The Key to Successful Support: Knowing Your TEX and LATEX Users*

Anita Z. Hoover

University of Delaware
002A Smith Hall
Newark, DE 19716
anita@brahms.udel.edu

Abstract

The primary emphasis of this paper is to address the issues related to supporting TEX and LATEX. One essential ingredient to successfully supporting any package is that you must know your users. In the case of TEX and LATEX, this is especially true, because the user base can be so diverse. This paper will focus on support strategies that address different types of users and what you can do as a TEX and LATEX support person to adopt these strategies in your organization.

Keywords: user support

1 Introduction

When I started working at the University of Delaware, one of my primary responsibilities was to build a sound structure to support TEX and LATEX. Well, five years have passed, and I can say that a good foundation for support is in place but not without a lot of learning and understanding. I started this job never having used TEX or LATEX. Shortly after getting involved, I realized that this was not going to be a short-term project, but one that would last forever. What I mean by this is that from week to week, I would learn of a new macro or style file, new previewer, new printer driver, new utility, new user and kept wondering how I was possibly going to stay above water and offer good support in such a changing environment.

In the preface of the *TEXbook*, I think that Knuth [6] summed up the transition one goes through when learning a powerful tool like TEX. I think his description can be taken a step further in defining the role of a support person—how he or she needs to grow and change in order to provide good support.

> ... When you first try to use TEX, you'll find that some parts of it are very easy, while other things will take some getting used to. A day or so later, after you have successfully typeset a few pages, you'll be a different person; the concepts that used to bother you will now seem natural, and you'll be able to picture the final results in your mind before it comes out of the machine. But you'll probably run into challenges of a different kind. After another week your perspective will change again, and you'll grow in yet another way; and so on. As years go by, you might become involved with many different kinds of typesetting; and you'll find that your usage of TEX will keep changing as your experience builds. That's the way it is with any powerful tool: There's always more to learn, and there are always better ways to do what you've done before. At every stage in the development you'll want a slightly different sort of manual. You may even want to write one yourself. ...

I think it is important to remember that you will always have users at these different stages of learning and, as such, the type of support you provide will also vary depending on the level of the user. It is also important to keep in mind that you will always have new users that will invariably require special attention when beginning to use TEX. Remember that you were once there yourself, and it is sometimes easy to forget and overlook the level of explanation required for new users. You will find that the time spent with these beginners will pay off in the long term because these users will become the support network in your organization, solving the basic questions within their own department.

With this in mind, I feel there are three areas of support that need to be addressed

*Presented at EuroTEX '92, September 14–18, Prague, Czechoslovakia.

1. Users
   What kind of users are using TEX and LATEX?
2. Style files and macros
   What particular style files and/or macros are necessary to fulfill the user's needs?
3. Tools
   What tools are necessary to provide a good working environment for using TEX and LATEX.

2 Users

Through my experiences, I can break down the different types of users into three groups:
1. Graduate students
2. Professors or professional staff who are writing books and/or journal articles
3. Technical secretaries

Our largest audience for all three groups listed above is in the "sciences." As a result, most of the users had already discovered the beauty of TEX and LATEX. They were convinced of its ability to offer high quality mathematics and overall typographic quality required for their documents.

However, the level and type of support required is quite different for each of these groups.

2.1 Graduate Students

Most graduate students familiarize themselves with the basics of TEX or LATEX, depending on which package what they want to use to write their thesis or dissertation. They usually purchase or have available to them a copy of the appropriate manual, *The TEXbook* [6] or *LATEX: Users Guide and Reference Manual* [8]. They also purchase a *Thesis Manual* [9] from the Office of Graduate Studies explaining the required specifications.

Based on the above analysis, I found that the best support for graduate students was to
1. develop a style file for LATEX and a set of macros for TEX that would meet the requirements of the Office of Graduate Studies at the University of Delaware. The style file and macros are called UDThesis.
2. write a document with examples that explain how to use the UDThesis style file for LATEX [1] and macros for TEX [2].
3. develop a short course (two hours) based on the documentation that would primarily be hands on. The course would focus on the creation of a basic front and body of a thesis or dissertation.

This method has proven to be very effective [5]. Even though graduate students are usually only around for several years, their experiences help considerably with support because they help new graduate students and encourage them to read the documentation and attend the short course. We also work very closely with the Office of Graduate Studies to ensure that our macros meet their specifications, and they too inform us of any problems that arise to determine if the macros are incorrect or the student made a mistake. This is a tremendous help believe it or not, as we have found problems with the macros that must have been there for months, but students just would get around them by using correction fluid or cutting and pasting in changes.

In general, you will begin to see that good support just doesn't come from one person (even if technically you are the only designated support person). It is crucial to be able to communicate and work with others in your organization to provide a successful support network.

2.2 Professors and Professional Staff

Again, this particular group of users is usually familiar with the basics of TEX and LATEX. They also purchase or have available to them a copy of the appropriate manual. However, I have found that two levels of support are necessary for this group based on whether or not they were provided with style files and/or macros to meet the requirements of the publisher [5]. I classify them into two groups:
1. User-defined macros
   The user must define macros to design the document to meet a publisher's specifications and then submit a final printed copy.
2. Publisher-defined macros
   The publisher supplies macros that meet the publication specifications. The user uses these macros to design the document and then either submits a final printed copy or sends in the TEX/LATEX file.

In terms of support, group two is relatively self-sufficient provided the documentation is written well and examples are included. In most cases, the only support required is interpretation of the publisher's terminology and documentation.

Group one, on the other hand, can sometimes be a real challenge to support. Although most of these users are familiar with TEX/LATEX, they are not familiar enough with modifying or creating style files or macros to follow the specifications of a publisher. Currently, I need to work with each user and make the necessary changes. This is the most time-consuming activity as a support person. However, if the information is shared with the user, then more than likely he or she can use this information for future documents. I feel that two manuals are desperately needed: (1) "How to modify LATEX style files" and (2) "How to set up a book/journal specification with TEX." I understand that the first of these two manuals is soon to arrive on the scene.

In both groups, you will find that the key to minimizing support lies in the ability of the author and publisher to communicate. Much time is wasted on issues concerning design and layout. The author says they are unhappy with a particular format, the author asks you to make the changes, the publisher sends back edited copies that change it back to the original specifications, and on and on. This process can get quite frustrating for the author, publisher and support person. Try to encourage the user to make these issues as clear as possible up front. This kind of support will be more valuable than getting into a design war.

2.3 Technical Secretaries

This particular group of users is not usually familiar with TEX and LATEX. More importantly, they are not accustomed to teaching themselves by using a manual. I have found that many times the problems are not even specific to TEX/LATEX, but more the basic unfamiliarity with the background in which the paper was written, Greek symbols, mathematical expressions, etc. These problems introduce an entirely different level of complexity based on support. Many of these types of problems were stated clearly by Kubek [7]:

> . . . TEX use seems to be largely confined to the technical professionals with degrees in the fields listed above, who have taught themselves with only the *TEXbook* to guide them (and wouldn't have a clue as to how to teach anyone the program who doesn't speak *at least* three programming languages). For those of us who had to look up the word "algorithm" several times in the dictionary and still wouldn't recognize one even if it asked us to dance, a major dilemma arose when faced with the prospect of learning TEX "by the book." Not only is the *TEXbook* oriented toward programming, but a lot of the commands don't even look like they're in English!

Based on similar experiences, I have found the following to be most successful in supporting this group of users:

1. Provide two short courses (three hours each) that teach the basics of TEX/LATEX. This course is mostly hands-on, where a particular formatting technique is discussed, and the participants in the class type in an example using the information they just learned. They process the file using TEX/LATEX and preview it to see their results.
2. Provide exact examples (as many as possible) that cover commonly asked questions like, "I need to change my margins," "I need to change my equation numbering," "I want a full page figure or table," etc.
3. Time permitting, take advantage of some of the common word processor features available to interface with TEX and LATEX and provide some short cuts for typing and error detection. For example, designing keyboard layouts and macros for WordPerfect [4].
4. Encourage users to share their experiences and examples with each other to solve their problems.

