Cahiers GUTenberg

THE STATUS QUO OF THE NTS PROJECT

Hans HAGEN

Cahiers GUTenberg, nº 39-40 (2001), p. 205-220.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_2001___39-40_205_0>

The status quo of the NTS project

Hans HAGEN

pragma@wxs.nl

Note de la rédaction. Le projet NTS a été présenté par Phil TAYLOR lors du précédent congrès GUTenberg à Toulouse [1]. Hans HAGEN, qui a été chargé par DANTE [2] de réaliser un audit de ce projet, donne ici son avis — assez critique — sur la manière dont il a été conduit et propose de réunir les efforts des développeurs de NTS, de Ω et de pdfTEX pour donner à TEX le successeur qu'il mérite [3].

The reason

In the last decade, several initiatives were started in extending TEX The Program. Most closely related to the original is ε-TEX. This program adds some primitives to TEX that provide more control over expansion, extends the range of registers beyond 255, and provides bidirectional typesetting at the paragraph level. The fact that ε-TEX is programmed within the original WEB concept makes it a close relative.

Donald Knuth's main motivation for writing TEX was the need to typeset his own books in the best of typographic traditions. Therefore, it will be no surprise that its typographic engine favours the English script over other, more complicated, scripts. Composed characters and glyphs, advanced ligatures, complicated input encodings, and tightly integrated multi-directional typesetting, are not handled well by TEX, but they are covered by Ω, yet another relative of good old TEX. Ω not only provides an advanced input translation processor, it also extends the range of registers. Opposite to ε-TEX, Ω can handle a large number of math font families. However, it is especially the multilingual capabilities that have given Ω a well deserved position in the family of TEX descendants.

The third major descendant of TEX is pdfTEX. Where ε-TEX demonstrates quite well that TEX can be extended, and Ω gives TEX its place in typesetting non-

1. Philip TAYLOR, Jiří ZLATUŠKA et Karel SKOUPÝ "The NTS project", *Cahier GUTenberg* 35-36, mai 2000, p. 53-78.

2. Groupe germanophone des utilisateurs de TEX, principal bailleur de fonds du projet (environ 43 000 €) ; GUTenberg a contribué à hauteur de 3 000 €.

3. Cet article, déjà paru dans *Die TEXnische Komödie*, nº 1 (2001) p. 36-53, est reproduit avec l'aimable autorisation de DANTE.

western languages, pdfTEX lets TEX survive in the turbulent internet environment. It does so by providing an alternative back-end, which enables TEX users to provide documents that can be distributed, viewed and printed without additional resources; in color, with graphics included, and enhanced with hyperlinks and widgets.

Because pdfTEX can be combined with ε-TEX it can also provide the ε-TEX goodies, but it offers more. pdfTEX extends TEX's paragraph building routines with character protruding (marginal kerning) as well as horizontal font expansion (*hz* optimization). In doing so, pdfTEX ensures that TEX is still quite up to date and ready for the near future.

There are a few more extensions, like those provided by mlTEX, which focuses on 8 bit encodings and mapping, but these extensions are small compared to the ones already mentioned. Being useful for European languages, they are often part of the mainstream TEX distributions, probably without users being aware of it.

So, to summarise the current state of TEX, we can classify the programs developed so far as follows:

– TEX: the stable and bug free ancestor
– ε-TEX: the useful successor
– Ω: the much needed extension
– pdfTEX: the successful descendant

pdfTEX differs from the other two TEX descendants in that it goes a step further in combining more tools into one. This is a logical consequence of the fact that it is a typesetting engine as well as a backend. It has to handle all aspects of fonts, images and resources. It does so by using new code, written within the WEB paradigm, but it also uses existing code, available as precompiled C libraries, while some of its subsystems are written from scratch in C instead of PASCAL.

