Do Open Source Developers Respond to Competition? The (L^A)T_EX Case Study

ALEX GAUDEUL *

School of Economics, University of East Anglia

Abstract

This paper traces the history of T_EX, the open source typesetting program. T_EX was an early and very successful open source project that imposed its standards in a particularly competitive environment and inspired many advances in the typesetting industry. Developed over three decades, T_EX came into competition with a variety of open source and proprietary alternatives. I argue from this case study that open source developers derive direct and indirect network externalities from the use of their software by others and must therefore consider non-developers' needs to make their software more attractive to a broader audience and more competitive with proprietary alternatives.

1 Introduction

This paper offers an analysis of the competition between open source (OS) and proprietary software systems through a study of the long-term development dynamics of the T_EX typesetting software.¹ The T_EX project was launched in 1978 under an open source license, long before other better known OS projects such as Linux. This case study thus draws on a long history.

In this article, I focus on the dynamics of competition between OS and proprietary typesetting software. I analyze the multi-period strategies by which they borrowed from each other, complemented each other and reacted to advances on the competing side. A variety of developers developed applications, proprietary or otherwise, for T_EX, an open source typesetting platform. On one side, users chose whether to adopt T_EX as their

* School of Economics, University of East Anglia, Norwich NR4 7TJ United Kingdom. E-mail: a.gaudeul@uea.ac.uk. I would like to thank William Adams, Jacques André, Nelson Beebe, Barbara Beeton, Karl Berry, Lance Carnes, Thomas Esser, David Fuchs, Bernard Gaulte, Hans Hagen, Yannis Haralambous, Jim Hefferon, David Kastrup, Donald Knuth, Leslie Lamport, Wendy McKay, Barry MacKichan, B. Mahesh, Frank Mittelbach, Oren Patashnik, Simon Pepping, John Plaice, Fabrice Popineau, Sebastian Rahtz, Denis Roegel, Chris Rowley, Joachim Schröd, Karel Skoupý, Hàn Thế Thành and all other participants in the (L^A)T_EX project whom I met and interviewed. Special thanks are due to Hans Hagen for his help in editing this paper. I would also like to thank Jacques Crémer, Bruno Jullien, Jean Tirole and Hal Varian for their advice, and an anonymous referee for helpful comments. The support of the CNRS and of the ESRC Centre for Competition Policy is gratefully acknowledged. All errors and omissions are mine.

¹ T_EX is the core of the typesetting program while most users of T_EX use the L^AT_EX set of T_EX macros. Both the terms T_EX and (L^A)T_EX are used to designate the whole of the project.

typesetting platform, or competing proprietary platforms, and if they chose T_EX, also chose what specific applications to use. On the other side, developers chose whether to develop for the T_EX platform or for another platform, and if they chose T_EX, chose under what license they would release their contributions. This depended on the nature of the application they wished to develop. It also depended on the presence of and ease of access to other development platforms in the proprietary world. Decisions of users and developers were complementary: the diffusion of T_EX was closely dependent on the dynamics of its development and on the technological changes that happened outside of the T_EX world. OS developers and users recognized this; they formed coalitions and took strategic decisions to influence the direction of the development of T_EX.

This paper illustrates a simple basic hypothesis: while OS developers improve open source software (OSS) for their own purposes, in order to better fulfil their needs, they also derive benefits from the use of their software by others (network externalities). This encourages them to broaden the software's user base by providing features that may be of no direct use to themselves or that may be low on their own priority lists. For example, they work to facilitate the initiation into the use of the software, or to make the software more accessible and easy to use. They also want to maximize the benefit of network externalities by agreeing on and following a variety of standards in the development, licensing and maintenance of their contributions to the software. They want the software to behave the same way on any user's machine, they want new contributions to fit guidelines so they can be integrated seamlessly into the software distribution and they want this "official" software distribution to be used by all. Those objectives are only achieved by agreeing with others on how the software should behave and by coordinating the ensuing development efforts with other developers. Developers have to balance their need for independence in their development decisions with the need to coordinate with other developers and they have to be receptive to the needs of users. OS developers also have to take account of the proprietary offering and negotiate the positioning of their software in the market in order to maintain or increase their user base. Not only do they have to think in strategic terms but they also have to impose a level of discipline and a sense of direction onto the development of the software, which is more difficult. This is the purpose of OSS organizations. Those organizations provide a platform where users can express their needs and where developers can express their ideas and goals. Users and developers make sense together of the software's position in its development area and negotiate development objectives. This process give rise to complex organizational dynamics whereby developers position themselves, form coalitions, try to convince others by appealing to shared interests or the common good, etc.

The basic hypothesis according to which developers and users consider network effects in their development and consumption decisions is used to explain a number of patterns in the development of the T_EX development platform. The next section is an analysis of the existing literature on the topic.

2 Literature review

This paper is a study of the competition in the provision of a potentially excludable but non-rival good, software. When a good is non-excludable, then it is generally assumed it has to be publicly provided. If it is made excludable, by putting it under copyright for

example, then it is assumed it is privately supplied in the same way as most standard goods. However, the situation in the software industry is different: some software is provided under open source licenses,² while some other software is provided under proprietary licenses. This is a mixed industry where for-profits and nonprofits coexist (Kuan, 2001). Open source software is privately provided through voluntary contributions (Bergstrom, 1986). OSS organizations differ from other nonprofits in that OSS production is not usually publicly subsidized, it does not rely in any significant way on private donations for its development, and its provision is not directed by the State or any other formal institutions or interest group. Open source projects are the result of the work of individually motivated developers and it is difficult for any institution to direct their development.

This paper extends the literature on public goods in the specific context of open source provision by examining the dynamics of the private provision of a public good in a competitive setting. How does an OS organization react to competition, both from other OS projects and from proprietary projects? How does an open source organization adapt to changes in its market environment? How do commercial organizations adapt to the presence of OS competition? Will each type of organization specialize in specific areas of software development? Will they serve different types of customers or develop different software parts? Are there exchanges and synergies between each type of organization?

I will build on the existing literature on OS methods of production, which deals principally with (1) the organization of development and (2) the incentives of OS developers. I will then be able to present (3) the literature on competition between OS and proprietary organizations.

2.1 Organization:

Far from being the “bazaar” evoked in Raymond (2001), open source development is hierarchic and subject to peer review. Small groups of developers contribute most of the “core” code. Other developers provide support for specific systems or functionalities while some other volunteers provide user support and help maintain the infrastructure of the project (Mockus, 2005; and Lakhani, 2002).³

Development is done module by module in an incremental way, with each module a relatively independent part of the complete software. Work is thus fragmented to the individual or small team level, with limited need for central coordination (Krogh, 2003a). Central coordination is done by the lead developers, who also contribute the most to the software’s development (see, for example, Kogut, 2001, Table 3) and are able to influence its course (see, for example, Bezroukov, 1999 on the role of the leader and conflicts between developers).

2.2 Incentives

A developer who develops software that is under OS licensing terms has two sets of motives (Lakhani, 2005). One set of motives is own-use or own-enjoyment; she wants to develop software that she needs or enjoys developing (Hippel, 2005). The other set of

² OS developers keep the copyright over their contributions to an OS project, but are limited in its use; it can be asserted only to prevent others from using their contribution without acknowledgment (Berkeley licenses (BSD)) or from using it as part of proprietary software (GNU Public License (GPL)).

³ Fitzgerald, 2001; Fitzgerald, 2002; and Krogh, 2003b introduce special issues on this topic.

motives is to get others to use her software, or software based on the code she developed. It may be that the software benefits from network effects (the more people use it, the higher is its utility, as is the case, for example, with communication software); or that she derives prestige and reputation from its success (Lerner, 2002); or that she expects users to convert into developers who will then improve the software, contribute their expertise and knowledge and provide peer review (Krogh, 2003a); or that she is an altruist or her effort is sponsored by a governmental entity or a foundation with altruistic goals.⁴ Finally, it may be that developers who work in teams come to identify with the group, its welfare and its objectives (Hertel, 2003). Early users then become advocates for their software and help its diffusion (Dalle, 2003).