Because of the different levels of learning, you need to keep in mind that nothing is obvious and at the same time, give the user the benefit of the doubt. Sometimes there is a fine line between the two, but as you spend time with your users, you will begin to know them and what particular method of support works best for them.

3 Style Files and Macros

What makes supporting TEX and LATEX even possible is the fact that so many people have contributed thousands of style files, macros, printer drivers, previewers, etc. One person can not begin to solve every problem without the help of others who have developed expertise in particular areas and who are willing to share their solutions.

My concept of support is based heavily on relying on others to solve common problems. I don't believe in re-inventing the wheel! I would rather spend the time looking around the network searching for a solution to a problem, than immediately trying to write a style file or macro to do the job. In most cases, I find the solution to my problem and a few additional helpful hints along the way.

In order to take advantage of these resources, you need to be familiar with and have access to network facilities. I feel this is an essential asset for a support person. At least you can provide a link to the outside world for your users. I'm sure, in part, the reason we have such a good support structure at the University of Delaware for TEX and LATEX is based on our ability to access the network. We can inquire and receive an immediate response. This would not necessarily be the case for those who do not have access to the network. However, at least having access to this information is better than re-inventing the wheel.

I would like to mention a few network resources I have found invaluable in providing support for TEX and LATEX.

- **Archie**
  Archie provides an electronic directory for locating software, documents, and archives on the Internet and was conceived and written by Alan Emtage and Peter Deutsch of the McGill University Computing Centre. Archie maintains a database of numerous FTP sites throughout the Internet. Each site is updated approximately once a month, ensuring that listings are kept up to date. Archie also provides a large collection of descriptions of public domain software packages and documents.

  From the "Supplementary TEX Information" file:
  The Archie server contains file listings for hundreds of ftp sites. Telnet to `quiche.cs.mcgill.ca(132.206.2.3)` and type `archie` when asked to log in. Then type `help`.

Check with your local systems administrator about an Archie server nearest to you or write to archie-l@cs.mcgill.ca with questions about Archie itself.

- **TEX Newsgroup**
  comp.text.tex is a newsgroup available via the USENET network news article and utility files system. This particular newsgroup focuses on all forms of TEX and allows for interactive question and answer discussions.
  If you mail questions and/or responses to INFO-TeX@SHSU.edu, then they are cross posted to the comp.text.tex newsgroup.
- **FAQ (Frequently Asked Questions)**
  Posted monthly by Bobby Bodenheimer (bobby@dry.mu.caltech.edu) is the "Frequently Asked Questions" about TEX. This file currently contains answers to 34 common questions asked about TEX and LATEX. Following is the question and answer to obtaining a copy of this information.

  How can I get a copy of this article?
  You're reading it aren't you? SAVE it :-). Seriously, though, this article is posted monthly to comp.text.tex and cross-posted to news.answers. It is therefore archived at any site that archives news.answers. news.answers is archived on pit-manager.mit.edu (18.72.1.58), and this article is available there via anonymous ftp in the directory ./pub/usenet/news.answers/tex-faq. If you do not have anonymous ftp, send an e-mail message containing the lines SENDME FAQ. to fileserv@shsu.edu (fileserv@shsu.bitnet). Another way to retrieve it via email is through the mailserver at pit-manager: send a message containing the lines help and index to mail-server@pit-manager.mit.edu for information on how to obtain it.

  Other news.answers/FAQ **archives are:** cnam.cnam.fr
  (192.33.159.6) in the anonymous ftp directory /pub/FAQ; ftp.uu.net (137.39.1.2 or 192.48.96.2) in the anonymous ftp directory /pub/usenet (also available via mail server requests to netlib@uunet.uu.net, or via uunet's 1-900 anonymous UUCP phone number); and archive.cs.ruu.nl (131.211.80.5) in the anonymous ftp directory NEWS.ANSWERS (also accessible via mail server requests to mail-server@cs.ruu.nl). Many of the archives mentioned in question 22 also maintain current versions of this article.
- **FTP Sites, Major Archive Sites, Mail Servers, and Mailing Lists**
  Posted monthly by Guoying Chen (chenguo@lab.ultra.nyu.edu) is the "Supplementary TEX Information" file. This is the so-called "Supplement to the Frequently Asked Questions" file; it is intended to complement Bobby Bodenheimer's "Frequently Asked Questions" file. Note that there is some duplication of material.
  This file is monthly (in early days each month) posted to USENET newsgroup comp.text.tex and cross-posted to news.answers. This file is in several parts. It is reposited in CS.NYU.EDU (128.122.140.24) at ~ftp/pub/tex/ and also in many good archivers, e.g. rusinfo.rus.uni-stuttgart.de at the /soft/tex/documentation/ as the file FAQ.supplement-mmm.yy (mmm-yy is the month and year). In that directory is also the comp.text.tex FAQ.

- **TEX Macro Index**
  Recently released by David M. Jones, is the first edition of the TEX Macro Index. It is available at six different anonymous ftp/or mail server sites.
  A brief description of the TEX Macro Index follows.
  This is an index of TEX macros. Its scope includes all macros that are available via anonymous ftp or mail-server or some similar mechanism. Commercial packages will be included only if a full Index entry is supplied to me by the vendor.
  Since the Index is devoted to macros, fonts and special-purpose programs are mentioned only when they are necessary to explain the purpose of a set of macros.
  Each entry is divided into several fields with the following functions:
  Name: The name of the macro package.
  Description: A short (usually 1-3 line) description of the package.
  Keywords: A list of keywords to facilitate searching for special-purpose macros, as well as to help describe the macros. A glossary of keywords can be found at the end of the file.
  Archives: A list of archives where the package can be found. Whenever known, the primary distribution site is marked with an asterisk.
  Author: The name and address (preferably electronic) of the author of the package.
  Latest Version: The date and/or version number of the latest release of the package.
  Supported: Whether or not the package is supported, that is, whether the author wants to receive bug reports and/or comments on the package.
  See also: A list of other packages with similar features.
  Note: Any additional information that seems pertinent.
  In addition to the list of packages, the Index

also contains a brief list of TEX Archives with descriptions of the services they offer.

How to Retrieve the TEX Macro Index?

1. `archive.cs.ruu.nl` (Netherlands)
   How to get `TeX-index.Z` from the archive at Dept. of Computer Science, Utrecht University:
   NOTE: In the following I have assumed your mail address is `john@highbrow.edu`.
   Of course you must substitute your own address for this. This should be a valid internet or uucp address. For bitnet users `name@host.BITNET` usually works.
   by FTP:
   (please restrict access to weekends or evening/night (i.e. between about 20.00 and 0900 UTC)).
   ftp `archive.cs.ruu.nl (131.211.80.5)` user name: anonymous or ftp password: your own email address (e.g. `john@highbrow.edu`) Don't forget to set binary mode if the file is a `tar/arc/zoo` archive, compressed or in any other way contains binary data. get `TEX/DOC/TeX-index.Z`
   by mail-server:
   send the following message to
   `mail-server@cs.ruu.nl`
   (or: `uunet!mcsun!hp4nl!ruuinf!mail-server`):
   `begin path john@highbrow.edu`
   (PLEASE SUBSTITUTE *YOUR* ADDRESS)
   `send TEX/DOC/TeX-index.Z end`
   NOTE: *** PLEASE USE VALID INTERNET ADDRESSES IF POSSIBLE. DO NOT USE ADDRESSES WITH ! and @ MIXED !!!! BITNETTERS USE USER@HOST.BITNET ***
   The path command can be deleted if we receive a valid address in your message. If this is the first time you use our mail server, we suggest you first issue the request:
   `send HELP`
2. `ftp.th-darmstadt.de` (Germany)
   The TEX Macro Index is available via anonymous ftp from
   `ftp.th-darmstadt.de(130.83.55.75)`
   directory `pub/tex/documentation`
   file `styles-and-macros.Index.Z`
3. `ftp.math.utah.edu` (USA)
   The TEX Macro Index is available via anonymous ftp from `ftp.math.utah.edu (128.110.198.2)` in the file `pub/tex/tex-index`. To retrieve it by e-mail server, send a message to `tuglib@math.utah.edu` with the subject or body
   `"send tex-index from tex"`.
4. `Niord.SHSU.edu` (USA)
   To retrieve the TEX Macro Index in 8 parts suitable for electronic mail handling, include the command: `SENDME TEX-INDEX` in the body of a mail message to `FILESERV@SHSU.BITNET` (`FILESERV@SHSU.edu`).
5. `TeX.ac.uk` (UK)
   The TEX Macro Index is available via anonymous ftp, JANET NIFTP and mail server from the UK TEX Archive, `TeX.ac.uk (134.151.40.18)` in the file `[tex-archive.doc]TeX-index.txt`.
   To retrieve it by mail server, send a message to `TeXserver@tex.ac.uk` containing the following line
   `FILES [tex-archive.doc]TeX-index.txt`
6. `theory.lcs.mit.edu` (USA)
   The TEX Macro Index is available via anonymous ftp and mail server from `theory.lcs.mit.edu (18.52.0.92)` in the file TeX-index in the directory `pub/tex`.
   To retrieve it by mail server, send a message to `archive-server@theory.lcs.mit.edu` containing the following line
   `send tex TeX-index`
   The Index is also available in compressed, zip'ed and zoo'ed format in the files `TeX-index.Z`, `TeX-index.zip` and `TeX-index.zoo`, respectively. Note that if you want to request one of the compressed files by mail server, you'll have to specify a method of ASCII encoding by including one of the following lines in your mail message:
   `encoder btoa`
   `encoder uuencode`
   `encoder rscs`

