16 years of ConTEXt

Hans Hagen

1 Introduction

When Karl Berry asked me to wrap up something for the 100th issue of *TUGboat* I didn't hesitate too long to agree. Of course you then end up with the dilemma of what to write down and what to omit, but it's worth a try.

When you're asked to look back it is sort of unavoidable to also look forward. In this article I will reflect on some of ConTEXt's history and spend a few words on its future. First I will try to describe the landscape in which we have ended up.

2 Perceptions

After being present for some 16 years, about half the lifespan of the TEX community and its *TUGboat*, there has been enough published about ConTEXt to give the reader at least an impression of what it is about. However, to some it might (still) be confusing, especially because for a long time in TEX publicity, 'TEX' was viewed as nearly equivalent to LATEX. In itself this is somewhat troublesome, because TEX is a generic system but such is the situation. On the other hand, nowadays banners for conferences, cartoons and other promotional material mention multiple engines, MetaPost, LATEX and ConTEXt, fonts and more, so the landscape is definitely more varied.

Over the past decades I have run into descriptions of ConTEXt that are somewhat curious and they give a good impression of what ConTEXt users run into when they have to defend their choice.

- "It is a package that can be loaded in LATEX." This perception is natural for LATEX users as packages are part of their concept. On the other hand, for ConTEXt users, the package concept is alien as we have an integrated system. A quick look at the way ConTEXt is embedded in the TEX directory structure will learn that it does not relate to LATEX and I'm sure that a simple test will show that loading it as a style will fail.
- "It is a variant of plain TEX and has similar concepts, capabilities and limitations." For sure there are a couple of commands that have the same name and similar functionality but there it stops. We used to load plain TEX as a base because it provides some infrastructure, but even there most was overloaded. On the other hand, we feel that when a user reads *The TEXbook*, he or she should not get an error on each command that is tried, especially not math.
- "It is meant for and depends on pdfTEX." Actually, we started out using DVI and when we switched to using outline fonts we used DVIPSONE. Other DVI backends were also supported, but when pdfTEX came around it was pretty convenient to have all functionality in one program. Maybe because we were involved in pdfTEX development, or maybe because ConTEXt always supported the latest PDF tricks, this myth arose, but all functionality was always available for all backends.
- "It is best used for presentations." It is a fact that at user group meetings, the presentation aspect was quite present, if only because I like making new styles as part of preparing a talk. However, it is just a matter of styling. On the other hand, it has drawn some users to ConTEXt.
- "I don't see you using math and you're discussing features that I never needed, so why use TEX at all?" This comment was given after I gave a presentation about ConTEXt doing MathML where I explained that I liked content markup more than presentational markup.
- "I've been present at talks but only recently realized that you were talking about an integrated macro package that is independent of other packages." This kind of remark is of course an indication that I'd forgotten to explain something. It also says something about TEX marketing in general.
- Some comments are baffling, like "I saw you talking to Frank. Isn't that competition?" As far as I know there is no competition, maybe different audiences at most. The community is large enough for multiple solutions. And most of all, we don't sell anything and I always try to keep my own (commercial) work separated from presenting ConTEXt.
- "We don't need this, we're satisfied with the way we do it now." Remarks like that occasionally come up when someone presents something new. I don't care too much about it myself because in most cases I know that I can achieve more with TEX than the person making such a remark, but it can be very demotivating for those doing their first presentation.

I'm sure that ConTEXt is not alone in getting such comments. I remember that there have been discussions about embedding the PDF backend into the TEX engine and strong argument for keeping it separated. Especially when arguments come up like "We should keep TEX as it is", which boils down to "We should not change Don's work", it gets nasty. It

is a fact that the DVI backend is just an example of a backend and there can be many others. The same is true for the `\pdfsomecommand` extensions: deep down they use whatsits and these are definitely meant for extensions. Any argument for using specials exclusively is wrong as specials themselves are an extension.

Right from the start Don Knuth made clear that extending TEX and friends was part of creating solutions. When Volker Schaa and I presented the Latin Modern fonts to Don Knuth, one of his first comments was "Why didn't you fix the mu?". He could not do it himself because Computer Modern is as frozen as pure TEX, but that does not mean that it cannot be done elsewhere! It's interesting to hear users defend a status quo while those they think they are defending definitely keep moving on. I'm sure that I don't speak for myself alone when I say that Don Knuth is one of the main reasons why I've chosen the TEX route and keep following it. It makes TEX and its community special in many ways.