Organizational factors and individual incentives combine in such a way as to allow some predictions on the likely outcomes of OSS development processes. User-developers who arrange to produce a good for themselves (Kuan, 2002) are precursors whose needs anticipate others' (Hippel, 1994; and Hippel, 1998). They focus on developing novel functionality rather than improving existing products. They develop software that is more flexible and offers better control of its internal working than what proprietary software can offer. OS developers develop software in those areas in which the proprietary offering is unsatisfactory and they group along common needs (Franke, 2003). This commonality of aims facilitates coordination among developers.

2.3 Competition

OSS has made significant inroads in many areas of software development, from servers (Apache) and mail management systems (Sendmail) to operating systems (Linux), browsers (Mozilla) and typesetting engines (T_EX). However, some argue quite reasonably that OSS is not used by the average user because it is difficult to install, use, maintain and update, because it is of lower quality or offers less functionality than proprietary software or finally because OS development is too unruly and undirected to provide the stability and support users' need (Nichols, 2003).

The problem that is most often evoked in the economic literature, however, is that of free-riding which leads to a tragedy of the commons where no one develops and all use low-quality free software. The problem worsens as OSS becomes more popular; it can lead experienced developers to quit (Foray, 2001; see also Foray, 2007) or to develop proprietary versions of OSS. That last option is indeed available if software is under the liberal Berkeley Software Distribution (BSD) license rather than the more restrictive General Public License (GPL) (Gaudeul, 2005).

In the following, I consider three different hypotheses: (1) OSS would mean the end of proprietary software because of it is available for free; (2) proprietary software would subsist as a complement to OSS because OSS is not fit for all uses or for all users; and (3) proprietary developers would maintain their position vis-à-vis OSS because they keep a strategic advantage over OS developers. This paper complements the existing literature by showing how OS developers also develop strategies to compete with proprietary software.

- (1) Consider in the first instance the hypothesis according to which, everything else being equal, OSS means the end of proprietary software. The logic is simple: consumers

⁴ An altruist is happy to develop software that is freely modifiable, with specifications that are open and that is free. Those developers who are employed by a firm to work on OSS have more pragmatic motives (Bonaccorsi, 2004; see also Hann, 2004).

prefer using a free product to a paying one. OSS thus reduces profits for proprietary firms and deters investment in software production and thus innovation, leading to a standstill in software development (Schmidt, 2003). Bitzer (2006) argues otherwise; case studies of the markets for operating systems, servers and web browsers show that increasing competition by OSS tends to accelerate the pace of innovation. Krogh (2003b) argues that OSS breeds a new and more efficient “private-collective” innovation model. Rather than disappear, proprietary firms adopt a range of for-profit strategies in the exploitation of OSS – examples are offered by Koenig (2004) for such companies as Oracle, IBM, HP or Red Hat. Proprietary firms develop and sell OS-related products and support services, or they launch OS products and encourage the development of a community of OS developers that improve them (Bonaccorsi, 2004; see also Wichmann, 2002). Proprietary firms encourage and support OSS when doing so promotes their preferred standard,⁵ or when it allows them to gain some control of the OS standard and influence its development (a strategy followed by IBM with Linux and by Sun Microsystems with Java) (Mustonen, 2005; see also Mustonen, 2003).

- (2) A second strand of literature thus argues how OS and proprietary software can cohabit. This is, for example, because OSS is developed by developers for the benefit of developers, whose purpose and need are different from those of the average user. Notably, OSS is developed specifically for those who are not satisfied with the proprietary offering so that by design, OSS is not fit for use by the majority of users who use proprietary software. Bessen (2004) argues that pre-packaged proprietary software addresses common uses with limited feature sets while OSS targets users with more specialized and complex needs. Nichols (2003) points out how developers differ from most users in their preferences in terms of user interface. For example, developers prefer command-line-based interfaces with many shortcuts that allow direct access to the basic functions of the software. Users prefer the more intuitive WYSIWYG interfaces that automate frequently used tasks but are less flexible.
- (3) This leads a third strand of literature to point out that proprietary developers keep a strategic advantage over OSS. This is through their ability to coordinate strategy and development in a market oriented way. Proprietary developers benefit from their ability to direct the work of others in a centralized way according to a well-defined and enforceable strategy (Herbsleb, 2003 or Yamauchi, 2000). Proprietary developers may exploit network effects strategically, subsidizing the first users to build a user base and then exploiting the late comers (Casadesus-Masanell, 2003). The proprietary developers attract developers to their development platform by cross-subsidizing developers with revenues from users or vice versa (Economides, 2006). This type of cross-subsidy, as well as other strategies that require financial means, such as advertising, is not available to OS developers as OSS generates only limited income streams. Proprietary developers are also able to differentiate in specific ways from OSS (Bitzer, 2004). They may identify new needs and devise targeted development strategies to address those needs. Proprietary developers may address the needs of a niche portion of the market if they are latecomers on a market where OSS is already

⁵ A strategy followed by Adobe, for example, which encouraged the development of pdfT_EX, a version of T_EX that produces pdf output.

present. Proprietary developers may use OS code or adopt OS standards to get a head start into specific development areas (Gaudeul, 2006). They may also hire OS developers away from OS development by offering them high wages, a strategy that may be at work if one considers such OS developers as Linux's Linus Torvalds or Sendmail's Eric Allman who did go on to work for proprietary outfits (Mustonen, 2003).⁶ Some companies such as (allegedly) Microsoft, may use a range of other strategies to counter the emergence of OSS. They may use scare tactics underlining the fact OSS is "unsupported".⁷ They may also lower their prices for those consumers who are most likely to be attracted by OSS. Administrations thus get preferential deals and less developed countries get Windows XP Starter Edition, a lower cost lesser quality version of Windows XP.

In this paper, I analyse the strategies, successful or otherwise, by which OS developers try to establish themselves in the market. As in Dalle (2003), much depends on the initial momentum gained from proselytizing by early users and on the ability of open source developers to organize efficient development processes (possibly with the help of ancillary business firms). There is a constant tension between coordinated development on a standard that brings high direct (for consumers) and indirect (for developers) network externalities in the long-term, and fragmented development that faster addresses a wider range of needs, though at the expense of coherence in development.

This paper is introduced by a presentation of the T_EX software and of the actors and institutions that influenced its development. The development of the software is analyzed over time, which leads to a discussion of how T_EX positioned itself in its environment. The main body of the paper deals with the strategic motives behind the evolution of the development of OS and proprietary software in the typesetting software industry since the emergence of T_EX. Information on the project's development was gathered from interviews with participants in the development of the T_EX typesetting system, with administrators of the T_EX User Groups (TUG) and with maintainers of the project's organizational infrastructure. Other sources include the TUG's journal (TUGBoat), the TUG's website (<http://www.tug.org>), the websites of the many national T_EX users' group, the websites of projects associated to T_EX, and also the T_EX related newsgroups. Several others sources were used to gain a better understanding of the typesetting software industry and of its evolution over time.

3 T_EX, the software and its history

T_EX is a computerized typesetting system. A T_EX file contains both the text to be published and instructions on how to format it for an output (dvi, ps, pdf, html, or xml files). That output may be distributed in digital or paper form. T_EX is used by authors, typesetters and publishers for a wide variety of tasks. T_EX became the standard submission

⁶ Lerner (2002) argues indeed that OS developers wish to signal their ability to firms by contributing to OSS development.

⁷ FUD tactics (fear, uncertainty, and doubt), compare with for example, the Halloween documents that underline responses to open source emergence that were considered by Microsoft (<http://www.catb.org/126esr/halloween/index.html>).

format for most mathematical journals and is commonly used for scientific papers; $\text{\TeX}$ is widely used in the publishing industry for the communication of printed work between authors, editors, and printers; $\text{\TeX}$ is frequently used as a back-end application in publishing, taking as input documents typeset in various mark-up languages (HTML, XML, $\text{\TeX}$) and outputting documents in various rendering formats for printers or for the web. Typesetters whose needs are not fulfilled by a proprietary offering, those who cannot afford proprietary tools and those with very specialized needs appreciate the flexibility, versatility and free availability of $\text{\TeX}$ systems. Finally, the $\text{\TeXinfo}$ format, based on $\text{\TeX}$, is the official documentation format of the Free Software Foundation (FSF).