In early 1991, I hoped that some type of document would be developed to provide information about macros, tools, etc. I'm glad to see that something has so emerged quickly onto the scene. I'm sure the entire TEX community will benefit from the time David has put into this Index. Many thanks David.

4 Tools

In order to support any package efficiently, you need to be able to specify a working environment that is acceptable for TEX and LATEX. For example, if you do a lot of telephone consulting, it can very time consuming and frustrating if someone calls and says "I have this error, Missing `}`, but I can't seem to find anything wrong." Sometimes you just want to say, "SO, what do you want me to do?" All joking aside, much of this frustration can be eliminated if the proper tools are available.

First of all, I would say that having the appropriate documentation is essential. As the support person, you should have a copy of as many of the popular TEX, LATEX, etc. manuals as possible. As a user, you should have a copy of the appropriate manual for TEX or LATEX (whichever package you would use). This avoids the translation of information over the phone, network, etc. You can specifically refer to page numbers for solutions and hopefully the user will realize that he or she had the answer to his or her problem all along in the manual and will use it the next time before they call upon you.

Following is a list of common books that I see floating around in the TeX user community.

- *The TeXbook*, Donald E. Knuth
- *TeX for the Impatient*, Paul W. Abrahams
- *A Beginner's Book of TeX*, Raymond Seroul and Silvio Levy
- *The Joy of TeX*, Michael Spivak
- *LaTeX: A Document Preparation System*, Leslie Lamport
- *LaTeX for Engineers & Scientists*, David J. Buerger
- *LaTeX for Everyone*, Jane Hahn
- *LAMS-TeX The Synthesis*, Michael Spivak

Secondly, I would say that there are many tools available to make using TeX and LaTeX much easier. Sometimes, knowing all these tools can be difficult especially when you are supporting several computer platforms. For example, some tools are available in UNIX and not for PC's and Mac's, and vice versa. However, there are a number of tools that go across platforms and certainly are worth implementing as part of a basic configuration [5].

- Check Matching
  `texmatch` is a program that checks for matching delimiters in TeX and LaTeX documents. I know this is available on most computer platforms; however, I am not sure about the Macintosh.
- Spell Checking
  `detex` is a filter that strips TeX and LaTeX commands from a document. This is needed for some computer platforms before using a spell checker program.
  On the PC and Macintosh platforms, spell checkers are available that allow the user to set up a dictionary of words (which can contain control sequences) related to TeX and LaTeX. This has the added value of reducing TeX errors because it is less likely to have a misspelled control sequence.
- Previewing
  Any previewer is fine as long as there is at least one available to preview a `.dvi` file. One may also consider having a previewer for PostScript, too. I have found this invaluable with trying to include PostScript into TeX and LaTeX documents. In general, having a previewer is a must. This should be true not only from the user's point of view, but especially the support person's point of view.
- Including Graphics
  If you have a method that works for you, GREAT! However, if you have not committed, then consider using PostScript. Of course this means you need to get a PostScript capable printer, a DVI→PostScript printer driver, and the software you are currently using to generate your graphics must provide an option to create a PostScript file.
  The DVI→PostScript printer driver of choice is `dvips` (written by Thomas Rokicki). It is well supported and has given the best and most consistent results when trying to include PostScript graphics from many different applications. `dvips` has been ported to many different computer platforms. It provides a set of macros, `epsf`, and also works well with the `psfig` macros, where either can be used to facilitate the inclusion of PostScript into TeX and LaTeX documents.
  Providing support for including PostScript into TeX and LaTeX documents will become essential in the future. Over the past two years, I have seen its popularity grow and, as a result, the support I was providing needed to grow, too [3].
  1. Learn enough PostScript to be able to interpret errors and modify an existing PostScript file.
     PostScript Language Reference Manual, Adobe Systems Incorporated
     PostScript Language Tutorial and Cookbook, Adobe Systems Incorporated
  2. Learn the two macro packages available for including PostScript, `epsf` and `psfig`.
  3. Learn as much as possible about what macros or programs are available to accommodate what users might want to do with PostScript. For example, rotating, two-up, etc.
  4. Get to know the applications and printers that users will be using to generate PostScript files and run a lot of tests.

5 Summary

Although providing good support for a package like TeX or LaTeX can be difficult, it is possible! It will take time and work to get to know your users, address their needs, and let them know what your expectations are as the support person.

There are three basics points to keep in mind:

1. Remember the different stages of learning as a user and support person. Try not to overestimate or underestimate your users.
2. Know what is available (or where to find existing solutions) to solve common and difficult problems. Don't re-invent the wheel!
3. Keep up with the technology.

All of these ideas are essential in providing accurate, efficient, and timely support. Hopefully, in return, you will develop a strong support network in your organization.

References

[1] CNS USER SERVICES, *The LaTeX UDThesis Format.* University of Delaware, Newark, June 1990.

[2] CNS USER SERVICES, *The TeX UDThesis Format.* University of Delaware, Newark, June 1990.

[3] CNS USER SERVICES, *Getting PostScript into TEX and LATEX Documents.* University of Delaware, Newark, June 1992.

[4] ANITA Z. HOOVER, "Using WordPerfect 5.0 to Create TEX and LATEX Documents", *TUGboat* 10(4), December 1989, pp. 549–558.

[5] ANITA Z. HOOVER, "LATEX/TEX User: A Typist, or Typesetter?", *TUGboat* 12(3), December 1991, pp. 397–400.

[6] DONALD E. KNUTH, *The TEXbook. Computers and Typesetting,* Vol. A., Addison-Wesley, Reading, Mass., 1986.

[7] ROBIN L. KUBEK, "TEX for the Word Processing Operator", *TUGboat* 10(4), December 1989, pp. 561–566.

[8] LESLIE LAMPORT, *LATEX: A Document Preparation System.* Addison-Wesley, Reading, Mass., 1986.

[9] OFFICE OF GRADUATE STUDIES, *Thesis Manual: A Degree of Excellence.* University of Delaware, Newark, June 1990.

The Pursuit of Quality*

How can automated typesetting achieve the highest standards of craft typography?

Frank Mittelbach† and Chris Rowley‡

November, 1991

Abstract

This paper compares high-quality craft typography with the state of the art in automated typesetting. The first part discusses several typographical conventions which cannot be implemented by means of any formatting model currently in use. The second part explains why the current paradigms of computerized typesetting will not serve for high-quality formatting and suggests directions for the further research necessary to improve the quality of computer generated layout.

Keywords: craft typography, automated typesetting, document formatting model, paradigm, typographic rules, visual context, logical context, global optimization.

