Frank Mittelbach

Frank Mittelbach has been the leader of the LaTeX Project for many years.

[Interview completed 7 February 2006.]

[Background of this unusual double interview: the *Free Software Magazine* (`http://www.freesoftwaremagazine.com`) and the TeX Users Group (TUG) both like to publish interviews. By chance, Gianluca Pignalberi of *FSM* and Dave Walden of TUG both approached Frank Mittelbach about interviewing him around the same time. Rather than doing two separate interviews, Mittelbach, Pignalberi, and Walden decided on a combined interview in keeping with the mutual interests already shared by *FSM* and TUG.]

Dave Walden, interviewer: Please start by telling us a bit about yourself and how you got involved with LaTeX.

Frank Mittelbach, interviewee: I have lived with my family in Mainz (Germany) since the early eighties, i.e., by now the larger part of my life. Besides my primary hobby (typography), which can effectively be called my second job, I enjoy playing good board games, listening to jazz music, and reading (primarily English literature). Professionally I work for Electronic Data Systems where these days I'm responsible for concepts and implementation for remote monitoring and management of distributed systems and networks.

While I was studying Mathematics and Computer Science at the Gutenberg-University Mainz, I was first introduced to TeX and later LaTeX and, eventually, this got me interested in typesetting and in particular in research on algorithms for automated high quality typesetting.

During my student days in the eighties, a friend brought back a source tape from Stanford University containing something like TeX 1.1 and fascinating news about the quality of that program (back then we did our theses using a typewriter and either hand-pasting symbols or, in case of some sophisticated IBM typewriter, changing the "ball" every couple of seconds). He tried to implement that program on the Multics system we had at the university and in fact succeeded — probably the first if not the only implementation of TeX on this operating system.

In this way I was introduced to TeX and AMSTeX and typed my first paper, achieving beautiful results. The only catch was that back then the Stanford tape only contained Almost Computer Modern fonts in 200 dpi resolution (and no METAFONT) and the only graphical printing device available to us had a resolution of 72 dpi. So the output we got was of the size of the formula in the middle of this paragraph (or bigger) — wonderful to plaster the walls, but not necessarily suitable for handing in your thesis. As a result, my friend finally had to type his diploma thesis in the traditional way, despite his efforts.

$$\sum_{i=1}^{n} x_i$$

Sometime afterwards I was asked by the department to install a commercial TeX

product on our shiny new PCs and to give a series of lectures to students and professors on how to use it. And that distribution came with LaTeX 2.08 and a loose-bound copy of the manual (which later became Leslie Lamport's book on LaTeX). LaTeX compared to plain TeX looked very good to me, but alas, when trying to produce any document with it, it died while loading the "article document style" due to running out of memory on those PCs. So my introduction to LaTeX stopped after I read the manual, and I was forced to develop my own TeX macro package that implemented similar concepts while requiring less memory. A year later it became possible to actually use LaTeX at the department, and we could retire my macro package.

But this initial exercise gave me a good insight into the inner workings and concepts of a system like LaTeX and enabled me later to constructively criticize certain aspects of LaTeX — something that eventually led to Leslie passing on the development and maintenance of LaTeX to me.

Gianluca Pignalberi, interviewer: Many of our readers are familiar with LaTeX, but for those who aren't, can you introduce LaTeX to our readers?

FM: LaTeX is a batch-oriented typesetting system that uses the typesetting engine TeX or one of its variants (ε-TeX, pdfTeX, Omega).

The TeX program itself (developed by Professor Donald Knuth in the early eighties) is a programmable low-level typesetting engine whose concepts and algorithms provide micro-typographic[1] knowledge of highest quality in these days when this knowledge is slowly declining due to the fact that more and more authors are forced to become their own designer and typesetter without proper training. TeX is especially known for its excellent paragraph breaking algorithm and for its math formula typesetting capabilities, both of which are unsurpassed even though the program and its algorithms have been freely available for more than twenty years.

LaTeX is a macro package written for the TeX engine which allows the user to step back from the low-level formatting capabilities of TeX by providing higher-level interfaces that give the author the ability to mark up the text with logical markup rather than procedural markup (e.g., specifying that something is a list or a section, rather than stating that something should be set in a bold typeface with a little space above and below). The actual transformation of a LaTeX source into a typeset document is done with the help of "style sheets" and configuration adjustments that allow even radical changes to the design and layout in a consistent manner without touching or changing the source. (Well, ideally, but see below.)