The reason for mentioning this is that when you are part of the flow of developments around TEX, you also have to deal with conservatism. In itself this is understandable as TEX is a tool that you will use forever once you've typeset more than a few documents with it. And there is a nice aspect worth mentioning here: as pure TEX will always be there, and as derived engines are as closely as possible downward compatible, you can stick to 25-year-old methods and customs as long as you keep your macros and fonts around! No one is forcing you to update or switch to another engine or use another macro package. And you can still produce the same perfect output as years ago. The best proof of that is the author of TEX himself, and you can bet that he knows pretty well what developments happen around TEX and friends.

In the following story I will mention some of the design decisions and reasons for going the ConTEXt route. I will use two qualifications there: ConTEXt MkII, the version meant for pdfTEX and XETEX, and ConTEXt MkVI, the new version for LuaTEX. (More about these in the last section.) The MkVI variant is a complete rewrite and as part of the process I threw away lots of code that I had always before considered to be final. Code that I spent weeks or more perfecting, and that evolved along with getting more experienced in the macro language, code that has been optimized to the max, code that I got emotionally attached to because I know for what occasion I wrote it. It gets frozen into MkII and is never used again by myself, but it can be run forever anyway. That's what TEX is about: accumulating experiences. In a few weeks I will travel to BachoTEX again. There, among TEX friends, it will be clear once more that we're still moving forward, that a 30-year-old TEX is not yet ready for retirement, even if some of its first time users are getting close to that.

3 Running into TEX

I always take 1996 as the year that ConTEXt showed up in public. That year sits between the two first international TEX conferences that I attended: EuroTEX 1995 in Arnhem and TUG 1997 in San Francisco. That means that this year ConTEXt is about 16 years old and as a consequence only half of all *TUGboat* publications can contain articles that refer to it.

We started using TEX a bit earlier. I still remember the bookshelves in the store where I first saw volumes A to E and because at that time I was programming in Pascal and Modula the content looked quite familiar. However, as I was neither involved in typesetting nor a mathematician it did look intriguing. Nevertheless I bought *The TEXbook*, and reading something without the ability to run the related program is somewhat special. In successive years, whenever I picked up the *The TEXbook* I was able to understand more of the neat tricks described in there.

4 The first experiments

The real reason for using TEX came when I was involved in a project where we had to get some advanced math on paper. The customer used a special typewriter for this but I remembered TEX and considered it a better tool for the job. We bought a copy of the program from Addison Wesley and got ourselves a decent printer only to find out that our customer preferred the typewriter method over typesetting.

This didn't stop us, and we decided to use TEX for typesetting our own reports and the courses that we developed along with experts in the field. I did an inventory of alternatives but they were either too expensive or closed (and obsolete within years after that moment) so in retrospect the choice for TEX was not that bad. Using TEX at that time definitely made our budgets for hardware rise: faster computers, larger disks, better and faster printers, suitable connections between them, etc.[1]

There is one thing that keeps coming back when I think about those times: we were acting in complete

[1] Before free versions of TEX came to desktops we had to actually buy TEX and friends. Interestingly I'm quite sure that it still accumulates to the largest amount of money we ever spent on software, but competing systems ran into five digit numbers so it was no big deal at that time.

isolation. There was no Internet the way there is now, and when it came our way, using it as a resource was no option with slow modems. We had no email and were unaware of user groups. The university that I had attended, and especially our department had top of the line equipment but TEX was simply unknown. We used ASCII terminals and I had written a formatter for the mainframe that helped making reports and could paginate documents: poor man's styling, pseudo-floats, tables of contents. I think that I still have the source somewhere. However, no one ever bothered to tell the students that there were formatters already. And so, when we moved on with our business we were quite unaware of the fact that something like TEX was part of a bigger whole: the TEX community.