When TEX was written, PASCAL was one of the favourite structured languages. In order to make TEX portable, Knuth sacrificed some of PASCAL's features and implemented his own memory management. Also, instead of relying on PASCAL data structures, he used his literate programming environment WEB as a wrapper. As a result, extending TEX is possible, but only to a certain extent. The main reason for this is that many data structures are reused and/or overloaded. Another handicap is that many variables have a global nature, so that one should be very careful in manipulating them. TEX is one of the few programs that really benefit from faster machines since the code is highly optimized, but sometimes these optimizations have the nasty side effect that they obscure what the code does. It is no secret that pdfTEX demonstrates quite well that the limits of extending TEX within its current concept are reached.

At the time when ε-TEX took shape, Ω prototypes started to show up, and pdfTEX was not yet invented, there was already a more structured discussion taking place on re-implementing TEX The Program. This re-implementation should be done in such a way that extending TEX would be more easy. This envisaged successor has been flagged as The New Typesetting System, or NTS for short.

For quite some time, the ε-TEX and NTS projects were combined and hosted by the German user group DANTE. Since the start of the project, DANTE has been funding it substantially. This makes the project unique in the TEX world, since the ε-TEX, Ω and pdfTEX projects were not funded at all, or at least not to that extent. Before discussing the NTS project, we will spend some words on the environment where these developments take place.

The environment

Visiting a TEX user group meeting is a special experience. Such meetings often look more like a gathering of family and friends than a conference of experts. This is not to say that the people present are not experts. Actually, they are an interesting mix of highly qualified professionals with many areas of interest. They share the feeling that TEX is special, and by using TEX they can express their knowlegde on paper in the way that they want. Although a minority of them has in-depth typographic knowledge by education, they embody quite some expertise in the, sometimes even dark, areas of high quality automatic typesetting.

Given that everything related to computers evolves fast, the TEX community is rather stable. Many users will stick to using TEX when they are permitted, and even when they are forced to use commercial software in their offices, they keep an eye on TEX. However, open source software is gaining attention and we may consider TEX and friends to be one of the oldest examples of open source. (It is in this respect interesting to observe that TEX distributions are always struggling with the public licences, that somehow do not fit them well. Many TEX distributions depend on stability and consistency and thereby sometimes pose some restrictions, mainly to guarantee their users a working system.)

One of the main drives for using TEX is that it makes one independent, especially if one also uses related or similar free tools. Although there are commercial versions of TEX available, some with quite interesting extensions, the wish to be independent implies that the successor of TEX has to originate from the user community and so far, the extensions mentioned before did so.

As a demonstration that TEX could be extended, Donald Knuth added the `\special` and `\write` primitives. I think that there is much truth in saying that although they can be qualified as 'just' extensions, both mechanisms have given TEX an edge over competitors. A decade after TEX was born, we make documents with lots of graphics, color, and extensive referencing, all of which would not be possible without those primitives.

This demonstrates that what can be regarded as an interesting example of an extension today, tomorrow can prove to be a necessity. Currently, pdfTEX has some extensions that are waiting to be used to the full extent some day in the future.

The number of people that understand enough of programming, typography and user interfacing to extend TEX The Program, is not that large. Therefore, the statement that TEX is extensible is rather an optimistic one. Even if a successor would be implemented using today's technologies, this would not change much. And if some limitations of the good old TEX can be qualified as fundamental shortcomings, this does not automatically mean that replacing them by better alternatives can be achieved in a couple of days programming. For some problems there are no simple solutions, and some of the current limitations are quite natural, given the solution space.

The development of pdfTEX is a good demonstration that, although many people are involved in testing the core program, only a few people are involved in the actual development of the program. Actually, the making of pdfTEX is mainly a one person job, namely Hàn Thế Thanh's. But, this one person can fall back on the experience embodied in the TEX community. Experts in the areas of fonts, images, PDF and macro writing can be consulted and when they see the potential of the extensions, they are willing to participate. The number of experts is small, but their expertise is available whenever needed. Those operating at the cutting edge of what TEX can do want to be involved, and often are involved. Fortunately only one person pulled the car, which meant that right from the start working prototypes were available, bugs were being fixed quite fast, and what was even more important, design decisions were made.