Historically speaking, LaTeX was largely influenced by a system called Scribe (by Brian Reid). In turn, LaTeX's concept of logical markup was quite influential on HTML and various SGML/XML DTDs, as were its approaches for turning such logical markup into visual representation.

One of the differences between LaTeX and many other similar approaches is that the LaTeX language is in fact a community development: new packages that augment (or modify) LaTeX's markup and typesetting functionalities are constantly appearing, so that these days LaTeX offers typesetting solutions for nearly every subject domain — as diverse as game typesetting (such as chess, go, or crossword puzzles), chemical formulas, or music. Another important difference is that, although LaTeX brought the concept of logical markup to a larger audience, it also provides ways to fine-tune the results (essentially

[1] Micro-typography is concerned with the detailed aspects of type and spacing, e.g., the kerning (shortening or enlarging space) between letters, generation and placement of ligatures, line breaking, etc. In contrast, macro-typography is concerned with larger structures, such as the design of headings, lists, or pages.

providing interfaces to procedural markup), acknowledging the fact that no automated transformation of logical markup into a visual representation is able to automatically resolve all problems produced by the physical restrictions of the output format (e.g., line width or page size). While in certain applications such fine tuning adds no value (like database content publishing where full automation is required), it is crucial for typesetting high quality books and journal articles.

GP: How many people are officially part of the LaTeX Project? And how would you define the "LaTeX project"?

FM: The LaTeX Project Team is a fairly small (slowly changing) group of people who look after the LaTeX kernel and a small number of core packages that provide a stable basis for a huge number of constantly evolving packages and add-ons. Providing and guarding a stable core is (although not necessarily popular with everyone) an important part in keeping LaTeX alive as a language for document exchange. Current and past members of the team include Javier Bezos, Johannes Braams, David Carlisle, Michael Downes, Denys Duchier, Robin Fairbairns, Morten Høgholm, Alan Jeffrey, Thomas Lotze, Chris Rowley, Rainer Schöpf, and Martin Schröder with varying degrees of involvement.

Historically, the project took over maintenance and development of LaTeX 2.09 from Leslie Lamport in 1991. At one time the system was split into several incompatible variants that often prohibited successful processing by LaTeX at one site of documents created by LaTeX from a different site, even though system independence was originally one of the important goals of LaTeX as a documentation language for the scientific community. Another goal for the team was to address apparent deficiencies in the concepts of LaTeX 2.09. The project team addressed both issues in the early nineties with LaTeX 2_ε which provided a stable and consolidated platform that offered further development possibilities outside the kernel code.

Although LaTeX 2_ε already addressed most, if not all, of the deficiencies identified in the first decade of LaTeX 2.09 use, it was originally thought that LaTeX 2_ε would only be an intermediate step towards a LaTeX 3 version. But over time it became clearer and clearer that the remaining open questions could not be adequately resolved within the constraints of: a) TeX as the underlying formatting engine, and b) no changes in the fundamental concepts deployed in LaTeX.

As a result, most of the efforts in the recent years by members of the LaTeX team have gone into research on features desirable for the underlying formatter engines as well as in development of experimental languages and concepts for a designer's interface to typesetting — a level of abstraction that is largely missing from today's LaTeX (which currently often requires TeX programming).

So one definition of the LaTeX project these days would be that it works on providing the foundation for the core concepts and implementation of a new typesetting system that is based upon the good aspects of LaTeX 2_ε (e.g., logical markup, extensibility), but that on the code level is not necessarily focused on providing compatibility. At the document syntax level the situation is clearly different, as reuse of older documents is certainly an important goal. But even there, the main focus will be on clean concepts and as a result compatibility may be restricted in certain cases to providing support for automated conversion.

DW: You have a long history in the world of TeX for collaborative work, e.g., famously with Rainer Schöpf in the early days of LaTeX. You must enjoy working collaboratively. Tell us a bit about your approaches to collaborative work.

FM: I do indeed like to collaborate and over the years worked successfully with many

different people (on various topics and in different subject domains). For me the main value of collaboration is during the development of ideas which, in my experience, are best produced in an open exchange. My mental picture here is a table tennis or similar game which only develops if one directly reacts to whatever your counterpart thinks of and "picks up the ball as played". People who have worked with me know that I like white board drawing sessions (I do need to visualize while I play along) and brainstorming and mind mapping methods.