$\text{\TeX}$ is a medium size software project; not as big as an operating system like GNU/Linux, but still a whole typesetting system with many interdependent modules. From statistics on the popularity of the project (Table 1, Appendix 7.1), gathered from several sources, $\text{\TeX}$ could be compared in size to Sendmail, the OS mail management software, or to QuarkXPress, a dominant proprietary typesetting software. It is, however, certainly less popular than mainstream word processing software, be they open source or proprietary. The initial version of $\text{\TeX}$ (1979) gained about 100 to 1,000 users, the second version (1982) had about 10,000 users while $\text{\TeX}$ was used by more than one million by the time its third version was released in 1990.⁸ $\text{\TeX}$ is widely available as it was ported to the Linux, Mac and Windows platforms, among others. The extent of the diffusion of $\text{\TeX}$ is reflected in the steady increase in the number of requests for support in $\text{\TeX}$ -related newsgroups, up to 4,500 posts per month in 2000 across newsgroups devoted to $\text{\TeX}$ in English, German and French.⁹ The creation of Local User Groups (LUG) across the world is another testimony to the popularity of $\text{\TeX}$ and the wide involvement of users in its promotion and development. There were about 6,500 active members in a dozen national $\text{\TeX}$ user groups across the world in the early 2000s.¹⁰

The $\text{\TeX}$ project was started by Donald E. Knuth in the late 70s at Stanford University.¹¹ In a first phase (1978-1985), development was heavily centred around Donald Knuth at Stanford University. This was when the core of the program was developed.¹² In a second phase (mid-80s), core development was complemented by the

⁸ Estimates provided by $\text{\TeX}$ administrators.

⁹ From its creation in early 1990, posts to comp.text.tex peaked to 2,000 per month in 1994. As postings to the English-speaking newsgroup reached a plateau, posts to newsgroups in other languages grew. The German de.comp.text.tex started in 1996 and equalled the English group by 2000, while the French fr.comp.text.tex which started earlier (1994) received 500 messages per month by 2000 (figure 2, p.139 in Gaudeul, 2003). $\text{\TeX}$ newsgroups are primarily user and not developer oriented as developers communicate through mailing lists. By way of comparison, a popular Linux newsgroup such as comp.os.linux.misc was created in 1993 and received about 5,000 posts per month in 2000.

¹⁰ The first $\text{\TeX}$ user group was established in the US in 1980 and now counts 1,800 members. It was followed by several LUG, first in Western Europe and then in Eastern Europe and Asia: Netherlands (1988, 300 members), Germany (1989, 2,000 members), France (1989, 500 members), the UK (1992, 150 members), Poland (1993, 300 members), the Czech Republic (1995, 400 members), India (1997, 100 members) and China (1999, 700 members).

¹¹ Table 2, Appendix 7.2 lists the most important $\text{\TeX}$ sub-projects, ranked by their date of first release.

¹² The core of $\text{\TeX}$ went through three versions, $\text{\TeX}$ 1.0 in 1978, $\text{\TeX}$ 2.0 in 1982 and $\text{\TeX}$ 3.0 in 1989, each a progressively smaller marginal improvement on the other. By 1990 then, the development of $\text{\TeX}$ was essentially stopped and by the mid-90s, limitations in $\text{\TeX}$ became more apparent. Various projects to extend the capabilities of $\text{\TeX}$ emerged: Ω (1994) for multilingual typesetting, pdf $\text{\TeX}$ (1996) for pdf rendering,

development of $\text{T}_{\text{E}}\text{X}$ sub-routines, that is, sequences of $\text{T}_{\text{E}}\text{X}$ commands that eased access to $\text{T}_{\text{E}}\text{X}$'s functionalities.¹³ A number of packages were developed in a rather decentralized way, leading to many replications in the development of similar features. Several different distributions of $\text{T}_{\text{E}}\text{X}$ competed on the market, each maintained by different persons and distributed under different licenses: freeware, OS, proprietary. The $\text{\LaTeX}$ package rapidly became the most widely adopted. In the third phase (late 80s, early 90s), priority was given to the development of tools and systems that made the $\text{T}_{\text{E}}\text{X}$ system easier to use and develop.¹⁴ Development became better coordinated thanks to the creation of coding banks (software packages repositories) and to the creation of a standard way of classifying those packages into the repository (directory structure). The $\text{T}_{\text{E}}\text{X}$ User Groups worked to improve usability by making $\text{T}_{\text{E}}\text{X}$ systems available as a complete set of inter-related $\text{T}_{\text{E}}\text{X}$ packages – the $\text{T}_{\text{E}}\text{X}$ Live distribution, structured along a standard (the $\text{T}_{\text{E}}\text{X}$ Directory Structure) and available on a common repository (the Comprehensive $\text{T}_{\text{E}}\text{X}$ Archive Network). There were efforts also in developing programs for easy access to the various $\text{T}_{\text{E}}\text{X}$ programs and commands (interfaces). This was, thus, a stage of rationalization and of integration. The fourth and present stage (90s to present) was characterized by a return to decentralization. The need for a central dedicated coordination system became less urgent as Web-based tools made it possible to efficiently coordinate OS projects in a decentralized way. The software came to be used in a wider variety of countries and by a wider variety of people. The software also came to be called upon for the fulfilment of needs that the original developers had not anticipated and that the standard $\text{T}_{\text{E}}\text{X}$ software could not respond to adequately. This was when there were several attempts at remodelling the core of $\text{T}_{\text{E}}\text{X}$ along new lines, with various levels of success.¹⁵ A stable $\text{T}_{\text{E}}\text{X}$ distribution of reference ($\text{T}_{\text{E}}\text{X}$ Live), used by most but developed and maintained only by a few, now

$\epsilon\text{-T}_{\text{E}}\text{X}$ (1997) and the “New Typesetting System” (NTS) (1998) that made $\text{T}_{\text{E}}\text{X}$ more flexible. $\epsilon\text{-T}_{\text{E}}\text{X}$ is now the standard $\text{T}_{\text{E}}\text{X}$ extension while $\text{pdfT}_{\text{E}}\text{X}$ is currently the most popular $\text{T}_{\text{E}}\text{X}$ package.

¹³Early sets of macros such as plain- $\text{T}_{\text{E}}\text{X}$ (1978) and AMS- $\text{T}_{\text{E}}\text{X}$ (1981) were soon replaced by $\text{\LaTeX}$ (1983-1985). $\text{\LaTeX}$'s development was taken over by the $\text{\LaTeX}3$ team (1989), a group development effort that led to the release of $\text{\LaTeX}2_{\epsilon}$ (1994), while a set of macros for educational publishing was developed (Con $\text{T}_{\text{E}}\text{Xt}$, 1994). The suite of programs became more complex and the number of programs linked to it rose. The programs came to fit increasingly specialized needs: programs to generate bibliographies (Bib $\text{T}_{\text{E}}\text{X}$, 1984) and indexes (makeindex, 1987) or to draw figures (PSTricks, 1992) were developed. In Europe, local user groups initiated adaptations of macro packages to non-American typesetting traditions; $\text{T}_{\text{E}}\text{X}$ the program was adapted to deal with 8 bit input needed for non English languages ($\text{T}_{\text{E}}\text{X}$ 3.0, 1989)

¹⁴In that third phase, the TUG (1980), along with LUGs, set up a common repository for $\text{T}_{\text{E}}\text{X}$ packages (CTAN, 1990) and a standard way of organising all the $\text{T}_{\text{E}}\text{X}$ -related files on a computer system (TDS, 1994). Several other initiatives gave coherence to $\text{T}_{\text{E}}\text{X}$ systems, such as the NFSS (1989) that allowed easier access to the wide variety of fonts that had been developed over time. Complete distributions were made available. $\text{T}_{\text{E}}\text{X}$ Live (1996), the OS $\text{\LaTeX}$ distribution based on $\text{teT}_{\text{E}}\text{X}$ for UNIX distribution (1994), was translated into $\text{fpT}_{\text{E}}\text{X}$ for Windows Distributions (1998), a distribution that competed with the independently developed Mik $\text{T}_{\text{E}}\text{X}$ Windows distribution (1996). Interfaces were also developed, the most popular being now the freeware WinShell (1998) which competed with the earlier shareware WinEdt (1993). Finally, the LPPL, a formalization of existing $\text{T}_{\text{E}}\text{X}$ license agreements, was imposed on most packages in a $\text{\LaTeX}$ distribution (1999). That license was adapted to the Debian Linux OS guidelines (2002) so as to facilitate the inclusion of the $\text{T}_{\text{E}}\text{X}$ system into Linux distributions. Routines to translate $\text{T}_{\text{E}}\text{X}$ in various alternative typesetting mark-up languages (html, xml, sgml ...) and convert its output into various document formats (bitmap, postscript, pdf ...) were developed. User interfaces became more sophisticated.

