The (LaTeX project: A case study of open-source software

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

September 17, 2004

*I would like to thank William Adams, Jacques André, Nelson Beebe, Barbara Beeton, Karl Berry, Lance Carnes, Thomas Esser, David Fuchs, Bernard Gaulle, Hans Hagen, Yannis Haralambous, Jim Hefferon, David Kastrup, Donald Knuth, Leslie Lamport, Wendy McKay, Barry McKichan, B. Mahesh, Frank Mittelbach, Oren Patashnik, Simon Pepping, John Plaice, Fabrice Popineau, Sebastian Rahtz, Denis Roegel, Chris Rowley, Joachim Schröd, Karel Skoupy, Hàn Thế Thành, and all other TeX project participants I met. This paper was directly inspired by the reflections that community kept having about its own culture, history, future and organization. All errors and omissions are of course mine and are the result of my bad understanding. I also owe many thanks to all the people who develop, maintain and distribute the TeX system and make it freely available to us researchers and academics; this project was an opportunity to learn how to better use that typesetting system that keeps amazing me. I would also like to thank Jacques Crémer, Bruno Jullien, Jean Tirole, Hal Varian and other academics who inspired or guided my reflections.

Contents

List of Figures

1 Introduction

This is a case study of TEX, a typesetting software that was developed by Donald E. Knuth in the late 70's. Released with an open source license, it 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 is part of a wider effort by academics to understand the open-source phenomenon. That development model is similar to the organization of the production of knowledge in academia; there is no set organization with a hierarchy, but free collaboration that is coordinated spontaneously and winds up generating complex products that are the property of all who can understand its functioning. The case study was led by gathering qualitative data via interviews with TEX developers and quantitative data on the TEX community – the program's code, the software that is part of the TEX distribution, the newsgroups dedicated to the software, and many other indicators of the evolution and activity in that open-source project. The case study is aimed at economists who want to develop models to understand and analyze the open-source phenomenon. It is also geared towards policy-makers who would like to encourage or regulate open-source, and towards open-source developers who wonder what are the efficient strategies to make an open-source project successful.

The primary aim of this case study is to define the relationship between consumers and producers of a software product in an open-source ("OS") organization; how can a product that was developed for a producer's own use become of use to others. The case-study is inspired by the work of Hippel (2002) that underlines how, while the development of open-source software ("OSS") may have own-use as their primary motivation, it is complemented by another, collective, process that reconciles private aims with the aims of the general public. (LA)TEX provides a good case study to illustrate that point, as its development was oriented from the beginning towards the end-user, and it was distributed under a liberal license that allowed developers to build commercial products from their work in developing the software. The open-source user/developer interface organization was set up very early, and worked alongside commercial ventures to popularize the TEX software. The study of the development of TEX thus provides an interesting sample of various strategies to bring open-source software development efforts to the market. The process by which software is popularized may be taken up by the developers who, to obtain a better product, must make it available to others so that they may contribute to its development, and must therefore take their needs into account. That process can also be taken up by a mediating organization, set up by users to exploit the work of the developers. That organization assumes the role that the marketer plays in the proprietary model of software development. The production and marketing of a product inside a firm are closely linked in a continuous feedback process; a product has no meaning other than the value it has on the market. The open-source phenomenon, as seen through this case study, disjoins the production and the marketing side of software development; the productive activity of developers is not directed towards the market, but rather towards the fulfilment of private needs, while the market appropriates the result of that independent productive activity and transforms it to make it available to the general public.

The analysis of the coordination between developers and users in an OS project is analyzed in more details in another paper (Gaudeul (2003b)) that is not included in the thesis. The present paper mainly deals with other matters: By examining the raw output of the project – the software's code and its evolution, the stages in the development of the core of TEX are delineated: there is a 'breaking' point in the development where a code that favored early development and usability of the software becomes an impediment to its further, independent, development into new applications and for new types of users. By looking at the process by which that code was produced – the project's functioning – the constant clash between a logic of centralization of the contributions and of competition between development initiatives is underlined. The independent development initiatives must be centralized so as to make them available to all, developers and users. However, there are cases where it is not possible to make a choice between two incompatible ideas, which leads to the choice being made later, based on the ensuing compared popularity of the two competing development strands. The two development logics overtake each another constantly in the chronology of the project. The limits and advantages of the open-source development and coordination methods over proprietary software are also examined. The independent role of the open-source nature of the project in explaining its successes and failures is identified by identifying the causes that are idiosyncratic to this project (chance events, personalities, context). Finally, the framework in which the development process took place is examined – the TEX organization, rules and culture. A chart of its changes over time is made, as initiatives were taken from different quarters to solve the coordination and organizational problems of the community. TEX developers and users experimented with various ways to encourage the development and distribution of the software, and various logics were at work depending on the identity of the community and project leaders. Professional typesetters and publishers, academics and other individual end-users, commercial and open-source software developers had different motivations and objectives for the TEX organizations, and this is reflected in the organization of the project.

Before outlining the plan of the paper, the OSS production model will be detailed further, so as to explain the perspective taken in this paper: A set of pre-existing tasks are taken up by developers, either with a view towards producing a commercial implementation (proprietary model) or with a view towards private use (OSS model). In this latter case, implementations of those tasks are then exploited by an intermediary layer of developers whose work results in a product that is better fitted to the satisfaction of the users' needs, because those developers/exploiters represent more closely the general composition of the potential user base for the software. They face a constraint that is not present in the proprietary model of development: the set of tasks they can integrate in the software has been selected by individual developers, with no account taken of the needs of the whole market (which determine the conception of the commercial product). [1] Not all OSS commercialization projects are led by the users, however – commercialization being meant here as making a product available to non-expert users, as opposed to that set of expert users who developed the product in

[1] Of course, the developers in an open-source project are also part of the user-base for the software, and there is therefore some similarity in what they do and what the market needs.

the first place. There are indeed commercial distributions of GPL software, such as Red Hat for Linux and BSD software can be exploited commercially by entrepreneurs. There can therefore be a mix of commercial and private motives in OSS commercialization projects, and the distinction between OS and proprietary software is not as sharp as frequently assumed.

In the schema adopted here, there are two steps in software development, one that is production – and that stage can either be commercially or privately motivated – and the other that is popularization or marketing of the software – that stage being also either commercially or privately motivated. In that last stage, users unite to make use of the part of the work that was made available for free after the first stage. There may for example have originally existed a lot of different tasks that could be taken up in one area of software development, and that would provide utility to users (the utility of each of those tasks not being necessarily the same for each of them). A profit motivated entrepreneur will choose to take up those that will maximize his revenues, i.e. the combination that maximizes the difference between revenue and costs. His own needs are mostly irrelevant in that calculation, and since he can recover his investment in the development of the software by selling it to a large base of users, he may take up important and costly tasks that are out of reach of a private developer. Privately-motivated developers will take up one or more tasks depending on their own needs, and while they may have limited resources, their own professional interest in the project may motivate them to take up the development of important tasks. They may also take up tasks that other people on the market do not value much, because they personally need a functionality that others are not interested in.

In the second stage when marketing is done, and in the case where the first stage was not aimed toward the second phase of development (OSS model), since private developers may not want to market their development work themselves, there is an array of functionalities that are available for free use, but that are not directly usable by the general public. Commercially or altruistic minded individuals or organizations may then choose among those tasks those they want to include in a product that will be meant for general use. A model based on that vision of software development should therefore compare three types of software production: pure commercial, semi-commercial and pure open-source. The comparison would be based on what tasks are taken up by each method of production and how much welfare is generated. The analysis should also take into account the interactions between each type of production. If the open-source development model is very efficient in one area of software development, then a commercially minded entrepreneur will choose to base his work on that body of work instead of developing a product from scratch, for example.

The second, marketing, stage does not involve only the distribution of existing software, it also involves building an interface to the software packages, making those packages work by the same standards, organizing the software project's functioning and facilitating or encouraging the development of the software in socially desirable ways.

Let us give a very simple mathematical example to understand the economic problems involved in the analysis of the OSS development model. Compare two options, developing two features independently in a modular way, as is typical of open-source development, or develop them in an

integrated way, planned centrally, as is typical of the proprietary development model. Suppose there are two projects, A and B, which cost c_A and c_B to develop independently, and c_{AB} if developed jointly, i.e. it costs c_{AB} to develop software that fulfils both projects' aims. There are two developers A and B who value projects A and B at v_A and v_B respectively, and if a software provides both functionalities, they both value it at v_{AB}. Social welfare if they each develop the software independently is therefore $v_A + v_B - c_A - c_B$, and if it is developed jointly it is $2v_{AB} - c_{AB}$. Joint development of the two tasks can be taken up for example by developer B who will sell the completed software to developer A at price P subject to $v_{AB} - P \geq v_A - c_A$ (A's incentives), and provided that $P + v_{AB} - c_{AB} \geq v_B - c_B$ (B's incentives). Solvinf for P, this means, in this simple model, that whenever a model of development is efficient from a social welfare point of view, it will be taken up. Open-source development would therefore arise when it is not efficient to develop two tasks together, either because developing software that can fulfil both tasks is very intricate and requires too many compromises, because there are coordination problems and it is too costly to establish communication between the two developers and get them to agree on a common structure and some common goals, or because setting up a commercial front to sell the software to A is not among the competencies of B.

That simple comparison, where developers have different value for tasks, must choose which tasks to develop in common and decide whether that common development is desirable, doesn't take into account the following points:

- Having many people using the product doesn't bring the same benefits in the proprietary model as in the open-source one; schematically, while in a proprietary model users bring money, in the open-source model, they bring expertise, suggestions for change, bug reports, and can become developers.

- There is an additional layer in the open-source development model, which is that taken up by the people who manage the software distribution. The software distribution is an essential part of an open-source project. It makes a whole of the different contributions that were made to the project. There are three ways software distribution maintainers can influence the development of an open-source project: influencing development by supporting developers and directing the effort of volunteers toward their aims; punishing developers who go against the common development effort (forking) or do not adopt the distribution's standards (license, compatibility) by excluding their software from the distribution and thus negating them the help and prestige that goes with acceptance; and finally, developing the infrastructure to make the different contributions work together, managing the meeting point for developers, directing the debates on which way the software development must go, developing standards for package writing, occasionally hiring people to take up some code writing that necessitates commitment over the long term, etc. In all this, the software distribution maintainer has, arguably, superior knowledge of the functioning of the whole software. In his role as a middleman, he frequently is the one who filters bug reports from the user to the developer,

and he has a direct interest in making the software work as smoothly as possible so as to minimize the amount of maintenance.

- There is frequently a mix of open-source and proprietary exploitation of open-source software. While the distribution maintainer has an essential role in GPL projects, because commercial distributions of GPLed software will never be much different from the open-source version (there is no point spending much in developing a product that will have to be made freely available to all), it has to compete with proprietary version of the software when the software is BSD, and that version can add a lot of functionalities that will not be put back in the common pot.

The rest of the present paper is a preliminary study, which is meant as a global analysis of the important features of the (LA)TEX project. It will serve as an illustration for further theoretical work. While the above mentioned goal of the case study is its primary focus, the case study doesn't limit itself to the question of whether and how OSS can be user-oriented. The case study also deals with other matters and can be of interest to those who want to test other potentially interesting topics in the study of the OSS phenomenon.

A first part of the paper puts this case study of TEX into its context. After presenting the motivations for this case study and the body of work that frames it, the history of TEX and the institutions that frame and encourage its development and spread will be presented. Two appendices at the end of the paper define the terms that will be used in the rest of the study, and give a chronology of the main events in the now long history of the project. The first part is also an attempt at outlining the major phases in the development of an open-source project, and will determine whether TEX is actually an open-source project — it outlines the difficulties in defining that concept practically. Finally, a comparison between the Linux and TEX open-source paradigms will be made. Other projects could have been chosen as a basis for comparison, notably some projects that are closer in size and purpose to the TEX project; but Linux has become a reference in the study of open-source and it is therefore interesting to show how different TEX and Linux are. A second part, which was presented to TEX developers for discussion, presents the most important and innovative findings of the study — it is not meant as a discussion of others' findings on open-source software, but as a presentation of some patterns and facts that are not usually discussed from an economic point of view in the literature. The conclusion of this paper outlines areas for future research, notably the interaction and comparison between proprietary and open-source software, the role of distribution maintainers in enforcing discipline and quality standards, and the patterns in the leadership styles of successful open-source projects.

2 The context of the (LaTeX case study

2.1 Motivation

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 open source software projects ('OSSPs') are managed, and also help policy makers gain a better understanding of the open source ('OS') phenomenon.[2] 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 a general-purpose software, its users' community was not necessarily technically sophisticated, and the software was not part of software 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 their functioning in economic terms. In the last few years, open source software economics has been the subject of a lot of 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 the well-trodden areas by studying a less well-known OS 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 OS. The conclusions from this study challenge 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 the previous case studies by going into the TeX success stories as much as into the problems encountered along the way.

This case study is sponsored by the GREMAQ, a CNRS research group in mathematical economics at the University of Toulouse in France, and the 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 Schools 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

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

unflappable kindness.

2.2 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 with other developers.

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 to define what license terms are best in what setting. License terms balance the need for control over the development of the software versus its necessary change under the influence of others, and balance private incentives versus group incentives: Proprietary license terms give individual developers more control over their work, GPL ones 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 non-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.

2.3 Definitions of TEX: a software and a community

2.3.1 TEX and LATEX

TEX is a mark-up language used to typeset scientific documents with complex mathematical expressions. It is a standard for the writing of linearized mathematics and the electronic filing of mathematical documents, and it greatly improved the efficiency of scientific journals production. LATEX is a set of macros operating on the TEX primitives (A.5.4) that was developed on top of TEX and is a somewhat restrictive but more user-friendly 'front-end' to TEX.