1 Introduction

The preparation of quality typeset documents requires highly skilled understanding and application of concepts developed over hundreds of years in craft typography [16]. As the constraints imposed by a particular document (i.e., its unchangeable textual and logical content) usually result in conflicts between the generally accepted rules of quality typography, some of these rules have to be violated in favour of others. The skill of the compositor lies in the resolution of such conflicts between ideals. This results in the transformation of an abstract document into a visual form which is both aesthetically pleasing and faithful to the intentions of the author.

The nature of the author's intentions, and the details of how best to realize them visually, are largely outside the scope of this paper. For us, it is sufficient to know that the placement of objects in a physical representation of some particular document is (or should be) the result of a set of rules together with some mechanism for resolving conflicts. In craft typography these conflicts are typically resolved by interaction between the typographic designer and the compositor [32].

The equivalent process for automated typesetting is a computer program whose input is a form of the document containing only the text and logical mark-up and whose output is a complete description of the detailed content of each page of the typeset form of the document. As RICHARD SOUTHALL [27] has written:

> In formatting a document on a computer-based system, the graphic characteristics of blocks of text and their arrangement on the printed page are dictated by presentation rules which are determined by the format designer. The detailed arrangement of characters and spaces within the lines of text that make up a block is also governed by rules, ...

To have any chance of emulating the traditions of craft typography a program must, in some sense, understand such rules and be able to implement them. Moreover, it must contain mechanisms for resolving conflicts between these rules. It is also important to note that the set of rules which form the basic description of the layout for a document, or a class of documents, (i.e., the concepts for positioning and shaping the contents) is potentially very large.

We have therefore split the requirements specification of a program for the production of high quality documents into two major components:

- Any of the possible rules which form the basic layout must be implementable and applicable by the program.

†EDS, Electronic Data Systems (Deutschland) GmbH, Eisenstraße 56 (N15), D-6090 Rüsselsheim, Federal Republic of Germany, Internet: `mittelbach@mzdmza.zdv.uni-mainz.de`

‡Open University, Walton Hall, Milton Keynes, United Kingdom, Janet: `ca_rowley@vax.acs.open.ac.uk`

- Whenever the application of the rules produces a conflict, a resolving mechanism needs to be definable.

The challenge which this paper poses is, in computer science terms, to find suitable data-types whose transformations model well both the rules of the craft and the heuristics for conflict resolution. A necessary prerequisite for this is a suitable codification of the rules of craft typography and an evaluation of methods for automating its heuristics. We are not suggesting that the software should mimic exactly the skills of a craft typographer but that it should aim to produce typographical results which are, to the trained eye, indistinguishable from what would be expected from a reputable compositor.

Furthermore, we do not intend to give the impression that there is no need for well-designed systems which support interaction between a human operator and the typesetting software. However, in this paper we wish to focus on the requirements of the *automated* part of any system which aims at typographic excellence.

We start by looking briefly at a few examples of the typographer's craft which are addressed imperfectly, if at all, by existing typesetting software. Section 3 then analyses some of the basic requirements of a model for the design of software to automate craft typography and Section 4 summarizes the current state of the art; finally we suggest some directions for future research.

The authors wish to thank Richard Southall for many helpful discussions on subjects related to this work and Reinhard Wonneberger for his comments on early drafts of this paper.

2 A brief look at some rules

The rules of craft typography were developed to help the reader to understand the contents of a document [25, 26, 28, 29, 31]: good layout does not distract the reader's attention from the main aspect of the document—its message. Good typography therefore is a silent art; not its presence but rather its absence is noticeable.

Naturally, few typographical rules are universal: they depend on the purpose of the document, the cultural background of the prospective readers and many other things. Examples of the cultural aspect of typography range from obvious differences such as the direction of the typesetting (left-right, right-left or vertical) to more subtle distinctions such as the use of a type-face suited to the language of the document. Quite noticeable changes in the grey-values of paragraphs occur when the same type-face is applied to different languages ([25, p.43]), e.g., German with many capitals, French with its accents, etc.

The rules and concepts also depend on the school of typography and on particular house styles [5, 21, 6, 11]. Therefore, the discussion in this section is intended only to give some examples of rules which a designer *might* want to specify for particular documents and is not meant to be a complete survey of the subject; it focuses on rules accepted by most western schools of typography but which are not, to our knowledge, achievable by the currently available systems for automated typesetting.

2.1 Line breaking

Line breaking and its concomitant, word-division, provide a good example of the need for balance between conflicting ideals. The breaking of paragraphs into individual lines is not done simply because of the constraints imposed by a given page width—it has many aspects, including the following.

- The choice of page-size, layout, text-measure and type-face appropriate to the subject matter: long eye movements should be avoid as they tire the reader and impede the reading process.
- The quality and quantity of hyphenated lines: hyphenated words, especially when they appear too often or use psychologically bad breaks, make reading difficult (see Section 2.1.1).
- The shape of the paragraphs and the distribution of white-space within them: getting this right often leads to conflicts with the previous constraint. The use of ragged-right typesetting can help this conflict but leads to other problems, which are discussed in 2.1.2 below.

2.1.1 Word-division

Hyphenation of words is a process deeply entangled with paragraph construction [17]. All modern computerized typesetting systems incorporate some automated system for hyphenating words when the need arises, but most treat all the possible hyphenation-points as equally acceptable. As RONALD MCINTOSH suggests in [18] (and has implemented in the *Hyphenologist* software [8]), this is incorrect as they usually vary in their 'goodness' and these differences will influence the overall quality of the document. A special case of this aspect of hyphenation is that an inserted hyphen should ideally be distinguishable from a hyphen that is part of the word (e.g., re-cover vs. recover), allowing the designer to set up rules which take into account whether such a word should be broken at such a point [7].

Additionally, the distribution of hyphens within a paragraph is normally subject to typographical rules such as "avoid more than three hyphens on consecutive lines".

2.1.2 Paragraph shape and white space

When words have to be shaped into some template, two basically different approaches are common: ragged-right or justified blocks of text [30].

With justified text the intention is to give a uniform appearance: the excess space on individual lines needs to be distributed as evenly as possible over the whole paragraph, which makes a global optimization algorithm for determining the break-points necessary [1, 22, 23]. To ensure a similar grey value over the whole paragraph, differences between the excess space in adjacent lines must also be taken into account. However, it is not only the *average* distribution of white space which is important: it is, for example, also essential to avoid 'rivers' of white space flowing vertically through a paragraph [31].

Another refinement is that white space between words needs to be different for different combinations of boundary characters in order to give a uniform appearance [19].

If, on the other hand, ragged-right shapes are desired then the typesetting system must be capable of producing aesthetically pleasing shapes [26, p.131], e.g., avoiding successive lines of equal length and extremely uneven shapes (if the layout description specifies this). The absence of literature on this problem and the failure of any current typesetting systems to implement such rules suggests that the underlying concepts of ragged-right typesetting are not at all well understood.

Most of the problems covered here can, with most currently available systems, be detected only by human eyes. Furthermore, their circumvention is generally difficult and, at best, involves a large amount of detailed ad hoc interaction with a skilled human operator. That they have not at present been automated is in large part because the concepts necessary to describe and evaluate them have not yet been defined.

We are aware that there exist automated systems which go some way towards tackling particular aspects of the production of visually optimal paragraphs. The methods they use include the following: assessment of the 'average grey value'; use of 'weighted word-division points'; control of multiple consecutive hyphenations; optically correct inter-word spacing and optical alignment of the margins. However, we know none which do all of these, and none which make any explicit attempt to avoid 'rivers'

2.2 Grid layout

The eye needs a regular grid of points on which to focus when searching for objects (e.g., the next line) on a page. Many designers therefore base all aspects of the page-layout of a document on an underlying grid [13]. Such layout constraints are fundamental to the typesetting paradigms of many of the widely used DTP software packages but, regrettably, this is the *only* concept relevant to high quality typesetting which is understood by most such systems!

The distribution of white space in such a layout is necessarily discrete, which throws optimizing systems based on the box-glue model [14, 12] out of balance.