¹⁵ Ω (1994), $\text{pdfT}_{\text{E}}\text{X}$ (1996), $\epsilon\text{-T}_{\text{E}}\text{X}$ (1997) or NTS (1998), see note 12.

coexists with a variety of specialized applications based on $\text{\TeX}$, where most of the development effort is concentrated.

The licensing of $\text{\TeX}$ systems is complex. The core of $\text{\TeX}$, `tex.web`, is copyrighted by D.E. Knuth, and no changes are permitted, although of course concepts used in the program may be re-used. “ $\text{\TeX}$: the program” is in the public domain, although the use of the name “ $\text{\TeX}$ ” is restricted to exact copies of Knuth’s software systems. Most programs that have added up to “ $\text{\TeX}$: the program” – are licensed under the $\text{\LaTeX}$ Project Public License (‘LPPL’). The LPPL can best be described as a BSD license. This means developers can develop closed-source proprietary versions of the software. However, the LPPL is slightly more restrictive than the BSD: any change to a file must be renamed and distributed with the original version. This is meant to guarantee the integrity of the $\text{\TeX}$ system: a user should always have the choice between the standard version of a file and its modified version. This means any user is guaranteed access to the exact same program as any other user and can thus output the exact same result from a `.tex` file as that other user. Finally, many of the binaries in $\text{\TeX}$ distributions are licensed under the GPL.

The next part goes further into an analysis of the dynamics of $\text{\TeX}$ ’s development but the argument can be summarized as follows: the growth of $\text{\TeX}$ ’s user base came with a progressive diminution in the dynamism of the software’s development; the weight of legacy users undermined attempts to adapt $\text{\TeX}$ to new uses, technologies and standards. There was then a certain amount of diversification in different projects that addressed ancillary needs. From competing projects addressing the same needs emerged a few clear winners. Those were gathered and combined by developers and users into a distribution,¹⁶ under the leadership of the $\text{\TeX}$ User Groups. Proprietary developers identified specific gaps in the OS offering and developed proprietary improvements on $\text{\TeX}$, provided WYSIWYG interfaces and add-ons or simply provided support in the use of the software. While some of those proprietary products did draw some users away from OS $\text{\TeX}$ and its community, they also often provided the non-programmer a first step in the use of $\text{\TeX}$ and thus broadened its audience. A variety of new independently developed proprietary software and standards then threatened the survival of $\text{\TeX}$. $\text{\TeX}$ developers had to adapt to and adopt those new standards to keep $\text{\TeX}$ up-to-date with changing technologies and standards in the typesetting industry. However, not all developers agreed on how to do so, which weakened the coherence of the $\text{\TeX}$ project. Different engines, packages and distributions appeared and users came to adopt the ones that responded best to the different specific new needs that were brought about by the evolution of the typesetting software industry. This fragmentation of user constituencies limited the incentives for developers to coordinate and maintain compatibility, which threatened further the coherence of the $\text{\TeX}$ project.

¹⁶A software distribution is a collection of software, already compiled and configured, that when installed on a computer constitutes an entire self-contained software system.

4 Competition with proprietary software

4.1 A map of the industry

$\text{\TeX}$ had to face intense competition from an array of competitors, from simple word processing software to professional publishing software. First aimed at professional typesetters, $\text{\TeX}$'s superior quality (better fonts, better hyphenations and justification algorithms, better rules for letter spacing) drove out less advanced proprietary competition from which it borrowed many ideas and features as well (Script (1970), Roff (1971), Scribe (1978)). It also replaced in-house proprietary typesetting systems such as the one that was used by the American Mathematical Society for mathematical typesetting.

$\text{\TeX}$'s code was then integrated into proprietary offerings. Distributions such as $\text{pc}\text{\TeX}$ (1985), $\text{Micro}\text{\TeX}$ (1985), $\text{True}\text{\TeX}$ and $\text{Y\&Y}$ (1991) provide(d) $\text{\TeX}$ systems that were pre-packaged and easy to install. Environments such as $\text{\TeX}\text{tures}$ (1989) for the Mac or $\text{Scientific Workplace}$ (1994) for Windows provide(d) user interfaces. Extensions such as $\text{v}\text{\TeX}$ added functionalities to open source $\text{\TeX}$ systems. Some companies provided additional font packages and printer drivers for $\text{\TeX}$ systems.

With the exception of Scientific Word and $\text{PCT}\text{\TeX}$, $\text{\TeX}$ -based proprietary competition did not maintain itself over the long-term as OS versions caught up with it. Distributions such as $\text{\TeX Live}$ (1996) and $\text{Mik}\text{\TeX}$ (1996) eased the installation and management of $\text{\TeX}$ distributions, and some projects such as $\text{L}\text{\TeX}$ (1999) or $\text{\TeX}\text{macs}$ (2002) developed WYSIWYG interfaces to $\text{\TeX}$.

Independently developed proprietary software was more serious competition: it was easier to learn and came with superior interfaces and took a large part of the market. This occurred in a two pronged attack: some proprietary systems fulfilled the needs of those professional typesetters who needed easy-to-learn standardized software (Framemaker (1986), $\text{Quark's QuarkXPress}$ (1987)). Some others satisfied the needs of those users who merely needed simple and easy-to-use text processors ($\text{Corel's WordPerfect}$ (1982), Microsoft's Word (1984)).¹⁷ Proprietary software that was developed independently from $\text{\TeX}$ thus came to replace $\text{\TeX}$ in its less specialized, general typesetting functionalities.

Typesetting standards and technologies evolved as well. Adobe's pdf technology (1993) was generally adopted as a document rendering format, replacing $\text{\TeX}$'s DVI, and the $\text{\TeX}$ mark-up language was replaced by new OS markup languages (SGML (1986), XML (1998), MathML (1999), XHTML (2000)).

Development at that stage was mainly influenced by the emergence of those open typesetting and document rendering standards. $\text{pdf}\text{\TeX}$ (1996) was developed to output pdf documents from $\text{\TeX}$. $\text{\TeX}$ was one of the first high-end typesetting programs to be adapted to handle new mark-up languages such as XML ($\text{XML}\text{\TeX}$, 2000).

In the end, $\text{\TeX}$ kept those users who could not afford proprietary typesetting software (users in emerging countries, students), those people who needed its unique mathematical capabilities (academics) and those typesetters who valued the flexibility OSS offers to

¹⁷Some of those companies (Arbortext (1982) or 3B2 (1986) for mathematical typesetting) borrowed parts of $\text{\TeX}$. $\text{\TeX}$'s conception principles, line breaking algorithm or syntax were widely copied. $\text{\TeX}$'s equation editor was integrated in MS's Word (1984) and $\text{\TeX}$'s hyphenation and justification algorithm was used in Adobe's InDesign (2000).

devise their own specialized versions of $\text{T}_{\text{E}}\text{X}$. Many typesetters kept using $\text{T}_{\text{E}}\text{X}$ as a back-end to handle documents typeset in a variety of typesetting languages. Development focused on specialized applications such as Ω (1994) for rare script and ancient documents with complex, multi-lingual typesetting or $\text{ConT}_{\text{E}}\text{Xt}$ (1994) for interactive educational material. There was also a drive to develop user friendly interfaces for $\text{T}_{\text{E}}\text{X}$ ($\text{L}_{\text{Y}}\text{X}$ (1999), $\text{T}_{\text{E}}\text{Xmacs}$ (2002), $\text{X}_{\text{E}}\text{M}_{\text{E}}\text{X}$ (2003)), following a trend towards improving OS usability; cf. for example, the emergence of open source word processors such as OpenOffice (2002) or Abiword (2000).¹⁸

Figure 1: The development of $\text{T}_{\text{E}}\text{X}$ over time: Dominance, squeeze and fragmentation.