TEX is thus a computerized typesetting system that allows both to generate a text to be published, but also to provide instructions on how it is to be typeset. This program greatly facilitates the communication between authors, publishers and typesetters, as it provides a common language for all of them to communicate with. But it also provides them with high quality typesetting; texts typeset with TEX still are recognized as being of very high standard.

TEX is a medium size software project, not an operating system, but not a single-purpose program either. It was a successful program, that generated a whole literature, an active users' community that organized meetings and conferences gathering people interested in the software: the TEX community was at the forefront of a revolution in publishing.

TEX was one of the first OS projects, it managed to survive many changes over many years and is still at the center of a very active users' community. Many firms based their activity on it (See A.1.1 for more on TEX, A.4 for some TEX specific terms, A.5 for programming and open-source definitions, A.6 for typesetting notions that will come back in the document).

2.3.2 The software, its users and developers

There are different classes of actors with different motivations in the TEX community: commercial users, developers, and authors.

- Typesetters and font designers, generally from publishing companies, but some also from public institutions, gave directions and comments to the first set of TEX programmers, those who developed the software, the core group of TEX developers. Among those are the AMS which was the first to typeset its journal with TEX, Addison-Wesley, an academic and technical books publisher which published most of Knuth's books, Elsevier which adopted LATEX as its standard for archiving documents for a while. There also are a variety of one person companies, such as Hans Hagen's Pragma ADE which developed ConTEXt, that base their activity on the use of programs derived from TEX.

- The early TEX programmers were computer science students of Knuth, later ones were publishing, typesetting or font-design professionals or academics, self-selected based on their interest for typography, and often self-taught in programming. Some of the core developers were hired by public institution or commercial companies to adapt TEX to their need; most of them program TEX as a hobby, but the core development is done by academics (Plaice and Haralambous, of the Omega project, at the ENST telecommunications school and the University of Sydney), professional programmers (Mittelbach, of the LATEX3 team at EDS), and they sometime took jobs as TEX consultants (Rahtz who worked at Elsevier Science)

- End-users — authors, secretaries, academics, etc — use the software under its various forms — for the PC, for the Mac, commercial, open-source, TEX, LATEX — with various levels of sophistication — from the basic user of a commercial version of TEX (See A.2.1) to the user who is able to program simple style sheets for LATEX so as to obtain customized results, going through those who use open-source implementations of TEX (See A.2.2). They are involved at various degrees in the users' community — from the occasional participant in TEX newsgroups (See A.3.9), to those that become members of the TEX organizations. Some gained influential positions in the TUG (See A.3.4) hierarchy even though they had no experience in programming.

2.3.3 The software, its developers' groups and support institutions

There are many centers around which developers congregate and organize their work; they can be divided between development projects and support institutions, although developers commonly straddle the divide.

The main TEX support institutions are:

- The TUG (A.3.4), which organizes the support and meetings for TEX users and developers. It also organizes working groups such as the one that established the TEX directory structure (A.3.2) or established the DVI standard (A.6.3).

- The CTAN (A.3.3) archives, where all TEX programs are archived and which serves as a point of reference for programmers, and

- The TEXLive project (A.3.1), which releases ready to install open-source implementations (A.4.1) of TEX for the end-users.

The TEX Users' Group ("TUG", A.3.4) was created in 1980 and was financially supported by the AMS. The TUG was intended for all those who had an interest in systems for typesetting technical texts, and for font design. It was also meant to encourage the exchange of information in that domain, to establish standards and share macro packages based on TEX. It produced TEX publications, such as the TUGBoat (A.3.6), and sponsored, developed and implemented TEX training programs, seminars and conferences. It provided tapes with the TEX system at cost, and provided the formation and the materials for people who wanted to set up TEX workshops in their universities, companies and other institutions.

When TEX became used internationally, Local Users Groups ("LUG", A.3.7) emerged, all under the umbrella of the TUG that became both the US LUG and the international LUG of reference. The first LUGs were set-up in Western Europe, in or around 1989 - the French GUTenberg, the Dutch NTG, the German Dante, the British UK TUG - and groups later emerged in developing countries — China or India — in 1997, and in Eastern Europe — Poland, Hungary. Those international LUGs were more interested than the TUG in LATEX, as it was easier to develop and use than TEX on which development they had no influence, and they focused on serving the individual end-users that were at their origin, as opposed to publishers, universities or TEX developers who created the TUG. Later LUGs in developing countries were also very interested in the open-source nature of the software as a mean to locally develop software instead of using pirated foreign software.

Development projects can be divided in three categories:

- Reworkings of the core of TEX or extension of the set of LATEX macros. (LATEX3, A.1.3, eTEX, A.1.4, NTS, A.1.5, Omega, A.1.6, pdfTEX, A.1.7, ConTEXt, A.1.9)
- Distributions and other infrastructure projects that make the programs available to the users. (A.3.1)
- Independent, non-core special purpose projects that include users interfaces, converters, etc. (LATEX-preview A.1.11, Bib-TEX A.1.12, some derived open-source systems A.2.2)

Those three types of projects have widely different ways of operating. The 'core' projects involve a variety of actors as their direction is of interest to all users of TEX. The infrastructure projects bring the programs to the users and are thus responsive to their queries, they act as user-developer interfaces. Special purpose programs form a galaxy of independently motivated developers that may have few interactions with other TEX actors.

The LATEX3 (A.1.3) developers - Frank Mittelbach, Chris Rowley, Sebastian Rahtz - are considered as the core developers whose decisions affect all other developers, and they work closely with the TEXLive (A.3.1) maintainers - Sebastian Rahtz, Fabrice Popineau - who decide what packages will be included in the standard (LA)TEX distribution (A.4.7). Other important core projects include pdfTEX (A.1.7) originated by Hàn Thế Thành, ConTEXt (A.1.9) around Hans Hagen, Omega (A.1.6) around Yannis Haralambous and John Plaice.

2.4 An history of (LA)TEX

Various forces direct the development of TEX, due to the different concerns, objectives and priorities of its developers, who came from different fields and made use of it in different ways. Over time, with the user base changing and the software's environment evolving, different types of priorities have emerged: in a first period, the objective of Knuth was to develop a software that could do computer mathematical typesetting worthy of the best manual typesetting tradition, then, when he felt his objective was achieved, the AMS (A.3.5) wanted to make this instrument available to the wider mathematical community, and sponsored the development of AMS-TEX (A.1.8) and its subsequent merging with LATEX. The objective of the following core developers' teams was to make use of new computer capacities and make TEX more easily extendable with the use of new programming tools (NTS, Omega), establishing a standard LATEX to prevent forking (LATEX3), and making TEX up to date with new document publishing standards (PDF, XML, A.6.3 and A.6.2). In the same time, competing and successful typesetting programs and standards (Framemaker, Adobe, Word, A.2.4) made some work necessary in making TEX able to produce, for example, and most importantly, pdf and web documents, but also improving its user interface to compete with WYSIWYG system (A.6.1)

The number of users of TEX can be estimated by looking at the different versions of TEX. By the estimates of Knuth, and based on the number of bug reports and feedback he got, TEX79 was developed while TEX was going from 100 to 1,000 users, TEX82 probably was developed for 10,000 users, and by the time TEX90 was developed, there were probably a million users. The direction of the development also shows the increase in the number of users, as more and more needs came to be covered by the software and it saw applications in many unforeseen domains (drawing graphs, managing databases, producing interactive web documents, etc...) The evolution in newsgroup activity around the world, and the setting up of LUGs in Europe and then in Asia, is also testimony to the rapid spread of TEX.

The history of (LA)TEX can be divided in three stages:

- The first phase (1978-1982 for TEX, 1982-1990 for LATEX) was the building of TEX and LATEX by Knuth and Lamport respectively, with the help of other developers and in conjunction with concurrent development; the TEX systems came to equal the functionalities of competing proprietary typesetting systems of the time such as TROFF or Scribe (A.2.3) while producing better quality documents with an interface that was more easily portable to a variety of systems. In this same time, the TUG was set up, primarily as a tool to disseminate the program, and help people use it.

- The second phase (1982-1990 for TEX, 1989-1995 for LATEX) was the solidification of TEX and LATEX as they came to fulfill all the needs that they were intended to fulfill at the beginning. This was accompanied by the setting up of some institutions and the progressive gain of independence of the organization; license terms were defined, a centralized repository was set up, developers came from outside the inner Stanford circle that developed the early TEX, the

TUG became financially independent from the AMS. Commercial distributions appeared, which allowed the end users to have prepackaged versions of the program. While TEX remained under the control of Knuth, this phase of solidification for LATEX was directed by the LATEX3 team, in which Lamport was not involved.

- The third phase (1992-now) was a phase of divergence, with new projects revealing the inadequacy of the program for some users. It was prepared by a flurry of new projects that emerged in the 1990s, and first tried to remain in the limits of the acceptable for the community. An official open source distribution that had been made possible by the centralization of all TEX programs progressively put the commercial firms based on TEX out of business.

2.4.1 The first phase: building the program and distributing it.

The first phase in the history of TEX was the building of the program that would be the point of reference for all further development. The development objectives were set by Knuth at Stanford, the AMS set up the TUG to help distribute the program, and the objectives were influenced by concurrent developments in other proprietary typesetting systems of the time — this is how sets of macros for TEX were developed to imitate competitors' functionalities.

TEX was developed in 1979 and a first version was delivered in 1980. Its paragraph-breaking algorithm was based on joint work with Michael F. Plass in 1980. (See A.1.1) Successive version of TEX were developed by Knuth, including a radical rewrite of TEX in 1982 (TEX2.0 also known as TEX82), and a minor one in 1990. That last reworking was done under the pressure of new uses for the program that could not be accommodated with the older versions (use of international characters, notably), but TEX82 had the same goals as TEX79, except that by that time Knuth had gathered the means and people to achieve his initial aims. TEX therefore kept being under the control of a limited set of developers chosen by Knuth and coming from an academic setting. Knuth sought suggestions and ideas for improvements and changes, getting inspiration from previous electronic publishing systems, and taking example from what he considered as the best manually typeset mathematical journals, but the implementation of those ideas was entirely under his control (See A.1.2).

There was a completely different process of development for TEX macros; TEX provided for a macro language that became very powerful with TEX82 and thus allowed for many independent developments. In fact, Knuth expected authors and publishers to write their own versions of TEX or at least their own style files[3], and provided for the means to do so. Lamport and others developed competing set of macros for TEX and it was Lamport's LATEX (A.1.3) that became the most widely used as his manual, published in 1983, became a reference, and it provided all the functionalities of Scribe, the main competing typesetting software for mathematics at the time. This manual was published by Addison-Wesley, which also published most of Knuth's books, and was one of the first users of TEX, alongside with the AMS which used TEX for typesetting its journals.

[3] A set of instructions to typeset a document automatically using TEX commands.

A final version of LaTeX, LaTeX2.09, was released in 1985, before it was taken over by Mittelbach, Schöpf and Rowley in a project that became known as LaTeX3 in 1989. This led to the introduction of LaTeX 2_ε in 1994, which is the current final version of LaTeX. In 1990, the AMS ported AMS-TeX into LaTeX, a move that brought back all development onto the pair TeX/LaTeX, a commitment that continued in 1995 with the adaptation of AMS-LaTeX to LaTeX 2_ε. The development of macro packages for TeX was thus a lot more open to competition and challenges than was the development of the core, AMS-TeX being the main competitor to LaTeX, and LaTeX being an independent project by Leslie Lamport.

The drivers for the development: origin, inspiration, orientation and sponsors Knuth decided in 1974 that a solution had to be found to the degrading quality of typesetting and typography, as typesetters were moving toward computer aided typesetting and there was no software that could achieve the same kind of quality as that of traditional typesetting of the time. TeX was inspired by previous computer systems, such as the ones used at the Universities Press in Belfast or the one developed by Computer Technology Inc. System in Cambridge, Massachusetts.

A precursor to TeX was ROFF, developed by the Bell Laboratories, which provided text formatting and worked under UNIX, a product of the same Bell Laboratories. ROFF came with adjunct programs like EQN to typeset mathematical expressions, REFER to make bibliographies, TBL to make tables. ROFF competed for a long time with TeX, as TeX progressively came to acquire all the functionalities of ROFF and became a complete and better system. BibTeX, for example, was released in 1985 to be the equivalent of REFER. Other systems such as Scribe, created by Brian Reid, or IBM's Script also existed (See A.2.3).

Knuth delivered in 1978 an AMS Gibbs lecture about mathematical typography, which gave the impetus for his project. TeX was from the beginning oriented towards the scientific and mathematical community, the typesetting-printing-publishing establishment, and suppliers of computerized typesetting equipment and text processing systems. Those last ones were the ones that were to make TeX work in practice.

The development and launching of TeX was planned in advance, and its initial development doesn't resemble the debut of more recent, 'spontaneous' open-source software projects. The launching of TeX followed a marketing plan based on the publication of the TeX manuals by Digital Press, an affiliate of Digital Equipment Corporation ("DEC"), and the provision by DEC of the TeX system on tapes. Knuth's manual about TeX and Metafont (an adjunct program to generate fonts) was published in 1982, but an unfinished version had been circulating before. More generally, the impulse for the development was from the top-down, with few independent projects at the beginning; work was assigned rather than chosen.

It was also the result of close cooperation, sponsored by the NSF, with the American Mathematical Society ("AMS") that had worked from 1965 to 1971 on the use of computer help for publishing its journals. The AMS already used a commercial typesetting system, STI, that was displaced by Knuth's open-source system. The AMS sponsored the development of a version of TeX, AMS-TeX,

by Spivak, which was an adaptation of TEX for sophisticated mathematical typesetting.

The TUG will not be talked much here, as the TUG and its role became more prominent and independent only later in time, but it was set up mainly as a way to help academics obtain and learn to use the program, so that they be able to submit papers, for example to the AMS, in that format. It was not meant in the beginning as a way to link or recruit developers, but as a way for users to share their experience. There was indeed at the time no concept of how independently motivated developers would come to join open-source organizations and contribute to the writing of software code. The infrastructure for such cooperation was not available anyway; no common programming language, no Web to coordinate work efficiently, no concept or need for an open-source community.

