft Word. Even though Word is WYSIWYG while TEX is not, and the audience is therefore very different, the two compete because TEX saw itself as a potential standard for document exchange. The main competing software for typesetting of complex mathematical documents in the publishing industry is 3B2. Adobe Framemaker and QuarkXPress are also popular alternatives.

A frequently asked question is whether OSS replaces proprietary software and whether it undermines innovation by imitating proprietary companies. In the case of TEX, it is quite clear which way the inspiration went. Some aspects of TEX were imitated, for example the equation editor in MS's Word and TEX's hyphenation and justification algorithm in Adobe's InDesign. Other commercial software eased the use of TEX by adding a graphical user interface and porting it to other platforms — this is the case of Personal TEX's PCTEX, the first IBM PC-based TEX system, or of MacKichan Software's Scientific WorkPlace which integrates TEX and Maple. It is TEX which inspired commercial development much more than the reverse.

It is also not clear whether commercial and OS products complement or substitute for one another. There are examples of dual use, some typesetting firms using TEX internally and delivering the finished product with 3B2. There are also examples of users and firms switching back and forth between OS and proprietary software. TEX Live, for example, did take some business from commercial implementations, especially since it is easier to maintain using Linux based network management software. The competition is very rarely frontal, and few TEX projects see themselves as ideologically opposed to commercial software. TEX did take the place of other commercial software though, but while it replaced obsolete proprietary typesetting software at the AMS, it also inspired other proprietary software (principal concepts, line breaking algorithm, syntax) and it paved the way for getting typesetting software in the hand of the users instead of that of the typesetter. It initiated a new workflow in publishing. Additionally, some of the first people to use TEX did not see commercial software as an alternative and TEX was a way for them to obtain functionality not present in (affordable) commercial software.

Finally, the development of TEX was encouraged by potentially competing commercial software. Hàn Thế Thành received a scholarship from Adobe to develop the pdfTEX program; this was in the interest of Adobe as it wanted to gain more general acceptance for its software and was also a way to encourage exchanges with the OS community. The competition between OS and proprietary software is based on subtle mechanisms that are deserving of further study.

[2] With the exception of a couple of very expensive, proprietary systems, e.g., Penta. *Ed.*

While a comparison of the welfare generated by TEX with that which would have been generated by a proprietary program may appear to be a futile academic exercise, TEX was developed as an alternative to commercial software that was used by the AMS in its publishing section, and the AMS did ponder what was the best option: wait for commercial software to be released that would fit their need, or give the impulse to a new, open source, software. The comparison between TEX and a hypothetical equivalent commercial software can be made in terms of innovation, responsiveness to users' needs, pace of development, capacity to integrate into existing systems and the efficiency with which the software is developed.

Proprietary software is sometimes out of touch with users, as developers are not users. But in OS, developers are sophisticated users, which means the software may not be at the reach of the average user. As far as OSS is a tool for average users whose needs are not fulfilled by proprietary firms, then its development may be as misdirected as that of closed-source software, although in other ways. However, the development of TEX and LATEX was made after consultation with professionals from the publishing industry and meetings with the AMS — the first sponsor and user of TEX. In the summer of '79, Barbara Beeton of the AMS and Michael Spivak — both of whom went on to important positions in the TUG organization and others — spent time in Stanford developing TEX macros to test TEX capabilities for such AMS requirements as generating indexes. Their work led to a series of suggestions for improvements, and to the AMS giving its backing to the project.

The LATEX3 project members also consulted with the AMS and various TEX user groups, publishers such as Addison-Wesley and Elsevier, and got support from companies, some that sold TEX-based software — Blue Sky Research, TCI Research, PCTEX — but also Digital Equipment Corporation, Electronic Data Systems, etc. David Rhead gathered the wishes of users from email discussion on the LATEX discussion list. Those wishes were mainly about the page layout specifications and the user-interface design, things that are of primary concern to users and not so much to developers. This close collaboration with professionals in the typesetting and publishing industry, which can be illustrated in many other examples, goes against the view that OSS that is too geared towards specialist use will not be successful (Schmidt and Porter [2001]).

It is often said that the pace of improvements is quicker in OSSPs. Improvements in proprietary software are not released frequently, since there is a cost to doing so, and their owners want the improvement to be valuable enough for existing users to buy it. But with OSS, it is difficult to coordinate the user community on the most recent improvement; this is a problem as the software is used for collaborative work, and people want stability. In the case of TEX, the solution was to design standards for the classification of packages and requiring new packages to be distributed with older, approved ones so as to guarantee the availability of a complete working set of packages to users.

Standards are difficult for proprietary software firms to adhere to because they want to protect their user base — prevent it from switching — and also because the source is closed, so that it is difficult to create applications linked to it. However, software firms that propose development platforms to programmers are also interested in the promotion of their platform, and usually are able to establish and maintain them as a standard. OS projects on the other hand generally find it difficult to coordinate on a standard. While this may not be a problem because OS is usually platform independent, it is difficult to keep code operational when there are constant changes to the underlying operating system and compiler platform (Torzynski [1996]).