Figure 1 illustrates the above paragraph. The three phases in the relation of $\text{T}_{\text{E}}\text{X}$ to its competition are outlined (dominance, squeeze and fragmentation) and the main actors in each phase are represented. Represented at the top are high-end applications for typesetting and publishing, and at the bottom are common applications for the end user (word processors). The right hand side illustrates the fragmentation in the development of high-

¹⁸ Some proprietary firms, such as Corel and its Wordperfect (1998), made their word processors open source, in the same type of reaction to Microsoft's dominance as that of Netscape vs. Explorer. Whether that type of software can really be considered open source is debatable, as there is generally little subsequent involvement by OS developers in those projects.

end $\text{\TeX}$ applications, as well as the development of integrated $\text{\TeX}$ systems and interfaces for the end user.

The next sections offer an analysis of the evolution of $\text{\TeX}$ with respect to its proprietary competition, and is divided in four parts, broadly along chronological lines.

4.2 Network externalities and OS development goals

How effective the OS organization was in getting work coordinated was largely dependent on the anticipated benefit of doing so. This was linked in the minds of developers not only to how effective coordination could be in achieving technical aims, but also to whether those aims would make $(\text{\La})\text{\TeX}$ more competitive vs. its proprietary alternatives. They thus considered the interests of users, as well as their own, in the choice of what features to develop and whether to participate in collaborative work instead of each working on their own.

The consideration of user needs was important to $\text{\TeX}$ developers for all the reasons evoked in 2.2, but also more specifically because $\text{\TeX}$ developers believe in the superiority of $\text{\TeX}$ as a standard and in the desirability of all using one standard to facilitate communication of typeset material in the scientific community. Developers wanted $(\text{\La})\text{\TeX}$ to keep a high market share in some domains – mathematical typesetting foremost among those. From the beginning, the development of $\text{\TeX}$ was user oriented and attracted active interest from users.

The user organizations' dynamism was highest when $(\text{\La})\text{\TeX}$ was under the highest threat of losing market share to its alternatives (1990s). The TUGboat, official journal of the $\text{\TeX}$ community, collected users' initiatives and outlined the problems they encountered. Several programmatic papers outlining goals and priorities for the development of $\text{\TeX}$ were published in the TUGboat and presented at TUG conferences.¹⁹ This was when OS developers worked on interfaces, distributions and a rationalization of development so as to keep their user base. Development then had to become ever more efficiently led – with the use of better coordination tools, the building of central code repositories and the writing of procedures to submit new codes.

Lesson: OS developers that are motivated, at least in part, by the popularity of their software see competition from proprietary software as a stir for the development of features they would otherwise find useless. Competition encourages developers to participate in collective development, to make their software accessible, to promote it actively and to respond to users' needs and queries.

4.3 Open source inertia

There was some inertia in this response to competition, however. $\text{\TeX}$'s development was dominated by the original set of users of the software, which meant new or potential new users found it difficult to influence the dynamics of the software's development. $\text{\TeX}$'s

¹⁹ For example, Mittelbach (1997) set out the program for the reworking of $\text{\LaTeX}$ by the $\text{\LaTeX}3$ project team. The Netherland Local User Group (NTG, 1998) called for more cooperative development efforts in the reworking of $\text{\TeX}$. Flynn (2001) set priorities for the broadening of $\text{\TeX}$'s user base. Beebe (2004) advocated the adoption of the new OS mark-up languages rather than $\text{\TeX}$'s native language. The debate over whether $\text{\TeX}$ should develop Windows-like, WYSIWYG interfaces was particularly heated (Cottrell, 1999).

license forbade different versions of $\text{T}_{\text{E}}\text{X}$ from calling themselves $\text{T}_{\text{E}}\text{X}$. This meant there was only one version of $\text{T}_{\text{E}}\text{X}$ which was controlled by its original developer, D.E. Knuth. The same pattern was repeated in the case of $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, the dominant set of $\text{T}_{\text{E}}\text{X}$ macros. First controlled by its original developer, Leslie Lamport, its control was then transferred to a group of developers, the $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}3$ team. This tight control made it difficult for outsiders to influence the development of new versions of $\text{T}_{\text{E}}\text{X}$ or $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$.²⁰

The user groups had an ambiguous influence on development, both encouraging better coordination between developers and discouraging development into new areas which would have required adaptation by “legacy” users. One of the TUG’s main early sponsors was the American Mathematical Society, which saw in $\text{T}_{\text{E}}\text{X}$ an archival format. It thus wanted to promote backward compatibility, that is, all new versions of $\text{T}_{\text{E}}\text{X}$ had to be able to process all $\text{T}_{\text{E}}\text{X}$ documents that were typeset in the past. The TUGs encouraged projects that were of interest to existing users, such as extensions of the software’s core, compatibility with alternative standards (PDF, HTML, XML), rationalization of the $\text{T}_{\text{E}}\text{X}$ distributions or standardization of the most popular packages. It did not get involved in specialized projects that were of interest only to a few of the existing users or that could lead to problems of compatibility between versions of $\text{T}_{\text{E}}\text{X}$. The TUG wanted to protect their constituency; a change attracting some users had to be balanced against the risk of alienating some established users.

This meant there was a fall in the membership and contributions to the original US TUG in the 90s (see figure 2 in appendix 7.4); existing users, satisfied with $\text{T}_{\text{E}}\text{X}$, stopped contributing to its development, and few new members joined. On the other hand, European user groups such as the NLG (Netherland) and Dante (Germany) that were more responsive to the needs of new users became very dynamic and worked to encourage the adoption of $\text{T}_{\text{E}}\text{X}$ in their respective countries.²¹

The inertia from having to keep on serving established users translated into tensions between $\text{T}_{\text{E}}\text{X}$ developers and a tendency for new developers to start their own projects, leading to a progressive fragmentation in the development effort. Fragmentation (forking) was due to initiatives by users whose needs were not accommodated by the $(\text{L}^{\text{A}})\text{T}_{\text{E}}\text{X}$ development teams. While this forking weakened the position of $\text{T}_{\text{E}}\text{X}$ as a standard and

²⁰ On the other hand, it did ensure coherence in the development of $(\text{L}^{\text{A}})\text{T}_{\text{E}}\text{X}$, which made $\text{T}_{\text{E}}\text{X}$ more stable and less difficult to maintain on one’s computer. One can notice that the release of new versions of $\text{T}_{\text{E}}\text{X}$ packages is a lot less frequent and more controlled than what is usually portrayed for a “typical” OS project; there were only three successive versions of $\text{T}_{\text{E}}\text{X}$; the $\text{T}_{\text{E}}\text{X}$ Live distribution is distributed only annually to a wider public; more generally, development in teams tends to be rather secretive with potential new releases being the subject of much anticipation, as in the proprietary world. Much of this can be explained by the wish expressed by both Donald Knuth (Knuth, 1991) and the $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}3$ team ($\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}3$, 1997) that $\text{T}_{\text{E}}\text{X}$ and $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ be typesetting standards. This meant that changes have to go through a long approval process and are released only once one has made sure they work well within existing $\text{T}_{\text{E}}\text{X}$ distributions.

²¹ The European Local User Groups were instrumental in making $\text{T}_{\text{E}}\text{X}$ more accessible to the users. A group of volunteers at Aston University in the UK established a central repository for $\text{T}_{\text{E}}\text{X}$ code from which it was possible to download the latest developments in the $\text{T}_{\text{E}}\text{X}$ system via the Net. This group inspired the development of a package classification system, the $\text{T}_{\text{E}}\text{X}$ Directory Structure, which served as the model for $\text{T}_{\text{E}}\text{X}$ distribution everywhere. The Aston initiative led to the creation of the CTAN archives in 1990. Another seminal initiative came from the Netherlands Local User Group which developed 4All $\text{T}_{\text{E}}\text{X}$ in 1993. 4All $\text{T}_{\text{E}}\text{X}$ was a $\text{T}_{\text{E}}\text{X}$ distribution on a CD that was intended for an end user with no programming background. It was a precursor and an inspiration for the development of $\text{T}_{\text{E}}\text{X}$ Live.

diverted developers from the main strand of development, it also adapted T_EX to the needs of a wider variety of users, thus contributing to the popularity of T_EX. On the proprietary side, there was a switch from entrepreneurs borrowing from T_EX to them developing software independently.