5 Personal usage

In fact this is also the reason why the first steps towards a macro package was made. The floppies that we bought carried LATEX but the rendered output was so totally un-Dutch that I had to change files that I didn't understand at all. Of course some localization had to happen as well and when we bought an update I had to do all that again. After a while I figured out how to wrap code and overload macros in a more permanent way. Itemizations were the first to be wrapped as we used lots of them and the fact that they got numbered automatically saved us a lot of time.

Because we were involved in writing course material, we had workflows that boiled down to investigating learning objectives, writing proposals, collecting and editing content, and eventually delivering a set of related materials. It is therefore no surprise that after a while we had a bunch of tools that helped us to do that efficiently. It was only around that time that we ourselves actually profited from a TEX-based workflow. We had our own editor[2] that provided project support based on parsing structure, syntax highlighting, as well as a decent edit–view cycle.

In my job I could chair a session, drive home, sit down and wrap up the progress in a document highlighting the most recent changes and the participants would have a print on their desk next morning. The time spent on writing new macros was nicely compensated by efficiency.

We're speaking of the beginning of the nineties now. We already had dropped LATEX after a few documents and via the more easily configurable LAMS-TEX moved on to INRSTEX which was even more configurable. The fact that these variants never caught on is somewhat sad, as it indicates that in spite of TEX being so flexible only a few macro packages are available.[3] Around 1995 we had a decent shell around INRSTEX and much code was our own. I didn't understand at all what alignments were all about, so for tables we used Wichura's TABLE package and as output routines were also beyond me, we stuck to the INRSTEX page builder for quite a while. We called the beast pragmaTEX simply because we needed a name and it didn't occur to us that anybody else could be interested.

6 A larger audience

It was around that time that I became aware of user groups and we also joined the Internet. Because we had to do a lot of chemical typesetting, in particular courses for molding latex and plastic, I had written a macro set for typesetting chemical structure formulas for my colleague (who coincidentally had a background in chemistry). As there was interest for this from Germany, represented by Tobias Burnus, it was the first piece of code that went public and because we used macros with Dutch names, I had to come up with a multilingual interface. Tobias was the first international user of ConTEXt.

In the meantime I had become a member of the NTG as well as of TUG. Around that time the 4TEX project was active and it carried the first version of the renamed macro set: ConTEXt. It is at that time that Taco Hoekwater and I started teaming up our TEX efforts.

We started publishing in MAPS and *TUGboat* and after being introduced to the Polish and German user groups also in their journals. So, around 2000 we were better aware of what was happening in the larger community.

At some point I had ordered copies of *TUGboat*s but I have to admit that at that time most of it simply made no sense to me so I never really read that backlog, although at some moment I did read all Don's articles. It might be fun actually going back in time once I retire from writing macros. But the fact that there were journals at least gave me a sound feeling that there was an active community. I do realize that much of what I write down myself will not make sense either to readers who are not at that moment dealing with such issues. But at least I hope that by skimming them a user will get the impression that there is an active crowd out there and that TEX keeps going.

[2] The editor was called texedit and was written in Modula, while its follow-up, called texwork, was written in Perl/TK.

[3] Of course TEX is not unique in this: why should billions of users use only a few operating systems, editors, drawing programs or whatever?

7 How ConTEXt evolved

For this reflective article, I spent some time hunting up the past before actually sitting down to write ... here we go. The first version of ConTEXt was just a few files. There was some distinction between support macros and those providing a more abstract interface to typesetting. Right from the start consistency was part of the game:

- there were define, setup, and start-stop mechanisms
- keywords and uniform values were used for consistent control
- layout definitions were separated from content
- there were projects, products, components and environments
- the syntax was such that highlighting in an editor could be done consistently
- we had support for section numbering, descriptions and of course items
- content could be reused (selectively) and no data or definition was keyed in more than once (buffers, blocks, etc.)

As a consequence of the structure, it was relatively easy to provide multiple user interfaces. We started out with Dutch, English (the first translation was by Sebastian Rahtz), and German (by Tobias Burnus). Because Petr Sojka gave students the opportunity to do TEX-related projects, Czech followed soon (David Antos). Currently we have a few more user interfaces with Persian being the latest.