2.4.2 The second phase: taking a life of its own

This was a phase where the TEX community organized itself independently from its originators, and set up goals for its survival and adaptation. The software was exploited commercially, with proprietary versions coming about because the community was not yet able to put enough structure to the development, and not strong enough to initiate the development and maintenance of an open source user-friendly software based on TEX. This was a phase where the originators of the project dis-involved themselves and commercial implementations of TEX came to fill the void, as the development of TEX packages was disorganized and it was difficult for the non-initiated to make sense of it. However, that phase was in the continuity with the previous one from the point of view of development, as it didn't require radical rewriting of the original software.

The most important commercial implementations of TEX were Blue Sky Research ("BSR")'s TEXtures for the Mac released in 1988, Y&Y's implementation of TEX, MacKichan Software's Scientific Word which first version was released in 1988. The first implementations of TEX for the PC were PersonalTEX, developed by Lance Carnes, and MicroTEX by Addison-Wesley Ed, developed by Spivak, both commercialized in 1985. All those commercial implementations provided users with a user-friendly implementation of TEX ported onto personal computers as those became more widespread in the 80s. (See A.2.1)

Among the other commercial ventures related to TEX, notable is the Imagen company that was set up in 1980 to make a fast, high quality printer able to generate TEX 's output for middle-end users. Lester D. Earnest and Luis T. Pardo, two collaborators of D.E. Knuth were the main shareholders of this company in which D.E. Knuth was also given some shares. This is one of the few examples where originators gained financially from their work on TEX, as many early developers went on to work as software developers or font designers: David Fuchs went on to participate in the setting up of Framemaker, an unrelated typesetting system, for example.

That period led to attempts to organize the open-source community, with the emergence of a central repository (CTAN, A.3.3), of a structure to be applied to all TEX distribution to facilitate their installation on any system (the TDS, A.3.2), and of users' group in Europe who took up the task of adapting TEX to their typesetting cultures (See A.3.7).

As for the development, this was a time where initial developers found new interests and their

projects had to be taken over; this led to the creation of the LaTeX3 team to keep on developing LaTeX. Unlike the third phase, though, the limits of TeX were not yet reached, and there wasn't such painful putting into question of the TeX tradition as that next period will show.

2.4.3 The third phase: going beyond (La)TeX

This was a phase where the developers who gathered around a common software project had to find a new purpose, adapting the software to a new environment and preserving its open-source nature. 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 so much concern for compatibility, while serving the needs of more specialized classes of users. The choice between stability and development was represented by the conflict of ideals between the LaTeX3 project (See A.1.3) - building upon TeX with the help of macros- and the NTS (see A.1.5) projects - a project that endeared to rewrite the core of TeX. It also confronted the usefulness of the changes to most users that was provided by the pdfTeX (See A.1.7) project, which output TeX into Pdf format – and the need for adjustments for fringe users provided by the Omega project (See A.1.6), which extended the capabilities of TeX to multilingual typesetting.

There was a dwindling impetus in the project, as the mainstream product came to cover most needs, and new developments were interesting only for a minority. Also, initial users saw most of their needs covered, and did not see why they would accommodate changes that were necessary for new users. Potential new users therefore chose other, more recent programs, instead of TeX, joining communities that were more responsive to their needs. The complexity of reworking the program was also a major impediment, and there was a conflict in strategies to solve that problem between a total reworking of the program (NTS), organizing the switch to other programs (pdfTeX), or adapting the program by gradual increments (LaTeX3).

This was accompanied by the progressive fading of the role of the formal organization that linked all developers together: A centralized coordination system lost some importance with the emergence of the Internet as a coordination tool, but also as the initial developers who had participated into it were leaving. The main developments in the program consisted in managing its legacy, and making it up to date with the new publishing standards (pdf, xml, mathml). However, at the same time, efforts from the central organization were made to make the open-source program more easily accessible to the general public, as the TeX Live project, a complete distribution of TeX on one CD, showed (See A.3.1 and A.2.2 for some example of independent or centralized efforts to make TeX more accessible).

2.5 TeX, an Open Source Software Project?

This part discusses whether TeX's license terms and the way it was developed qualify it as an open source project.

2.5.1 TEX's license terms

In its ideal form, an OSS is software which source code is available to all and independent development is allowed by its license terms. It is distributed for free and derived works may also be required to be made available to all. This means that such software is freely available to all for use or for development. They are an alternative to proprietary software, not only because they can replace existing proprietary software, but also because the way they are developed differs. Indeed, OSS developers organized themselves into communities based around the sharing of the improvements to the code of specific software projects. Such communities have been successful in developing complex software projects, even though they lack the formal organization and legal protection that traditional firms use (See A.5.5 for some open-source terms definitions).

At the time when TEX was written, the 'Open Source' concept didn't exist, and the license of the program written by Prof. Knuth doesn't properly fit in any of its present categories: of course, the program's source code was published and anybody can rewrite it, but the name 'TEX' is trademarked and any change to the program that are not approved by Knuth must be renamed. The core of TEX, tex.web (A.4.5) is copyrighted by Knuth, and no changes are permitted, although of course concepts used in the program may be re-used, but "TEX: the program" (A.4.6) is in the public domain, although the use of the name 'TEX' is restricted to exact copies of Knuth's software systems. Any change to the program must pass a series of tests (A.4.8) that guarantee they will give the exact same output as TEX if they want to call themselves a 'TEX system'.

The license that is used for most TEX programs — that include all programs that have added up to 'TEX: the program' is the LATEX Project Public License ("LPPL", A.3.8). Any program belonging to a TEX distribution — a distribution is any set of TEX packages that can input `.tex` files and output a result that passes the TRIP tests (A.4.8) — must fit with the Debian Free Software guidelines, which define the term 'open-source'. The LPPL can best be described as a Berkeley Software Distribution ("BSD") license as closed-source proprietary versions are allowed, but it requires that any change to a file be renamed and distributed with the original version, so as to guarantee the integrity of the system: any user must be able to have access to the exact same program as any other user so as to output the exact same result from a `.tex` file. That requirement caused problems vis-à-vis some Debian administrators, who said it contradicted some tenets of open-source, but is not unlike some requirements in the Apache project. Any modification of Apache code may not call itself Apache, that name being the property of the Apache software foundation (See A.5.5 for more on open-source licenses).

2.5.2 The organisation of TEX's development

Professor Donald E. Knuth initiated the development of TEX, a typesetting program for mathematics, in 1977 in Stanford (See A.6 for some typesetting related terms). This was initially a personal project. As the project became more complex, an organization was built to support its development. It was a non-academic project, which groundwork was done during sabbatical periods, but the way

the project was developed borrowed many features from the way academic projects are developed in American universities. The source code was distributed, first to a limited circle of colleagues, and sponsors (publishers like Addison-Wesley, associations like the AMS and government bodies like the NSF) financed the buying od some equipment, tested the code and provided funding to hire assistants who came in to develop TEX's core code (See A.1.1 for more on the development of TEX)

It is only later in the development that TEX came to acquire the features of more familiar Open Source software projects. Knuth wrote a manual explaining the working of TEX, which was widely distributed from 1982 on. The program attracted a lot of attention from developers and users outside the initial Stanford inner circle. This is when Knuth had to define under what license TEX would be distributed. He encouraged the setting up of an users' organization to help in spreading the program, but also to give a direction to the various development efforts that came from many parties.

The motivations of Knuth when developing TEX were however clearly of an open-source nature. TEX was meant first as an academic experience designed to test a programming language (WEB, A.5.2) and provide a model for building large programs that students could learn from. Every step in the code writing was carefully noted down in a document called "The Errors of TEX" (A.4.2) and lessons were taken to improve system programming. It also filled a need for high-quality computer typesetting, and this is why it attracted a lot of contributions, improvements and additions after the main work was done. This need met with the readiness of Knuth to provide the source free of rights, and his willingness to spread the program. That willingness was based on the belief that the program would attract more contributions and require less effort from him that way. In fact, the program's core development never really became independent of Knuth, in part because of its license term that prohibited rewriting it and naming the new version TEX, but also because the core was sufficiently well-written, with many ways to hook up to it and add new features, that developers could make it do what they wished by writing macros. This is how the core never attracted much independent development until the late 80s, when Knuth made the last few changes to accommodate the needs of new and more sophisticated users and then decided to freeze the program's core, something the license terms permitted as the only one authorized to make changes in it was Knuth.

The 90s saw the full emergence of TEX as an open source software project as it became independent from its orginator and attracted independent development.

2.6 A reference point: Linux.

Before going on a presentation 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 Linux example. 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 were distributed under different license terms. The TEX project's size, evaluated by the number of people who develop TEX and LATEX, is smaller than the GNU/Linux project's size and as a matter of fact, GNU/Linux distribution generally include TEX and LATEX. Finally, the goals of the projects were different.

Donald E. Knuth developed the TEX software in the late 70s, before the Internet came to be the tool it is today to organize open-source communities. The community surrounding the software went through several changes over many years, and accompanied the evolutions in the standards used for publishing and in the way software developers' communities work. Linux on the other hand was started in the 90s, relied on the existing open-source community that freed Unix, and used tools developed by the Free Software Foundation in the GNU project. The TEX project provides a long-term view of the history of an open-source software project. Its relatively self-contained developers' 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 still is a whole typesetting system with many interdependencies. TEX provides a sufficient level of complexity to be the subject of a self-contained case study, but still can be studied as a whole. The project can be understood without relying on catch phrases and slogans, unlike what has been the case with many studies of Linux.

The community that built around both software was 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 on TEX was pragmatic, funded by governmental 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 the Microsoft monopoly—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 was distributed was 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 have to compete with proprietary systems that are 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 diffusion of an unique 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 predominant and became part of the standard distribution. Therefore, no one asserted itself 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 pretty much alone as no one could make those changes other than him. Linus Torvalds on the other hand had to integrate changes in the code that were proposed by others, because anybody could take the kernel and make its 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 on; it was meant to provide an interface between the authors and the 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 respect compatibility with older versions of the software and that they did not contain bugs. The development of Linux was made in the open while TEX packages were developed 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.

3 Some preliminary findings.

This part is organized in three main sections. The first one 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 first begin 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.

3.1 The output: The characteristics of the code and its quality.

3.1.1 The importance of the initial coding of the software for its future.

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 in the case of 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 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 the control of the TEX's original hierarchy.

This is why it is only quite late in the development of TEX that problems with the core's limitations appeared and it was necessary to make it easier to change, for example by organizing its modules into a set of 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 as Knuth didn't designate a successor who would be responsible for his program, there was no focus point for developers on which to 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 that one finally was 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 like pdfTEX.

In summary, independent development of the program was delayed because TEX evolved into a monolithic-looking program that was intended to become a standard in publishing and was developed in a closed academic setting. While the objectives of Knuth did get realized, subsequent developments were made difficult in a new setting where TEX users and developers had to coordinate through electronic means and the OSS developers community was established around concepts and tools that were different from those of the first TEX implementors. TEX's program had to get translated in the standard 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 a proof 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 developers' community at the time.

3.1.2 The impact of the OS software on welfare and innovation. Its quality seen from various point of views.

In this part, 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 barely up to the standards of TEX when it was developed. The program that was 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 those two software programs. The main competing software for the casual user is now Word by Microsoft. 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 the typesetting of complex mathematical documents in the publishing industry is 3B2. Framemaker of Adobe and QuarkXPress of Quark are also popular alternatives.

A frequently asked question is whether OSS takes the place of proprietary software and whether it undermines innovation in the field 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 an user interface and porting it to non-Unix 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 opposite.

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 also are examples of users and firms switching back and forth between OS and proprietary software. TEXLive for example did gain a lot of business on commercial implementations of TEX, 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 (conception principles, 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 functionalities that were not present in 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.

While a comparison of the welfare generated by TEX with that which would have been generated by a proprietary software may look like a futile academic exercise, TEX was developed as an alternative to a commercial software that was used by the AMS in its publishing section, and the AMS did ponder what was the best option: wait for a 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 an hypothetical equivalent commercial software can be made in terms of innovativeness, 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. If OSS is a tool for the 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 and Michael Spivak—both of the AMS and who went on to important positions in the TUG hierarchy—spent time in Stanford developing TEX macros to test TEX capabilities for such AMS requirements as generating indexes, for example. 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 users' groups, publishers such as Addison-Wesley or 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 to get 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 a standard. In the case of TEX, the solution was to design standards for the classification of packages (the TEX directory structure) 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 also difficult for proprietary software firms to adhere to because they want to protect their user base—prevent it from switching—but also because the source is closed, so that

it is difficult to create applications linked to it. However, software firms propose development platforms to programmers, are also interested in the promotion of their standard, and usually are able to establish and maintain them. 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 platform independent, it is difficult to keep it 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 kept on 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 a 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.

3.2 The process.

3.2.1 The dynamics of the project and its limits.

The dynamics. Various forces direct the development of TEX, due to the different concerns, objectives and priorities of its developers who came from different fields and made 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 a first period, the objective of Knuth was to develop software that could do computer mathematical typesetting 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 (Omega, NTS), while also establishing a standard LATEX to prevent forking—LATEX 2_ε by the LATEX3 team. In the same time some work was necessary in making TEX able to produce 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 in-

stead of Adobe fonts, and mark-up-based document writing instead of Word-like WYSIWYG programs. There were however some open-source projects trying to attain more users' friendliness—Kastrup's preview-LaTeX package to ease editing, LyX, a document processor for Unix platforms 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 changing the code—rewrite the WebPascal 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)).