Because TEX has its own DVI output format, the whole suite of related programs (think of DVI viewers and converters and font generators) is rather independent from commercial developments. Because pdfTEX is used also to produce PDF output, it is more dependent on the outside world. It is no secret that Hàn Thế Thanh has spent quite some time in keeping (buggy) viewers happy and figuring out the real PDF specs. One (maybe only philosophical) question we should ask ourselves is if we want to be that dependent. Both alternatives ask their price.

In the last few years we have seen that (finally) the TEX community managed to get a hold of its multitude of files and resources. There is a well defined TEX directory structure and there are some de facto standard distributions with binaries, fonts, macros and more. As a direct result, extensions like ε-TEX, Ω, and pdfTEX are available for everyone who uses TEX and on many platforms.

This also means that the maintainers of those resources (distributions) can ensure that such extensions are being integrated into the current framework of TEX in a natural way. For instance, when Ω is part of a distribution, its unique (re)encoding and font resources are available too. Or, when pdfTEX is on someones system, one can also be sure that the right configuration files are around somewhere. Development of new technologies is integrated into the constant process of updating and distributing TEX.

I already mentioned TEX user groups. They are organized by country or language and many of them have regular meetings and journals. Although the number of members differs from hundreds to thousands, the number of users that attend meetings is often not more than 75-100. A survey by the NTG showed that many members, when asked for the reason to be a member, responded that they are a member out of sympathy. Although many of them do not understand everything that is published, they are happy to be kept informed that there are developments. It shows them that TEX is alive. Of course members also like the regular distribution of CDROMS and the support that mailing lists provide.

So, whereas the large audience wants to be kept informed and is willing to support the TEX community, a small group actively attends meetings where issues like the future of TEX, extending TEX, and writing of macros are discussed. It will be no surprise that this group harbours many of the people that also take part in the developments.

We can summarise the main characteristics of the TEX community as follows:

– the developers want to be involved,
– the maintainers want to be in control,
– the users want to be kept informed, and
– they all want to be independent.

It is in this framework of TEX developments and the TEX community that I will discuss the current state of the NTS project. So far I have been rather general in my remarks, but I will be more explicit from now on. The following observations can therefore be seen as personal ones, and I express my sincere hope that future developments may benefit from them.

The project

I started this article by mentioning a few extensions to TEX of which the NTS project was planned to become one. It has been started in the early nineties, and after some years of discussion the decision has been made to re-implement TEX The Program using a modern programming language and applying today's software technology.

In spite of the fact that the project runs for nearly ten years, it is quite unknown. One reason for this is that for a long time it has been only a mental exercise. Where each of ε-TEX, Ω and pdfTEX at a certain point lead to a real usable product NTS only existed in the minds of a few people. I don't know much about what took place in those early days, but I am told that NTS was discussed by a broad audience, but at the moment when I joined the team, the group of people that took part in it had become rather small.

At a certain point in time the NTS dream became an official project and there are not that many of them in the TEX community. Most efforts are concentrated around a rather active group of developers, and driven by users who see the benefits from those efforts. The community is rather open, and the lines of communications are short. This means that when someone becomes aware of an effort that is of common interest, this knowledge spreads rather fast.

Knowing that TEX has some limitations and that ε-TEX could not solve them all, it should not be a surprise that NTS became the magic successor that was supposed to solve those problems. Its official project state gave it a reputation on forehand. The magic resulted from the fact that for a long time there had been talks of a successor, but no real progress was seen. It is interesting to observe that meanwhile some extensions have been implemented in ε-TEX, Ω and pdfTEX in a quite acceptable way, which proves that demand can lead to solutions quite effectively.

In many user groups, or sometimes between user groups, projects are being launched with ambitious goals. Some of these projects keep rolling while others get stuck in the conceptual phase or merge with other efforts. Most projects in one way or another contribute to the constant developments, if only because their ideas merge with others. None of these projects is really official, and as far as I know, none of them is like the NTS project.