There is an interesting technical note to make here. Because ConTEXt is keyword-driven and uses inheritance all over the place it put some burden on memory. Just in time we got huge emTEX and I think in general it cannot be underestimated what impact its availability had: it permitted running a decent set of macros on relatively powerless personal computers. Nevertheless, we ran out of string space especially but since the hash was large, we could store keys and values in macros. This was not only space-efficient but also faster than having them as strings in the source. It is because of this property that we could relatively easily provide multiple interfaces. Already in an early stage a more abstract description in XML format of the interface was added to the distribution, which means that one can easily create syntax highlighting files for editors, create helpers and include descriptions in manuals.

Right from the start I didn't want users to even think about the fact that a TEX job is in most cases a multipass activity: tables of contents, references, indexes and multipass optimization means that unless the situation didn't change, an extra run is needed. It is for this reason that a ConTEXt run always is managed by a wrapper. When I realized that ConTEXt was used on multiple platforms I converted the Modula version of texexec into Perl and later into Ruby. The latest version of ConTEXt uses a wrapper written in Lua.[4] I think that the fact that ConTEXt came with a command line utility to drive it for quite a while set it apart. It also created some myths, such as ConTEXt being dependent on this or that language. Another myth was that ConTEXt is just a little more than plain TEX, which probably was a side effect of the fact that we kept most of the plain commands around as a bonus.

It was long after using LATEX that I understood that one of its design decisions was that one should write styles by patching and overloading existing code. In ConTEXt it has always been a starting point that control over layout is driven by configuration and not by programming. If something special is needed, there are hooks. For instance, for a very special section title users can hook in macros. Ideally a user will not need to use the macro language, unless for instance he or she wants some special section header or title page, but even then, using for instance layers can hide a lot of gory details.

I think that switching from one to the other macro package is complicated by the fact that there are such fundamental differences, even if they provide similar functionality (if only because publications have so much appearance in common). My impression is that where LATEX draws users because they want (for instance) to submit a paper in a standard format, the ConTEXt users come to TEX because they want to control their document layout. The popularity of for instance MetaPost among ConTEXt users is an indication that they like to add some personal touch and want to go beyond pure text.

As a consequence of consistency ConTEXt is a monolithic system. However, special areas are dealt with in modules, and they themselves are also monolithic: chemistry, MathML, presentations, etc. For a long time being such a big system had some consequence for runtime or at least loading time. Nowadays this is less an issue and with the latest and greatest MkIV we even seem to get the job done faster, in spite of MkIV supporting Unicode and OpenType. Of course it helps that after all these years I know how to avoid bottlenecks and optimize TEX code.

As I'm somewhat handicapped by the fact that in order to understand something very well I need to

[4] One can argue that this is a drawback but the fact that we use TEX as a Lua interpreter means that there are no dependencies.

write it myself, I have probably wasted much time by (re)inventing wheels. On the other hand, finding your own solution for problems that one deals with can be very rewarding. A nice side effect is that after a while you can 'think' in the language at hand and know intuitively if and how something can be solved. I must say that TEX never let me down but with LuaTEX I can sometimes reimplement solutions in a fraction of the time I would have needed with the pure TEX way.

8 Fonts and encodings

When we started with TEX a matrix printer was used but soon we decided to buy a decent laser printer (duplex). It was a real surprise to see that the Computer Modern Fonts were not that bold. Our first really large printer was an OCE office printer that was normally sold to universities: it was marketed as a native DVI printer. However, when we tried to get it running, we quickly ran into problems. By the time the machine was delivered it had become a PostScript printer for which we had to use some special driver software. Its successor has already been serving us for over a decade and is still hard to beat. I think that the ability to print the typeset result properly was definitely a reason to stick to TEX. The same is true for displays: using TEX with its font related capabilities is much more fun with more pixels.

At the time of the switch to OCE printers we still used bitmaps (and were not even aware of PostScript). First of all, we needed to tweak some parameters in the generation of bitmap fonts. Then we ran into caching problems due to the fact that each DVI file relates id's differently to fonts. It took the support people some time to figure that out and it tricked me into writing a DVI parser in Lisp (after all, I wanted to try that language at least once in my lifetime). We decided to switch to the Y&Y previewer and PostScript backend combined with outline fonts, a decision that we never regretted. It was also the first time that I really had to get into fonts, especially because they used the texnansi encoding and not the usual 7-bit TEX encoding. It must be said: that encoding never let us down. Of course when more language support was added, also more encodings had to be supported. Support for languages is part of the core so users don't have to load specific code and font loading had to fit into that approach.