Does OS development process, but also the specific OS institutions that support that development, put some limits to the growth and success of open source software? Growth and success are important because even if the software does function in accordance with the stated aims of the project initiator and the initial users' community; it will quickly become obsolete and useless to those same people if it doesn't keep being developed to make it up to date with the changing software environment. This can justify changing the aims of the software's community, even in ways that are 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. This 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 encoding in 1989. Since the whole TeX system refers back to the core of TeX, its pace of development is indicative of what happens in the wider TeX community.

Figure 1 p. 29

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 can be trained to use it. However, it is still true that it will be more difficult to diffuse it if there is nobody ready to make the necessary tinkering in the software code that will permit adaptation to new usages.

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, or 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.

Figure 2 p. 29

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

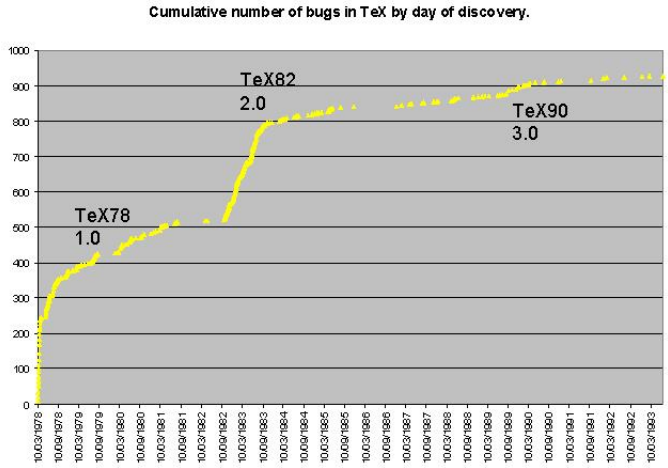

Figure 1: The bugs of TEX

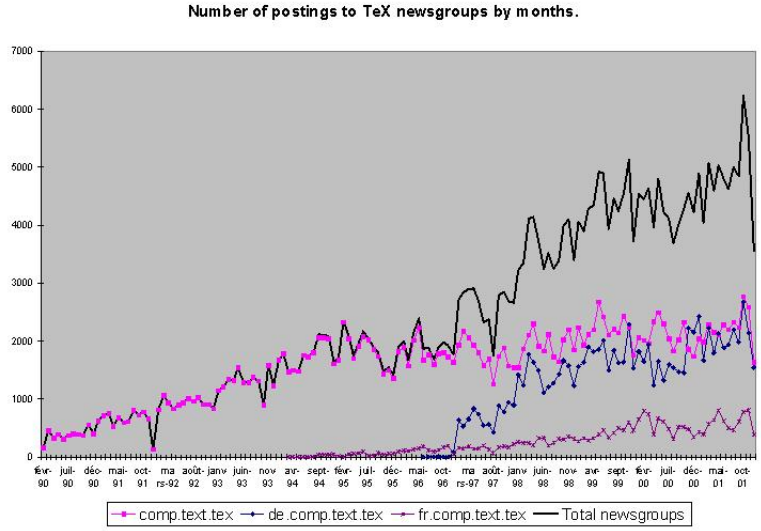

Figure 2: The monthly posting to TEX-related newsgroups.

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 a difficulty in getting 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 its appeal. The software progressively reaches all of its intended audience, or is supplanted by another software for that audience.

There are aspects of the software's concept that are difficult to change, for example its typesetting mark-up language, and this put limitations to its appeal. The concept fatally becomes 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's 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. The ConTEXt project and the pdfTEX project were attempts at broadening the capabilities of TEX to make them up to date with what was necessary for online publishing. TEX was oriented toward printing, and was not able to provide the kind of interactive color complex documents with figures that were needed for online publishing, and was also based on a document exchange standard (DVI) that had been abandoned in favor of the pdf format. The pdfTEX and ConTEXt projects had to make changes in TEX's conception to adapt it to new needs. Other projects made TEX XML-compatible. In summary, TEX first was at the forefront in mathematical publishing, but it 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 to itself some limits: the originator is ready to support only a limited number of people; Knuth had other priorities, the writing of his series of books about the Art of Computer Programming. TEX was 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.

3.2.2 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 then releasing it into the public once 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 advanced.

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

As 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 output TEX into pdf files. 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 early on its goals, 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 is 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 into. 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 programmers, 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 ultimate goal's importance was not discussed. 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 does not get 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 succesful 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 TEX served the mathematical community, the TUG, which ensured it was user-friendly. The AMS was the main leader for TEX's development. It provided financing for users' group, made propositions to developers, 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

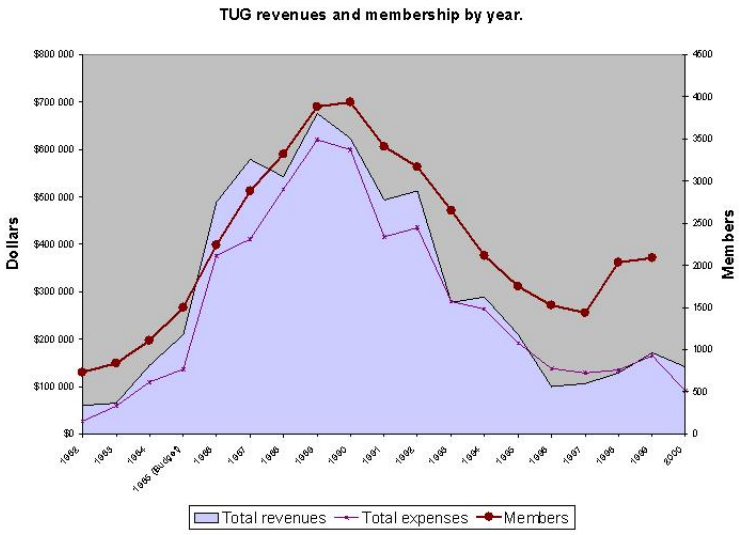

Figure 3: TUG's revenues and membership.

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 have wanted to. 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 version of TEX that would fit its own use (AMS-TEX) and also sponsored the TUG. Its role was to form people who would be able to use the tools recommended by the AMS. The AMS couldn't hope for academics to 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 was far beyond the means involved because it served as a signal for other sponsors that TEX was a valuable project that would ultimately, willy-nilly, be a complete system. This is why the TUG also attracted sponsorship from various hardware companies and universities. The AMS progressively lowered its financial contribution to the 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 the TUG, and the influence the TUG had on TEX's development.

Figure 3 p. 32

Leadership was also provided by users and intermediated by users' 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, which here was the goal of an association distinct of the TEX community. TEX had no purpose of its own but the 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 some aims, which are that of its sponsors. The TUG served as a meeting point between users and developers of TEX, the two not being exclusive. The articles in the TUGboat were often made 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 from the part of the users' 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—Omega project. New categories of users who did not have the same legacy issues succeeded in breaking the status quo. Various users constituencies, 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 built under the lead of typesetters, publishers, software companies and university institutions who were interested in mathematical publishing, while the European ones were led by individual users, sometimes active in universities or educational publishing. Many of those users' needs were not satisfied by the users' organization that was 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 tapes 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 European 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 the contribution from the American organization too.

Another example of users' 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 TEXLive project. This one adopted many of the ideas in the 4AllTEX distribution—choice of program, organization of packages.

The tensions between TEX constituencies did not only translate in users' 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 so much concern for compatibility, while serving the needs of more defined classes of users.

3.3 The framework: The TEX rules and culture and how it 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 parsing unnecessary developments, but the fact that Knuth was not clear whether he had stopped developing TEX or not did not help in encouraging 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 the DVI standard, 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 submissions 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 serious. 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 e-mail inside an 'old boy' network, 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 of TEX a different case from other well-known OSS that have been studied before (Apache, Linux), and will help to determine if an OSS can be user-oriented, a

mass-product. From the beginning on, the program was intended to be used by non programmers: secretaries, researchers... In fact, one of its main 'selling' point at the beginning was how easy it was to install and learn; the syntax was meant to be natural, and Knuth wrote a manual for the program in 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 out of the lucky coincidence into one person of 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 nor 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 TEXLive distributions that a complete easy to install TEX distribution was made available for Linux.

There also were individual OS initiatives to make the software easier to use; preview-LATEX or TEXview provided graphical interfaces. Other collective initiatives were driven by ideology; LyX is such an attempt at providing a convenient TEX interface for Unix systems. 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 interface 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 a close coordination with leading core developers.

A stable users' platform was needed because TEX documents had to be easy to share for it is to be considered a standard and adopted widely. Therefore a core product had to be defined, on which other things added up 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 TEXLive 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 users' community.

4 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 developed upon. A more careful analysis of the determinants in the success of an OSS sub-project could for example be made. This would allow a better definition of quality, leadership and support in an OS environment.

References

(LA)TEX related articles

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

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

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

[4] Hàn Thế Thành (1998): "The pdfTEX program", Cahiers Gutenberg, 28-29, Mars 1998

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

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

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

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

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

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

[11] Mittelbach F. and C. Rowley (1997): "The LATEX3 project", TUGboat 18(3), 1997 Annual Meeting.

[12] Skoupy K. (1998): “NTS: A New Typesetting System”, TUGboat 19(3), 1998 Annual Meeting.

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

[14] Taylor P. (1997): “Présentation du projet eTEX”, Cahiers Gutenberg, 26, Mai 1997.

[15] Torzynski M. (1996): “Histoire de TEX sous Dos et Windows à l’ENSP de Strasbourg”, Cahiers Gutenberg, 25, Novembre 1996.

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

[17] Interview with Leslie Lamport (2000): “How LATEX changed the face of Mathematics”, DMV-Mitteilungen, January 2000

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

[19] The TEX Users Group (TUG) at `http://www.tug.org`

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

[21] “Modifying LATEX” TUGboat 18(2), June 1997

[22] NTG TEX future working group (1998): “TEX in 2003”, TUGboat, 19(3), 1998 Annual Meeting.

General articles on Open-Source

[23] Anderson R. (2002): “Security in Open versus Closed Systems—The Dance of Boltzmann, Coase and Moore”, Working Paper.

[24] Behlendorf B. (1999): “Open Source as a Business Strategy”, Open Sources, O’Reilly editors

[25] Benkler Y. (2001): “Coase’s penguin, or, Linux and the Nature of the Firm”, Yale Law Journal, 112, October 2001

[26] Bessen J. (2002): ‘OSS: free provision of a complex public good’, `http://www.researchoninnovation.org/`

[27] Dalle J-M and N. Jullien (2002): ‘OS vs. proprietary software’, `http://opensource.mit.edu/`

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

[29] Brady R., Anderson R. 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.

[30] 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.

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

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

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

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

[35] 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.

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

[37] Hertel, G., Niedner, S. & Herrmann, S. (2002): "Motivation of software developers in open source projects: An internet-based survey of contributors to the Linux kernel." Research Policy, in print.

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

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

[40] 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/`

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

[42] 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

[43] Lerner J. and J. Tirole (2002): “The Scope of Open Source Licensing”, Draft, 2002.

[44] Lerner J. and J. Tirole (2000): “The Simple Economics of Open Source” NBER Working Paper 7600

[45] Mockus A., Fielding R. and J. Herbsleb (2000): “A case study of open source software : The Apache Server” International Conference on Software Engineering, pp.263-272, 2000.

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

[47] Mustonen M. (2002): “Why do firms support the development of substitute copyleft programs?”, Working Paper, October 2002

[48] Mustonen M. (2002): “Copyleft—the economics of Linux and other open source software”, Working Paper

[49] Nakakoji K.et alii (2001): “Toward Taxonomy of Open Source : A Case Study on Four Different Types of Open Source Software Development Projects”

[50] O’Mahony S. (2003): “Non-Profit Foundations and their Role in Community-Firm Software Collaboration”, Working Paper, Harvard Business School.

[51] Peyrache E., Crémer J. and J. Tirole (2001): “Some reflection on Open Source Software”, Communications & Stratégies, 40, pp. 139-160.

[52] Pressman R. (1997): “Software engineering”, 4th edition, Mc-Graw-Hill editors.

[53] 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.

[54] Scotchmer S. and Samuelson P (2002): “The Law and Economics of Reverse Engineering”, Yale Law Journal, April 2002.

[55] Silverman D. (1999): “Doing Qualitative Research: A Practical Handbook”, Sage.

[56] Varian H. (1993): “Economic Incentives in Software Design”, Computational Economics, 6(3-4) pp.201-17, 1993

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

[58] GNU General Public License at `http://www.gnu.org/copyleft/gpl.html`

A Glossary

A.1 Projects and programs

A.1.1 TEX

TEX is a computerized typesetting system, meaning that a TEX 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 may be distributed in digital or paper form. TEX compiles that file and generates an output with the desired fonts and location of characters.

Building this typesetting engine required many layers of programming: first writing a library[4], then a (macro) language, and then a high level interface.

The first version of TEX was developed in 1978-1979, changes in TEX were made, with a major re-implementation of TEX culminating in the release of TEX82, and then, other changes leading to the release of TEX3.0, which is the final version of TEX, the last one developed by Knuth and therefore the last one to be called "TEX".

In order for a distribution to use the TEX name, it must pass the TRIP tests that were devised by Donald Knuth in order to guarantee they can process files in the same way and with the same output than the original TEX. However, there are other definitions of what constitutes a TEX program. For example, some projects are based on the TEX core developed by Knuth and made some changes to accommodate new needs (eTEX, Omega, ConTEXt, pdfTEX). They are called extensions to TEX, and work in the same way and with the same packages than other TEX distributions. However, like in the case of Omega, they do not necessarily pass all tests required by Knuth, and may not be 100% compatible with TEX files produced by other distributions. Other programs integrate some ideas of (LA)TEX (TEXmacs, LyX) but necessitate some conversion tool to translate the file they produce into TEX files. They are said to be 'based' on TEX, and can be used as a self-contained alternative to TEX or LATEX. The TEX distribution can decide to include some implementation of TEX and not others, as for example ConTEXt is included in the TEXLive distribution, while Omega is not.