But I'm also a stickler for details and can spend a lot of energy and effort in actually finishing something (to my own satisfaction) when I consider it worthwhile. Collaboration on that level — after the initial concept and design development work has finished and the nitty gritty detail work starts — normally takes one of two forms: either I restrict myself largely to mentoring and let others work on actual implementations, or I put so much energy into a certain task that it outweighs other people's involvement by a large factor. My base motto here is "Es gibt nichts Gutes, außer man tut es" (free translation: Nothing good will come into existence unless you actually do it) by Erich Kästner which at least in the German language nicely rhymes.

A lot of collaboration necessarily happens via email (due to living in different countries, etc.), but I find it extremely valuable to interrupt this method of working at irregular intervals with face-to-face meetings to flesh out ideas and make them concrete enough to go ahead for a while in semi-isolation with only email and or phone calls as the means of "direct" communication. This also explains why most of the more fundamental work that is associated with Rainer Schöpf's and my names dates from the time when we both studied at the University and had a chance for a more regular exchange of ideas in front of white boards (drinking gallons of tea).

In general I think that Frederick Brooks is right when he argues in *The Mythical Man-Month* [4] that to run a successful software project you need a fairly small and structured team that is responsible for making the final design decisions. Large scale "committee" design only leads to bad results by compromising too much between different factions or by incorporating incompatible design concepts.

GP: TEX is widely considered the best typesetting system, but professional typesetters seem to prefer using commercial, visual software. Why would you advise them to use LATEX instead of another system? Or, conversely, why not?

FM: There is no doubt that TEX has superior qualities in a number of areas compared to other typesetting systems (e.g., paragraph breaking quality, or math formula presentation, etc.). But it was designed as a batch processing program, that is, it does not allow for direct interaction with the user. In WYSIWYG typesetting systems the user can make visual corrections which are then instantaneously reflected, while with TEX (or LATEX) you have to modify your source, and then reprocess and check that your correction produced the desired result.

Especially in the last stage of book production, TEX's tendency for making far reaching changes to achieve high quality can actually be a hindrance rather than an asset (at least if you do not account for it and adjust your working method). For example, due to global optimization in paragraph breaking, a *removal* of a single word in one paragraph will usually result in a complete reflow of the whole paragraph and might in fact make the paragraph even one line *longer* because TEX decided that a slightly looser setting of all lines produces the best possible solution. If this happens when correcting last-minute typos in an otherwise finally formatted document, it can be rather annoying.

Most graphic designers and professional typesetters are used to working visually with

immediate feedback and control, so for them systems like LaTeX appear difficult to handle and they do not see any benefit in this unfamiliar working model.

Whether or not the use of LaTeX would be advisable really depends on the job at hand and cannot be answered without context. In a nutshell I would suggest using LaTeX or a similar system whenever one or more of the following factors play an important role in the job:

- User's preference is to think in logical structures
- Designs that require consistency
- Documents whose designs are not yet fully defined or that need to be presented in several layouts in parallel
- Documents that require high-quality paragraph breaking
- Documents that contain heavy mathematics
- Automatically produced content (e.g., from databases)
- Long material

On the other hand, the following factors move the balance towards using a (good quality!) visually oriented system:

- User's preference is to think in visual structures
- User not at all comfortable working with programming languages (a high-level front end for TeX, e.g., LaTeX or ConTeXt, helps here but . . .)
- Designs that require a lot of visual flexibility rather than consistency (e.g., headings are designed one-off according to nearby objects)
- Designs that require text to flow around arbitrary shapes (TeX is simply not designed for this)
- Designs that change the horizontal measure from column to column
- Short material

What actually tips the balance may differ in different circumstances; for example, in *The LaTeX Companion* [9] with its nearly 1000 in-line examples, it was an enormous plus to be able to redesign example layout without touching the individual examples.

DW: Is there any relationship between TUG, DANTE, or any of the other TeX user groups and the LaTeX Project, and how might the user groups help maintain the viability of TeX and its derivatives?

FM: The relationship between the LaTeX project and TeX user groups can probably be best described as loose and informal. Several project team activities have been supported in one way or the other by a user group (e.g., by providing meeting space at a conference), but most LaTeX team activities have drawn on non-user-group resources such as support from ZDV (the computing laboratory at the Gutenberg University Mainz), royalty payments from *The LaTeX Companion*, and to a small extent from individual user contributions. The biggest joint venture with a user group was probably the development of a model for a set of extended math fonts, where the French user group financed a student for three months to work with me on this topic. The outcome of this work [13] is now finally bearing some fruits as it helped in developing the STIX fonts.

