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

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.

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.

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 financial closure

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

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.

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 de-activated. 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*.

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.

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.

Bibliography

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

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.

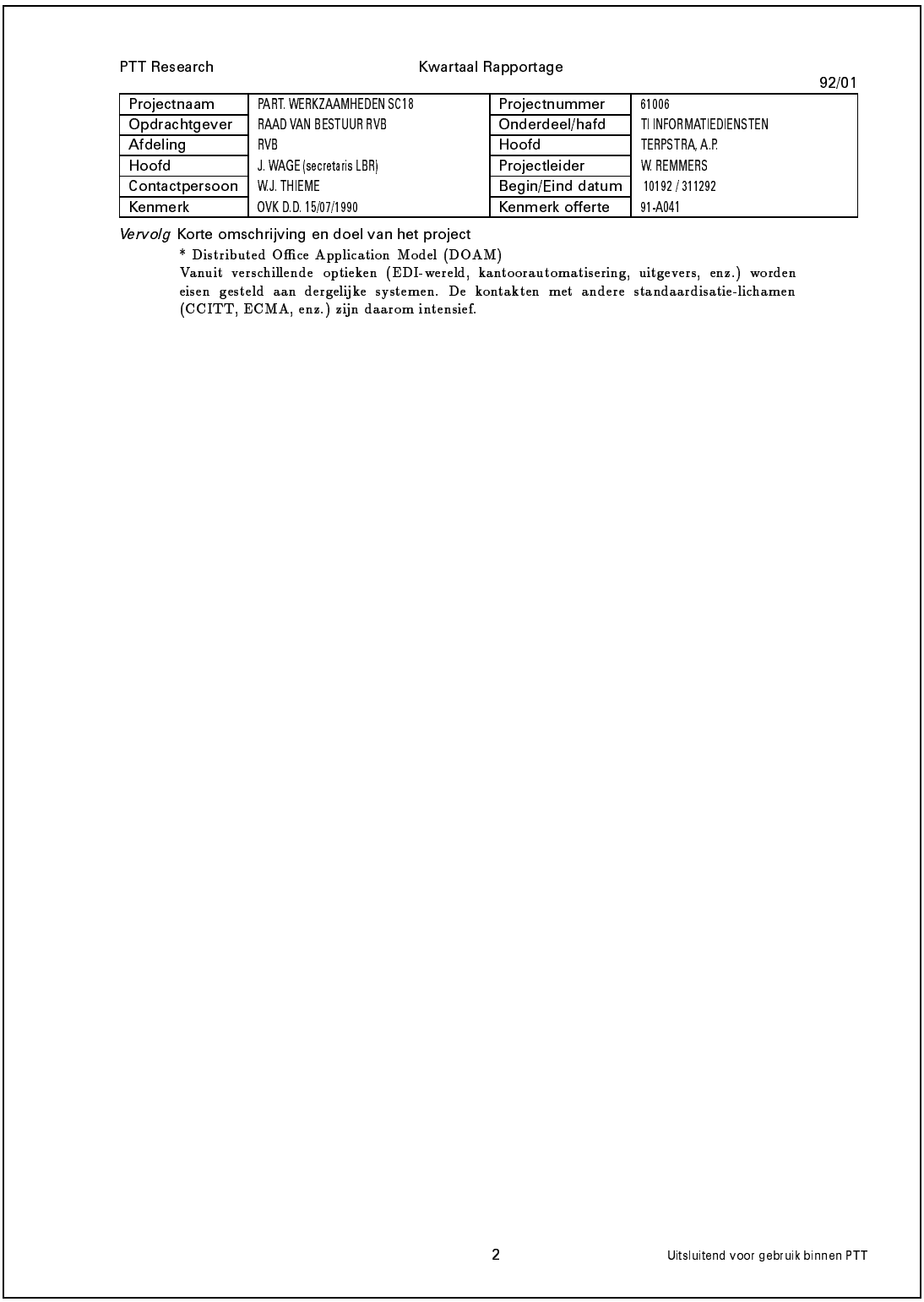
Gesteld doel voor het volgend kwartaal

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

Figure 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

Figure 2: The second page of the sample report