The NTS project, for instance, has an interesting structure. There is a managing director, a project manager, a technical director, about three members and (since recently again) a treasurer. The real work is done by *one* paid programmer. Although undoubtedly the original ideas behind this structure were sound, in practice it does not work out that well. One reason is that this is not a real project in the sense of projects that are being run within institutions or companies. There are no clear roles, and there are no clear functions

amid the structure. The project is not embedded in research, but there have even been suggestions to organise the project as a legal body. Apart from occasional email exchanges, there is no day-to-day communication, no formal responsibilities and there is no planning.

However, there is progress, which is mainly due to the fact that there is a professional programmer involved. Thanks to DANTE, the project was able to hire such a programmer. One of the quoted reasons behind making the conversion work into a paid job was that it would speed up the process. Another reason was that it would lead to a consistent redesign. We can safely agree with the second reason, but right from the start it has proven to be impossible to estimate how much time was needed.

The latter is in itself interesting. Given that TEX is considered to be a well documented program, and given that it is almost bug free, the first very optimistic estimate was that a conversion would take a few months using a rapid prototyping language. This later became more than two years because the prototyping stage was omitted. So far, each intermediate estimate for the moment when the first stage could be finished has been wrong.

This has its (in itself valid) reasons. As I mentioned before, users want to be in control, and part of this control is in using stable tools. And, TEX The Program is as stable as a program can be, both in terms of functionality and in terms of reliability. This is clearly proved by the fact that during the process of re-implementation, no bug has shown up in the original TEX, although there are certainly questionable areas. However, in the process of cleaning up and reaching full compatibility, a real bug in TEX surfaced when processing the TEXbook. [4]

For many users stability means that any future extensions, like NTS, should be able to compile existing documents and macros. For some users, this also means that the result should be 100% compatible, both in terms of DVI output as well as the log file content.

A considerable amount of time has been spent on making the re-implementation 100% TEX compatible. As a side effect, the new code is not as beautiful as it could be due to some strange dependencies resulting from the requirement that also the log files should be identical. However, this also resulted in the new implementation being quite bug free, because the programmer had to test every tiny aspect in order to get exactly the same DVI and log files as TEX does. Full compatibility is only the starting point, and future (extended) versions would be upward compatible in functionality, but not necessarily produce the same output.

4. The bug is related to `\xleaders` and makes the last leading box disappear in an inconsistent way. Karel Skoupý and Bernd Raichle did an in depth analysis of this bug and will report on this.

I will not elaborate on the pros and cons of the conversion, the problems encountered, the joy and frustrations of the programmer, the quality of the code, portability and the performance of the re-implementation. In due time Karel Skoupý, the NTS programmer, will share his insights with us in a more systematic way, as he already did at several user group meetings. However, I think that the project missed a chance to research in a systematic way why it took so long to go from one implementation to another, especially since the language of choice, JAVA, qualified as an highly portable and easy to use language.

In an earlier stage of the project a rapid prototyping language was considered but this option has been rejected in favour of JAVA. Given some negative experience with this language, in terms of sub-optimal performance, lack of portability and an insufficient design (features), Karel has been discussing alternatives with some experts in object oriented programming. It is his strong belief that, given the object oriented design of the current re-implementation, switching to another language is not a real problem.

Because we were dealing with a program that is very well documented — which does not automatically mean that the subject at hand is easy and trivial— it is an interesting question why the re-implementation took so much effort. Since no systematic data has been gathered during the project, we will never know the complete answer to this question.

Another fact that became clear, especially in the final stage of the re-implementation, was that good old TEX runs much faster. The JAVA re-implementation is far more memory hungry and about 30 times slower in processing the TEX book, and thereby much slower when used in large applications. When Knuth wrote TEX, department computers were much slower than today's desktops. So what exactly is slow? Anyhow, when one watches the page numbers appearing so slowly on the screen, one gets a good impression on how precise Knuth must have been in writing code in order not to waste much time waiting. One may argue that speed is not an issue, but evolving macro packages are getting more and more demanding and new features in the typesetting engine will ask for much more processing power.