While the sharing of information may be done less efficiently in OS projects than in proprietary firms, the pool of information that can be shared is expanded. Many contributors to TEX would probably never have worked in a commercial firm, and even when they were hired in commercial firms, such as Elsevier, they continued contributing their improvements to the wider community. Overall, if it is possible to prove that OS developers would not be able to do what they do in a closed environment and that what they do would not be done by proprietary software, then OSS is beneficial. As an OSSP develops however, it can grow to come into competition with closed source: there is competition at the fringe, when users could use both.

The situation is complicated by the fact that some proprietary software may be based on OS and compete with pure proprietary software. Some work would need to be done to compare publishing firms that use OS software (Hans Hagen's Pragma ADE in the Netherlands, B. Mahesh's Devi Information Systems in India) versus firms that use proprietary software. There is a difference in the nature of upfront cost, maintenance efforts, level of support, possibility of improvements, capabilities, etc.

4.2 The process

4.2.1 Dynamics of the project

Various forces direct the development of TEX, due to the different concerns, objectives and priorities of its developers who come from different fields and make use of TEX in different ways. Over time, with the user base changing and the software's environment evolving, different types of priorities have emerged: in the first period, Knuth's own objective was to develop software that could do mathematical typesetting by computer that was worthy of the best manual typesetting tradition. Then, when he felt his objective was achieved, the AMS wanted to make this instrument available to the wider mathematical community, and sponsored the development of AMS-TEX and its subsequent merging with LATEX.

Later on, the objective of the subsequent core developer teams was to make use of new computer capacities and make TEX more easily extendable with the use of new programming tools (Ω, NTS), while also establishing a standard LATEX to prevent forking — LATEX 2_ε by the LATEX3 team. At the same time, some work was necessary in making TEX able to produce not only pdf and html documents, but also Framemaker and Word documents. The work on making TEX compatible with proprietary standards was first done by commercial companies. Among the priorities, keeping up with competitors' functionality, such as Adobe or WordPerfect, does not seem to have been important, as TEX developers advocated the use of free source fonts instead of commercial fonts, and mark-up-based document writing instead of Word-like WYSIWYG programs. There are, however, some open source projects trying to achieve greater user friendliness — David Kastrup's preview-LATEX package to ease editing, LyX, a document processor using LATEX in the background, GNU TEXMacs, inspired by TEX and GNU Emacs, etc.

Competition between different development philosophies also worked to determine what works and what doesn't and which way the overall project had to go. An illustration is the difference in philosophy between the NTS project and pdfTEX: the NTS team wanted to keep compatibility with the initial version of TEX, while totally rewriting the code — rewrite the WEB Pascal program into the Java programming language. pdfTEX, on the other hand, was based on the C implementation, less generalized in scope, but easier to work on (Taylor [1998], Hàn Thế Thành [1998]).

4.2.2 Limits of the project

Does the OS development process or the specific OS institutions that support development put limits on the growth and success of open source software?

Growth and success are important because even if the software functions in accordance with the stated aims of the project initiator and the initial user community, it will quickly become obsolete and useless, even to those same people, if it is not maintained to keep up to date with the changing software environment. This can justify changing the aims of the software's community, even in ways not to the advantage of the project initiators, if that can make the software more attractive to new developers.

The pace of development slowed over time. The graph represents the number of bugs found by Knuth in the core program through time. After TEX82 was released, Knuth stopped implementing general user requests, except for allowing 8-bit input in 1989. Since the whole TEX system refers back to the core of TEX, its pace of development is indicative of what is happening in the wider TEX community.

There is however a difference between development and diffusion. As the software's main tree development is blocked, it can still be adapted to new platforms, translated, and people trained to use it. Nonetheless, it is still true that it will be more difficult to diffuse if there is nobody ready to make the necessary tinkering in the software code to permit adaptation to new usage.

The diffusion of TEX can be evaluated by looking at the number of requests for support in TEX-related newsgroups, the number of TUG members, and the number of academic papers written with TEX. While postings to the English-speaking newsgroup reached a plateau — probably because most questions were already answered in English and referenced in FAQs! — newsgroups in other languages attest to the vitality of the international growth of the user base.

There are some technical limits to the development of an OSSP, and those are different from those that limit the growth of proprietary software. Those limits are due to coordination problems in the development and support. Initial choices in the software programming are hard to change because that requires more coordinated effort over a longer period of time than most OSSPs are able to provide. This means a project can get stuck with outdated standards. There is also difficulty in keeping the original programmers to remain committed to the project.