However, the user groups are extremely important to projects such as the LaTeX Project in that they provide a research framework for contacts and face-to-face discussion at conferences and journal publications. This aspect of providing a research framework cannot be underestimated; and, if the user groups would become unable to provide it, it

might eventually result in the death of the community. I certainly enjoy (and I'm sure so do others) the fruitful exchange that is only possible in such a framework.

The question about what the user groups long term can do to sustain the viability of TEX and its derivatives is difficult to answer — I'm unfortunately not sure the user groups themselves will survive in the long run. The role of the user organizations has changed over the last two decades. In the beginning, just getting a TEX system installed was a major effort, and user groups were formed by interested people to help each other and exchange knowledge and ideas, and to support development effort. Back then the role of the user groups was fairly clear and the benefit for each member was immediately visible, e.g., obtaining information otherwise not available, getting help, etc.

Over time, access to a TEX installation and all its accompanying goodies (such as LATEX packages, etc.) has drastically changed — nowadays installations are prepackaged, access to all software is available in large archives, and there is much more documentation available. As a result, the typical (LA)TEX user has no need to understand the underlying mechanisms and isn't (unfortunately) any longer interested in sharing in their development — the users have largely changed from actively participating members of a group of like-minded people excited by the possibility of doing high-quality typesetting to consumers of a "finished product" who get very upset if the product does not do precisely what they want it to do. For this new kind of users, the user groups do not play an important role since, at this point in time, the user groups have no resources to actually help individuals with their problems.

This is somewhat ironic, since it was largely members of the user groups that initiated all the changes that now appear to be leading to the downfall of user groups' accepted mission and reason for existence in the eyes of the average user.

In theory, I think the best way that user groups could help these days would be in the following areas:

- Recruiting and providing the resources that keep the "product" alive and well-maintained
- Providing a suitable forum for the active development community
- Obtaining and managing research funds
- Attracting new users to broaden the base

Unfortunately none of this is easily achievable. It would need an amount of capital (and resources) noticeably beyond what is currently available to the groups, and it is not clear that this — as a charter — will attract enough new members who then could share those costs. After all, to most people the "product" and its support appear to be available free of charge, so today's consumer thinking is: "Why pay a (substantial) recurring membership fee when all that is needed is connecting to the Internet and asking a question on `comp.text.tex` or downloading some software from CTAN? — I don't go to those conferences 'they' go to, so why should I finance 'them'?" It would be necessary to break that thinking and make people understand why the user groups nevertheless are beneficial for them; but unfortunately many people take a free lunch if they can get it, without considering the consequences.

GP: Your project generated a license: the LATEX Project Public License. Thanks to the last modifications to it, LATEX can be considered a real free software. How did this fact improve the diffusion of such a tool?

FM: To be honest I always considered and still consider LATEX *real free software* regardless of the license under which it was distributed in the past or is distributed now. The term "free" has definitely different meanings for different people, and I do not necessarily agree

with the understanding of some people that their freedom to be able to arbitrarily change things without any restrictions should be considered a more important good than the right of others to get what they expect when they use a certain product.

LaTeX is not just a single user product but a language being used for communication of information, and one of the important points here is that it enables processing a document at different sites with identical results, provided that the same version of LaTeX is used. This is a feature a large proportion of the community is relying on, so the original LPPL (LaTeX Project Public License) [3] in a nutshell said: do whatever you like with file X but if you change it (i.e., modify its behavior in the system) change its name to something else, so that people relying on the communication feature of LaTeX will not be affected by your modification.

Technically, this allowed for any modification and any desired change, but it gave the people using LaTeX a conscious choice to apply a changed version to their documents or not. In some cases it would have meant some extra effort for the person doing modifications but on the whole I feel it provided a nice balance between the people who think "free" means their right to change what they like and people who think they have a right to a reliable means of communicating information.