I already mentioned that ε-TEX, Ω, and pdfTEX have been created by individuals but were developed with the help of users and experts. As a result, these programs are really used. The NTS project on the other hand has had a rather low profile. When the first alpha versions were made available, only a few people did a few tests. One reason for this is that the implementation is uncomfortably slow, is not as portable as the development environment promises, is not yet embedded in the existing file structures, and, most of all, does not offer anything new. I believe that there are also a few more reasons for this isolation on which I will elaborate later.

Some of the ideas behind the original project were to boost TEX into the future by providing a successor with more advanced features, as well as providing means to add a user interface. A third objective was that anyone could take the code and extend the program.

Even if we can envision those more advanced features, these are not goals that are reached fast. There are a few good ideas about areas of extensions. But to say for instance that, given a nice re-implementation, we can build a stable and full functional multi-column mechanism is a gross oversimplification of the problems at hand. Giving TEX a nice user interface is not per definition something that goes hand in hand with its batch processing character. And, how many people really understand the issues that TEX has to deal with to the extent that he or she can extend the program?

People use word processors for everyday tasks, and these programs have become better over time. In typesetting, WYSIWYG page layout programs have become more sophisticated, and some of the features that made TEX famous, like its paragraph builder, have made it into some of those. On the other hand, TEX is one of the few programs which can deal with today's document encoding formats, like for instance XML, in advanced ways. It is also one of a few programs that can handle database output with ease and speed. And, in the math arena it is still the best.

Times are changing, both in terms of demands and usage patterns. The main objective for a TEX successor is to provide better and more flexible general purpose routines to handle any input, typeset any document, in any language. In this respect the NTS project is far more ambitious than its predecessors ε-TEX, Ω and pdfTEX. But while all of these are already available, used and appreciated, the full NTS implementation is still a dream.

The status

One could expect that an effort like NTS would make other developments obsolete. But the opposite can be observed. Even after 20 years of TEX, user group meetings show that TEX is far from being dead. At such meetings, users often demonstrate new applications. They demonstrate specific ε-TEX, Ω or pdfTEX features and demonstrate new and advanced macros. When discussing those features, and possible future extensions, NTS never is part of the discussion.

In spite of being overloaded with official functions, the project team has not managed to get a good and promising reputation. In general, publicity has been handled at a bare minimum. And, even where the project is known, it is not so per definition in the positive sense.

One reason for this is that at a certain moment in time, politics entered the project. I must admit that I am only partially aware of the fine details of the political issues, since much of what I know comes from secondary sources. Surely some of the DANTE internal affairs influenced the project. On the other hand, the generous contributions and positive attitude of past and present DANTE boards towards the NTS project have ensured that at least the first main objective, the TEX re-implementation has been achieved. Unfortunately the project lost some valuable German participants already in its early stage, what in my opinion has damaged the project.

I already pointed out that this project has quite a number of official tasks in its organisation. Since I am participating in more 'projects' than NTS alone, I can safely conclude that this has been contra productive rather than productive. No other project in the TEX world has such a formal structure, no other project has spent so much user groups money, and no project has such a vague reputation as the NTS project. Instead of having a stronghold in the TEX community, this project has isolated itself beyond an acceptable limit.

I want to summarise the previous observations as follows:

– the NTS effort is largely unknown,
– the project is not really managed,
– the re-implementation is not embedded in research,
– the project objectives seem to be out of sync with reality,
– publicity has been handled badly or not at all, and
– the project is too isolated from other developments.

It may be clear that most of the conclusions result from the fact that the project was organised in such a way that the key players in the TEX community were only minimally involved. In this respect, I think that one way or another, the project became a hostage of its own structure. In spite of this, one of the objectives, namely the re-implementation of TEX The Program has been achieved. In the next section I will therefore elaborate on the future of the project as I see it.