There are only a limited number of people who may use the software, even if it tries to broaden

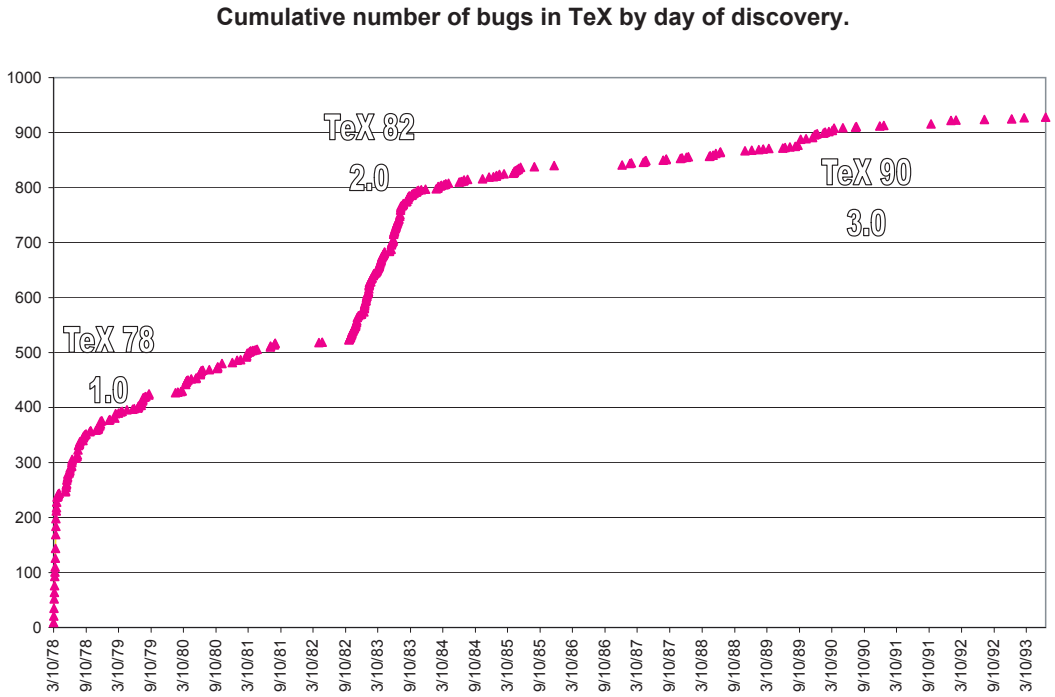

Figure 1: The bugs of TEX.

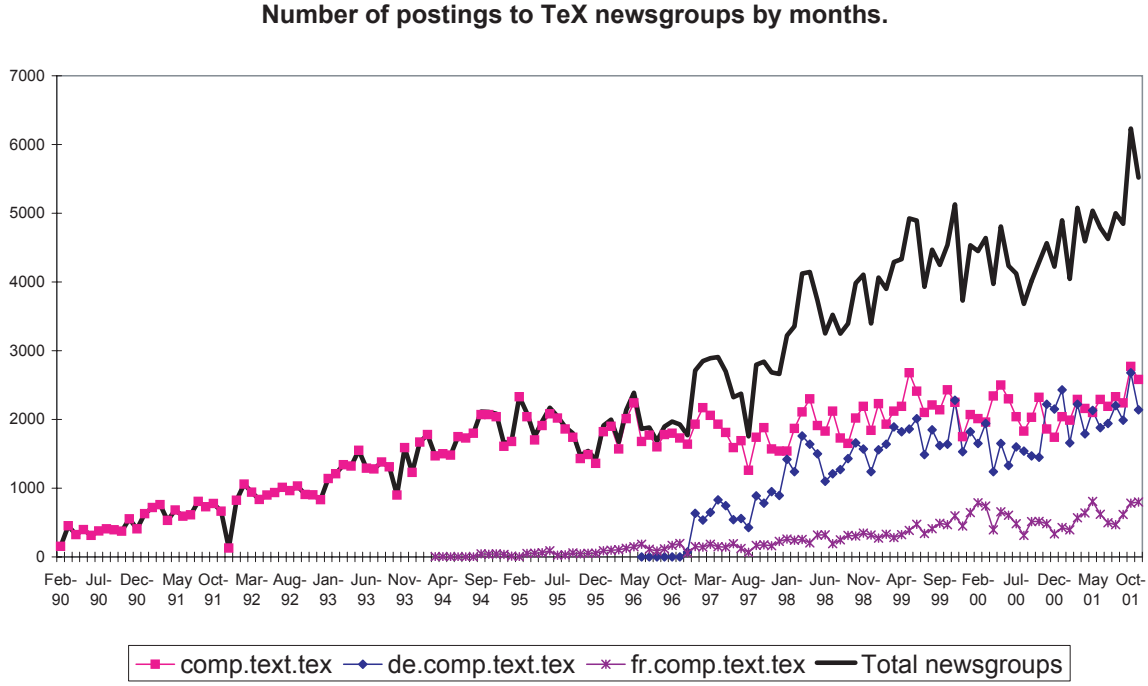

Figure 2: Monthly postings to TEX-related newsgroups.

its appeal. The software progressively reaches all of its intended audience, or is supplanted by other software for that audience.