Lesson: Institutionalized tight control by relatively closed development teams and the conservative influence of established users combine to make changes in OSS difficult. This is not all for the worse, as it promotes stability and continuity in development. However, this slows down the response to technological shocks (new standards, new concepts). Faced with the obsolescence of their software, developers find it difficult to agree on needed changes because any change threatens at least some users or can weaken the influence of at least one development group. This leads to dissensions so that development coherence is threatened. Inertia in the development of OSS leads to the fragmentation of its user base and of its development groups. Developers start independent projects and each project attracts a part of the existing user base.

4.4 Proprietary responsiveness

As explained above, the TUG encouraged the diffusion of new T_EX developments to a broad audience and prevented unnecessary replication in the development of new features. However, it cared mostly about existing users and improving the existing software, not necessarily finding new domains of application or increasing T_EX's market share. OS developers frequently anticipated the need for some features but quite often, however, their needs were too specialized and were of no interest to the common user, while on the other hand, what would have been interesting for the common user was of no interest to them. For example, they were not interested in a graphical user interface as they were already comfortable with the T_EX language and there was considerable reluctance in considering the needs of Windows users.²² T_EX developers also lacked the means or the will to conduct radical restructuring of T_EX; a project such as the New Typesetting System (NTS), whereby T_EX users' groups financed the development of a successor to T_EX, was not successful. Successful projects, such as pdfT_EX, developed only through small extensions of T_EX and focused on practical typesetting problems.

All this meant that proprietary developers were often first to spot new market opportunities and act on them. Proprietary development, in contrast to OS development, was directed toward providing new functionalities so as to attract new segments of the user population, at the expense sometimes of improving existing functionalities that were of use to their existing partially locked-in users. Commercial developers focused, for example, on making a coherent whole from OS development efforts or filling in its missing features. Commercial software found a relative advantage in facilitating the use of T_EX by adding a user interface and porting T_EX to non-Unix platforms. They did relatively little work in improving the core functionalities of T_EX. Proprietary software based on T_EX was frequently able to offer new and original features that were in demand but outside of the realm of interest of existing T_EX users. T_EXtures catered to the Mac users, PCT_EX catered

²² T_EX Live was always considered to be a superset of the Unix teT_EX distribution. Changes in that Unix distribution always took precedence over changes in the fpT_EX distribution for Windows, which led to this later distribution being abandoned.

(and still caters) to the PC users, Y&Y provided sets of proprietary font outlines, Scientific Workplace integrates a computation software (Maple and then MuPAD).

One can wonder at this point why proprietary developers did not get involved in the development of $\text{\TeX}$ itself but rather concentrated on its marketing. In this, $\text{\TeX}$ indeed differs from other OS projects where proprietary involvement took other forms (support, development, patches and add-ins). This difference was probably due to the nature of $\text{\TeX}$'s license (the LPPL, which is similar to the BSD license and allows for proprietary exploitation of the code), to the fact $\text{\TeX}$ was relatively easy to market as-is, as it was designed for use by a wide public of end-users with no required programming background, and to the fact that most changes in $\text{\TeX}$ could be handled without touching its core programming. Another factor was the stability of $\text{\TeX}$ over time and its status as a standard. This encouraged its take-up by commercial companies that then made $\text{\TeX}$ available to a wider public.

Lesson: Proprietary developers take advantage of a certain amount of inertia in OS development to offer their own solutions to the problems faced by the OS community. Unimpeded by a responsibility to established users and motivated by profits, proprietary developers offer their own private “OS solutions”, targeted at those users who are outside of the area of interest of OS developers. In doing so, they patch OS deficiencies and help make OSS more accessible.

4.5 Open source catch-up

While proprietary developers were more responsive to users' needs, this was only a transitory advantage as competition triggered a reaction by $\text{\TeX}$ developers and encouraged them to address user needs. $\text{\TeX}$ was made available in a complete easy-to-install distribution, integrated new font sets and also came to offer pdf format output. $\text{\TeX}$ Live, the OS distribution of $\text{\TeX}$, and user interfaces such as the shareware WinEdt or the freeware WinShell spelled the end of many proprietary versions of $\text{\TeX}$. Only these proprietary software that included features that the OS community was not able to develop, such as the WYSIWYG equation editor of Scientific Workplace and its integration of the Maple computation program, were able to survive.²³

The fate of those entrepreneurs who chose to borrow from $\text{\TeX}$ and that of others who chose to innovate independently of OS development differed markedly. Entrepreneurs who borrowed from $\text{\TeX}$ saved on development cost and, as $\text{\TeX}$ was already established, could offer users of their proprietary product the benefit of being able to use an established standard. Proprietary products that were based on OSS often served as a Trojan horse for

²³ While OS interfaces improved over time, they never became as aesthetically pleasing, complete or ergonomically efficient as that of competing proprietary WYSIWYG text editors such as Microsoft's Word, of competing proprietary typesetting software such as Quark, or of some software based on $\text{\TeX}$ such as Scientific Workplace.

The main interfaces were the freeware WinEdt (1993) and the OS WinShell (1998) but they required knowledge of the $\text{\TeX}$ syntax and commands. Some developers tried to go further and develop WYSIWYG interfaces based on their own implementations of $\text{\TeX}$ (LyX (1999), $\text{\TeX}$ Macs (2002)). This disconnected them from the main branch in the development of $\text{\TeX}$. There were, therefore, individual OS initiatives trying to adopt the middle ground ($\text{\TeX}$ shop (2000) for Mac Operating Systems, preview- $\text{\LaTeX}$ (2001) for Unix OS or $\text{\XeLaTeX}$ (2003) for Windows OS). Those interfaces could be used on top of the standard $\text{\TeX}$ distributions.

open source products. The enhanced diffusion of T_EX, thanks to its better marketed and more accessible proprietary versions, helped to promote its standard. Users of the proprietary versions of T_EX, which T_EX would not have reached if it had been distributed only in its OS version, switched to OS T_EX when the premium offered by proprietary versions weakened. Most firms that based their strategies on offering augmented versions of T_EX were driven off the market when the OS T_EX distribution caught up with them in terms of ease of use and functionalities. Commercial developers who did not differentiate from T_EX, those who developed niche products in areas which OS development neglected or those who supplied easier access to the newest developments in T_EX only gained a fleeting competitive advantage. Those entrepreneur who developed software independently from T_EX, on the other hand, faced more expenses in development and had more difficulties establishing themselves but they were not threatened by every advance in T_EX's open source development, eschewed T_EX's technological limitations and did not depend on the continued development of T_EX for their own survival.

Lesson: OS developers catch up with proprietary solutions that use OSS. Proprietary leadership encourages them to develop missing applications, develop compatibility with newly established standards and make OSS more accessible to the end user. Proprietary developers face a choice between using OS code and building a fragile lead over OSS developers or incurring high development costs by developing software independently and establishing their own user base.

5 Conclusion

The open source T_EX typesetting program was developed over three decades and came into competition with a variety of open source and proprietary alternatives. T_EX was an early and very successful open source project that imposed its standards in a particularly competitive environment and inspired many advances in the typesetting industry. T_EX's rather unique success in a domain applications for the end user that is usually reserved for proprietary software gives it a status as a role model for the OS projects that want to attain success beyond the rarefied world of current users of OSS.

A strong user base increased the value of the software through direct and indirect network effects. Direct, demand-side network effects were particularly strong for T_EX, a standard format for document exchange, while indirect, production-side network effects were strong as users could relatively easily convert into developers who then contributed themselves to the improvements in the software.

The TUG, staffed by volunteers, used members' contributions to finance and encourage development of a more complete and accessible T_EX distribution as well as a more user-friendly interface. This made T_EX more attractive to a broader audience and more competitive vs. proprietary alternatives. The resulting wider use of T_EX, in turn, increased the benefit of network effects to existing developers and users.

Network effects motivated developers to consider users' needs. However, those users were those in their own constituency – be it defined as the group of early developers, as early users of T_EX, as academics or as specialized typesetters. This situation gave the opportunity for commercial software developers to offer typesetting software for other types of users.

Over time, as commercial software came to offer increasingly high typesetting standards, T_EX lost a part of its user base. Compatibility with its standard thus became less valuable. Those OS developers who previously had to abide by the T_EX standards for fear of losing compatibility of their systems with the main strand of T_EX development were no longer so reluctant to launch their own versions of T_EX, extending its capabilities to, for example, multilingual typesetting or educational publishing.

Network externalities thus played an important role in the dynamics of T_EX's development, imposing coherence in times of dominance, leading to fragmentation in times of weakness.