Since such a layout limits variation in the placement of constructed objects (like broken paragraphs, constructed figures with captions, etc.), to achieve good quality within such constraints there needs to be a high level of interaction between the building mechanisms for these objects and the page make-up mechanism (or, more generally, the algorithms for constructing higher level units). Such interaction is not implemented in any current systems (Section 3 contains a more detailed discussion of this topic).

2.3 Placing floating objects

Floating objects (e.g., footnotes or figures) are linked to the main body of a document by means of cross-references or other visual clues intended to identify them for the reader. Rules for their positioning can become quite complex to implement, even when they are easy to state [21]: e.g., that the distance between the reference and the float should ideally be small, or that the float should be visible from the point of its major reference.

Since floats should be easily distinguishable by the reader as individual units, clear visual separation from the main body of the text is usually necessary. This is typically achieved by font changes, extra space, rules, etc. thus such objects can quite drastically alter the visual appearance of a page. For this reason some designers prefer, for example, to lay out all the figures on a double-page spread as a single visual unit (placing individual figures and captions according to their size and form) to produce a balanced look.

To allow for such designs the software needs to be driven by heuristic rules as opposed to the procedural routines used in all current systems. Again, to emulate craft typography it is essential that complete interaction between this part of the system and all the other formatting routines is maintained: for example, it is necessary to allow paragraphs which flow around arbitrary figure shapes to, nevertheless, have globally optimized line-breaks.

Again, no currently available software implements more than a minimal amount of automation in this area.

3 Choosing the context

Each of the rules from the last section (along with many others) poses an intrinsically demanding and interesting implementation problem worthy of research effort—but it is not only the specification of certain important typographical conventions which is difficult or impossible within current systems. There is a more fundamental limitation which will prevent these systems from approaching craft standards, however good their formatting mechanisms may be—they have all been developed under the following paradigm:

> *That the formatting parameters for any object in some document can be determined by sequentially parsing its logical form.*

In other words, that these parameter values can be predetermined in such a way that the document can be processed in one pass (except for things like forward cross-references, which are resolved in a subsequent pass).

We claim that this approach will never suffice for the high quality standards of craft typography. As RICHARD SOUTHALL [27] went on to point out:

> With complex text, the achievement of character arrangements in the printed output that help to clarify its meaning to the reader may require differing rules of composition to be used in the formatting of different parts of the text.

We would extend his observation by pointing out that each object in a document should be formatted according to the context in which it finally happens to fall. This context has two components: its logical context and its visual context.

By **logical context** we understand the placement of the object (i.e., of its tag in the logical mark-up of the document) with respect to the other objects of the document. This component of the context is, in principle, fixed for a particular document (e.g., in a simple model for the logical structure of a document, it is given by the nesting and sequencing of the logical tags) but its analysis may involve substantial look-ahead or even a preliminary pass over the whole document.

The **visual context** of an object consists of the visual concepts and rules which are active at the place in the formatted document where the object actually appears (this includes, for example, the page(s) on which it falls, whereabouts on a page it appears and what else is on nearby pages). This visual component usually cannot be determined without complete knowledge about the placement (and consequentially the formatting) of all the other objects in a particular document. For example, the placement and measure for a figure caption might depend on whether that figure falls on a verso or on a recto page or whether there are other figures present on this particular page. Thus, if a 'floating figure' floats from a page of mixed text/figures to an 'all figures' page, its logical context will not change (since this is determined by the first reference to it in the text) but its visual context may change significantly.

Whilst the correct logical context for any object can in principle be determined after a single scan through the whole document, this is certainly not possible for its visual context since this depends on the formatting of all other objects within the constraints given by the applicable typographic rules. The above paradigm must therefore be replaced by one which recognizes that:

- formatting a document is a global process which must take into account variation in the visual context of each object;
- an iterative process is required to achieve an optimal (or even a high-quality) result.

Our basic model of automated document formatting is thus very similar to the model described in Section 1.2 of [10]. However, our refinement of the model is radically different from that developed in Section 1.3 of that article: this is probably an accurate reflection of the fundamental tension between the needs of a WYSIWYG system and those of automated high-quality typography. To achieve craft quality, detailed decisions must be made concerning the typographical treatment of all aspects of a document (from the layout of a spread down to the individual character glyphs). Such decisions depend, for a particular object, not only on its logical position in the abstract document but also on its visual relationship to the rest of the formatted document.

4 The current situation

The only major project which has addressed the problems of automating high quality typesetting remains that of Knuth (and developments thereof). There seems to have been nothing published on the theoretical side of this subject since 1982. The progress to that date is well described in *Document Preparation Systems* [20] from which we would particularly recommend the comprehensive survey by RICHARD FURUTA et al [9]. In this article the distinction between the editing process and the formatting process is clarified and a number of systems are described—both pure formatters and integrated editor/formatter systems. The authors then point out, in Section 4.5.1, that since all systems leave much of the task of producing satisfactory formatting to the user, close integration of the editing and formatting is essential since this makes "the generation of the concrete document part of a single document creation process".

Perhaps this explains the rapid advances over the last decade in the development of integrated editor/formatters, resulting in the present-day sophistication and range of systems incorporating the WYSIWYG paradigm of document formatting. By contrast, there has been very little progress in automating the formatting process itself and thus rendering unnecessary the human time and skills needed to produce high quality work within the WYSIWYG paradigm. It is encouraging to note that the emerging DSSSL standard [2] does allow expression of our paradigm—but this establishes no more than a common language in which to express these ideas.

In the TeX system the concept of global formatting, allowing for variation of visual context, is realized up to a certain point. In particular, the paragraph builder implements our paradigm to a limited extent: it evaluates a large number of variant possibilities for the visual context of the objects (characters and inter-word spaces) and globally (up to the paragraph boundary) optimizes over them within a quite complex parameter space. For example, even if a character sequence (such as `ff`) would be typeset as a single ligature glyph in the visual context of a line of type, when it is broken (by hyphenation) across a line-boundary the visual context of the sequence is changed so that two separate glyphs are typeset. TEX's paragraph builder has recently been extended by allowing, when no sufficiently good solution is found, a 'rescaling' so that a larger region of the parameter space is searched for an optimal solution [15].

Following on from the work of Knuth, but still working under the assumption that the visual context, and thus the formatting parameters, of certain entities could be predetermined, a globally optimizing pagination algorithm was analysed and implemented by MICHAEL PLASS in his thesis [24] and such an algorithm has been incorporated in the Type & Set System developed by GRAHAM ASHER [3, 4]. Both use a two step procedure in which the first run is used to produce a galley form of the document. In this form, textual items are already composed into lines, figures, etc. and their formatting is not therefore subject to any further refinement. In the second phase these preformatted objects are used to construct the final pages. This phase is controlled by an optimizing algorithm which takes into account the "distances" from references to figures together with "grey-values" (glue-stretching) for individual pages.

Even though these systems implement considerably more quality-oriented features than most other systems, they are still severely limited in many ways: e.g., they have no explicit constructs, or rules, designed to detect and prevent 'rivers'. Also, they do not conform to our paradigm in other important areas. For example, the page-building mechanism receives paragraphs already irrevocably broken into lines with no possibility of requesting a retry to find a variant which improves the page-breaking. The mechanism used by TEX itself to discourage a hyphen at the end of the last line on a page is simply to penalize a possible page-break after this line: it does not recompose the paragraph to avoid the problem hyphen by use of a slightly different sequence of line-breaks.[1]

5 New directions

The problems involved in the automation of typesetting are numerous and none of the currently available computer programs addresses more than a small number of them. As a result, documents produced by present-day software (without a significant amount of human interaction) do not come anywhere near the standards of craft typography. (The use of TEX in conjunction with the very simplest of typographic designs is perhaps an exception to this statement.)

The suggestions we make in this section for future research are directed mainly at computer scientists. However, this does not mean that we believe that they can solve the problems by themselves—on the contrary, much collaborative work with designers and compositors is essential to this task.

Implementing in full the concept of global formatting either means optimizing the formatting of the document as a whole (i.e., storing information about *all* possible variants in the formatting of every entity) or it means emulating this process by means of an iterative process—the latter appears to be the only practical option.