Some aspects of the software's basic concept are difficult to change, for example its typesetting mark-up language, and this put limitations to its appeal. The concept becomes fatally obsolete, even if it made sense when it was first thought up. Here, other mark-up languages appeared (MathML), and other typesetting engines did not necessitate as much learning — Microsoft Word is less powerful than TEX but has a more gradual learning curve — or were more tightly integrated with new standards and necessary capabilities — Adobe's InDesign to produce pdf files or to use Quark XPress document files. The ConTEXt and pdfTEX projects were attempts to broaden the capabilities of TEX to keep them up to date with what was necessary for online publishing. TEX was oriented toward printing, and was not able to easily provide the kind of interactive color complex documents with figures needed for online publishing. It is also based on a document exchange standard (DVI) that has never achieved the popularity of the pdf format. The pdfTEX and ConTEXt projects thus had to make changes in TEX's conception to adapt it to new needs. Other projects made TEX XML-compatible.

In summary, TEX was at first in the forefront of mathematical publishing, but then had to adapt and borrow concepts from new and popular software projects — and this process met with some resistance. The number of people who were interested in those improvements was limited to a fringe, and they found it difficult to advertise their projects beyond the people who already were using TEX.

Finally, the OS organization imposes some limits: the originator is ready to support only a limited number of people; Knuth had other priorities, the writing of his monumental series *The Art of Computer Programming*. TEX was in fact originally meant only to typeset those books.

Limited explicit mechanisms (interface specifications, processes, plans, staffing profiles, reviews), extensive reliance on implicit mechanisms (personal relations, customs and habits) and on one-to-one interactions in small teams (communications only mechanism), mean that the development process did not scale easily. Choosing an OS development process put limits on some areas of the software's development.

4.2.3 Leadership

There is a need for a leader in an OSSP. The production of an OSS cannot be described as being peer-based. Patterns in the history of the projects related to TEX provide lessons on what constitutes good leadership in an OSSP because they provide a broad sample extended though time. The reason for the projects' successes and failures, which can only be determined through time, can thus be analyzed. The most effective type of leadership seems to consist in first developing independently some implementation of an original idea and only releasing it into the public when it is already well advanced. Projects that began by announcing their goals without backing their ideas with some implementation generally failed because other developers contested their technological decisions or couldn't contribute. There is therefore a limit to the power of consensus building and cooperative development; it is frequently better to go it alone and then ask for help once the project is well along.

Knuth's leadership was characterized by a heavy involvement in the beginning and the choice to leave later development to others. That leadership style was very successful for the beginning of the software's development, but the desire to preserve some stability in the program produces the danger of impeding its development. This could have led to forking if the community built around TEX had not been so cohesive.

Although there is a need for a leader, there also are problems in coordinating on one leader. An example of a successful leader was Hàn Thế Thành, who initiated the pdfTEX project to directly output pdf files from TEX. This is seen as a successful project because Thành released his work only after having done the preliminary groundwork, and was then able to let other developers take the initiative in applying and enhancing his work. The Omega project encountered problems because, while it communicated its goals early, and implemented innovative ideas to enhance the multi-lingual capabilities of TEX, it did not at first attract developers beyond the initiators and had problems convincing the TEX community it would one day become fully implemented. That project was first presented in 1995 and it is only now that it is gaining momentum and being supported by the TEX community.

A large part of the difference between those two projects is often attributed to the leadership style of their initiators; the fact that the Omega developers did not deliver on their claims rapidly made the established leaders in the TEX community doubt that project was worth getting involved in. As we will see below, however, the main difference between the two projects was perhaps not the difference in the way they were led and in the ability of its program-

mers, but in the acceptability of the project to the existing developers and users. The pdfTEX project did encounter some resistance at its beginning from people who thought other ways to generate pdf files from TEX input were preferable, but the importance of the ultimate goal was not in question. This was not the case of the Omega project.

In short, a project leader will be seen as a good leader depending on whether he is allowed to work within the existing system. If he is not accepted and does not get the support of other developers, then his project may end up badly in a self-fulfilling prophecy. A project will find it difficult to thrive if it doesn't get the support of the establishment, so that most successful projects will serve the needs of existing users and developers, and not those of potential newcomers.

The most consistent leadership was given by organizations: the AMS, which ensured that TEX served the mathematical community, and TUG, which ensured it was user-friendly. The AMS was the main leader of TEX's development. It provided financing for a user group, made propositions to developers, and gathered them to establish objectives. The involvement of the AMS was thought out well in advance, as they wanted to become involved in helping to develop a document preparation system, instead of waiting for a commercial system to be provided to them. They needed a system that was compatible with most hardware, simple, flexible, and cheap; it was to run on mainstream computers. The AMS, as well as other TEX sponsors, were conscious that there was competition between the various possible uses of the software. Because of limited human resources, the product's development could not be led in the way each constituency would like.

