ELECTRONIC PUBLISHING, VOL. 2(3), 169–173 (OCTOBER 1989)

Can structured formatters prevent train crashes?

JACQUES ANDRÉ

IRISA/INRIA
Campus de Beaulieu
F-35042 Rennes, France

SUMMARY

A typographic layout error is analysed for its likely effect as being one of the causes of a train crash. Arguments are put forward to show that this error could not have occurred if a structured text formatter had been used.

KEY WORDS Structured formatters Document reliability Typographic errors

Paris, Gare de Lyon, 27 June, 1988, 18:47. A crowded suburban train is ready for departure when, suddenly, another train arrives in front of it. There is a crash with 56 dead and hundreds injured. Obviously the killer train had no brakes. Why?

The French government immediately set up a commission to analyse the disaster. This commission published its analysis in a report in September 1988 [1]. First of all it appears that 'Such a disaster is not due to a single cause. Rather to a chain of different circumstances'. Someone pulled down the emergency lever which caused the train to stop in an unscheduled station; an air-brake pipe was faulty and the train driver was unable to bleed it; the radio alarm system was out of order, etc. However the commission noticed that *the maintenance manuals were particularly complex to use*, and it even noticed *an error in the layout of the document describing the process of repairing brakes*.

The French text of that document, in its original layout, is shown in Figure 1. Figure 2(a) exhibits the main points of the document in that original layout, while Figure 2(b) shows how they should have appeared[1].

Whenever the '1st CASE' of Figure 2(a) applies, the layout tells us that the driver processes only the *xxx* actions and nothing else. Apparently he does not need to look at the second case, nor to notice the hidden phrase 'In both cases' with its associated actions. Thus, he does not switch on the taps, nor does he check the air brakes, and so on. By contrast, the layout of Figure 2(b) indicates that, after obeying the '1st CASE' *actions*, the driver has to follow on with the 'In both cases ... ' actions which, in turn, involve checking the brakes.

We can surmise that the original document was typeset using a second-generation typesetting machine. Let us assume that, on such a machine, there are three tags for controlling indentation, of the form:

<**IX**> to right-indent the text that follows (i.e. to shift right the left margin),
<**QL**> to feed the current line and to start the next line at the current margin,
<**EP**> to feed the current line and to restore the left margin to its previous value.

[1] Though it could be argued that the lines *1st CASE* and *2nd CASE* should be even more indented

0894–3982/89/030169–05$5.00 *Received 6 November 1988*
Revised 13 November 1989

b) Plusiers véhicules sont bloqués, le mécanicien :
S’assure que ce blocage n’est pas la conséquence de la fermeture
d’un robinet d’arrêt de la conduite générale situé avant la partie
de train bloquée :
1^{er} CAS : Aucun robinet d’arrêt CG n’est fermé :
Il actionne la commande de la valve de purge le temps suf-
fisant pour provoquer le desserrage sur chaque véhicule
bloqué.
2^{e} CAS : Un (ou plusiers) robinet d’arrêt est trouvé fermé :
Il ouvre le robinet
Dans les 2 cas, le mécanicien :
— ouvre le robinet d’arrêt CG situé en arrière du dernier
véhicule relié à la CG.
— vérifie le serrage des freins du dernier véhicule freiné.
— referme le robinet CG
— vérifie en se dirigeant vers la tête du train :
• le désserrage des freins de tous les véhicules,
• que le blocage n’a pas provoqué d’avarie aux roues.
Il applique les mesures concernant le signalement et la reprise de mar-
che (article 385).

Figure 1. Part of a train maintenance manual with a layout error: the second line after 2^{e} CAS (Dans les 2 cas = In both cases) should not be indented

Figure 3(a) shows how the text has probably been typeset, to give Figure 2(a), while Figure 3(b) shows how it should have been typeset. This example shows how a very small bug (<EP> instead of <QL> on line 5) can cause a disaster.