The short term objective of the NTS project was to re-implement TEX. At the time of this writing, NTS can process the TEX book. As Karel and I demonstrated at the DANTE October 2000 meeting, there is still a small problem in processing the METAFONT book, and the trip test is passed largely, but not completely. Personally I presented the program with some more complicated situations and apart from a few not so dramatic bugs I am impressed by what Karel has achieved so far.

In the week before DANTE 2001 Karel announced that NTS has reached the beta stage. An important milestone was reached, namely that NTS can operate in the de facto standard TDS (the so called `texmf` tree). From that moment on NTS could be really used as a replacement for traditional TEX.

In the continuous process of debugging, the programmer will also clean up some messy code, improve the performance where possible and document the source to the extent needed for further development. Because the team is very aware of the fact that users expect any TEX to be stable, and will expect the same from a re-implementation, the official release date is left to the programmer.

We can safely assume that in the summer of 2001 the code will be present in the TEX archives and become a part of distributions. At that moment we can start evaluating if the result has been worth the money spent so far. This may be a good place to mention that the main official contributions to the project were from DANTE (85.000 DM), GUTenberg (3.000 €), TUG (5.000 $), CSTUG (20.000 CZK for Karel's expenses), and an unknown donator (5.000 DM), and the NTG (3000 HFL) which means that until now the whole project has consumed over 100.000 DM. The finances were managed by DANTE, and the regular payments to the programmer went through Masaryk University in Brno (Czech Republic). This university also provided Karel with an email account and internet facilities, for which it deserves the team's gratitude.

As a sidenote I want to remark that at DANTE 2001 the membership decided to provide a regular budget for projects related in any kind to TEX, METAFONT, METAPOST and friends. For a couple of years, the NTG has a similar budget for projects. The NTS project has demonstrated the need for such financing. One obstacle has been the requirement to handle transactions in such a way as to fit into the tax regimes of the countries involved. This topic is a good candidate for the agenda of future cross-usergroup board meetings.

So, we can now safely conclude that:
– NTS version zero is there as a beta release, but still being debugged and cleaned up,
– some basic documentation will be provided,
– soon everyone can take the source and go ahead,
– so far the project did cost about 100.000 DM, and that
– thanks to Masaryk University we were able to transfer the money to the programmer.

Especially the fact that there is not much money left, causes the need to look into the future.

In recent publications in the Gutenberg magazine (spring 2000) and the TUG proceedings (fall 2000), some team members have drawn conclusions with regard to the project, its history, status and future. These conclusions were not discussed within the team, so a less informed reader could understand them as the voice of the whole team. Unfortunately, I don't share the views aired in those articles and, if I am right, also some other team members disagree. To

state it clearly, the following section reflects my own thoughts and therefore should not be taken as the views of the whole NTS team.

The future

At a certain moment in time I became involved in discussions with regard to ε-TEX, which at that time were also related to NTS. I must say that those discussions were quite interesting, and each proposal was considered in detail. Some decisions made it into ε-TEX already, other could make in into future versions of ε-TEX, but those that were too complicated were put on the agenda for NTS.

After a while, I became involved in the more ambitious NTS project, first as a reviewer for DANTE, later as a project member with the obligation to report to the DANTE membership about the progress of the project, since reporting had proven to be a weak spot of the project.

I have only been involved in the last stage of the project, a period when not many fundamental discussions were taking place within the team. Nonetheless, I carry pleasant memories of the discussions concerning the design that I had with Karel whenever I was visiting him in Brno. I saw it as my main contribution to make sure that this stage was finished and tried as hard as possible to be of help to him.

So, in the light of my experience, how do I see the future of NTS, or to be more specific, how do I think a TEX successor should be developed? What lessons can be learned from the past, and how should we proceed?

I already remarked that the project is rather isolated from the rest of the TEX community and I see no indication that this will change soon. Given this, and given that I don't regard myself as being a real member of the NTS team any longer, if only because I am not one of the founding members, I feel that my role will be finished as soon as the first official release is there.