In this article, I focus on the competitive dynamics in the development of the T_EX system. Further work would examine the ecology of this OSS project: how did the various projects within T_EX compete, how was a constant level of commitment obtained from developers, what were the specific coordination/management skills that successful project leaders within T_EX possessed? The T_EX experience certainly is a fertile ground for further research into the dynamics of community wide efforts in the development of a public good.

6 References

L^AT_EX3 Project Team (1997) "Modifying L^AT_EX," *TUGboat*, 18: 127-130.

Beebe, N. H. (2004) "25 Years of T_EX and METAFONT: Looking Back and Looking Forward," *TUGboat*, 25: 7-30.

Bergstrom, T., L. Blume, and H. Varian (1986) "On the Private Provision of Public Goods," *Journal of Public Economics*, 29: 25-49.

Bessen, J. (2004) "Open Source Software: Free Provision of Complex Public Goods," Working paper, Boston University School of Law and Research on Innovation.

Bezroukov, N. (1999) "Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism)," *First Monday*, 4, http://www.firstmonday.org/issues/issue4_10/bezroukov/

Bitzer, J. (2004) "Commercial versus Open Source Software: The Role of Product Heterogeneity in Competition," *Economic Systems*, 28: 369-381.

Bitzer, J., and P. Schröder (2006) "The Impact of Entry and Competition by Open Source Software on Innovation Activity," in J. Bitzer and P. Schröder (eds.) *The Economics of Open Source Development*, Elsevier: Amsterdam.

Bonaccorsi, A., and C. Rossi (2004) "Altruistic Individuals, Selfish Firms? The Structure of Motivation in Open Source Software," *First Monday*, 9, http://www.firstmonday.org/issues/issue9_1/bonaccorsi/.

Casadesus-Masanell, R., and P. Ghemawat (2006) “Dynamic Mixed Duopoly: A Model Motivated by Linux vs. Windows,” *Management Science*, 52: 1072-1085.

Cottrell, A. (1999) “Word Processors: Stupid and Inefficient,” Essay,
<http://www.ecn.wfu.edu/cottrell/wp.html>

Dalle, J.-M., and N. Jullien (2003) “‘Libre’ Software: Turning Fads into Institutions?” *Research Policy*, 32: 1-11.

Economides, N., and E. Katsamakas (2006) “Two-sided Competition of Proprietary vs. Open Source Technology Platforms and the Implications for the Software Industry,” *Management Science*, 52: 1057-1071.

Fitzgerald, B., and J. Feller (2001) “Open Source Software: Investigating the Software Engineering, Psychosocial and Economic Issues,” *Information Systems Journal*, 11: 273-276.

Fitzgerald, B., and J. Feller (2002) “A Further Investigation of Open Source Software: Community, Co-ordination, Code Quality and Security Issues,” *Information Systems Journal*, 12: 3-5.

Flynn, P. (2001) “ $\text{\TeX}$ – A Mass Market Product? Or Just an Image in Need of a Makeover?” *TUGboat*, 22: 137-139.

Foray, D., S. Thoron, and J.-B. Zimmermann (2007) “Open Software: Knowledge Openness and Coalition in Cyberspace,” in E. Brousseau, and N. Curien (eds.) *Internet and Digital Economics*, Cambridge University Press: Cambridge.

Foray, D., and J.-B. Zimmermann (2001) “L’économie du Logiciel Libre: Organisation Coopérative et Incitation à L’innovation,” *Revue Économique*, 52: 77-93.

Franke, N., and E. von Hippel (2003) “Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software,” *Research Policy, Special Issue on Open Source Software*, 32: 1199-1215.

Gaudeul A. (2003) “The (IA) $\text{\TeX}$ Project: A Case Study of Open Source Software,” *TUGboat*, 24: 132-145.

Gaudeul, A. (2005) “Public Provision of a Private Good: What is the Point of the BSD license?” Working Paper, University of East Anglia, Norwich and ESRC Centre for Competition Policy.

Gaudeul, A. (2006) “Vertical and Horizontal Differentiation with Network Effects in the Competition between Open Source and Proprietary Software,” Centre for Competition Policy Working Paper 06-12.

- Hann, I.-H., J. Roberts, S. Slaughter, and R. Fielding (2004) "An Empirical Analysis of Economic Returns to Open Source Participation," Working Paper, Carnegie Mellon University.
- Herbsleb, J., and A. Mockus (2003) "An Empirical Study of Speed and Communication in Globally-Distributed Software Development," *IEEE Transactions on Software Engineering*, 29: 481-494.
- Hertel, G., S. Niedner, and S. Herrmann (2003) "Motivation of Software Developers in Open Source projects: An Internet-based Survey of Contributors to the Linux Kernel," *Research Policy*, 32: 1159-1177.
- Knuth, D. E. (1991) "The Future of T_EX and Metafont," *TUGboat*, 11: 489.
- Koenig, J. (2004) "Seven Open Source Business Strategies for Competitive Advantage," IT Manager's Journal, <http://www.itmanagersjournal.com/articles/314>
- Kogut, B., and A. Metiu (2001) "Open-Source Software Development and Distributed Innovation," *Oxford Review of Economic Policy*, 17: 248-264.
- Kuan, J. (2001) "Open Source Software as Consumer Integration into Production," Working Paper, Haas School of Business, University of California-Berkeley.
- Kuan, J. (2002) "Open Source Software as Lead User's Make or Buy Decision: A Study of Open and Closed Source Quality," Working Paper, Stanford Institute for Economic Policy Research, Stanford University.
- Lakhani, K., and E. von Hippel (2002) "How Open Source Software Works: "Free" User-to-user Assistance," *Research Policy*, 1451: 1-21.
- Lakhani, K. R., and R. G. Wolf (2005) "Why Hackers do What They do: Understanding Motivation Effort in Free/Open Source Software Projects," in J. Feller et al. (eds.), *Perspectives on Free and Open Source Software*, The MIT Press: Cambridge, MA.
- Lerner, J., and J. Tirole (2002) "Some Simple Economics of Open Source," *Journal of Industrial Economics*, 50: 197-234.
- Mittelbach, F., and C. Rowley (1997) "The L^AT_EX3 Project," *TUGboat*, 18: 195-198.
- Mockus, A., R. T. Fielding, and J. D. Herbsleb (2005) "Two Case Studies of Open Source Software Development: Apache and Mozilla," in J. Feller et al. (eds.), *Perspectives on Free and Open Source Software*, The MIT Press: Cambridge, MA.
- Mustonen, M. (2003) "Copyleft – The Economics of Linux and Other Open Source Software," *Information Economics and Policy*, 15: 99-121.

Mustonen, M. (2005) “When Does a Firm Support Substitute Open Source Programming?” *Journal of Economics & Management Strategy*, 14: 121-139.

Nichols, D., and M. Twidale (2003) “The Usability of Open Source Software,” *First Monday*, 8, http://www.firstmonday.org/issues/issue8_1/nichols/

The NTG T_EX Future Working Group (1998) “T_EX in 2003,” *TUGboat*, 19: 323-329.

Raymond, E. (2001) *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly Media, Inc.: Sebastopol, CA.

Schmidt, K., and M. Schnitzer (2003) “Public Subsidies for Open-Source? Some Economic Policy Issues of the Software Market,” *Harvard Journal of Law and Technology*, 26: 473-505.

von Hippel, E. (1994) ““Sticky Information” and the Locus of Problem Solving: Implications for Innovation,” *Management Science*, 40: 429-439.

von Hippel, E. (1998) “Economics of Product Development by Users: The Impact of “Sticky” Local Information,” *Management Science*, 44: 629-644.

von Hippel, E. (2005) *Democratizing Innovation*. The MIT Press: Cambridge, MA.

von Krogh, G., S. Spaeth, and K. R. Lakhani (2003a) “Community, Joining, and Specialization in Open Source Software Innovation: A Case Study,” *Research Policy*, 32: 1217-1241.

von Krogh, G., and E. von Hippel (2003b) “Special Issue on Open Source Software Development,” *Research Policy*, 32: 1149–1157.

Wichmann, T. (2002) “Firms’ Open Source Activities: Motivations and Policy Implications,” Free/Libre and Open Source Software Final Report, Part II, International Institute of Infonomics.