In traditional ConTEXt the user can mix all kind of fonts and input encodings in one document. The user can also mix collections of fonts and have several math font setups in parallel and can have different math encodings active at the same time. For instance the Lucida fonts had a different setup than Computer Modern. The pattern files that TEX uses for hyphenation are closely related to font encodings. In ConTEXt for languages that demand different font encodings in the same document we therefore load patterns in several encodings as well.[5] Because we mostly used commercial fonts as part of MkII we provide some tools to generate the right font metrics and manipulate patterns.

The reason for mentioning all this is that a font subsystem in a TEX macro package always looks quite complex: it has to deal with math and due to the 8-bit limitations of traditional TEX this automatically leads to code that is not always easy to understand, especially because it has to suit limitations in memory, be efficient in usage and behave flexibly with respect to weird fonts. In MkIV we're using Unicode, OpenType, and wide Type 1 fonts so much of the complexity is gone. However, font features introduce new complexities if only because they can be buggy or because users want to go beyond what fonts provide.

As a side note here, I want to mention the font projects. The real reason why Volker Schaa and I took the first initiative for such a project (currently Jerzy Ludwichowski is leading the project team) is that we came to the conclusion that it made no sense at all that macro packages were complicated by the fact that for instance in order to get guillemets in French, one has to load fonts in an encoding most suitable for Polish just for these glyphs. Because in ConTEXt by default all languages are loaded and no additional special language packages are needed, this was quite noticeable in the code. What started out as a normalization of Computer Modern into Latin Modern Type 1 fonts and later OpenType variants, moved on to the Gyre collection and currently is focusing on OpenType math fonts (all substantially funded by TEX user groups). The ConTEXt users were the first to adopt these fonts, not only because they were more or less enforced upon them, but also because the beta releases of those fonts are part of the so called ConTEXt-minimals distribution, a subset of TEX Live.

9 Interactivity

The abovementioned Y&Y previewer supported hyperlinks and we used that in our workflow. We even used it in projects where large and complex documents had to be related, like the quality assurance

[5] It was one of the reasons why we moved on to patterns in UTF encoding so that users were free to choose whatever encoding they liked most. Nowadays UTF encoded patterns are standard in TEX distributions.

manuals fashionable at that time. As a result, by the time that PDF showed up, we already had the whole machinery in place to support Adobe Acrobat's interactive features. At that time PDF had two different audiences: prepress (printing) and online viewing and we were able to provide our contact at Adobe with advanced examples in the second category.

Unfortunately, with a few exceptions, none of our customers were much interested in that kind of documents. I even remember a case where the IT department of a very large organization refused to install Acrobat Reader on their network so we ended up with products being distributed on floppies using a dedicated (ASCII) hypertext viewer that we built ourselves.[6] The few projects that we used it for were also extremes: hundreds of interlinked highly interactive documents with hundreds of thousands of links. Those were the times that one would leave the machine running all night so that in the morning there was some result to look at. At that time we set up the first publishing-on-demand workflows, using either TEX input or XML.

One of the more interesting projects where interactivity came in handy was a project where we had to identify lots of learning objectives (for some 3000 courses) that needed to be categorized in a special way so that it became possible to determine overlap. With TEX we generated cross-linked dictionaries with normalized descriptors as well as documents describing the courses. It was a typical example of TEX doing a lot of work behind the screens.

Personally I use the interactive features mostly in presentations and writing a (new) style for an upcoming presentation is often the first step in the preparation. In my opinion the content and the form somehow have to match and of course one has to avoid coming up with the same style every time.[7] Maybe ebooks will provide a new opportunity, given that they get better quality screens. After all, it's a pretty trivial and brain-dead activity to produce ebooks with TEX.

10 XML

There is some magic that surrounds XML, and it is often part of a hype. I can waste pages on stories about structure and non-structure and abuse of XML and friends, but I guess it's not worth spending too much energy on it. After all, the challenges can be interesting and often the solutions come right on time, although I admit that there is some bias to using tricks you've just implemented.