There was therefore a need to define an allocation process for development resources. This is how the AMS decided to sponsor the development of a modern system based on TEX that would fit its own use (AMS-TEX), and also sponsored TUG. Its role was to form a group of people who would be able to use the tools recommended by the AMS. The AMS couldn't hope that academics would use the TEX system if it didn't also provide them with the means and training to do so. This sponsoring by the AMS had an impact on the development of TEX that went far beyond the means involved because it served as a signal to other sponsors that TEX was a valuable project that would ultimately, willy-nilly, be a complete system. This is why TUG also attracted sponsorship from various hardware companies and universities. The AMS progressively lowered its financial contribution to TUG and it is interesting to note that TUG's revenues and membership declined after the release of the final version of TEX in 1990. It would be interesting to look further into the impact AMS's support had on TUG, and the influence TUG had on TEX's development.

Leadership was also provided by users and mediated by user group organizations. This part exposes the many initiatives coming from the TEX users. The identification that is often made between users and developers is not correct, as even users who are not developers have an impact on the development. An OS project is not led through a competition mechanism where the best project wins, it is led by the user who is ready to devote time and effort to the project toward a defined goal, in this case the goal of an association distinct from the TEX community. TEX had no purpose of its own, but TUG did have a mission statement: To be a meeting place for users and developers, of course, but also to use that central position to serve the aims of its sponsors. TUG served as a meeting point between users and developers of TEX, the two not being exclusive. Articles in *TUGboat* were often written by users explaining the use they made of TEX in their respective fields, and outlining the problems they encountered. The various uses of the program led to various pulls (queries to developers) and pushes (independent development) on the program. The problem was then to harness those, integrate interesting contributions into the main distribution, and not have dead-ends. This was done by encouraging independent developers to work with the core developers so as to attain compatibility with the existing TEX system.

Users wanted to protect their investment in the software. Given the weight of legacy, there was reluctance on the part of the user community when faced with changes in the TEX system. Indeed, many of them had written their own modifications of TEX to fit their own use, and were not willing to abandon those in favor of a new system that would provide only minor improvements to them; multilingual typesetting for example was of no use to most users but requires extensive changes. There was no need for change for most users, as any change they needed could be done with macros — even though the TEX macro language itself led to very complicated and badly structured programs. It was very difficult to attract older users on an alternative, and indeed, the main hope for some developers who pursued changes to TEX came from new users in non-Latin countries — the Omega project. New categories of users who did not have the same legacy issues succeeded in breaking the status quo. Various user con-

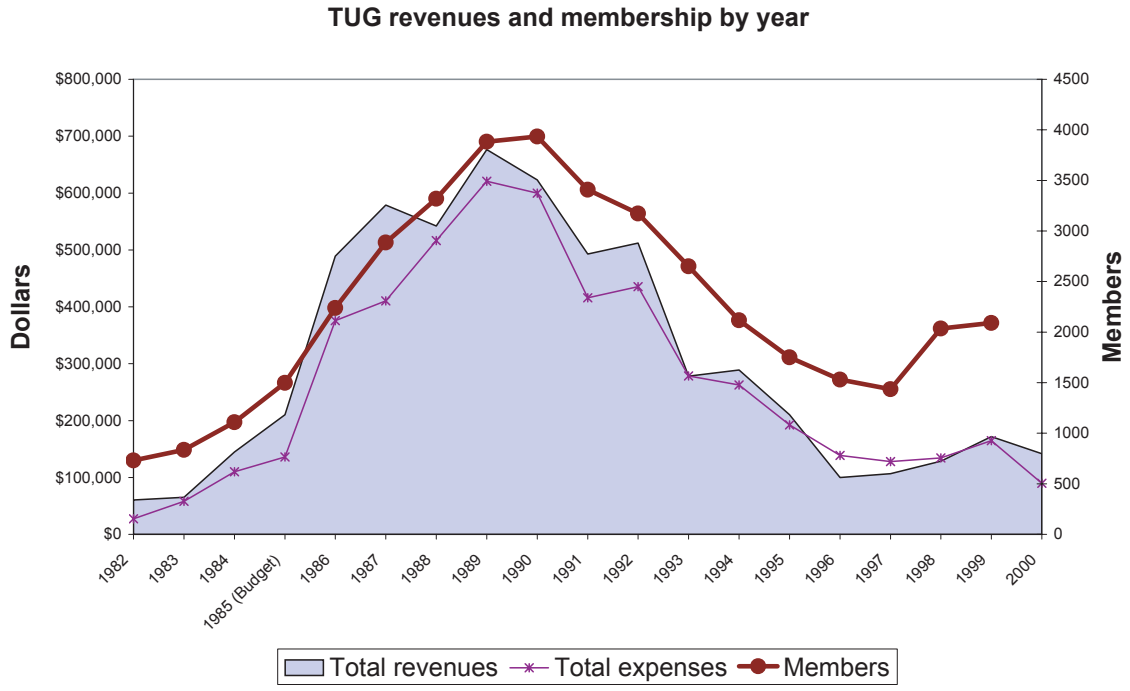

Figure 3: TUG's revenues and membership.