However, some developers in the free software community think that such a simple rule restricts their rights too much (not being able to change things in arbitrary ways, including ways that hide the modification to later users — even if that is not the intention), and so a discussion started about whether or not such a rule makes software non-free — the main obstacle for many being the requirement to change names if you change content. Clearly, this requirement is quite different from those posed by the majority of free software licenses, but then those licenses have been written with quite a different software model in mind (one where the focus lies on individual software components where differences at different sites do not restrict the usability of the software).[2] But since we were not interested in enforcing a name change per se (even though we still think that it mediates nicely between all different needs), I entered a longish discussion with debian-legal and, as a result, we came up with a new license which softened this requirement while still preserving the community need for stability and reliability. In essence I think the new license is better in many parts (and I'm very grateful for some folks from debian-legal helping achieve this), but it is also, perhaps unnecessarily, more complex than it could have been in other parts. In the TeX world, the original license was trying to codify what was standard and accepted behavior, i.e., when you changed or improved a package you called the result something else so that older documents would compile as expected while newer ones could make use of extended or changed features and both could co-exist.

So did the license improve the diffusion of the tools? As far as the TeX world as such is concerned, I would say "no", as even the original license was already simply codifying what most people thought to be a good model for software in the TeX domain. As to the wider world of free software in general, the modifications probably helped people to understand that LaTeX and friends are also "free" software and provided common ground for some understanding that different usage requirements may need somewhat different interpretations of "free" to be useful.

GP: Are you (or were you, or will you be) involved in other free software projects?

FM: The answer to this probably depends on the definition of the terms "project" and "involvement". Many of my interests these days are of a more theoretical nature and will

[2] For a discussion of why we think that something like the GPL is not a good licensing model for free languages, which is one aspect of LaTeX, see [2].

not necessarily directly lead to software or not to software where I will directly participate in implementations; and those projects where I most likely will participate could be labeled under the broad heading of (LA)TEX development. For example, just a couple of weeks ago Hàn Thế Thành (the main developer of pdfTEX), Morten Høgholm, and I spent a productive weekend at my home working on ideas for grid typesetting (which describes designs that are based on an invisible underlying grid restricting the placement (and size) of objects); so, even if I most likely will not participate in actual implementations, there is and will be involvement in projects outside of LATEX. And who knows, as I'm doing completely different work in my professional life, perhaps that too one day will lead to one or another free software package in that area.

DW: You, among others, have written about the need to move beyond the limitations of TEX and suggested improved approaches. In your biography in *The LATEX Companion* [9] you say you want to work at bringing extensions such as Omega and ε-TEX together as a base for an actual LATEX 3. Obviously, you have a track record for accomplishing big, complex, TEXy projects. Do you envision getting involved with something like the ε_{χ}TEX project or starting your own low-level implementation project for an enhanced TEX?

FM: When I wrote "E-TEX: Guidelines for future TEX extensions" [7] in 1990, the time was not yet ripe for improving TEX, and many people actually considered it an affront to Don that I suggested there could be something worth improving in his product (I remember, for example, public musing about strange theories from unknown and obscure German typographers — well, those "strange theories" had been suggested to me by none other than Hermann Zapf, who, though German, may not be precisely called unknown let alone obscure). But be that as it may, what I was challenging in that paper was the typesetting quality at the micro-typography level; but, as TEX was technically so much better than anything else at that time, my challenge was probably premature, and it took nearly a decade until the first real experiments were conducted on that level and moved things ahead in that domain (largely with the development of experimental versions of pdfTEX but also experimental code by others, e.g., Matthias Clasen).

When discussing improvements to TEX one needs to distinguish three largely disjunct areas. First is the area of the programming language and the fact that this language is incomplete and for certain tasks difficult to use (or, as some people state, "a mess"). On that level (without diverting from the fundamental paradigms of TEX), extensions like ε-TEX and to some extent Omega tried to ease the programming task by providing missing primitives that bridge the obvious gaps in the base language. But since such additional functionality was only easing the programmer's life without actually improving the typeset results and the function of (nearly) all the new primitives could be achieved with some extra effort in the base language, we decided to stay away from them in the LATEX development as their use would have resulted in a LATEX version that would then only run on a small fraction of the installations without any practical gain for the user. The LATEX Project together with people from ConTEXt and ε-TEX actually made some effort to produce an enriched syntax definition for ε-TEX [5] that we thought would provide enough benefits to switch to a TEX successor implementing this extended set. Sadly, shortly after this proposal, work on ε-TEX effectively came to a standstill, and so none of this was ever implemented. Nevertheless, something has changed since then: the installed base of ε-TEX-enabled installations did grow beyond critical mass (largely because of pdfTEX which included the ε-TEX extensions), so that some time ago the LATEX Project officially announced that it will base future LATEX versions on this extended set of primitives — and recently started to actually produce code that made use of these extended features

(although so far only outside the kernel code). In essence we never wanted to go away from being able to have LaTeX run "out of the box" on a large base of installed interpreter programs and valued this higher than a potentially easier or better adjusted programming language that nearly nobody could use.[3] So instead of only trying to influence the TeX language by extending it, I and some others in the LaTeX Project also worked from the inside by developing the "Experimental LaTeX programming language" [6]. This was done over several prototypes, the first already done in 1993 or so; and the current version is something we think can be successfully used and we have started to provide the first public packages in this language [1].