In all currently available systems, formatting is implemented by procedural routines based on bottom-up concepts, i.e., larger units are constructed from smaller ones whose internal format is already fixed. To achieve higher quality in automated typesetting, the optimization part of the process cannot be confined in this way to individual layers of complexity, e.g., first all paragraphs, then all pages (as in Type & Set). Instead, each individual routine which formats a particular object needs to produce as its output a range of variant formattings (each locally optimized) for that object under one or more assumed context possibilities. For example, a paragraph formatting routine might return information such as "this paragraph could be laid out in the given context into 5 lines with a forbidden break after line 2, or into 6 lines with . . . ".

The information thus gathered can then be the input to a global optimization process which produces a new formatted view of the document with, in general, altered visual contexts for certain entities. This process should then be iterated until a fixed-point is reached, i.e., until successive iterations give the same mapping of entity and context pairs.

The specification of layout rules such as those discussed in the previous sections by means of a large collection of parameters is certainly not intuitive and requires a thorough understanding of the underlying algorithms in order to predict the results of even small changes to the values of the parameters. Moreover, this approach is defective because the only rules which are specifiable are those whose underlying concepts are present in the model and algorithms used.

In view of the nature and quantity of these rules, and hence the computational complexity of the parametric optimization problems to which they lead, it would seem sensible to investigate whether other areas of computer science can be applied to this task. In particular, we suggest the following questions.

- Can certain aspects of the rules and heuristics be effectively modelled using an expert-system approach?

[1] It is interesting to note that the TEX model for optimizing the line-breaks within a paragraph is flawed in this case since it erroneously takes into account the visual incompatibility of two neighbouring lines which end up separated by a page-break!

- Can conventional optimization algorithms be replaced by algorithms which will, with a high probability, quickly produce a near-optimal solution?

A necessary precursor to any such investigations must be the study of new conceptual models of document formatting together with accompanying description languages designed to capture the rules and heuristics in a natural manner whilst being precise enough to drive the formatting software.

We are working on the formulation and refinement of such a model of document formatting and we are studying the data-types and transformations required to implement it. However, much work needs to be done in collaboration with typographers and typographic designers in order to adequately understand the rules and heuristics of the craft which need to be built onto our model and its description language.

References

[1] James O. Achugbue. On the line breaking problem in text formatting. *Proc. of the ACM SIGPLAN/SIGOA*, 2(1,2), 1981.

[2] S. Adler, project editor. *ISO/IEC CD 10179: Information technology—Text and office systems—Document Style Semantics and Specification Language*. American National Standards Institute, New York, 1991.

[3] Graham Asher. Type & Set: TEX as the engine of a Friendly Publishing System. In Malcolm Clark, editor, *TEX applications, uses, methods*, pages 91–100. Ellis Horwood, Chichester, 1990.

[4] Graham Asher. Inside Type & Set. *TUGboat*, 13(1), (to appear).

[5] Judith Butcher. *Copy editing: the Cambridge handbook*. Cambridge University Press, Cambridge, second edition, 1981.

[6] *The Chicago Manual of Style: Rules for authors, Printers and Publishers*. University of Chicago Press, Chicago, 13th edition, 1982.

[7] Carl Dair. University of Toronto Press, Toronto, 1967. Paperback reprint 1985.

[8] David Fawthrop. *Hyphenation by algorithm of English/American and other languages*. Computer Hyphenation, Bradford, 1990.

[9] Richard Furuta, Jeffrey Scofield and Alan Shaw. Document Formatting Systems: Survey, Concepts and Issues. In [20].

[10] Bo Stig Hansen. A function-based formatting model. *Electronic Publishing*, 3(1):3–28, 1990.

[11] Sue Heinemann and Virginia Croft, editors. *Xerox Publishing Standards: A Manual of Style and Design*. Watson-Guptill, New York, 1988.

[12] Mary Anne Holstege. *Marking and the Design of Notations*. PhD thesis, Stanford University, Department of Computer Science, Stanford, CA 94305, June 1989. Report No. STAN-CS-89-1270.

[13] Allen Hurlburt. *The grid: A modular system for the design and production of newspapers, magazines, and books*. Van Nostrand Reinhold, New York, 1978.

[14] Donald E. Knuth. *The TEXbook*, volume A of *Computers & Typesetting*. Addison-Wesley, Reading, Massachusetts, May 1989. Eighth printing.

[15] Donald E. Knuth. The new versions of TEX and METAFONT. *TUGboat*, 10(3):325–328, 1989.

[16] Hans-Joachim Koppitz, editor. *Gutenberg Jahrbuch*. Gutenberg-Gesellschaft, Internationale Vereinigung für Geschichte und Gegenwart der Druckkunst e.V., Mainz. Contains results on the past and present history of the art of printing. Published since 1926.

[17] Franklin Mark Liang. *Word Hy-phen-a-tion by Com-put-er*. PhD thesis, Stanford University, Department of Computer Science, Stanford, CA 94305, August 1983. Report No. STAN-CS-83-977.

[18] Ronald McIntosh. *Hyphenation*. Computer Hyphenation, Bradford, 1990.

[19] Frank Mittelbach. E-TEX: Guidelines to future TEX extensions. In Lincoln K. Durst, editor, *1990 Conference Proceedings*, pages 337–345, September 1990. Published as TUGboat 11#3.

[20] Jurg Nievergelt, Giovanni Coray, Jean-Daniel Nicoud and Alan C. Shaw, editors. *Document Preparation Systems*. North-Holland, Amsterdam, 1982.

[21] *Hart's Rules; For Compositors and Readers at the University Press, Oxford*. Oxford University Press, London, 39th edition, 1991.

[22] Michael F. Plass and Donald E. Knuth. Breaking paragraphs into lines. *Software—Practice and Experience*, 11:1119–1184, 1981.

[23] Michael F. Plass and Donald E. Knuth. Choosing Better Line Breaks. In [20].

[24] Michael Frederick Plass. *Optimal Pagination Techniques for Automatic Typesetting Systems*. PhD thesis, Stanford University, Department of Computer Science, Stanford, CA 94305, June 1981. Report No. STAN-CS-81-970.

[25] Emil Ruder. *Typographie; Ein Gestaltungsbuch*. Niggli/Hatje, Heiden, Stuttgart, fifth revised edition, 1988. Contains complete English and French translation.

[26] Manfred Siemoneit. *Typographisches Gestalten*. Polygraph Verlag, Frankfurt am Main, second edition, 1989.

[27] Richard Southall. Presentation rules and rules of composition in the formatting of complex text. In *SGML & TEX Conference 1990, Groningen*. Abstract of talk on Page 18 of the conference program.

[28] Jan Tschichold. *Ausgewählte Aufsätze über Fragen der Gestalt des Buches*. Birkhäuser Verlag, Basel, 1987. Second printing.

[29] Jan Tschichold. *Leben und Werk des Typographen Jan Tschichold*. Saur, München; New York; London; Paris, second edition, 1988.

[30] Alex White. *How to spec type*. Watson-Guptill, New York, 1987.

[31] Jan White. *Graphic Design for the Electronic Age*. Watson-Guptill, Xerox Press, New York, 1988.

[32] Hugh Williamson. *Methods of Book Design*. Yale University Press, New Haven, London, third edition, 1983.

Writing Reports with More than a Hundred People*

Walter van der Laan and Johannes Braams

PTT Research Neher Laboratories
P.O. Box 421
2260 AK Leidschendam
The Netherlands
W.vanderLaan@research.ptt.nl; J.L.Braams@research.ptt.nl

June 1992

Abstract

This paper describes a system that produces project status reports using LaTeX. The reports contain both textual and financial information. The textual part of the status reports is written by over a hundred people who don't need to know what LaTeX is. The financial information is retrieved from a database.

1 Introduction

Each quarter of a year the clients of our research center receive status reports concerning all of their projects. As you might expect these reports are typeset using LaTeX. What might be more of a surprise is the fact that the status reports are written by over a hundred project managers who don't need to have any knowledge of LaTeX. They write their status reports at different sites, using various computer systems, word processors and editors.