A.1.2 TEX1.0, TEX2.0 and TEX3.0

The first version of TEX was developed by Knuth alone, and Knuth thought it would be finished quickly. Two students of his developed a prototype during the summer of 78, based on a sketch of the program and its specifications. Knuth then realized that he would have to write it alone, as while this prototype was well written, it was nowhere near completion and it would have taken a lot of time just communicating those. The problem with first generation systems is indeed to communicate your ideas and requirements, especially if other developers do not have the same background, or the program is not organized in independent modules. There is a need for boundaries between people so that they do not have to interact too much, but in the case of TEX, every part of the program was

[4] A programming library is a collection of subroutines and functions stored in one or more files, usually in compiled form, for linking with other programs. Libraries are one of the earliest forms of organized code reuse. They are often supplied by the operating system or software development environment developer to be used in many different programs.

dependent from the other. The goal of Knuth was to minimize the total amount of time spent on the program, as it was simply meant as an instrument to typeset TAOCP, a book he considers as his main contribution to the programming field, and there were not anymore any typesetting system at the time able to produce a book up to his standards. Knuth decided not to use 'rapid prototyping', which consisted in writing the program bits by bits and testing those independently so as to check their correctness, and then assemble the parts later. This is because he realized this would require too much time testing. Therefore, he wrote the program all in one block, and began testing it only after 6 months of programming. This explains why TEX is written in a monolithic way, and not with modules. TEX was one of the first large program written using 'structured programming', a technique that is described in TAOCP, Vol. 1, p.191. Knuth felt confident doing so as it was a shorter program than one he had written before using this technique. This technique involves developing some 'unifying principles' in the program, which can be used for many different tasks, but the problem is that it makes the program more difficult to understand for somebody who doesn't understand those principles. On the other hand, it makes it easy to grasp the program once it has been understood. This was helpful for Knuth as after 6 months he had to come back on his program and begin debugging it, and needed to be able to understand what he had written 6 months before. It is only for TEX82 that 'literate programming' was used for the writing of the program. This came about as Tony Hoare, who was working at the Oxford University Press, suggested that he publish the program for use as course material in the formation of programmers. There already was at the time versions of UNIX that were circulating and were used as an example of the writing of large systems, but this was not done officially. Addison-Wesley Publishing Cy, Knuth's long time publisher, was also interested in publishing a book about a program. TEX82 came into being as TEX79 was in transition from being used by about 1,000 people to about 10,000. Knuth was receiving a lot of feedback from users of his program, among those, Gaile Steele, visiting from MIT, who had made a report to his university about this typesetting system. Then also, contributing to valuable feedback, were his distribution of a TEX manual, and his invitation to the prestigious Gibbs lecture, organized by the AMS in January of 78. Right after that presentation, 4 people from the AMS came during two weeks in Stanford to discuss with Knuth on what were the problem they were facing in typesetting their mathematical journal. The AMS was already at the time an important publisher of mathematical research documents, and wanted to move to acceptable standards in electronic publishing. TEX82 also incorporated ideas from the implementation of TEX made in 1979 in MESA, Xerox's Palo Alto Research Center. TEX82 was developed as the TUG was already in place since February 80, and TEX already had about 1,000 users. A version in Pascal had been written, which made it easier to port the program to other computer systems, and had helped draw a lot of new users to the program. TEX82 was meant as 'what TEX should have been from the beginning on'. By then, Knuth already had a program to work on, and thought he would use literate programming, not only to give an example of good program documentation, but also because it helped him write a better program, as there is nothing more helpful in organizing one's thoughts than trying to explain them to others. The use of literate programming was not meant so as

to encourage other people to work on the program. In fact, Knuth was already trying to disengage from the program, which explains why he put some hooks on the program for other people to get control of the inner working of the program and add new features by themselves. He wanted the new program to be easy to adapt to new typesetting projects, but also wanted to finally produce a stable system on which no work would be anymore needed. By that time, Knuth had plenty research support, with several students working as research assistants on the system. For example, Liang did a lot of work on the word hyphenations. Every week during the development of TEX82, a working group of about 20 people convened to talk about the problems Knuth needed to solve. While Knuth still did all the coding, this working group was useful in relaying the suggestions of users, designing fonts, creating some algorithms, working on parts of the project that were relatively independent, such as the DVI drivers, and the adaptation and testing of TEX on new systems. The project got volunteers to help, but research associates were the ones who interacted with those. Knuth acted as filter to prevent the project from diverging. Knuth owned the code of TEX and Metafont, drivers were made by other people as well as the interface with other micro-processors, the DVI system was made by David Fuchs, and in general, people just commented on the work of Knuth so that he could make changes on it. Later on, a system to filter contributions to the program and bug reports was set up. Volunteers who had shown sufficient understanding of TEX were designated by Barbara Beeton, to decide whether a report of a bug was really a bug or normal functioning of the program, and then passed on those reports to Knuth. Most of those 'filters' worked for about 5 years at a time in that function, and this was not a perfect system, as there is at least one example of a bug report that was filed away, rediscovered independently, and then attributed to its 'original owner'. Finding a bug is the source of great prestige, as Knuth maintains a public list of all the errors found in TEX with their discovery date and the name of their finder, and sends a reward check to the finder which amount grows exponentially with time. Other than this system of filter, Knuth now generally answers only to people with whom he already worked on TEX, and people who have friends in common with him.

A.1.3 LATEX, LATEX2.09, LATEX 2$_\varepsilon$, LATEX3

Leslie Lamport developed LATEX from 1983 to 1985. LATEX is a set of macros designed to facilitate the use of TEX and was developed concurrently with other sets of macros, most notably that of the American Mathematical Society with AMSTEX. All those effort finally merged into the language fostered by Lamport. LATEX was very successful in Europe, because when the TEX system was beginning to be adopted there, LATEX already was fully developed, while in the US, most developers chose to work directly on TEX, since it was the available program at the time. This led to major 'cultural' differences between European programmers and American ones, the last ones emphasizing total control over the output, while the previous ones emphasized ease of use and the generation of useful macros to automate the creation of various types of documents. The LATEX program was taken up in 1989 by the LATEX3 project team with Frank Mittelbach et alii. The most current version of LATEX is LATEX 2$_\varepsilon$ that was released in 1994.

A.1.4 eTEX

eTEX was a project by the same group that was behind the NTS to expand the capabilities of TEX, notably to 16-bit encoding, which was useful to make use of more complicated fonts. eTEX was meant as a stop-gap between TEX3.0 and the total reworking of TEX by the NTS team. It ran into problems as it was not completely compatible with the previous version of TEX by Knuth and users preferred keeping that one even though eTEX provided for more functionalities.

A.1.5 NTS: a New Typesetting System.

The NTS project was begun in 1993 when Jiri Zlatuska, who was supervising the thesis in computer programming of Karel Skoupy, hired him to rewrite the core of TEX, a project which had gained financing by the German LUG, Dante, and also got financial contributions from various other LUGs, including the TUG. It was directed by a group of TEX developers including Peter Bieternach, Hans Hagen and Phil Taylor. The idea was that as Knuth had written TEX alone, a rewriting of TEX was only possible if one person devoted himself full-time to that project. The program was very idiosyncratic, and uncovering all its inter-dependencies required the full-time devoted job of one programmer who could concentrate on disentangling it. The final objective was to obtain a modular program where each part would be independent of the other; improving TEX would become much easier as developers would be able to work more independently and concentrate on one part of the program instead of having to deal with the consequences on the whole each time they made a change. This didn't take account of the fact that Knuth was working in a supportive environment, with the help of a variety of student and computer scientists at Stanford, and was a leader in the project, while Karel Skoupy often found himself alone without the support of a community of programmers and his objectives were set by a group of developers who sometime didn't understand the realities of his job. Java was chosen as a programming language because of its promise as a cross-platform program (one of the big problem with previous TEX versions was the need to adapt it to new platforms, but this was ill advised, since it led to a slow program which has been used up to now only for experimentation purposes. The NTS team also wanted the program to be 100% compatible with the previous TEX, i.e. it had to pass the TRIP test. There also were conflicts about the objectives and design of the project because of the peculiar power structure in the project, with a set of developers heading one developer and directing his work.

A.1.6 Omega

John Plaice is at the origin of the Omega project, a project aimed at extending the multilingual capabilities of TEX beyond western typesetting. He and Yannis Haralambous developed TEX-Unicode, an adaptation of Unicode, an international effort to encode every characters, to the TEX setting, and a set of pre-processors (OTP) to handle complex Chinese, Indian or other scripts. Plaice wanted to have a system that would fit the needs of students at his university, who were coming from many Asian countries and whose typesetting needs were only partially taken into account by proprietary

western software. Haralambous had a typesetting company that specialized in rare scripts and ancient documents with complex typesetting, and wanted a program that could handle those types of documents automatically, instead of having to use a series of ad-hoc pre-processors. Omega is a rewriting of TEX's core and does not maintain full compatibility with TEX. It is mainly a two person project as it didn't get much support from the TEX community, which didn't see multilingual typesetting as very important, but also because the originators of the project wanted to rewrite TEX from the ground up and wanted to have total control over the outcome; they didn't believe in the benefits of open-source as a source of contributors and didn't want to integrate code written by others, but were more interested in the possibility that open-source provides to be in total control of the code. Omega is not yet integrated in standard LATEX distributions because it lacks an interface with TEX to translate its routines. There is a problem as the LATEX3 team does not know enough about Omega to do this and the Omega team want to finish its work on Omega before thinking about an interface. The problem is that the Omega developers made many promises about deadlines, but failed to meet them repeatedly so that TEX developers are wary to commit work on it. Contributors to Omega were mainly students of Plaice and Haralambous at the University of Sidney in Australia and the National School of Telecommunications (ENST) in Brest, France, respectively. It recently got financial contributions (grants) from the TUG.

A.1.7 pdfTEX

pdfTEX was developed by Hàn Thế Thành as part of his thesis in computer science on microtypography. He made TEX's already very good line-breaking algorithm look much better, by (amongst other things) fiddling character widths very slightly and allowing certain characters (hyphen, comma, etc) to protrude very slightly into the right margin. This program generates Portable Document Format (PDF) directly from TEX source. This is a derivative made from the TEX source which allows to bypass DVI output generation, and to produce documents in Adobe PDF directly.

The goal was to make documents produced with TEX more portable by switching to the dominant page description langauge. This also allowed to add functionalities like clickable links in documents or colors, that were difficult to obtain with TEX.

A.1.8 AMSTEX

AMS-TEX is a macro package originally written by Michael Spivak for the AMS between 1983 and 1985 and documented in 'The Joy of TEX'. It is based on plain.tex but is more specifically directed to mathematicians. AMS-LATEX is a port of AMS-TEX to LATEX, providing AMSTEX functionalities in the LATEX syntax.

A.1.9 ConTEXt

ConTEXt is a complete rework of LATEX, does not have to support the LATEX legacy, and produces sophisticated HTML and PDF documents to serve the purpose of its developer, Hans Hagen, who

heads Pragma ADE, a typesetting firm specialized in educational web documents.

A.1.10 Web2c

Web2c is an implementation of TEX which translates the original WEB sources into C, so they can be readily compiled on modern systems. Tomas Rokicki originated the TEX-to-C system in 1987, working from the first change files for TEX under Unix, which were done primarily by Howard Trickey and Pavel Curtis. Tim Morgan then took over development and maintenance for a number of years and changed the name to Web-to-C somewhere in there, and Karl Berry took his place in 1990 and started using the shorter name Web2c. Web2c is under the GPL, unlike most TEX programs, because the Free Software Foundation sponsored the initial development of the Kpathsea library that Web2c uses. Kpathsea is a library providing path searching and other common functionality.

TEX was originally written in a language that very few people are now able to program in, WEB — actually, it was first written in Sail, but the version in WebPascal quickly came out and is the one that was worked on by outside programmers. There have been several attempts at solving this problem, the two main ones being the Web2c package, which translates the source code into the C programming language and vice-versa and the rewriting of TEX in C or in Java, but this was also unsatisfactory as it was not possible to retain all characteristics of the original TEX in those, so that they were not widely accepted as a work basis.

Before the mid-80s, TEX was very difficult to install, as there was a need for a Pascal compiler good enough to compile the Pascal of Knuth. The C translation of TEX with Web2C made things easier. This way of doing things makes it difficult to write extensions and debug them, as you first have to write it in Pascal, and then compile it in C, and then see if it works. This makes it very difficult to debug a program.

A.1.11 preview-LaTeX

Preview-LATEX was developed by David Kastrup and makes it easier, while typesetting a document with LATEX, to see the end result as it will appear on the final document to be distributed. This is one of many projects that aim at adding some Word-like WYSIWYG abilities to TEX. Rather typically, most of the coding was done by one person, others contributing to the porting of the program into other applications, or to the promotion of the program by making it easier to install and understand. Interactions with other projects were rather limited, as for example preview-LATEX was integrated in a Linux version of TEX, LyX, without much need for the help of the original developer.

A.1.12 BibTeX

Another project that with LATEX made of TEX a system with all the functionalities of Scribe and TROFF, the two proprietary typesetting systems of the time. Like with LATEX, the goal was quite clearly to offer all the functionalities those competing systems offered. Knuth indeed wanted his system to be widely used, and thought its open-source nature would better serve the interest of the

community than proprietary systems, which he believed would make closed standards and impede the free exchange of documents. Unlike Lamport, Patashnik was directly contacted by Knuth or one of his assistants to work on this package to automate the creation of bibliographies in documents.

A.2 Other typesetting systems

A.2.1 Derived commercial systems: PCTEX, MicroTEX, Scientific Word by McKichan Software, Y&Y, TEXtures by Blue Sky Research,...

(LA)TEX spawned various commercial applications, some of them profitable. Prominent among those are *Scientific Word* by MacKichan Software, a 'what-you-see-is-what-you-get' ("WYSIWYG") program using TEX in the background, Y&Y Inc., which sells TEX systems for Windows, and Blue Sky Research, which sells TEXtures for Macintosh.