The second area is the one dealing with micro-typography issues, e.g., those that I was mainly concerned with when discussing shortcomings of TeX in [7]. In this area my involvement was largely confined to initiating work by others.

The third area is the one that concerns itself with the generally open and unsolved questions of computer typography, e.g., models for representation of logical [8] and visual content material [12]; transformation between logical and visual representation using automated methods that nevertheless provide highest quality according to a defined metric [10] to give some examples. Part of that research is to understand and codify typography rules and to develop concepts and algorithms that can be driven by parameterized rules, e.g., to produce high-quality float placement.

Do I envision starting my own low-level implementation project to improve on TeX? Most certainly not, but I do envision getting (re)involved with the developments currently happening and hope to bring some of those developments together. Whether this will be in a project like ε_{χ}TeX or pdfTeX is largely irrelevant. At this point in time there are still many unresolved questions, and it is still the time for experiments (which may happen in different projects in parallel), but one important goal would be to bring the various developers together to talk to each other about their ideas and the concepts behind the ideas.

GP: You mentioned two operating systems LaTeX was ported to, and we know it runs on several free and non-free (whether commercial or not) OSes. Which kind of OSes and programs do you mainly use? And why?

FM: To be precise, LaTeX is interpreted so it is not software that needs porting to any OS; LaTeX runs everywhere where TeX has been ported to — and TeX to my knowledge has been ported to more or less every operating system ever in existence (with the exception of something like the Palm OS), e.g., I have used it on mainframes, VMS, Unix, Multics, and Windows.

I use both free and commercial operating systems; it largely depends on the environment and the task at hand. At home I run mainly Linux with VMware to access certain programs only available on Windows. On my laptop the situation is reversed: here I run XP native and use Cygwin for a decent command line environment with all the benefits of a good Unix system. My favorite editor is Emacs which I use on nearly every platform. I like to structure things using mind maps and here the only really good program I found is commercial and works only on Windows — it is one of the reasons that these days I use Windows fairly regularly.

In the professional world, where I earn my living, the predominant OS on the desktop

[3] I learned that this is a critical factor when we tried to introduce LaTeX 2_{ε} in 1994 which required the installation of T1 encoded fonts (i.e., fonts containing characters with diacritics). The switch to the new system nearly collapsed because users in the US saw absolutely no benefit in a system that contained all these useless characters only needed by Europeans. Fortunately enough, LaTeX 2_{ε} had other benefits that eventually won over nearly all LaTeX users, but it was a close shave.

is Windows; and in the server world you'll find commercial Unix variants but also a growing number of Linux servers.

What I use depends largely on the task at hand: for some, e.g., project management, the Windows world simply offers the better tools; in others free software (running on commercial or free OSes) provides better quality or features otherwise not available. Examples would be Perl, Apache, CVS, Subversion, and others.

DW: I remember Knuth saying that writing *The TEXbook* led to hundreds of changes in TEX, because he was forced to explain things to the reader, and when he couldn't, he changed the program. You have written a number of books as well as been a major developer. Did your work as an author influence your work as a developer, or vice versa?

FM: I think Don is absolutely right in making such a statement: I think it is extremely valuable to combine the development of software (actually anything) with the task of writing about it in some way. Trying to explain to others the functions and concepts behind a creation helps a lot in finding out whether or not something is going to work in practice. If you can't explain it or if the explanation turns out to be horribly complicated, then there is something fundamentally wrong with your creation and you should return to the drawing board.

Very important here is that one does not stop at simply documenting functions or menu items (though that is a start) but effectively tries to document the usage flow and the reasons why one would do things in one way or the other. Often enough (with free software as well as commercial software) you find only rudimentary documentation that tells you that such and such feature exists but never explains why one would want to use the feature in the first place. That type of documentation, while necessary, will not help in improving your tool (and often enough it turns out that such features only got implemented because they were easy to add without providing any real benefit).