The language.

I think that at this stage, the positive conclusion can be drawn that at least there is a working re-implementation, possibly with all the flaws that the language of choice imposed, but a major goal has been reached. This means that we have a pretty good starting point for further development.

At a certain stage in the project, the decision has been made to use the JAVA programming language. Such a decision is not easy, especially since everyone has his or her favourite language. At that time, JAVA was brand new and promising, and the public relations were good.

In every discussion I had so far, this choice has been highly criticized and not without reason. An interesting aspect is that when discussing alternatives, the availability comes up as a criterium. When NTS started the re-implementation, JAVA's future was yet unsure and portability was (and to some extent is) still an issue. Since we cannot foresee the future yet, any choice can be the wrong one.

In the current version of NTS some lines are commented out in order to let the program run on all platforms. In due time Karel will reflect on the re-implementation with respect to the language used and I'm sure that he will discuss how JAVA compares to other languages and how well it suits proper object oriented programming.

I think that in order to succeed, a group of very dedicated people is far more important than the programming language, especially if languages are chosen that compile to the heavily portable C language. It may even be of a certain charm if the language of choice is special, and very well suited for the task. A strong belief in the virtues of a language is equally important to the success as dedication to high quality typesetting. It is my opinion that the project should be directed by those who do the work. This is not to say that there is no need for advisers in any of the areas involved.

The fact that TEX was programmed in WEB and PASCAL did not stop it from becoming available on nearly all platforms. An important aspect of Knuth's efforts was the documentation. Flagged as literate programming, the WEB system stimulates a particular way of programming. Programmers may like it or not, this has its charm, and it has certainly given TEX its place in the history of software development.

One thing that strikes me when people discuss a re-implementation of TEX, the language of choice is a major item. Of course we can wonder why we should re-implement TEX, and if re-implementing NTS is an issue, but at least I want to remark that the people involved in extending TEX should feel comfortable with the language that is used. There have been attempts to rewrite TEX, and I know of at least one other re-implementation project going on, but going from idea to the full concept is not trivial, not to mention the right structuring for future extensions. Current TEX has some flaws, but is nevertheless rather powerful (and often underestimated), so a successor had better be really good in order to succeed.

At TUG 2000 in Oxford, a number of people involved in maintaining and extending TEX were present (among them some well known TEX experts like Hàn Thế Thanh, Karel Skoupý, Fabrice Popineau, John Plaice). Since the descendants of TEX have all reached a more or less mature state, their creators shared their views on the future of TEX with the others experts present. Apart from the shared vision that those developments should converge in the near future,

they all have strong opinions about the languages that are most suitable for a re-implementation. Most people involved in less trivial TEX programming agree on the fact that in order to extend, we need to re-implement. But in what language and architecture is a non-trivial decision.

Functional languages seem to be the first choice, but this choice is more an (challenging) academic one, and it is understood that such languages are not the most stimulating candidates for users who want to extend TEX themselves. After some discussion, the object oriented language EIFFEL emerged, with especially John Plaice considering it to be a good candidate for a re-implementation of Ω.

Although I am completely new to EIFFEL, I cannot deny that reading the specs alone already gives me the good feeling that it suits such a project well. It compares to what I felt when for the first time I read the TEX book, the METAFONT book, the (real) books about Modula, SmallTalk, Lisp and alike.

But is a functional language, or a language with a vision like EIFFEL the best choice in the long run? Here I owe much to Fabrice Popineau for sharing with me his balanced visions on ideal languages versus practical languages (like C++). Whatever the outcome of merging these efforts into the worthy and stable successor will be, I am sure that those talented people will make the right decisions with regards to the tools to use.

The design.

Some time ago Karel and I discussed the viability to implement a successor in layers, like for example an efficient core in a pure imperative object oriented language, a programming layer in a functional language, and on top of that the macro language. Whatever choices will be, the languages to be used should be able to interface to other languages. Especially pdfTEX demonstrates how useful it is to fall back on existing libraries, like those that deal with font embedding, bitmap and PDF inclusion and compression.