Those are users-interface building on LATEX, adding some functionalities from other programs - for example Maple for scientific calculations in Scientific Workplace - and working on the smooth interaction of TEX with other platforms (Mac, PC) than the one it was originally developed for (VAX, Unix).

A.2.2 Derived open-source systems: TEXMacs, LyX... and interfaces: WinShell, WinEdt...

TEXMacs and LyX are two projects that develop independently from the main development, and their objective is to use TEX as a platform for further development. Their main work is to build a Word-like interface, TEXMacs using Gnu's emacs as the basis.

WinShell is included in the TEXLive distribution and provides a closed source interface with LATEX. Developed by Ingo de Boer, it is therefore not open-source.

A.2.3 Concurrent typesetting systems: TROFF/NROFF, Scribe, Script...

A program that was used to typeset mathematics when TEX was developed was called ROFF, a text formatting language/interpreter associated with Unix, and there was during a long time some competition between the partisans of TEX and TROFF.

LATEX copied many of its functionalities and syntax on Reid's Scribe, another program that was popular to typeset mathematics in the 80s.

A.2.4 Competing typesetting systems: Adobe's InDesign, QuarkXPress, 3B2, Framemaker, Word...

The main competing software for the casual user is now Word by Microsoft. 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 - which explain why TEX developers attach so much importance to having compatibility between TEX implementations.

The main competing software in the publishing industry is 3B2 by Advent Publising for the typesetting of complex mathematical documents. Framemaker and Pagemaker, and more recently InDesign, by Adobe, and QuarkXPress of Quark are also popular alternative. They benefit from an established users community, all typesetters being trained on one of those commercial software; there are thus large investments in human capital that forbid quick changes from one typesetting system to another. Adobe benefits from its dual status as the owner of a standard for the exchange of documents (PDF), and a typesetting software designer.

A.3 Infrastructure

A.3.1 The main (LA)TEX distributions

MiKTEX is the dominant TEX distribution for Windows users, maintained by Christian Schenck. teTEX is the main present Unix distribution of TEX, which organisation is based on the TDS. It is based on the Web2c implementation of TEX and is maintained by Thomas Esser. TEXLive is the official open-source distribution of TEX, a project headed by Sebastian Rahtz, developed since 1996, and includes distributions for most systems. fpTEX is a port of teTEX for Windows that was done by Fabrice Popineau, maintainer of the Windows part of TEXLive. 4AllTEX was a precursor of TEXLive for Windows, done by the NTG (Netherland's LUG), and WinGut was another precursor done by Gutenberg (The French-language LUG).

The most used TEX distributions are of course the Windows ones, since Windows is the dominant PC systems, and PCs are the most common computers. The two main ones according to CTAN download datas are MiKTEX and fpTEX, with fpTEX behind but gaining ground. fpTEX provides the advantage that it is easily maintainable on computer systems that run Unix in the background to manage individual users' Windows systems, because its organization is the same as teTEX which is an Unix-system. It is therefore becoming popular in universities and research institutions, while MiKTEX is more used by individuals on their own computers.

A.3.2 TEX Directory Structure

The TEX Directory Structure ('TDS') is a directory hierarchy for macros, fonts, and the other implementation-independent TEX system files. The role of the TDS is to stabilize the organization of TEX-related software packages that are installed and in use, possibly on multiple platforms simultaneously. Its role is different from the Comprehensive TEX Archive Network (CTAN) archives as the role of CTAN is to simplify archiving and distribution, not installation and use. The CTAN predates the TDS as it was created in 1990 as an extension of the Aston University Archives in the UK that were created in 1986, while the TDS working group was launched in 1994. The TDS was part of a ongoing process by the TEX community organized around its users groups to make TEX more widely available, first making packages easier to find (central repository with the CTAN), then easier to distribute (standardization of the distribution structures with the TDS), and finally easier to install (4AllTEX, WinGut and finally the TEXLive distribution that installs a TEX system in an

automated way).

A.3.3 CTAN

The Comprehensive TEX Archive Network (CTAN) is a repository of all TEXrelated software which is held on three servers in the UK, US and Germany. It is open to download by any person needing a TEX package, and is updated by developers themselves who give their program for upload to the CTAN maintainers. The archive is searchable via CTAN specific categories, including fonts, systems for bibliographies, printer drivers, systems to include graphics, indexes, plus of course the various macro packages for TEX such as ConTEXt or LATEX and the original TEX programs. It does not provide TEX distributions, and instead refer back for this to the TEXLive CD or individual distributions' web pages.

The usage statistics of CTAN, month-by-month summaries of downloads, can be a proxy for the number of new users of TEX, as they come to download the TEX packages. But those statistics may be in fact more of an indicator of the level of activity in the TEX developer communities. More specifically, the downloads made there can be made as well for sharing purpose (developers), as for simple use by individuals. The statistics available on the web begin in January 2001, for the US CTAN, and the UK CTAN archives have data beginning in 1996, which will be used as the proxy. Also, the US CTAN have been unreliable for a year or two before it moved to its present location. So a person could argue that any growth in usage of this site simply reflects a previously artificially depressed usage.

As for the UK CTAN usage, they show an increase in use in the first few years, and then stabilization. It is difficult to know if the increase reveals new users or simply more users becoming aware of the existence of CTAN, or finally, some users switching from commercial distribution of LATEX to TEXLive, which effectively was the cause of the disappearance of many other TEX distributions.

Looking at the data on the US CTAN archives, showing the number of downloads of the most popular packages, MikTEX and TEXLive are the most often downloaded packages, and it is very likely that the vast majority of downloads of those packages are for personal use. This would therefore serve as a good proxy for the evolution of the number of users — as opposed to developers — of TEX, i.e. those who will ask for support over newsgroup — as opposed to mailing lists that are where developers interact. However, those data are not available on a long enough period to compare them with the number of postings in TEX newsgroups.

The statistics showing the date of uploads in the CTAN archives also are a good indicator of the level of activity in the development of each package. If the number of uploads greatly declined over time (or greatly grew) then that would say something about interest in TEX in the world. That's not a one-package-at-a-time number, though. Some authors upload a lot, some only upload major upgrades.

This shows how the term 'level of activity' in the development of a program means. If it asks for example if the author of ulem.sty ever updates it, the answer is that he rarely does. If it instead

asks whether it is an abandoned project that has fallen into disrepair, the answer is that it is not. Because TEX is frozen and LATEX is also relatively unchanging, ulem works great and the author hasn't updated it because it still works.

The number of downloads is also not fair because it is part of larger packaging (like MiKTEX and teTEX) and so the fact that people got it that way means they won't ask for it as a single download.

A.3.4 TUG

The TUG is the "TEX users' group", was set up in 1980 with Richard Palais of the AMS as its first president. The role of the TUG is to organize meetings, publicize TEX, make people aware of each other's work, sending out mails, organizing conferences and workshops. TUG does not need much money, as most developers are self-financed. Developers generally agree that the role of TUG has diminished with the rise of the Internet as a medium for independent, non-centralized coordination of projects related to TEX. The early days TUG controlled distribution, sold books, resold software, took in charge education of new users. There was business going through them, but the Internet took it away. It used to organize working groups, for example to erect standards, like the one for DVI, the precursor to PDF, or to establish a common directory structure for all TEX setups, the TEX directory standard ("TDS"). One third of its revenues in its golden days (end of the 1980's) was from the organization of conferences and training sessions, one third from the sale of tapes, advertising in the TUGboat, and manuals, and the last third which now composes 90% of its revenues, from institutional and individual memberships.

A.3.5 AMS

The American Mathematical Society is an association that was founded in 1888 to further mathematical research and scholarship. It has 28,000 individual members and 550 institutional members, organizes conferences, publishes "Mathematical Reviews", over 3,000 books in print and mathematics databases. It has 209 employees.

A.3.6 TUGboat

The TUGboat is the journal of the TUG, and this journal was the only way to know what was happening in the TEX developers community in the 80's. If you wrote a package, you would write some thing about it in the journal. Now, there is less of an incentive to do so, when the Internet makes the distribution of programs and the coordination of developers much easier. This may explain the problems of TUGBoat in getting sufficient numbers of papers now. Barbara Beeton, editor of TUGboat, is one of the persons to have stayed the longest in the TUG organization, with Pierre McKay. LUGs also have their own journals in their own languages, and there are exchanges of articles between those journals.

A.3.7 LUG

Various LUGs ("Local Users' Groups') were created in Western Europe in the 1980s, with the same objectives and in collaboration with the TUG, among which DANTE (German-speaking), GUTenberg (French speaking), NTG (Netherland), the UK TUG and GUTH for Spanish-speaking countries. Some of those played a major role in establishing the CTAN archives, and in making a precursor of the TEXLive (4AllTEX by the NTG). Other LUGs have appeared in the 90s in Asia (China, India) and in Eastern Europe (Poland, Hungary).

The main concerns of LUGs have been to adapt TEX to their own typesetting traditions, adapt the hyphenation algorithm to their languages, develop LATEX as opposed to TEX, spread TEX through courses and manuals in the national language, and the organization of conferences, and follow and encourage the creation of document processing standards (SGML).

A.3.8 LPPL

The LATEX project public license was introduced in 1999 in preparation for the release of TEXLive 4 to regularize the licensing terms under which various packages on the CD are distributed. The restrictions it imposes are aimed at making the software more reliable, portable and predictable: any licensed item can be freely distributed as long as all the items that are covered by the license are included in the (re)distribution — this guarantees that every user has access to the same tools — all modified files are renamed and all the modified items are clearly identified and the author of those changes directs error reports to his/her own address — this guarantees that an author does not have to deal with many version of his software, and independent development is clearly marked as such, thus guaranteeing that the original version remains certifiably pristine. That last point is further helped along by the requirement that the unmodified version of the file (or a link to it) is distributed with the modified version.

The LPPL was introduced after LATEX2.09 — the last version of LATEX that was developed by Lamport in 1985 — got into dozens of incompatible versions. This required a change from the top, specifying what license the LATEX distribution should adopt. Many developers profess that they do not care much about license terms, however, since the official distribution is important for the success of their program, and it requires that they adopt a specific license, then that gave importance to the license terms chosen. The change in license terms was gradual, as there are still controversies by developers who do not want to get rid of some restrictions in their package that are not compatible with the LPPL. For example, Gaulle, who developed the "French" package to specify French typographic rules, didn't want to make his contribution open-source, which led to it being gradually put to the sidelines.

There have been some controversy with some maintainers of the Debian Linux distribution, who are the self-proclaimed guardians of the open-source philosophy; the LPPL was deemed too restrictive to be open-source, as changes were subject to too many conditions. The argument back was that this was an adequate license for projects such as TEX that support a document exchange

format, and where compatibility and standardization is key; the LPPL is designed to protect the common good that results from being able to exchange documents and be sure the receiver will see the exact same document as the one intended. TEX is not only an open-source software project, it is also a standard for linearized mathematics, and as such warrants special protection.

A.3.9 TEX newsgroups

The first TEX newsgroup emerged on the Usenet in the early 1990s and was called `comp.text.tex`, while other national groups emerged in France and Germany in the middle to late 90s. Such groups provide for support and discussion about TEX features and programs, and were until the easy set up of websites a favored way to put users and developers into contact. They also provide for a good way to follow the evolution of the practical concerns of the TEX users community.

The level of support given by the TEX community must be put into relation with the number of users of that software. A study by Lakhami and von Hippel (2000) on the users support provided through newsgroups for Apache users claims that the postings to Apache newsgroups may die off after a while, but the postings to `comp.text.tex` are still pretty lively, and an interesting quantitative exercise was to chart the postings to that newsgroup per year.

The graph that is shown in the text adds newsgroups postings from different countries as they emerge through time, since it appears those groups were created for the same purpose as the original `comp.text.tex` did, except in a different language.

This can be used as a proxy for how many people use TEX, and gives some indication of where they come from.

Data for newsgroup usage are available through various means: `http://groups.google.com` goes back to February 90 in the history of posts to `comp.text.tex`, and shows an increase, then stabilization since 1995 at 2000 threads per month. However, other newsgroup for French, German and Hungarian users have appeared.

A Microsoft research program `http://netscan.research.microsoft.com/` provides all possible statistics on newsgroup usage for various TEX newsgroup in different countries, allowing to replicate the Lakhami/Hippel study in a very easy way. It does not go very far back in time (09/99), but provides data such as the most active supporters, the number of answers to a question, the number of supporter vs askers, etc.

Those data, when compared to other newsgroups, and following their evolution in time, do not show any meaningful difference or change in the quality of support given, as the number of replies to posts does not decline over time, and the number of information providers versus information seekers does not go down either.

However, a general remark that can be made for all TEX newsgroups is that, when taken individually, country by country, there seems to be a saturation point, different for each country, with the number of posts and participants in the newsgroup not growing anymore after some years. This could either indicate a limitation in that type of support technology, overload, a limit in capacity on the support side, i.e. a newsgroup may not be able to support more than 60 support request per

day, for various reason — limits in the filtering capabilities of news-reading software, meaning that supporters stop following a newsgroup when it requires too much time just reading it, or/and all TEX users that may possibly be interested in supporting TEX users in a country finally come to be aware of the existence of the newsgroup, and then, can answer only a limited number of people each. This seems to be the best explanation, as each national TEX newsgroups reaches a different saturation point which is directly related to the number of supporters vs. information seekers.

Since the quality of support (questions vs. answers) doesn't decline over time, the most likely reason for the saturation point is that the number of new users that use the newsgroup as a source of information stabilizes over time as does the number of supporters.