Currently most of what we do involves XML one way or the other which is a consequence of the fact that ConTEXt can process it directly. As TEX is rather related to math typesetting we supported MathML as soon as it came around, and although we use it on projects, I must say that most publishers don't really care about it.

Apart from the fact that angle brackets look cool, advanced reuse of content seldom happens. This is no real surprise in a time where the content changes so fast or even becomes obsolete so that reuse is no option anyway. On the other hand, we manage some workflows for publishers that need to keep the same (school) method around for more than a decade, if only because once a school starts using it, you have to support it for some five years after the first year. In that respect it's hard to find a system that, after some initial investments, can stay around for so long and still provide occasional updates as dirt cheap as a TEX can. Unfortunately this way of thinking is often not present at publishers and the support industry happily sells them pages (each time) instead of workflows (once set up the price per page is close to zero). It does not help that (driven by investors) publishers often look at short term profits and accept paying a similar amount each year instead of paying a bit more upfront to save money later.

Maybe I'm exaggerating a bit but most projects that we run involve someone with vision on the other end of the table. Some of our customers take real risks by introducing solutions that go against the flow of time. The simple fact that TEX-based systems somehow guarantee a constant result (and at least some result) makes that succeed. Already several times we were surprised by the fact that by using TEX a solution could be provided where all previous attempts so far had failed: "This is the first automated publishing project that actually works." This might come as a surprise for TEXies who see such automation daily.

We also support companies that use ConTEXt as part of a workflow and the nice thing about that is that you then deal with experts who know how to run, update and integrate TEX. Of course specific code written for such customers finally ends up somewhere in ConTEXt so that maintenance is guaranteed.

11 Design

In the beginning we used TEX mostly in products where we were responsible for the result: no one really cared how it looked like in the end (read: no money could be spent on typesetting) so it was just

[6] In the beginning even the reader cost money so it is no surprise that it took a while before PDF took off.

[7] This is especially true when I know that Volker Schaa, one of my benchmarks in the TEX community, will be present.

an added value and we had complete freedom in design. For the last decennium we have dealt only with the rendering of whatever input we get (often XML) that no one else can render conforming to the specs of the customer. We implement the styles we need and set up workflows that can run for ages unattended. Of course we do use TEX for everything we need to get on paper, so there's also the personal fun aspect.

In that respect there is an interesting shift in usage of ConTEXt: for a long time we ourselves were the ones that drove new functionality, but nowadays it's regular TEX users that request specific extensions. So, where for us an efficient XML machinery is relevant, for users high-end typesetting in their specific field can be the focus. Of course we can do what is asked because most functionality has already been there for a while and often extending boils down to adding a key and a few lines of code. It is my impression that ConTEXt users really like to come up with a personal touch to their documents' look and feel, so fun is definitely part of the game.

Currently there are interesting developments related to the Oriental TEX project which in turn trigger critical edition support and more modern follow-ups on that kind of typesetting. Currently Thomas Schmitz is taking the lead in this area and I expect interesting challenges in the next few years.

12 The upgrade

A couple of years into this millennium I ran into a rather neat scripting language called Lua. This language is used as extension language in the SciTE editor that I use most of the time and after a while I wondered how it would be to have something like that available in TEX. As I don't touch the engine myself I asked Hartmut Henkel to patch pdfTEX into LuaTEX and after some experiments it didn't take much to convince Taco to join in: the LuaTEX project was born.

While initially just a little bit of access to some registers as well as a way to print back data to the TEX input was provided, we quickly started opening up the whole machinery. Once the potential became clear it didn't take much before the decision was made to make a special version of ConTEXt for LuaTEX. It was at the second ConTEXt user meeting that those present already agreed that it made much sense to freeze the ConTEXt for pdfTEX and XETEX and focus development on the next version for LuaTEX.

Although I have used pdfTEX for a long time (and was also actively involved in the development) I must admit that already for some years I only run it when a user reports a problem. In that respect we have already crossed the point of no return with ConTEXt. Since I never used XETEX myself, support for that engine is limited to additional font support in the MkII code and I know of some users using it, if only because they needed Unicode and OpenType while waiting for MkIV to reach a more stable state. Of course we will forever support the older engines with MkII.