stituencies, or categories of users, had different requirements: The Europeans came to adopt LaTeX, while the Americans used TeX, because LaTeX had not reached a sufficient stage of development to be interesting to them at the time they adopted the TeX system. The American organization had been built under the lead of typesetters, publishers, software companies and university institutions who were interested in mathematical publishing, while the European groups were led by individual users, sometimes active in universities or educational publishing. Many of those users' needs were not satisfied by the user group based in the US. This is how many initiatives were made in some European LUG before being accepted by the main organization.

For example, while the US organization had an established TeX tape distribution system, and while American developers knew each other well enough to coordinate development on a one-to-one basis or in conventions, the European users saw the need for a TeX code central repository. This was realized by a group of volunteers at Aston University in the UK, which made it possible for users to download the latest developments in the TeX system. This group inspired the development of a package classification system, the TeX Directory Structure, which served as the model for TeX distribution everywhere; this common system facilitated the installation of the TeX system. That initiative led to the creation of the CTAN archives in 1990, which drew contributions from the American organization too.

Another example of user-led initiatives is the initiative by the Netherlands LUG to develop and distribute 4AllTeX in 1993, a TeX distribution on a CD that was intended to be of use to an end user with no programming background. While the Netherlands LUG could not possibly have taken up the task of delivering a complete user-friendly distribution, it gave the impetus to a wider concerted effort, the TeX Live project. TeX Live adopted many of the ideas in the 4AllTeX distribution — choice of programs, organization of packages, and more.

The tensions between TeX constituencies translate not only in user initiatives, but also in different objectives for groups of developers. There was a conflict between pursuing a standardization of LaTeX that would fit most users' needs and allow easy interchange of documents between all users, and going forward without an overriding concern for compatibility, to serve the needs of larger classes of users.

4.3 The framework: The TeX rules and culture, and how they evolved

The TeX rules and culture differ from those of other communities, first due to the license terms, but also because Knuth gave authority to some lieutenants

to act on his behalf. Initiators, like Knuth or Lamport, effectively blocked development until pressure built up to a convincing expression of collective choice, when for example users and developers came to Knuth with requirements for TEX82 or TEX90. That strategy did succeed in reducing the workload on the initiators and in reducing unnecessary developments, but it did not encourage independent development.

The development of the software became more and more decentralized because of two effects. The first one is that the program became more complex, and the number of specialized programs linked to it rose — programs to make figures, indexes, nationalizations of the software, programs to translate TEX input into outputs other than DVI, various interfaces, various rewriting of the software into other programming languages, etc. The second one is that technology made it possible to coordinate projects one-to-one through electronic means, instead of all contributions going though a central body which then reflected it to all after having filtered the noise (errors, unwanted developments, etc.).

For example, the way submission of changes to the TEX software were made evolved through time. There is a contrast between new systems and older ones. For TEX, bugs were submitted to Knuth via filters, people who had to make sure the submission was valid. Changes in the software were suggested to Knuth, who then determined how those changes were to be embodied in the software. In the LATEX3 project team, developers exchanged code via private e-mail and met person to person to discuss changes. Of the newer project, some like MikTEX were listed on Sourceforge, and used all the tools now available to coordinate OS projects — CVS files, central repository — and many, like preview-LATEX, accepted contributions by totally unrelated volunteers.

One important part of the TEX community's culture was that it tried to develop user-friendly software based on OSS. This makes TEX a different case than other well-known OSSPs that have been studied before (Apache, Linux), and will help to determine if OSS can be user-oriented, a mass-market product. From the beginning on, the program was intended to be used by non-programmers: secretaries, researchers, ... In fact, one of its main 'selling' points at the beginning was how easy it was to install and learn; the syntax was meant to be natural, and Knuth wrote a complete manual for the program at the same time as he developed it. Even the programming language was meant to make the coding easy to understand; the coding was documented along the way using a language and a method, literate programming, developed by Knuth.

Also, TEX was developed with non-programming concepts in mind, i.e. concepts that were of no use for the greater programmers' community. Indeed, TEX was meant to be a translation of the best typesetting practices into a programming system. This served the needs of the typesetters, the publishers, the academics, but not those of the typical programmer who is not involved in typesetting. TEX came from the lucky coincidence of one person having the need for better typesetting and the ability to follow up on that need.