So, given that we can organize an enthusiastic group of people who want to spend time and effort on a TEX successor, and given that we have a reasonable starting point in the well organised TEX re-implementation called NTS, there is a good chance that in the near future a real successor will be created.

At the TUG 2000 conference as well as preceding conferences the basis for a cooperation has already been laid. But, we are talking of another project, with another name, this time properly embedded in the TEX community, and (again) carried by the user groups. Given the complexity of the typographic problems at hand, this should not be a naive effort to come up with a collection of a thousand classes for everyone to extend, but a stable, flexible, and still

extendable program, that can carry on the tradition started by TEX for another 20 years. As said, the existing extensions combined with the NTS redesign of TEX, provide a pretty good starting point.

Whatever course the developments take, the results should be highly usable, (intermediate) distributions have to be stable, and the system should be open for future extensions. Of course it should also solve our most persistent typographic problems.

The environment.

Another interesting development is that at TUG 2000 in Oxford, Karel was offered the opportunity to join the ETH in Zurich. There can be no doubt that a project like NTS or its successor will benefit from the possibility to embed it in proper research. We will learn more about those options when Karel has moved to Zurich (around the summer of 2001).

A result of a closer cooperation with the developers of TEX's multi-lingual follower Ω might also mean that developments will be related to the fundamental research that will follow the next release of Ω (this was presented at TUG 2000).

Apart from the fact that the (new) project could benefit from more fundamental research, an academic environment also gives access to all kinds of resources. Given that for developers such environments can be inspiring in themselves, this will enlarge the chance of success.

The organisation.

One thing that can be learned from the current NTS project is how not to organise a project in the TEX community. The ε-TEX, Ω and pdfTEX projects demonstrate clearly how a successor can be developed successfully, while the NTS project demonstrates the contrary. And, at a much higher cost.

At TUG 2000, I have participated in discussions between the developers of pdfTEX and Ω and experienced programmers and users from the TEX community. To some extent, these discussions were a continuation of discussions at previous user group meetings and from email exchange.

For me, it is always a great experience to see how people share their ideas about future TEX's, the languages of choice, and the possibilities to integrate ideas. It demonstrates the real power of the TEX community when it comes to combining efforts. It also shows the way in which the next stage in developing a successful successor should take place.

One of the leading mottos of the NTS project is that "anyone can take the source and go forward". Given that the current team —except the programmer— is not functioning in optima forma and seems to be unable to keep up its promises, this seems to be the right moment to take it at the word and start a new project.

Informal discussions at user group meetings have also demonstrated that it is quite possible to organise those who play a role in developments into a new team. I would not suggest this if I were convinced that the current team could be reorganised. Unfortunately there is too much historic ballast involved to guarantee success. Therefore I think that as soon as NTS version zero is released, the moment has come to start a new thread in the development of the successor. We need a fresh restart, run in such a way that user groups are involved in the proper way. We cannot do without a team, but apart from a group of people who can represent their user group, we also need dedicated teams for research, development and testing.

Let's do it.

The current NTS team has managed to re-implement TEX in an object oriented way, so in a sense it has accomplished its main objective. It is my strong belief that in order to achieve the more ambitious goals, a new team of enthusiastic and active people is needed. During the last couple of years I have received enough signals that such people are there waiting to get going.

At Bachotek 2001 as well as EuroTEX2001 there will be NTS related sessions. Especially the (on forehand memorable) Bachotek meeting will provide the right ambiance to make such a fresh start. There, in the woods along the lake, team members Jerzy Ludwichowski and Karel Skoupý will present NTS in its full glory and invite us to discuss the future. I hope that you will be there too.

I want to express my thanks to Jerzy Ludwichowski, Karel Skoupý, and Volker Schaa for proofreading this article, improving the English and providing suggestions. Don't confuse my opinions with theirs.