The first section gives an impression of the environment and history of this reporting system. The second explains the least that a project manager needs to know when using the system. A mail server is used to collect all project status reports. This server is discussed in the third section. The fourth section describes the generation of reports. Some general conclusions are listed in the last section.

2 Environment and History

With about 95,000 employees, PTT is the largest company in the Netherlands. The business of PTT is selling postal and telecommunication services. The reporting system described in this paper was made for PTT Research, a division of PTT with about 800 employees. PTT Research does most of its work under contract to other PTT divisions; typically there about 350 research projects for about 30 PTT divisions.

Each quarter, all project managers write a one-page status report to keep their client informed. These status reports consist of the following four sections:

- a short description of the project,
- the targets for the reporting period,
- the work realized in the reporting period, and
- the targets for the next period.

For each project the text written by the project manager is pasted into a form supplied by our financial department. This form shows information from a financial package, e.g.:

- the names of the client, project and project manager,
- important dates related to the project,
- the amount of money spent so far, and
- the budget.

The completed forms are bundled and presented to our clients.

The production of these status reports used to be manual. The texts supplied by the project managers, showing all kinds of fonts and printing qualities, were literally pasted onto the form using scissors and glue. Throughout our company, about a dozen people used to be busy collecting status reports and completing forms. It took more than a month before we were able to present the status reports to our clients. During this production process it was almost impossible for a project manager to make any corrections.

The system described in this paper offers much more flexibility. People, both with and without any LaTeX experience, send their status reports to a mail server. A report generator is used to produce LaTeX files containing a mix of information from the financial database and the status reports that have been received through the mail server. It enables us to present a uniform and beautiful report to our clients about ten days after the

*Presented at TUG '92, July 27–30, Portland, Oregon,USA.

financial closure of a quarter year. Within these ten days, project managers and their managers get several chances to correct the status reports.

3 Writing Status Reports

The status reports can be written as plain ASCII text or as LaTeX text. The first option is default and foolproof, it protects a project manager from any LaTeX errors. While this option is easy to use it also leaves the writer with only a few of the expressions available in LaTeX, namely paragraphs and some special characters common in the Dutch language. This has proved to be sufficient for 99 percent of the reports but leaves a few special cases, e.g., some project managers want to put formulas in their report and others have a need for symbols used in physics. In these cases the project managers write their report as LaTeX text and they have to know how to use it. The remainder of this section explains what a project manager needs to know when using the foolproof mode, i.e. plain ASCII text.

A report sent to the mail server may contain nothing but plain ASCII characters. This is a necessary constraint because the mail server receives reports written with about twenty different kinds of word processors and editors on about five different kinds of platforms. Fortunately all word processors have an option to save a text in ASCII. This means that all project managers must know how to produce an ASCII file containing their report. They must also be aware of the limitations of ASCII; e.g., no underline, no boldface, no special characters.

Project managers must be able to send their reports to the mail server. This hasn't been a problem in our organization as everybody is connected to the local network and most people are using electronic mail.

Each report must contain a few keywords. This simple syntax is needed to

- assign a project number to each report,
- separate the four pieces of text in each report, and
- separate the report from the mail headers and footers.

The next example shows the report for project number 12345. As you can see the syntax consists of uppercase keywords with four pieces of text in between:

```
PROJECT 12345
DESCRIPTION
We are working on a project.
CURRENT TARGET
Our plan was to finish the project.
REPORT
We've had a lot of problems but the
work is almost done.
NEXT TARGET
We'll finish the project in the next
quarter year.
END
```

Many people were having problems with this combination of a textual report and a strict syntax. We eliminated this problem by extending the mail server to accept all syntax errors that occurred.

Project managers have to understand the effect of an empty line in their texts: All text is aligned on the left and right margin. An empty line causes the alignment to restart at the beginning of the next line, because texts are set without paragraph indentation.

In the Dutch language one frequently finds characters such as é, è and ï. For this reason the project managers have the option of using the sequence `backslash accent vowel` whenever they need to put an accent over a vowel.

4 The Mail Server

All mail sent to a dedicated network address is processed by a program, which replies to every message received. The reply consists of two parts. The first part contains success and error messages, e.g.: "`i found a report about project 12345`", or: "`i removed some control characters from your text`". The second part is a copy of the received message in which all recognized texts have been removed. If all went well the second part contains nothing but keywords and mail headers. This construction is clear even to people who aren't used to syntax errors.

In the beginning we had trouble with errors in the project number. People would erroneously send us a report for project 12435 instead of 12345. This meant one could overwrite another report by mistake. We solved this problem by saving a list of the mail addresses of the senders for each report. We accept reports from any address on the list. A report from any other address is rejected but the address is added to the list. In such cases the sender receives a message saying the report has been rejected but will be accepted if the report is sent again. This warning eliminates the typing errors in project numbers but still allows different people to send updates for the same project.

In the foolproof mode all texts must be transformed into valid LaTeX texts. This transformation is described next.

Many characters special to LaTeX have to be deactivated. We handle those characters as four separate cases:

- Double quote characters are replaced by " and ", respectively.
- A backslash is added in front of the following characters: #, $, %, &, _, { and }.
- The math characters $<$, $>$ and $|$ are placed between dollar signs.
- The expression `{\tt\char92}{}` is used to insert a backslash. Note that 92 is the ASCII value of a backslash. The same procedure is used to insert

the characters ^ and ~.

The sequence `backslash accent lowercase vowel` is allowed. This sequence isn't changed by the transformation; only when the vowel is an *i* is it replaced by a dotless ı. This change makes the sequence very uniform and easy to understand for people not used to LaTeX.

LaTeX does a great job at automatic hyphenation. However, when words are joined together by characters such as the slash and minus, LaTeX doesn't hypenate the resulting string. This causes a lot of overfull hboxes. To solve this the transformation program looks for the sequence `letter slash` or `minus letter`. Within sequences like this the slash or minus is transformed into the parameter of the `nw` command (nw stands for *new word*), e.g.: `man/woman` is replaced by `man\nw{/}woman`. The `nw` command is expanded to let “man” and “woman” be separate words divided by a slash. The LaTeX definition is:

```
\newcommand{\nw}[1]
  {\hspace*{0pt}#1\hspace*{0pt}}
```

This transformation has proved to be sufficient in avoiding almost all overfull hboxes.

Last but not least, all characters unknown to LaTeX are removed during the transformation.

The *Lex* specification below (Lesk and Schmidt, 1975) produces a program that performs the transformation described above:

```
AN             [a-zA-Z0-9]
AC             [`'"^.=]
%%
               int dq = (0 == 1);
\"             { dq = !dq;
                 if (dq) printf ("``");
                 else    printf ("''"); }
[#$%&_{}]      { printf ("\\%s", yytext); }
[|<>]          { printf ("$%s$", yytext); }
[~\^\\]        { printf ("{\\tt\\char%d}{}",
                   yytext[0]); }
\\{AC}i        { printf ("\\%c{\\i}",
                   yytext[1]); }
\\{AC}[aeou]   { printf ("%s", yytext); }
{AN}[/-]{AN}   { printf ("%c\\nw{%c}%c",
                   yytext[0], yytext[1],
                   yytext[2]); }
\t             { putchar (' '); }
\n             { putchar ('\n'); }
.              { if ((yytext[0] >= 0x20)
                 &&  (yytext[0] < 0x80))
                   printf ("%s", yytext); }
```

Text sent in LaTeX mode is not transformed by the mail server. A copy of the text is transformed into a complete LaTeX document by adding a header and a footer. The mail server passes this document to the TeX compiler and afterwards checks the log file for errors. If an error occured, it rejects the text and adds the compiler messages to the reply.

The mail server program was written using *TPU*, the VAX/VMS Text Processing Utility (Digital VMS manuals, 1988). *TPU* and LaTeX make a great team and have allowed us to build this system in a short time. If you ever need a utility to process text, try *TPU*.

5 Report Generation

The four pieces of text per project are stored in a separate directory as four files per project. All of these texts have been tested or transformed by the mail server. These texts are ready to be typeset in any textwidth or font.