That stabilization in the number of new users coming to newsgroups does not necessarily indicate a stabilization in the number of new users of TEX, which could very well keep growing: The Usenet was a major form of exchange of information up to the 90s, when web-based support became easier to organize. There are now many other ways to offer support for software, via forums and more user of web-browser to search for individual projects pages. Additionally, it has become easier to search newsgroup archives, through the use of search engines such as Voila (now `groups.google`), so that less recurring questions may be asked — even though this may be countered by the democratization of the Internet, which leads to less experienced users asking question without prior searching. The base of existing repertoried knowledge increases, and most question have already been answered. Additionally, the program becoming more international, while the original users abandon TEX, leads to questions being asked in more dispersed forums. Finally, less and less development means there are less changes to explain. All this may explain what Lakhami/Hippel report and what we see in TEX data.

A.4 TEX related terms

A.4.1 TEX implementation

All of the standard programs developed by the Stanford TEX project directed by Donald E. Knuth: Metafont, DVItype, GFtoDVI, BibTEX, Tangle, etc., as well as TEX itself. Other programs are also included: DVIcopy, written by Peter Breitenlohner, MetaPost and its utilities (derived from Metafont), by John Hobby, etc.

A.4.2 The Errors of TEX

"The Errors of TEX" is an analysis by Knuth of the various bugs he uncovered in TEX during its development, and that he noted down along with their date, origin and solution. That log spans the whole development of TEX from its first runs in 1978 to its final version in 1990. Knuth divides the discovery of bugs in 3 phases, the first one in 1978 along with the writing of an user manual led to the debugging of the original program, the second one until 1980 saw new applications for the program which was faced with test cases from the AMS, the third phase until 1982 was the time when users began to report bugs independently and TEX82's coding was prepared. The errors

of TEX82 are in a separate log, its phase one corresponding to phase 3 of TEX78 as there already were many developers helping to improve the code as it was being written. Phase 2 began with the publication of the TEXbook in 1983 and lasted until 1986, and led to better integration with TEX adjunct programs to create fonts. Phase 3 marked the end of the development of TEX82 as bugs were reported less and less frequently and Knuth worked toward his goal of having a bug-free program.

The article includes interesting insights into the programming of medium-size software systems and their evolution, notably the importance of the program's new applications by new users in finding new bugs. It is also a good reference to understand in what spirit Knuth developed TEX, trying to provide for ways for them to extend the system by themselves, and writing manuals so as to alleviate the need for formation and explanation.

Ref: "Notes on the errors of TEX", TUGBoat, 10(4), 1989, "The errors of TEX", Chapter 10 of "Literate programming", CSLI Lecture Notes, no. 27., 1992 "The error log of TEX", an updated version of the appendix of the paper "The Errors of TEX", last updated in March 1995. It includes a chronological listing of all changes and bug fixes in TEX (both TEX78 and TEX82) that have been applied since March 1978.

A.4.3 TAOCP

The Art of Computer Programming, a reference book in the computer science that is written by Donald Knuth. TEX was developed to typeset the TAOCP.

A.4.4 (LA)TEX

TEX is a text processing markup language for scientific applications, and LATEX is a macro language built on TEX. When talking about both, the term (LA)TEX will be used. That term also include other versions of TEX and other macro packages.

A.4.5 tex.web

tex.web is what can be defined as the core of TEX and has a specific license which prohibits any change to it. It is written in the WEB language developed by Knuth and is a programming language for typesetting. Most development on TEX was made on plain.tex and not on tex.web.

A.4.6 plain.tex

plain.tex is one particular macro package written by Knuth which allows for a simplified use of TEX. LATEX is another macro package written by Lamport with a view towards making mathematical typesetting more user friendly.

A.4.7 Core LaTeX

A subset of a LaTeX distribution that includes all what is deemed as absolutely necessary to its functioning. It is defined and maintained by the LaTeX3 team, and its file are regrouped in the tex/latex directory on the TeXLive distribution CD.

A.4.8 TRIP tests

The TRIP tests are test routines that are described in "A torture test for TeX", a document written by Knuth. Programs that claim to be implementations of TeX must be able to process those test routines and produce the output contained in that report — the 'TeX' name was filed under copyright by the AMS and its use is under the control of Knuth. The tests were originally devised by Knuth to make sure the changes he made in the program when it was under development affected the output in the desired way. They were devised to test the program in its entirety, and not only those parts that are most used. Note that this testing philosophy contrasts with that of open-source software where bugs are supposedly found by users, and therefore, errors that are of no consequence to the average user are never fixed. Knuth chose another stance; all errors would eventually be found, so that it was better to design the program with no errors from the beginning on so as to avoid painful changes afterwards.

Ref: Knuth D.E (1990) "A torture test for TeX", Version 3 Knuth D.E. (1984) "A Torture Test for TeX" Report No. STAN-CS-84-1027, Stanford University, Department of Computer Science.

A.4.9 The bug report filter system

The bug report filter system is a system established by Knuth to filter bug reports on TeX so as to limit his work on it to what is certified as a bug by a team of experienced TeX developers coordinated by Barbara Beeton, the editor of the TUGBoat and an employee of the AMS. It was set up after the release of TeX82. In that system, bug reports are sent to Beeton who then sends it to the appropriate TeX 'guru' who may contact the original reporter to determine if this is really a bug or a feature of the TeX system. He then reports to Beeton whether to communicate the bug report to Knuth and may propose a solution. Beeton sends a report to Knuth once there are enough bugs to warrant it. Knuth then annotates that report, fixes those bugs that he thinks are bad enough, and sends the annotations back to Beeton who then communicates the outcome to the TeX implementers list and announces the new version of TeX.

A.5 Programming and OS related terms

A.5.1 Programming languages: Pascal, WebPascal, C, Java

Pascal was created by Niklaus Wirth in 1970 and was chosen to program TeX82. It was not standardized, unlike C. C was developed in the 70s by the Bell Laboratories along with the development of the Unix operating system, and was standardized in the middle 1980s. It has now spread beyond

Unix and is the most commonly used language in the computer industry. Like Pascal, it is a low-level non-modular programming language, but an extension of C, C++, allows for object oriented programming. Web2c was developed to translate Knuth's WebPascal code into C so as to facilitate its porting to new platforms and make TEX programming more attractive. Java was chosen by the NTS team to rewrite TEX. It is object-oriented and a simplification of C++. It is designed to be easily portable to any platform, and is especially appropriate for web applications.

A.5.2 WEB and literate programming

The WEB programming language combines a programming language, Pascal, with a document formatting language, TEX to write programs that are not only understandable to machines but also to humans. The program can be processed either by WEAVE to produce a TEX file that is then processed into an human-readable document in DVI, PDF or other format. It can also be processed by TANGLE to produce a Pascal file that can then be compiled into a binary file so as to be read by a computer. The main advantage Knuth gives for such literate programming is that it allows to write better programs because programmers have to think about the audience and must therefore organize their program clearly. The WEB language also facilitated the porting of applications to other systems, a long-term concern of programmers, and made their maintenance easier, because other developers could understand its working by just printings its human readable file. An advantage that was perhaps not a concern to Knuth, was that programmers were better able to understand each other's programs and therefore collaborative work was easier to organise.

Literate programming did not prove a big success, maybe because developers are usually not great writers like Knuth is, but mostly because another system was put into place in the open-source programming community: every one learns a common programming language that they can read almost like their mother's tongue, and explanations and documentation are done in mailing lists and elaborated as a results of discussions and frequently asked questions. However, the ideas behind literate programming and the problems it adressed were those of the later Open-Source community.

Ref: "The WEB System of Structured Documentation" and "Literate Programming", The Computer Journal, 1984, both articles by Knuth.

A.5.3 Monolithic, modular and object oriented programming

Usually, people start learning programming by writing small and simple programs consisting only of one main program. Here "main program" stands for a sequence of commands or statements that modify data that is global throughout the whole program. Monolithic programming allows for faster programs as they are more tightly integrated. TEX was first though of in a modular fashion, but Knuth introduced many interdependencies between parts of the program so as to make it faster; this was important at the time due to the computers' memory and processing power limits.

With modular programming procedures of a common functionality are grouped together into separate modules. A program therefore no longer consists of only one single part. A program is

modular if any of its part can be changed without requiring changes in other parts of the program, while it is monolithic if parts are interdependent. Open-Source programming relies on modularity to ease the coordination of independent teams who are interested in different parts of the program.

Object-oriented programming is an evolution of modular programming whereby the parts of the program are made even more independent of each other and can work by themselves. Java and C++ are object oriented programming languages.

A.5.4 Macros and primitives

TEX is usually supplied with additional information in the form of separate input files or information preloaded with TEX: such files are called macro packages. Macro packages provide information assigning values to the TEX primitives, or define control sequences, i.e. logical combinations of TEX primitives or of other control sequences that together perform frequently used formatting functions.

A.5.5 The free/open source licenses: GPL, BSD, Debian guidelines, FSF

The Debian Free Software guidelines are spelled out at `http://www.debian.org/social_contract#guidelines`, the GPL or 'copyleft' is explained at `http://www.gnu.org/copyleft/copyleft.html` while an example of a BSD license is exposed at `http://www.debian.org/misc/bsd.license`. The Debian project is a GNU/Linux distribution that adheres to Eric S. Raymond's concept of open-source, and his open source initiative (`http://opensource.org/`). The open-source initiative supports the use of the BSD license, which allows for the integration of open-source code into commercial software. On the other hand, Richard Stallman and his Free Software foundation (`http://www.gnu.org/`) recommend the use of the GPL which forbids such commercial use of open-source software. Commercial software companies support the use of the BSD in open-source products, while the GPL is the most popular license for open-source software. The opposition between the two is part philosophical, and part due to a conflict of personalities and of different open source developers' communities. Most of the TEX programs fall under the BSD subcategory, but some of them are GPL (Web2c for example).

A.6 Typesetting and electronic publishing

Common parlance defines 'typography' as the design of typefaces, and 'typesetting' as the design of a page's layout.

> Photocomposition preceded digital typesetting (start 1950-60, end 1975-85) The first photocomposition devices made their debuts as early as 1944, but didn't really catch on until the early 1950s. Typeface masters for photocomposition are on film; the characters are projected onto photo-sensitive paper. Lenses are used to adjust the size of the image, scaling the type to the desired size. In some senses this technology was an "improvement," allowing new freedoms, such as overlapping characters.

However, it also pretty much eliminated optical scaling, because in the rush to convert fonts to the newformat, usually only one design was used, which was directly scaled to the desired size. This is how the need for digital typesetting arose, brought about by the higher definitions allowed by new printer machines. Digital typesetting started in the years 1973-83. The earliest computer-based typesetters were a hybrid between the above-mentioned photocomposition machines and later pure digital output. They each had their own command language for communicating with output devices. Although these machines had advantages, they also had problems. None of these early command languages handled graphics well, and they all had their own formats for fonts. However, some of these devices were still in service as of 1995, for use in production environments that require more speed and less flexibility (phone books, newspapers, flight schedules, etc.).

In the late 1980s PostScript gradually emerged as the de facto standard for digital typesetting. This was due to a variety of reasons, including its inclusion in the Apple Laserwriter printer and its powerful graphics handling. When combined with the Macintosh (the first widely used computer with a what-you-see-is-what-you-get display) and PageMaker (the first desktop publishing program), the seeds were all sown for the current dominance of computer-based typesetting.

Most high-end typesetting still involves printing to film, and then making printing plates from the film. However, the increasing use of high-resolution printers (600-1200 dots per inch) makes the use of actual printing presses unnecessary for some jobs. And the next step for press printing is the elimination of film altogether, as is done by a few special systems today, in which the computer can directly create printing plates.

Today, although PostScript predominates, there are a variety of competing page description languages (PostScript, HP PCL, etc.), font formats (Postscript Type One and Multiple Master, Truetype and Truetype GX) computer hardware platforms (Mac, Windows, etc.) and desktop publishing and graphics programs. Digital typesetting is commonplace, and photocomposition is at least dying, if not all but dead. Digital typefaces on computer, whether Postscript or some other format, are generally outline typefaces, which may be scaled to any desired size (although optical scaling is still an issue).

There has been considerable economic fallout from all this in typography. Although some digital type design tools are beyond the price range of the “average” user, many are in the same price range as the mid- to high-end graphics and desktop publishing programs. This, combined with the introduction of CD-ROM typeface collections, has moved digital type away from being an expensive, specialized tool, towards becoming a commodity. As a result of both this and the brief photocomposition interregnum, the previously established companies have undergone major shakeups, and even some major vendors, such as American Type Founders, have failed to successfully make the

digital transition, and gone bankrupt instead (although at this time ATF appears to be undergoing a resurrection). More recently, even major digital type foundries have–dare one say foundered?–on the shoals of ubiquitous cheap typefaces (even a licensing deal with Corel Corp seems to have been insufficient to save URW).

Although there is a new accessibility of type design tools for hobbyists and professional graphic artists, the decreasing value of individual typefaces has resulted in a decrease in the number of working type designers per se (both independents and company-employed).

From `http://www.potasky.com/visco/assignments/typotxt/` which has an history of typography and typesetting.

A.6.1 The WYSIWYG controversy

The TEX community generally advocates markup-based document writing instead of Word-like WYSIWYG (What you see is what you get) programs, although there have been open-source efforts to provide a WYSIWYG GUI (Graphical users' interface) for (LA)TEX. However, not all people want to implement the same types of functionalities, and it is very difficult to make all of them wysiwyg. Each user makes a user interface to fit his needs, but not those of others. There is therefore no complete solution, all the more so as TEX is expanding and evolving, and it is not possible to follow all of those changes and make them wysiwyg. Allin Cottrell's "Word Processors: Stupid and Inefficient" (`http://www.ecn.wfu.edu/~cottrell/wp.html`), shows why typesetting a document and writing it should be kept separate processes, thus requiring the writer to include in the document some indications of how he wants it to be typeset. Perhaps more importantly, there is not much demand for wysiwyg on the part of people who could develop such an application, because they are used to using the shortcuts that Emac-type script oriented editors provide.