Two questions are relevant to this kind of instructional text:

1. How can we make-up a text to be both legible and understandable?
2. How can we edit reliable texts?

According to T.R.G. Green and S.J. Payne, when they were studying concurrency [2], ‘the well-documented use of typographical cues to illuminate instructional text has in the past been limited to illustrating *containment* relations (sections within chapters or subsections within sections) and *succession* relations (after one chapter we come to the next). No other relations have been studied’.[2]

It is worthwhile to consider the layout rules for a specific set of instructional texts, namely, those for programming languages. Thanks to Dijkstra’s paper ‘*go to statement considered harmful*’ [5], many studies have been made of the ‘best way’ to edit conditional statements (such as *if* statements or *case* statements, loops etc.). All the proposed solutions are based on horizontal indentation of internal parts of the statement, and on

[2] Since that time, other papers have been published, e.g. Pat Norrish [3] uses a typographical approach while Virbel [4] looks at lists using a linguistic approach.

```
1 The driver ...  checks ...                 The driver ...  checks ...
2 1st CASE:       xxx                        1st CASE:       xxx
3                 xxx                                        xxx
4 2nd CASE:       yyy                        2nd CASE:       yyy
5                 yyy                                        yyy
6                 In both cases the driver   In both cases the driver
7                             - zzz                          - zzz
8                             - zzz                          - zzz
9 Then, the driver restarts the train.       Then, the driver restarts the train.

                  (a)                                        (b)
```

Figure 2. (a) How the text was printed; (b) how it should have been printed

```
1 The driver checks ... <EP>     The driver checks ... <EP>
2 1st CASE: <IX> xxx <QL>        1st CASE: <IX> xxx <QL>
3 xxx <EP>                       xxx <EP>
4 2nd CASE: <IX> yyy <QL>        2nd CASE: <IX> yyy <QL>
5 yyy <QL>                       yyy <EP>
6 In both cases the driver       In both cases the driver
7       <IX> - zzz <QL>                <IX> - zzz <QL>
8 - zzz <EP>                     - zzz <EP>
9 Then, the driver restarts ...  Then, the driver restarts ...

               (a)                             (b)
```

Figure 3. (a) How the text was edited; (b) how it should have been edited

the vertical lining up of ‘parentheses’ (where an **end** would be the right parenthesis of a **begin**). However, a given statement may be edited in many ways, depending on the language, and in the end there is no definitive preferred template. There is still much work to be done on program understandability and its relationship to legibility factors.

The same problems occur in the formatting and make-up of texts. For example, many papers have been published on the legibility of bibliographical references, but how many of those definitive answers, using electronic publishing facilities, would be useful and usable today? Equally, one cannot give precise indentation rules for conditional statements in repair manuals for train brakes. Indeed, it may be the case that understanding would be helped by other visual clues, in addition to the indentation.

We can give rather more guidance regarding the second question, of how to edit reliable texts. More precisely, we wish to be sure that an error cannot occur when editing a manual. First of all, let us imagine that our manual for repairing train brakes was formatted using a WYSIWYG system. Obviously, when typing in the rules, errors of the kind we have just described would be quite impossible (one sees on the screen that the indentation is wrong). However, no one can guarantee that such errors might

```
1   The driver checks ...
2   \begin{description}
3   \item[1st CASE:] xxx\\
4   xxx
5   \item[2nd CASE:] yyy\\
6   yyy
7   \end{description}
8   In both cases, the driver
9   \begin{itemize}
10  \item zzz
11  \item zzz
12  \end{itemize}
13  Then, the driver restarts the train.
```

Figure 4. LaTeX *code for figure 2.a,*

not occur if the text is subjected to some global ‘search and replace’ operation (or any operation without human check), on some given text string.

Again, let us turn to examples from the realm of programming languages. J.J. Horning [6] tells us that ‘The experience of the last thirty years shows that it is not easy to produce nearly-correct programs. . . . The goal of reliable programming is to minimise the number of faults in completed programs. . . . Checking always relies on a certain amount of redundancy built into the language’. We note that the high level of redundancy in structured programming languages allows them to be practicable solutions to this problem.