Let me stress that the LuaTEX project is not about extending TEX but about opening up. Of course there are some extensions, for instance in the math engine as we need to support OpenType math, but the fundamentals are unchanged. Hard coding more solutions into the core engine makes no sense to me. First of all it's quite convenient to use Lua for that, but most of all it saves endless discussions and makes maintenance easier.

I would like to stress that the fact that most users having already switched to this version helped a lot. I'm pretty sure that the beta version is more popular than the regular (current) version. This is not only a side effect of active development, but also of the fact that the so-called minimals are quite popular. The original minimals were our self-contained, ConTEXt-only subset of TEX Live that we also put on the website, but at some point Mojca Miklavec and friends adopted it and already for some years it is the de facto reference implementation that can easily be synchronized to your personal workstation. Another important factor in the support chain is the Wiki, also known as the ConTEXt garden, an initiative by Patrick Gundlach. One of its spin-offs is Taco's additional TEX Live package repository. The minimals and garden also play an important role in providing up-to-date binaries of LuaTEX and MetaPost.

A for me quite interesting experience was that a few years ago on the ConTEXt list some users showed up who know the ConTEXt source pretty well. For a long time only Taco and I touched the code, apart from language related issues, where users sent us corrections of label translations. Most noticeable is Wolfgang Schuster. Not only is he posting many solutions on the list and writing nice modules in a style that perfectly matches the code base, but he's also amazingly able to nail down problems and I can integrate his patches without checking. Another developer worth mentioning is Aditya Mahajan. It's great to have someone in the team who knows math so well and his website `http://randomdeterminism.wordpress.com` is worth visiting. I could and should mention more, like Luigi Scarso, who is always exploring the frontiers of what is possible, or Thomas Schmitz, who not only makes beautiful presentations but also is a great tester. And

of course Willi Egger, the master of layout, composition and binding. And we are lucky to be surrounded by specialists on fonts and PDF standardization.

13 The future

So, what is the current state of ConTEXt? As we now have a complete split of the code base between traditional ConTEXt (MkII) and the new version (MkIV) we can go further in upgrading. Although one of the objectives is to be as compatible as possible, we can try to get rid of some inconsistencies and remove mechanisms that make no sense in a Unicode age. Some parts are rewritten in a more modern and flexible way and there are cases that more Lua code is used than TEX code (although of course at the Lua end we also use TEX core functionality). Also, all the tools that come with ConTEXt have been migrated to Lua. Eventually the code base will be completely redone.

In addition to coding in TEX a user can code in Lua using a user interface similar to the one in TEX, so if you know the ConTEXt commands, you can also use Lua and create so-called ConTEXt Lua Documents. At the TEX end we go a step further. Apart from some upgraded interface-related macros, for instance we have a better although somewhat less efficient inheritance model, we also support some extensions to the macro coding, like more extensive namespace support and named parameters. Files using these features are classified as MkVI. This numbering scheme is not a ratio scale — although one can argue that MkIV is twice as good as MkII, the difference between MkIV and MkVI is mostly cosmetic. It is an interval scale, so MkVI is definitely a bit better than MkIV. So for the moment let's qualify it as a nominal interval scale of numbering, one that also works out quite well in file names.

Some of the adventurous module writers (like Aditya and Wolfgang) have adopted this strategy and provide useful input to the directions to choose. It must be noted that at the time of this writing it is because of the active participation of Aditya, Luigi, Mojca, Peter, Taco, Thomas, Willi, Wolfgang and whomever I forget to mention that we can undertake such a major rewrite. On the agenda is a rewrite of code not yet scrutinized, of output routines (including various multi-column support) additional (rewritten or extended) support for tables, better access to the internal document model, an extension of the multiple stream model, maybe some CSS and DOM support, and whatever else comes up. Eventually most code will be in MkVI format. As we proceed, for sure there will be articles about it in this journal.

Of course I should mention my colleague Ton Otten, who has always been very supportive and patient with whatever I came up with. He is responsible for convincing potential customers to follow our TEX route to solutions and often he is the first to suffer from updates that for sure come with bugs. Without him we would not be where we are now.

That leaves me mentioning one person who has always been extremely supportive and open to new developments: Karl Berry. Without him you would not be reading this and I would not even have considered wrapping this up.

◇ Hans Hagen
http://pragma-ade.com