The problems that have been classically invoked to say that OS programming could not produce user-friendly, mass-market programs are that it cannot generate a good user interface, that a users' orientation requires a different turn of mind than that of OSS developers, that it is not possible to coordinate developers efficiently enough in an OSSP so as to release a fully functional pre-packaged product, and finally, that OSSP developers do not have the means or the will to communicate with end-users.

A good user interface requires a lot of work and is time consuming but is not needed by somebody who already is able to use the software. Ulterior (profit) motives must come into play: this is where commercial organizations have a role, at least until the OS organization is strong enough to be able to produce an easy to install and use program. This is what happened in the TEX community, where commercial implementations of TEX appeared in the mid-80s and provided the only user-friendly alternative during at least 10 years. MikTEX was the most popular open source user-friendly interface to TEX, but worked with Windows only, and it is only with the release of the first TEX Live distributions that a complete easy to install TEX distribution was made available for Linux.

There were also individual OS initiatives to make the software easier to use; preview-LATEX and TEXview provided graphical interfaces. Other collective initiatives were driven by ideology; LyX is such an attempt at providing a convenient TEX interface. All those encountered difficulties in providing an interface that would work with all different possible and ever-changing installations of TEX. They were surprisingly loosely coordinated with the core group of TEX developers, maybe because the development of those interfaces cannot possibly take account of all the changes in the core program, so that they were based on older versions; their development thus didn't necessitate close coordination with leading core developers.

A stable user-level platform was needed because

TEX documents had to be easy to share for it to be considered a standard and adopted widely. Therefore a core product had to be defined, to which other things were added and with which they had to be compatible. This is where a central authority, and defining limitations in what will be supported or not, came into play. The LATEX3 team played that role by defining a 'core' LATEX package, defined as what the core team thought it had the time to maintain. The TEX Live distribution helped in focusing energies by defining what would be distributed more widely to the end-users, thus guaranteeing to developers that their work would have some impact in the user community.

5 Conclusion: Work to be done

While many participants in the TEX community were interviewed, and a large documentation on the TEX system and the history of its development was gathered, that knowledge still has to be organized in a systematic way. Some salient features will be further developed; for example, a more careful analysis of the determinants in the success of an OSS subproject could be made. This would allow a better definition of quality, leadership and support in an open source environment.

6 Bibliography

6.1 (LA)TEX related articles

Bodenheimer B. [1996] "Questions souvent posées sur (LA)TEX", Cahiers GUTenberg, 23 April 1996.

Clark M. [1989] "Olde Worlde TEX", TUGboat 10(4), 1989 Conference Proceedings.

Gaudeul A. [2003] "The (LA)TEX project: A case study of open source software", Working Paper, January 2003.

Hàn Thế Thành [1998] "The pdfTEX program", Cahiers Gutenberg, 28–29 March 1998.

Knuth D.E. [1989] "Notes on the Errors of TEX", TUGboat 10(4), 1989 Conference Proceedings.

Knuth D.E. [1989] "The Errors of TEX", Literate Programming, CSLI Lecture Notes, no. 27, 1992.

Knuth D.E. [1989] "The new versions of TEX and Metafont", TUGboat 10(3), October 1989.

Knuth D.E. [1991] "The future of TEX and Metafont", TUGboat 11(4), January 1991.

Knuth D.E. [1998] "The Final Errors of TEX", Digital Typography, CSLI Lecture Notes, no. 78, 1999.

Lammarsch J. [1999] "The History of NTS", EuroTEX '99 Proceedings.

LATEX3 Project Team [1997] "Modifying LATEX, " TUGboat 18(2), June 1997.

Mittelbach F. and C. Rowley [1997] "The LATEX3 project", TUGboat 18(3), 1997 Conference Proceedings.

Skoupy K. [1998] "NTS: A New Typesetting System", TUGboat 19(3), 1998 Conference Proceedings.

Taylor P. [1996] "A brief history of TEX" in "Computer Typesetting or Electronic Publishing? New trends in scientific publications", TUGboat 17(4), October 1996.

Taylor P. [1997] "Présentation du projet ε-TEX", Cahiers Gutenberg, 26 May 1997.

Torzynski M. [1996] "Histoire de TEX sous DOS et Windows à l'ENSP de Strasbourg", Cahiers Gutenberg, 25 November 1996.

Advogato interview with Donald E. Knuth [2000] `http://www.advogato.org/article/28.html`.

Interview with Leslie Lamport [2000] "How LATEX changed the face of Mathematics", DMV-Mitteilungen, January 2000.

LATEX Project Public License at `http://www.latex-project.org/lppl.html`.

The TEX Users Group (TUG) at `http://tug.org`.

The LATEX3 project at `http://www.latex-project.org`.

NTG TEX future working group [1998] "TEX in 2003", TUGboat, 19(3), 1998 Conference Proceedings.

6.2 General articles on open source

Anderson R. [2002] "Security in Open versus Closed Systems—The Dance of Boltzmann, Coase and Moore", Working Paper.