Yamauchi, Y., M. Yokozawa, T. Shinohara, and T. Ishida (2000) “Collaboration with Lean Media: How Open Source Software Succeeds,” in *Proceedings of Computer Supported Cooperative Work*, ACM Press: New York, NY.

7 Appendix

7.1 Popularity estimates for T_EX

T _E X vs. other OS projects					
	Website	Page Rank	Back Links	Word Count	Alexa Rank
T _E X	www.tug.org	8	1,740	52,700,000	42,363
Apache	www.apache.org	9	106,000	125,000,000	535
Mozilla	www.mozilla.org	10	238,000	110,000,000	541
Linux	www.linux.org	9	23,700	355,000,000	12,276
Sendmail	www.sendmail.org	8	1,650	16,600,000	46,147
T _E X vs. other typesetting and word processing software					
	Website	Page Rank	Back Links	Word Count	Alexa Rank
T _E X	www.tug.org	8	1,740	52,700,000	42,363
QuarkXPress	www.quark.com	8	1,660	5,570,000	27,403
InDesign	www.adobe.com/products/indesign	9	5,730	11,200,000	100
Microsoft Word	www.office.microsoft.com	8	6,780	182,000,000	16
OpenOffice	www.openoffice.org	8	49,200	31,200,000	2408

Table 1: T_EX vs. its peer groups

Methodology: The name of the software was typed into Google's search box. The number of websites mentioning the software (word count) was collected. The first website in Google's list of results was selected. The number of websites linking back to that website (back-links) and its page rank was collected. The page rank is a complex function of the number of back-links that determines a website's rank in Google's list of results. It ranges from 0 to 10, on which scale higher is better. The website's Alexa Rank, determined by the number of visitors to the website and their comments was also collected (where lower is better).

In case the software name used common words, the relevance of the search results was assessed by checking that the 10th page of results on Google contained a reasonable concentration of references to the software under study. If not, a more precise keyword was chosen.

7.2 T_EX sub-projects

Name	Object	Category	Lead Developer	Inception
T _E X	Mathematical typesetting	Core	Donald Knuth	1978
Plain T _E X	Basic macros	Macros	Donald Knuth	1978
AMS-T _E X	Macros for AMS journals	Macros	Michael Spivak	1981
L ^A T _E X	General purpose T _E X macros	Macros	Leslie Lamport	1983
BiB _T E _X	Bibliographies	Extension	Oren Patashnik	1984
MakeIndex	Indexes	Extension	Pehong Chen	1987
NFSS	Easy access to a complete selection of fonts	System	Frank Mittelbach and Rainer Schöpf	1989
CTAN	Online repository of T _E X packages	System	CTAN team	1990
PSTricks	Figures and graphs	Extension	Timothy Van Zandt	1993
4AllT _E X	Plug and Play T _E X distribution for MSDOS PCs	Distribution	Netherland LUG	1993
WinEdt	Shareware Interface	Interface	Aleksander Simonic	1993
Ω	Multilingual typesetting	Core	Yannis Haralambous and John Plaice	1994
L ^A T _E X 2 _ε	Extension of L ^A T _E X	Macros	L ^A T _E X3 team	1994
ConT _E Xt	Macros for educational publishing	Macros	Hans Hagen	1994
TDS	Directory structure	System	TUG Working Group	1994
teT _E X	T _E X distribution for Unix	Distribution	Thomas Esser	1994
pdfT _E X	pdf output	Extension	Hàn Thế Thành	1996
T _E XLive	Official distribution	Distribution	TUG and LUGs	1996
MikT _E X	Distribution for Windows systems	Distribution	Christian Schenk	1996
ε-T _E X	Updating T _E X	Core	Peter Breitenlohner	1997
NTS	Rewriting T _E X	Core	Karel Skoupý	1998
fpT _E X	Windows version of teT _E X	Distribution	Fabrice Popineau	1998
Winshell	Freeware interface	Interface	Ingo de Boer	1998
LyX	WYSIWYM (what you see is what you mean) interface	Interface	Matthias Ettrich	1999
T _E Xshop	T _E X previewer for Mac OS X	Interface	Richard Koch	2000
Preview-L ^A T _E X	Instant previewing of documents	Interface	David Kastrup	2001
T _E Xmacs	WYSIWYW (what you see is what you want) interface	Interface	Joris van der Hoeven	2002
X _E MT _E X	T _E X system for the education sector and public administrations	Distribution	Marie-Louise Chaix and Fabrice Popineau	2003
XeT _E X	T _E X with Mac OS X and Unicode	Core	Jonathan Kew	2004

Table 2: T_EX sub-projects by year of inception and development category

The list of projects in table 2 is not comprehensive. A more complete list of contributors is given below. Categories include core, macros, distribution and interface development. Additional categories are systems development, which are improvements in the accessibility and structure of T_EX systems and extensions, which are programs extending the capabilities of T_EX without changing its core.

ε-T_EX: Peter Breitenlohner and Phil Taylor.

CTAN team: Rainer Schöpf, Joachim Schrod, Sebastian Rahtz, George Greenwade, Robin Fairbairns, Jim Hefferon.

LaTeX3 team: Johannes Braams, David Carlisle, Robin Fairbairns, Frank Mittelbach, Chris Rowley, Rainer Schöpf, Thomas Lotze, Morten Høgholm, and Javier Bezos. Previous members: Denys Duchier, Alan Jeffrey, Michael Downes.

T_EX Live/Collection coordinators: Sebastian Rahtz, Karl Berry, Staszek Wawrykiewicz, Manfred Lotz.

ConT_EXt: Hans Hagen, Taco Hoekwater, Patrick Gundlach, Adam Lindsay and others.

pdfT_EX: Hàn Thế Thành, Martin Schröder, Hartmut Henkel, Taco Hoekwater, Hans Hagen, Heiko Oberdiek.

7.3 Chronology

See tables 3 and 4 pp.262 and 263, respectively. Development is divided into three areas: core, macro, and others (organization, distributions and interfaces). Competitors are divided into four types: independently developed software, OS or proprietary, and software inspired by T_EX, open source or proprietary. Freeware and shareware are classified as proprietary.

7.4 The T_EX user group

Figure 2: TUG revenues and membership by year: 1982-2000

DEVELOPMENT OF T _E X													
	1970-1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989
Core		T _E X 1.0 in Sail	DVI format			T _E X 2.0 in Pascal					T _E X2C: T _E X in C		T _E X3.0
Macro		Plain T _E X			AMS-T _E X		L ^A T _E X	BibT _E X			makeindex		L ^A T _E X3 team
Organization (O), distributions (D) and Interface (I)				TUG (O)									NFSS (O)
EVOLUTION IN THE SOFTWARE ENVIRONMENT													
	1970-1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989
Independent proprietary	Script by IBM (70), Roff by Bell Labs (71)					Corel Word Perfect		Microsoft Word	Pagemaker, Adobe Postscript	Framemaker, Ventura, 3B2	QuarkXPress	MS Word for Windows	
Independent OS										SGML			
Inspired proprietary									PCT _E X, MicroT _E X				T _E Xtures
Inspired OS													

Table 3: Chronology of the development of T_EX and of its competitive environment from 1970 to 1989

DEVELOPMENT OF T _E X														
	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003
Core	Web2C				Ω		PdfT _E X	ε-T _E X	NTS					XeT _E X
Macro			PSTricks		L ^A T _E X 2 _ε , ConT _E Xt						XMLT _E X	JadeT _E X (SGML)		
Organization (O), distributions (D) and Interface (I)	CTAN (O)			4AllT _E X (D)	TDS (O), teT _E X (D)		MikT _E X (D), T _E X Live (D)		fpT _E X (D)	LPPL (O)	T _E Xshop (MacOS) (I)	preview- L ^A T _E X (Unix) (I)	Debian guidelines	X _F M _E T _E X (Windows) (I)
EVOLUTION IN THE SOFTWARE ENVIRONMENT														
	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003
Independent proprietary				pdf by Adobe							InDesign by Adobe			
Independent OS		Groff (roff for GNU)	HTML						XML, Corel Wordperfect for Linux	MathML	XHTML, Abiword		OpenOffice	
Inspired proprietary		Y&Y	Scientific Word	WinEdt					WinShell					
Inspired OS						Lyrix				LyX			T _E XMac	

Table 4: Chronology of the development of T_EX and of its competitive environment from 1990 to 2003