Our relational database system stores a lot of project information. This information is transformed into LaTeX strings just as the texts supplied by the project managers are transformed.

With an application program our financial department can define which projects are to be included in a report and in what sequence they are to be included. Every set of projects defined can be printed in several ways; e.g., a report containing:

- all information about a project on one page,
- only the financial information, and/or
- only the description of each project.

When we started work on automating the report generation process we first agreed upon an interface between the report generator and LaTeX that consists of a few special commands.

In Figure 1 an example of a status report is shown. At the top of the page some general information about who ordered the project and who runs it is shown. This information is repeated on a continuation page as you can see in Figure 2. Then follows a short description of the project and its targets. In this case the description is too long and is therefore continued on a second page. The next texts discuss the targets for the reporting period, the work done in the reporting period and the targets for the next reporting period. At the bottom of the page some financial information on the project is included. Both this financial information and the general information at the top of the page is extracted from the database.

For each of the text fields a LaTeX environment is defined. The task of these environments is to store the text parts in four boxes of the right size. To pass the information that is printed in the header and footer of a report we defined a few LaTeX commands with parameters. All information that belongs to a particular project should be inside yet another LaTeX environment. The task of this environment is to:

- start a new page,
- select the correct page style,
- write some information about the project to a table

of contents file, and

- make sure that all information accumulated is put on the page.

Originally, the layout of the report form was defined in terms of a number of characters per line, and a number of lines for each of the four text fields. It was also specified that a report for any project should occupy not more than one page.

With typeset text, the specification of the number of characters per line is not particularly useful, because it can vary with the kind of characters that are used in the text. Also, the manual process of putting together the status reports had shown that sometimes a project manager would produce more text than would fit in the field for which it was meant.

Because the sizes of the fields are fixed, we had to choose what to do. We could typeset the portion of the text that would fit into the field and either

1. let the rest print over the next field, or
2. discard the rest, or
3. store leftover text and print it on a second page.

Option 1 clearly is unacceptable; the result would be both ugly and unreadable. Option 2 would possibly result in texts ending in a weird manner. Choosing this option would, however, force the project managers to be brief. It was finally decided to use option 3. One of the reasons for this decision was that the implementation of options 2 and 3 is almost the same.

The implementation of this part of the LaTeX style file is based on the use of the `\vsplit` command. The four environments discussed before scoop up the text and typeset it in a `\vbox` `\pickup@box` of the appropriate width. The contents of the `\pickup@box` are copied into a different box for each environment using the `\vsplit` operation. The resulting height of the `\pickup@box` is then measured. If it is not zero there is more text than fits into the field. In that case an indication that there is more text is appended to the first part of the text, and the rest is stored away to be put on a second page.

6 Conclusions

The system imposes no special organization upon its users. Our users write and send their reports using the computer system that they use for their everyday work. In some departments the reports are gathered by one person who sends all reports and report updates to the mail server. In other departments all project managers send and update their own reports. The nice thing about a mail server is that it doesn't mind where a report was made or who made it.

The two modes, LaTeX or foolproof, have made the system both very flexible and very easy to use. No special training is required for people who use the system in the default foolproof mode. People with special requirements, on the other hand, are very happy with the LaTeX mode.

LaTeX separates the contents of a document from the form in which it is presented. This separation was of great benefit to us during the development of the system. It allowed us to make a very clean and easy division of the work to be done. After defining the different styles needed for our system one of us would work on the generation of LaTeX reports using the defined styles, while the other would work on the creation of the styles.

At first we implemented the reports to be generated interactively. This didn't work very well because LaTeX consumes large amounts of cpu time. At the moment the production of reports is implemented as a lower priority background process.

References

[1] Lesk, M.E., E. Schmidt. "Lex — A Lexical Analyser Generator." Bell Laboratories. Murray Hill, New Jersey, October 1975.

[2] *VAX Text Processing Utility Manual.* Digital VMS Manuals, **5B**, April 1988.

PTT Research Kwartaal Rapportage

92/01

Projectnaam	PART. WERKZAAMHEDEN SC18	Projectnummer	61006
Opdrachtgever	RAAD VAN BESTUUR RVB	Onderdeel/hafd	TI INFORMATIEDIENSTEN
Afdeling	RVB	Hoofd	TERPSTRA, A.P.
Hoofd	J. WAGE (secretaris LBR)	Projectleider	W. REMMERS
Contactpersoon	W.J. THIEME	Begin/Eind datum	10192 / 311292
Kenmerk	OVK D.D. 15/07/1990	Kenmerk offerte	91-A041

Korte omschrijving en doel van het project

Binnen ISO/IEC JTC1 is SC 18 gericht op de standaardisatie-aspecten van creëren en manipuleren van documenten. SC18 omvat de volgende gebieden:
* document architecturen en semantiek (ODA, SGML, DSSSL)
* document interchange formaten (ODIF, ODL, SDIF, DTAM)
* basis inhoudsarchitecturen (o.a. teken, raster, muziek) en weergave-aspecten (kleur, fonts)
* hypermedia (HyTime, SMDL)
* printeraansturing (SPDL, DPA)
* message handling en message store (X.400 of MOTIS)
* document filing & retrieval (DFR)
* User System Interfacing (keyboard layouts, Icons & Symbols, dialoog standaarden)

Zie volgende pagina

Gesteld doel voor dit kwartaal

Gezien het belang van de SC18-onderwerpen, de breedte van het aandachtsgebied en de doelstelling de werkmaatschappijen meer bij dit werk te betrekken, dient er een duidelijk plan van aanpak geformuleerd te worden. Vanwege een wisseling in de bezetting is deze aktiviteit vorig jaar opgeschort. Hieraan zal binnenkort wel veel aandacht besteed worden.

Korte rapportage

Op dit moment zijn een groot aantal standaarden en/of uitbreidingen op de basis-standaard (zie boven) in bijna-finale vorm. Tegelijkertijd blijken er nog grote definitie-problemen te bestaan, en blijkt tussen verschillende standaarden grote overlap te bestaan. In vorige rapportages is al genoemd de relatie tussen 'character, graphic character, font en glyph', Message Store Extensions versus DFR en DTAM, ODA en SGML. tegelijkertijd worden er in hoog tempo nieuwe werk-items voorgesteld. De wens naar stabiele standaarden is momenteel strijdig met de momenteel gehanteerde ISO-planning en het aflopen van de studie-periode van CCITT. De voorjaarsvergadering is bij OC\'E gehouden, waar een interessante presentatie over de geschiedenis van het bedrijf en de bemoeienis met ODA werd gegeven.

Gesteld doel voor het volgend kwartaal

1. Het aktief volgen van ontwikkelingen binnen SC18.
2. Afronden plan van aanpak (tweede kwartaal 1992)

Inzet S&O	afgelopen kwartaal	t/m afgelopen kwartaal	overeenkomst	perc. gerealiseerd
Aantal Uren	81,00	81,00	320,00	25,30
Bedrag (fl)	16.038,00	16.038,00	63.360,00	25,30

1 Uitsluitend voor gebruik binnen PTT

Figuur 1: *A sample report for a project*

PTT Research Kwartaal Rapportage

92/01

Projectnaam	PART. WERKZAAMHEDEN SC18	Projectnummer	61006
Opdrachtgever	RAAD VAN BESTUUR RVB	Onderdeel/hafd	TI INFORMATIEDIENSTEN
Afdeling	RVB	Hoofd	TERPSTRA, A.P.
Hoofd	J. WAGE (secretaris LBR)	Projectleider	W. REMMERS
Contactpersoon	W.J. THIEME	Begin/Eind datum	10192 / 311292
Kenmerk	OVK D.D. 15/07/1990	Kenmerk offerte	91-A041

Vervolg Korte omschrijving en doel van het project

* Distributed Office Application Model (DOAM)
Vanuit verschillende optieken (EDI-wereld, kantoorautomatisering, uitgevers, enz.) worden eisen gesteld aan dergelijke systemen. De kontakten met andere standaardisatie-lichamen (CCITT, ECMA, enz.) zijn daarom intensief.

2 Uitsluitend voor gebruik binnen PTT

Figuur 2: *The second page of the sample report*