Open source exceptions include TEXMacs, which was inspired by both TEX and GNU Emacs, and allows to write structured documents via a wysiwyg and user friendly interface, but is a free-standing program which works quite independently of LATEX, preview-LATEX which is an add-on package to LATEX to automatically preview the effect of changes in the LATEX document, and LyX, which is an open-source GUI for LATEX. Then, most commercial applications based on TEX provide a WYSIWYG user interface. However, all those retain the LATEX syntax for structuring the document in the background, so that users are not free as in Word to make inconsistent changes in the document's layout.

A.6.2 The web document formats: html, sgml, xml

HTML is a very high level, content-oriented, markup language; it can specify the content of a document with great precision, but an HTML author as little control over the final appearance of his document on the reader's screen. Both SGML and XML are "meta" markup languages; they are used for defining markup languages. A markup language defined using SGML or XML has a

specific vocabulary (labels for elements and attributes) and a declared syntax (grammar defining the hierarchy and other features). SGML and XML are thus more general than HTML, and TEX can be defined using XML or SGML. The main practical difference between SGML and XML is that most efforts that were devoted to the development of SGML have turned to XML, even though XML is only a subset of SGML, and because XML was adopted as a standard by the W3C, which defines standards on the web. LATEX and TEX can readily be used for formatting/typesetting documents structured with SGML defined languages, while they are not themselves suitable as formal languages for structuring information in the same manner (formal rigor and versatility) as SGML. A lot of work in the TEX community is therefore directed to making it XML compliant; several facilities are being designed and developed to directly support the processing of SGML/XML-encoded documents through LATEX — JadeTEX a SGML to TEX converter by Rahtz and Megginson, ActiveTEX by Fine to make TEX syntax more rigorous, TeXML, xmltex by Carlisle to parse xml into TEX, PassiveTEX by Rahtz to produce XSL style sheets (XSL is a standard W3C stylesheet language), etc...

A.6.3 The document sharing formats: DVI, PDF, PostScript...

PostScript is a page description language introduced by Adobe in 1985. Its main purpose is to provide a convenient language in which to describe images in a device independent manner. This device independence means that the image is described without reference to any specific device features (e.g. printer resolution) so that the same description could be used on any PostScript printer (say, a LaserWriter or a Linotron) without modification. It is therefore very similar in concept to DVI, the main TEX output format, DVI meaning 'DeVice Independent'; A DVI file contains all the information that is needed for printing or previewing except for the actual bitmaps or outlines of fonts, which must be on the viewer's system. This is where the PDF format has a big advantage, as a PDF file does not require the viewer to have those fonts, and this is why the pdfTEX project to output pdf files from TEX input was so important: PDF (Portable Document Format) is a strict subset of PostScript, it is

> "a universal file format that preserves the fonts, images, graphics, and layout of any source document, regardless of the application and platform used to create it."
>
> Adobe information

Its main advantage over DVI is that it doesn't require that the system it is viewed on has the same fonts that the document were created with; Acrobat Viewer, which is available for free, can generate substitutes for any font specified in a PDF document. However, unlike PostScript, it is not programmable, and unlike TEX or HTML, you do not have control over the content once it has been translated in PDF format. PDF is not a page description language, but a page rendition format.

B A chronology

This chronology is focused on the development of TEX and of its users' and developers' community, but also provides reference points in the career of Donald Knuth who initiated TEX, in the evolution of the computer science and industry, programming, typesetting systems and industry, the software industry and the open-source software movement, and finally the Internet and its document standards.

The main sources of the chronology are papers by Nelson Beebe and Phil Taylor that are mentioned in the bibliography and the TEX Stanford archives about the early days of TEX.

1965

The AMS begins its reflection on the use of computer aids for typesetting and publishing (1965-1971) This will end up in a set of requirement for a document preparation system.

1968

Niklaus Wirth develops the Pascal programming language (1968-1973). Donald Knuth joins the Stanford faculty as professor of Computer Science.

1969

D. Ritchie develops the C programming language (1969-1970) The US Defense Advanced Research Project establishes the ARPANET, a precursor to the Internet. Universities are linked through that network

1970

Roff and Script, two typesetting systems, are developed during the 70s, at the Bell Laboratories and IBM respectively. They are proprietary systems. Roff works under Unix and is used to promote that proprietary system.

1971

1972

1973

1974

Donald Knuth receives the ACM Turing Award. He stops submitting papers to the AMS because of its journal's falling typesetting standards.

1975

Kernighan and Chery develop eqn, a roff module to typeset equations (1975-1978)

1976

J. Ossanna develops nroff and troff, extensions of roff.

1977

Donald Knuth is appointed to Stanford's first endowed chair in computer science and takes a sabbatical to write the 'classic' TEX, TEX1.0 and Metafont in Sail, an Algol derivative at Stanford. Metafont is used to create fonts while TEX is used to typeset books, notably The Art of Computer Programming, Knuth's magnus opus. (1977-1978) Donald Knuth develops in parallel the WEB language for literate programming.

1978

Lesh develops bib and refer, roff modules to do bibliographies and automate references. B. Reid develops Scribe, a typesetting system for mathematics at Carnegie Mellon. (1978-1980) Donald Knuth is invited to deliver the prestigious Gibbs lecture at the AMS where he presents the TEX typesetting system and Metafont, a tool to create fonts. Major innovations include its hyphenation and justification algorithm and the design of fonts based on mathematical formulas.

The AMS sends some of its people to Stanford to test Knuth's system and discuss its features. Among those is Barbara Beeton who will went on to be the main editor of the TUG's journal, the TUGBoat, as well as Richard Palais who will be TUG's first president. TEX will displace STI's proprietary typesetting system that was used at the AMS.

1979

Lesh develops tbl to do tables in ROFF. Donald Knuth is awarded the Medal of Science by President Carter. Digital Equipment Corporation and the AMS jointly publish Knuth's book "TEX and Metafont: new directions in typesetting".

1980

The TEX users' group is created as a tool to disseminate the knowledge of TEX. Donald Knuth and Michael Plass's paper on "Breaking Paragraphs into Lines' is published. The Imagen company is created by assistants of Knuth (Luis Pardo, Blair Stewart, Lester Earnest) to develop and commercialize a microprocessor controllers for a Canon medium-end laser printer that will render appropriately TEX's output on paper.

1981

Kernighan develops pic to insert pictures in roff. Charles Bigelow initiates Stanford's program in Digital Typography that will be behind most improvements in Metafont. Students in this program go on to work for Adobe (Carol Twombly), Microsoft (Franklin Liang), Apple(Neenie Billawala) or IBM (Arthur Samuel) on font design, participate in the creation of Framemaker (David Fuchs) or go on to work as independent consultants (David Siegel).

1982

C. Van Wyk develops ideal to include figures in roff. Donald Knuth and his assistants develop the 'final' versions of TEX, TEX2.0, and Metafont in Pascal. Another possible programming languages at the time could have been Fortran but C, which became the standard open-source programming language was not yet out of Bell Labs. The development of TEX is financed by grants from the NSF and the SDF (Systems Development Foundation). Addison-Wesley, a publisher, and the AMS, the American Mathematical Society, provide tools and people to test the system. TEX82 comes about as TEX79 is in transition from being used by about 1,000 people to about 10,000.

1983

Leslie Lamport develops LATEX, a set of macros for TEX, which aims at providing the same functionalities than Scribe for TEX users.(1983-1985) He writes a manual along with it. Michael Spivak writes AMS-TEX for the AMS to adapt of TEX to their typesetting needs.

1984

Oren Patashnik developes BibTEX to do bibliographies in TEX. Adobe Systems develops PostScript, a predecessor to pdf for electronic documentation that was initiated at Xerox's Palo Alto center. DVI is the equivalent standard for viewing documents in TEX, but TEX will have to be able to output documents in the PS and PDF standard as they will become dominant. Addison-Wesley publishes Knuth's "The TEXbook" which becomes the definitive TEX reference.

1985

There are wide changes and re-organization of the typesetting industry due to the introduction of computer typesetting (1985-1990) Leslie Lamport predicts that by 1995 most people will have switched from (LA)TEX to something else. He releases LATEX2.09, his last version of the LATEX macros. Addison-Wesley publishes the first edition of Lamport's reference manual "LATEX, a document preparation system". Microsoft's Word begins to make inroads in TEX's clientele. Lance Carnes commercializes PCTEX, a version of TEX for the PC. The PC introduces the computer to a whole new set of users who are not programmers. Ferguson develops TEX styles and macros for

French at the University of Québec. Richard Stallmann creates the Free Software Foundation to promote the use of the GPL and of free software.

1986

Bently and Kernighan develop grep

1987

1988

1989

Knuth develops a new version of TEX, TEX3.0, and Metafont, that provide for 8-bit encoding that are necessary to handle European scripts. Knuth announces that TEX and Metafont will not be developed any further, and that any changes will be limited to bug fixes. Berner and Lels develop the HTML and HTTP standards at CERN (1989-1991). TEX will later have to adapt to the Internet standards, and its developers will contribute to the SGML project that will define a standard for document typesetting on the Web and is a precursor to XML. First postings to the `comp.text.tex` usenet newsgroup. The activity on that newsgroup rises up to 2000 postings per months presently. The usenet provides support for TEX users until the web provides other tools, such as web pages that answer the most frequently asked questions on the newsgroup. It is also a good way to report bugs and problems with packages. Leslie Lamport announces after the TUG conference in Stanford that the responsibility for the development and maintenance of LATEX is transferred to a team headed by Frank Mittelbach. Mittelbach and Schöpf release the New Font Selection System which allows for easier handling of PostScript fonts in TEX. First release of Word for Windows by Microsoft. A MS-DOS version existed since 1983.

1990

Hobby develops Metapost, an adaptation of Metafont to the Postscript standard. The number of members of the TUG begins its sharp decline from a high of 4000 in 1990 to 1500 in 1997, before recovering slightly afterwards to 2000 members in 2000. The CTAN is created as a central repository of TEX materials, hosted on three servers in the UK, Germany and the US. It is modeled on the UK Aston University archives and will allow for a wider distribution of the TEX in open-source form, beyond the tapes that were sold by the TUG or the commercial distributions. Hans Hagen begins the development of ConTEXt. Spivak's AMS-TEX is ported to LATEX by Mittelbach, Schöpf and Downes.

1991

CERN originates the World Wide Web.

1992

The eTEX project to extend the capabilities of TEX is initiated at a Dante (German user group) meeting.

1993

Andreeson at the NCSA develops the xmosaic web browser. Adobe Systems develops the PDF standard, which is based on Postscript but is not programmable. Jiri Zlatuska hires Karel Skoupy on behalf of the NTS working group to rewrite TEX in Java. The eTEX project works in parallel to provide for 16-bit encoding for TEX. Both are essentially European LUGs initiatives. The NTG Dutch LUG distributes 4AllTEX, a TEX distribution for MS-DOS systems on a CD. This will inspire the TEXLive project to provide TEX distributions for most systems in an easily installable CD, as well as the TEX Directory Structure that will allow to organize a central repository of TEX materials on the Web. The Gutenberg French LUG was also distributing TEX for Windows systems on a CD, WinGut. The CTAN archives in Germany, England and the USA are automatically synchronized.

1994

Work begins on the TEX directory structure ('TDS'). Its latest version 0.9995 is published in 1998, but the working group's guidelines are applied in most TEX distributions. Franck Mittelbach et alii. release LATEX 2_ε, an extension of Lamport's LATEX which latest version was LATEX2.09 (1985). Addison-Wesley publishes the accompanying manual "The LATEX companion" by Goosens, Mittelbach and Samarin. Haralambous and Plaice begin working on Omega and Delta, which are adaptation of TEX and LATEX respectively to typeset complex oriental scripts. First postings to the `fr.comp.text.tex` usenet newsgroup, the French-language equivalent of the `comp.text.tex` newsgroup. The activity rises up to 500 postings per months presently. A Dutch user manual is released with the first stable version of Hans Hagen's ConTEXt.

1995

Development of Webwork, a grading system based on TEX (1995-2000) Sebastian Rahtz is hired by Elsevier, the most important publisher for scientific books, to build an archive in TEX and then help Elsevier switch to the SGML standard. The typesetting industry goes through a new crisis as the Internet permits the outsourcing of basic typesetting functions to developing countries, notably India. The final draft of the TEX directory structure is released. The TEXLive project is launched and will be based on Thomas Esser's Unix distribution, teTEX.

1996

Breitenlohner et alii release eTEX which provides for 16-bit encoding for TEX. It is meant as a stop-gap between TEX3.0 and NTS. Hàn Thế Thành develops pdfTEX to produce pdf files directly

from TEX input. The first edition of TEXLive is released. Future versions will be based on Web2c by Karl Berry and Olaf Weber, Thomas Esser's teTEX.

1997

First postings to the `de.comp.text.tex` usenet newsgroup, the German-language equivalent of the `comp.text.tex` newsgroup. The activity rises up to 2,000 postings per months presently. This is a testimony of the increasing role of German developers in the TEX project, as Dante, the German TEX users' group, comes to rival the TUG in terms of members and revenues.

1998

Karel Skoupy releases the Java version of TEX, NTS. Bruce Perens and Eric Raymond set up the Open-Source Initiative that is a less militant version of the FSF.

1999

A license is adopted for the base LATEX distribution and is written in a separate document. The LATEX Project Public License (LPPL) regularizes the various licensing terms of the packages that are included in the new version of TEXLive.

2000

Carlisle develops XMLTEX The LATEX3 team release the latest version of LATEX 2_ε. The new release of the TEXLive CD is stripped of all non-free software so as to conform to the Debian Free Software guidelines.

2001

Rahtz and Megginson develop JadeTEX, a SGML to TEX converter.

2002

Changes in the LPPL are adopted so as to fit the Debian guidelines. Debian is an open-source distribution of Linux that acts as an unofficial guarantor of the open-source nature of software as it only include software that fit with Eric Raymond's Open Source Initiative definition of open-source.