For the same reason, structured formatters allow text to be formatted safely. Indeed, parenthesized structures (with **end** completing some part of text starting with **begin**), recursivity (parentheses are to be balanced) and redundancies (an **end** must have the same ‘label’ as the corresponding **begin**, e.g. **begin**{**section**} has to be closed by an **end**{**section**}) are good tools for reliability, even if they are boring to type in. More information on structured documents is given in [7].

Let us now imagine that the train manual for repairing brakes had been written in some structured formatter such as LaTeX[8]. A standard way for editing the text of Figure 2(a) is shown in Figure 4. Let us assume also, that the user made an error, e.g. forgetting the **end**{**description**} at line 7 or mistyping it. In these circumstances an error would be detected by LaTeX due to mis-matched parentheses, and although that error message would not be too easy for a beginner to understand, the fact remains that a message would be posted and no printed output would be produced. Redundancies of this sort lead to fault intolerance, and hence to reliability.

By using programming systems such as Mentor, the newer document manipulation systems such as Grif are even going further. Thanks to their internal structure and to the use of windows, one no longer needs to type in ‘parentheses’: they are now so implicit in the overall document structure that it is very difficult to use them erroneously.

REFERENCES

1. *Rapport de la Commision d'étude de l'accident de la Gare de Lyon*, Ministère des Transports et de la Mer, Paris 1988. See also J. André, 'LaTeX ou SGML pouvaient-il faire éviter l'accident de la gare de Lyon?', *Cahiers GUTenberg*, **1**(1), 21–25, (April 1989).
2. T.R.G. Green and S.J. Payne, 'The woolly jumper: typographic problems of concurrency in information display', *Visible Language*, **XVI**(4), 391–403, (1982).
3. P. Norrish, 'Semantic Structures of Text', in *Structured Documents*, eds. J. André, R. Furuta and V. Quint, Cambridge University Press, Cambridge, 1989, pp. 143–159. See also P. Norrish, *The Graphic Translatability of Text*, British Library R & D Report 5854, 1988.
4. J. Virbel, 'The contribution of linguistic knowledge to the interpretation of text structures', in *Structured Documents*, eds. J. André, R. Furuta and V. Quint, Cambridge University Press, Cambridge, 1989, pp. 161–189.
5. E.W. Dijkstra, 'Go to statement considered harmful', *Communications of the ACM*, **11**(3), 147–148 (1968).
6. J.J. Horning, 'Programming languages', in *Computing Systems Reliability*, eds. T. Andersen and B. Randell, Cambridge University Press, 1979, pp. 109–152.
7. J. André, R. Furuta and V. Quint (eds.), *Structured Documents*, Cambridge University Press, 1989.
8. L. Lamport, LaTeX, *A Document Preparation System*, Addison-Wesley, Reading, Mass., 1986.

Call for papers

SPECIAL ISSUE OF ELECTRONIC PUBLISHING: ORIGINATION DISSEMINATION AND DESIGN ON HYPERTEXT

Electronic Publishing: Origination Dissemination and Design (EP-odd), is pleased to announce a special issue on Hypertext, to appear in 1990. We now wish to call for papers in all aspects of hypertext and in related hypermedia models and systems. We especially invite papers that give a broader perspective of hypertext (well-grounded in experience), describe unified models of hypertext, show the strengths and limitations of hypertext, convincingly demonstrate the interrelationships between hypertext and other areas of electronic publishing, show how hypertext can be used to organize very large amounts of information, describe the appropriateness or inappropriateness of higher-level structuring of hypertext, or in other ways help define what hypertext is and help to show what its inherent strengths and weaknesses are. Papers may be as long or as short as authors desire, but we expect that most papers will be in the range of 10 to 20 pages in length. All papers will be refereed. Authors are asked to take special care to ensure that their manuscripts are in final form, as the limited amount of time available for reviewing contributions to the special issue will not permit very many passes through the editing cycle. Authors will be asked to follow the EP-odd author guidelines, with the exception that four copies of the manuscript are requested, and they are asked to include a copy of the EP-odd transmittal form with their submissions. Copies of the guidelines and transmittal form may be found in each issue of EP-odd or may be obtained by mail from the guest editor (address below).