So yes, documenting ideas and work flows has always been an integral part for me of developing and/or improving software, both my own as well as software from others. In *The LATEX Companion* [9], for example, a good proportion of what I describe is software developed by others, and the process of trying to explain how to use this software and finding good usage examples led in many cases to improvements in syntax or features after some discussions with the authors.

Therefore my advice to developers is to always try their hands at documenting their own creations or at least find somebody who does it for them (starting from the initial development!) — and carefully evaluate the findings from this process: it will result in noticeable improvements in the product.

DW: Thank you very much, Frank, for taking the time to do this interview with us. Your insights about TEX, LATEX, and the development and diffusion of complicated systems in a distributed development environment are fascinating.

And thank you, Gianluca, for agreeing to let me share this interview with you. It has been a pleasure to work with you.

GP: Thank you, Frank, for giving the *Free Software Magazine* readers a very well explained essay about an important piece of free software. Moreover, your LPPL explanation clarified some obscure points in a previous article [11]. And thank you, Dave, for giving me the possibility to do a combined interview, which is much more interesting than a "normal" interview.

FM: Thanks Dave and Gianluca for conducting this interview in the way it was done. I enjoyed seeing it unfold, question after question — despite the time it took (my fault)

and the fact that we lived far apart it felt like doing a live interview face to face, which, I think, is the way it should be.

[1] LaTeX3 Project Team. The LaTeX Project CVS Repository. http://www.latex-project.org/cgi-bin/cvsweb/.

[2] LaTeX3 Project Team. Modifying LaTeX. Available with all LaTeX distributions, file modguide, December 1995.

[3] LaTeX3 Project Team. The LaTeX Project Public License (Version 1.3). http://www.latex-project.org/lppl/, December 2003.

[4] Frederick P. Brooks, Jr. *The Mythical Man-Month; Essays on Software Engineering*. Addison-Wesley, Boston, Massachusetts, 2nd edition, 1995.

[5] David Carlisle. Notes on the Oldenburg ε-TeX/LaTeX3/ConTeXt meeting. http://www.latex-project.org/papers/etex-meeting-notes.pdf, 1998.

[6] David Carlisle, Chris Rowley, and Frank Mittelbach. The LaTeX3 Programming Language—a proposed system for TeX macro programming. *TUGboat*, 18(4):303–308, December 1997. http://tug.org/TUGboat/Articles/tb18-4/tb57row1.pdf.

[7] Frank Mittelbach. E-TeX: Guidelines for Future TeX Extensions. *TUGboat*, 11(3):337–345, September 1990. http://tug.org/TUGboat/Articles/tb11-3/tb29mitt.pdf.

[8] Frank Mittelbach. Language Information in Structured Documents: Markup and rendering—Concepts and problems. In *International Symposium on Multilingual Information Processing*, pages 93–104, Tsukuba, Japan, March 1997. Invited paper. Republished in *TUGboat* 18(3):199–205, 1997. http://tug.org/TUGboat/Articles/tb18-3/tb56lang.pdf.

[9] Frank Mittelbach and Michel Goossens. *The LaTeX Companion*. Tools and Techniques for Computer Typesetting. Addison-Wesley, Boston, Massachusetts, 2nd edition, 2004. With Johannes Braams, David Carlisle, and Chris Rowley.

[10] Frank Mittelbach and Chris Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In C. Vanoirbeek and G. Coray, editors, *EP92—Proceedings of Electronic Publishing '92, International Conference on Electronic Publishing, Document Manipulation, and Typography, Swiss Federal Institute of Technology, Lausanne, Switzerland, April 7–10, 1992*, pages 261–273, New York, 1992. Cambridge University Press.

[11] Gianluca Pignalberi. The LaTeX Project Public License. *Free Software Magazine*, (7):52–54, 2005. http://www.freesoftwaremagazine.com/articles/tex_license/.

[12] Chris A. Rowley and Frank Mittelbach. Application-independent representation of multilingual text. In Unicode Consortium, editor, *Europe, Software + the Internet: Going Global with Unicode: Tenth International Unicode Conference, March 10–12, 1997, Mainz, Germany*, San Jose, CA, 1997. The Unicode Consortium. http://www.latex-project.org/papers/unicode5.pdf.

[13] Justin Ziegler. Technical report on math font encoding (version 2). Technical report, LaTeX3 project, June 1994. http://mirror.ctan.org/info/ltx3pub/l3d007.*.