Behlendorf B. [1999] "Open Source as a Business Strategy", Open Sources, O'Reilly editors

Benkler Y. [2001] "Coase's penguin, or, Linux and the Nature of the Firm", Yale Law Journal, 112, October 2001

Bessen J. [2002] 'OSS: free provision of a complex public good', `http://www.researchoninnovation.org/`.

Bezroukov N. [1999] "Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism)", First Monday, 4(10), October 1999.

Brady R., R. Anderson and R. Ball [1999] "Murphy's law, the fitness of evolving species, and the limits of software reliability", Cambridge

University Computer Laboratory Technical Report no 471, September 1999.

Crowston K. and B. Scozzi [2002] "Exploring the strengths and Limits of OSS Engineering Processes: A Research Agenda", Second Workshop on Open Source Software Engineering, 24th International Conference on Software Engineering, Orlando, USA, May 25, 2002.

Dalle J.-M. and N. Jullien [2001] "Open Source vs. Proprietary Software", Working Paper, October 2001.

Dalle J.-M. and N. Jullien [2002] 'OS vs. proprietary software', http://opensource.mit.edu/.

Dalle J.-M., P. David and W. Steinmueller [2002] "An agenda for integrated research on the economic organization and efficiency of OS/FS production", Working Paper, October 2002.

Gaudeul A. [2003] "Open Source Software Development Patterns and License Terms", Working Paper, February 2003.

Ghosh R. and V. Prakash [2000] "The Orbiten Free Software Survey" at http://orbiten.org.

Halloran T.J. and W.L. Scherlis [2002] "High Quality and Open Source Software Practices", Position Paper, Second Workshop on Open Source Software Engineering, 24th International Conference on Software Engineering, Orlando, USA, May 19, 2002.

Hann I.-H., J. Roberts, S. Slaughter and R. Fielding [2002] "Delayed Returns to Open Source Participation: An Empirical Analysis of the Apache HTTP Server Project".

Hertel, G., S. Niedner and S. Herrmann [2002] "Motivation of software developers in open source projects: An Internet-based survey of contributors to the Linux kernel". Research Policy, July 2003, vol. 32, iss. 7, pp. 1159–1177(19).

Hippel E. [2002] "Open Source Software as horizontal innovation networks — by and for users", MIT Sloan School of Management WP No. 4366-02.

Johnson J.P. [2000] "Some Economics of Open Source Software", http://opensource.mit.edu/, December 2000.

Kuan J. [2002] "Open Source Software as Lead User's Make or Buy Decision: A Study of Open and Closed Source Quality", 2002 OSS Conference, Toulouse, http://www.idei.asso.fr/.

Kuwabara K. [2000] "Linux: A Bazaar at the Edge of Chaos", First Monday, 5(3), March 2000.

Lakhani K. and E. von Hippel [2000] "How Open Source software works: 'Free' user-to-user assistance", MIT Sloan School of Management Working Paper 4117, May 2000.

Lerner J. and J. Tirole [2000] "The Simple Economics of Open Source", NBER Working Paper 7600.

Lerner J. and J. Tirole [2002] "The Scope of Open Source Licensing", Draft, 2002.

Mockus A., R. Fielding and J. Herbsleb [2000] "A case study of open source software: The Apache Server", International Conference on Software Engineering, pp. 263–272, 2000.

Mockus A., R. Fielding and J. Herbsleb [2002] "Two case studies of open source software development: Apache and Mozilla", Working Paper.

Mustonen M. [2002] "Why do firms support the development of substitute copyleft programs?", Working Paper, October 2002.

Mustonen M. [2002] "Copyleft—the economics of Linux and other open source software", Working Paper.

Nakakoji K. et al. [2001] "Toward Taxonomy of Open Source: A Case Study on Four Different Types of Open Source Software Development Projects".

Peyrache E., J. Crémer and J. Tirole [2001] "Some reflections on Open Source Software".

Pressman R. [1997] "Software engineering", 4th edition, Mc-Graw-Hill editors.

Schmidt D.C. and A. Porter [2001] "Leveraging Open Source Communities to Improve the Quality & Performance of Open Source Software", Position Paper, First Workshop on Open Source Software Engineering, 23rd International Conference on Software Engineering, Toronto, Canada, May 15, 2001.

Scotchmer S. and P. Samuelson [2002] "The Law and Economics of Reverse Engineering", Yale Law Journal, April 2002.

Silverman D. [1999] "Doing Qualitative Research: A Practical Handbook", Sage.

Varian H. [1993] "Economic Incentives in Software Design", Computational Economics, 6(3–4) pp. 201–17, 1993.

OSS conferences in Toulouse [2001, 2002, 2003], http://www.idei.asso.fr/Commun/Conferences/Internet/

GNU General Public License at http://www.gnu.org/copyleft/gpl.html