Because space is limited in the special issue, it may not be possible to publish all deserving papers. Unless authors request otherwise, such deserving papers will be referred to the normal EP-odd editorial process and will be considered for publication in a subsequent issue.

The guest editor for the special issue on hypertext will be Professor Richard Furuta of the University of Maryland. His address is:

Richard Furuta
Department of Computer Science
University of Maryland
College Park, MD 20742 USA

E-mail:	furuta@cs.umd.edu (Internet)
Telephone:	(+1) (301) 454-1461
FAX:	(+1) (301) 454-8346

The important dates are as follows:

15 July 1990:	4 copies of papers due to address above
1 September 1990:	Notifications of acceptance sent
1 October 1990:	Final revisions due

Editorial

The present issue completes Volume 2 of EP-odd, and on this occasion the Editorial bears only one name. The reason for this is that the end of Volume 2 marks the point at which our US co-editor, Richard Beach, is handing over the US Editorship of EP-odd to Richard Furuta, one of our Editorial Board members. Devotees of the 'Monty Python' series would be justified in concluding that just as everybody in Australia is called 'Bruce' so also is it *de rigeur* for US Editors of EP-odd to be called 'Rick'. Whatever the truth may be, I shall have to distinguish them as 'Rick B.' and 'Rick F.' from this point on.

This solo Editorial enables me to thank Rick B. for all he has done in helping to set up EP-odd and in making it such a success. Shortly after the first issue of Volume 1 appeared he was promoted to the position of manager of the Electronic Documents Laboratory at Xerox PARC. It would be flattering, if rather unrealistic, to believe that the two were connected, but in any event the increasing administrative load associated with his new post, coupled with existing commitments to SIGGRAPH, have led Rick B. to the reluctant conclusion that he cannot devote the requisite editorial time to EP-odd, now that we have entered full-scale production. However, I am glad to report that he will be staying on as a member of the Editorial Board.

The next pleasant duty is to welcome Rick F. as our new US Editor. Many of you will already know of his work in areas such as structured documents and hypertext. It was particularly brave of him to take on new responsibilities at a time when he is embroiled in the myriad details of being Program Chairman of the EP90 conference (to be held at Gaithersburg, Maryland in September 1990). Rick F. comes to the job with the excellent pedigree of having two papers already published in EP-odd (one of them in this issue) and bearing a second name which is an anagram of a classic sans-serif typeface. What more could one ask? (Joking apart, I note that a certain font supplier, anxious no doubt to avoid trademark litigation, is distributing a PostScript version of a popular sans-serif typeface under the name 'Futaru'. This prompts the following conundrum: When designing the exhibition poster for EP90 will Richard Furuta use Futura or Futaru?)

Electronic Publishing is a strange business, and despite all the new technology available to us it is remarkably difficult to do it well. Perhaps there is something, after all, in the cynical observation that 'new technology rarely saves you money — it just helps you to do more and to stay competitive'. Certainly, EP-odd's policy of allowing editors and authors to participate in the production process adds quite a few headaches and, looking back over the six issues that have now appeared, we still see the classic symptoms that the first 95% of producing an issue of EP-odd is utterly straightforward — the last 5% drives the editorial and production staff to distraction. On the other hand it is true that the freedoms afforded, for example, by having PostScript as a common final form, allow us to publish 'difficult copy' with relative ease (the `triroff` paper, in the previous issue, was a good instance of this). In the end, what makes it all worthwhile is the occasional unsolicited testimonial, such as the following (from a member of our Editorial Board)

0894–3982/89/040177–02$05.00 *Received 16 May 1990*

> “A measure of the quality of a publication is the length of time that it stays on one’s shelves before being ‘borrowed’ (never to return). You will be pleased to know that every copy of EP-odd has now disappeared from my shelves”

It is a stern responsibility to ensure that EP-odd remains so compulsively readable but we shall certainly try. The next issue of EP-odd will see Rick F’s name on the cover and we also hope to publish the first of our Book Reviews.

DAVID F. BRAILSFORD