M.I.T. Laboratory for Computer Science July 12, 1976

Computer Systems Research Division Request for Comments No. 121

PROBLEMS OF OFFICE AUTOMATION, OR WHY WE DON'T USE MULTICS FOR TYPING IMPORTANT DOCUMENTS

by J. H. Saltzer

In a recent conversation with Robert Kahn, he asked why a document I had just written wasn't "on the network" so he could look it over immediately rather than wait for the post office to carry a copy to him. At the time, my only response was "we don't usually put rush jobs on-line," which only raised more questions. After giving it some thought, it became apparent that there are several problems that must be overcome before the office automation concept is going to be a success.

First of all, availability of terminals and cost of computer time to do the job happen not to be problems in our current environment. We have a large supply of both, provided to carry out the larger research goals of the division, and document preparation has been encouraged as a legitimate means of harnessing the computer to make more effective use of people. Also, scheduled availability of computer time is not a problem. The M.I.T. Multics service is scheduled to be available at essentially all times that document preparation or review might take place.

So what is the problem? Why do I tell my secretary "Let's type this off-line" when I am in a hurry to see the result? There are several things to observe :

1) Keyboard quality. My secretary is a very fast and very accurate typist, when working with her IBM Model D office electric typewriter. When she switches to a computer terminal, her typing speed goes down and her error rate goes up; both changes produce a noticeable effect on office routine. The primary cause seems to be that computer terminals come with keyboards ranging from poor to awful. Even the best terminal keyboards we have been able to find are obviously inferior to a good office typewriter.

This note is an informal working paper of the M.I.T. Laboratory for Computer Science, Computer Systems Research Division. It should not be reproduced without the author's permission and it should not be referenced in other publications.

The problems with the keyboard are many: probably the most important is that there is no mechanical interlock among the keys, so a tactile feedback mechanism that is unnoticed by the hunt-and-peck artist but is important to the rapid touch typist is missing. "Electronic rollover", while sometimes provided, is not a replacement for mechanical rollover as far as a fast typist is concerned. And, there are usually other problems: the key springs are too weak or too strong; there is no noticeable "breakthrough" as the key is struck; the noise of the terminal type element (on hard copy terminals) is not synchronized with the key stroke; sometimes the keyboard is planar rather than concave; the carriage return key may be small or located so far from the home position that the hand must be moved. Before office automation can be widespread, someone with experience in designing keyboards for office typewriters is going to have to be involved in the design of a computer terminal.

2) Human engineering of the document preparation system. There seem to be two kinds of document editors in the world: relatively easy-to-learn but not-very-facile line or context editors (such as Multics "edm") and extremely powerful editors that can do everything and more, but that were designed by systems programmers for systems programmers (such as "teco" and "qed"). The contrast is disturbing: my secretary has a choice of being underequipped (which means that editing takes much longer than it should) or of being snowed under by hard-to-learn conventions. APL and BASIC have proved that it is possible to put a clean, human engineered interface on a computer used for computing. But there is no

correspondingly powerful yet easy to learn and easy to use document preparation subsystem. I suspect that such a system, if it is ever designed will come not from a system programmer, but from someone with a broader "outside" perspective, as were the cases with both APL and BASIC. Meanwhile, my secretary uses scissors, Scotch tape, and Snopaque to edit my memos and get them back to me for a review in practically no time at all; the on-line approach is almost always slower.

3) Memo debugging. This problem may be just a subtopic of the previous one, but is worth a harangue of its own. The current strategy of document preparation is that an editor is used to create a file containing text and format control strings; a separate program is invoked to print the text in the intended format. This pattern is essentially analogous to that used by most programmers: use an editor to construct a program, then a computer or interpreter to run the program. And the effect is analogous: documents, like programs, have "bugs", which are detected by examining the output, going back to the editor to repair the original text or format controls, and trying the output program again. The repair may introduce a new "bug", and the cycle is repeated.

The most important impact of this strategy is that time is lost in repeated printouts. The first printed version always has a surprise, and must be repaired; the second one usually does too. Thus a document may be printed three or more times before it is ready for detailed review. In contrast, with the typewriter, the copy produced by the initial keyboarding is ready for review with no waiting.

Modern display technology should be adequate to allow the finished document to always appear on the screen, during initial input and during editing, so that the effect of all changes can be instantly observed, and so that editing can be expressed in terms of the appearance of the finished document rather than in terms of an intermediate file that bears little resemblance to the finished document. I suspect that such a display/editing system is another absolute prerequisite to successful office automation.

4) Availability. Although the local time-sharing system is scheduled to be available at the time it is needed, it often fails to meet its schedule. The monthly crash log will likely report that the mean time between failures has been 24 hours or more, but if there is an important document to get out, one can expect a crash every hour (making up for last week, when we had no documents to prepare, and the system crashed only once all week.) Although a crash rarely causes significant loss of work, it does mean a disruption in office routine, and delay until the system returns to operation, during which time some other office activity is usually initiated. When the system is again available, either this second activity must be interrupted or the on-line activity must be delayed till the second activity is completed. Finally, some time is lost in figuring out "where we were" in the originally interrupted activity and in assessing the extent of damage, if any.

Availability is reduced by failures of all kinds; mostly of shared equipment not under control of my office: system crashes, telephone line outages, failures of modems in the machine room, and so on. Probably the biggest contributor to failures is the shared main frame, used by 1300 customers for diverse purposes, and stressed in many ways. Of all the functions of the shared main frame, the document preparation task requires only the ability to communicate with others; it could otherwise get along superbly on a private computer whose unavailability could be smaller by orders of magnitude.

5) Equipment compatibility. The usual format for documents prepared in our office, as in most business operations, is with elite type (12 columns per horizontal inch) and "space-and-a-half" (4 lines per vertical inch). We use for high quality output of on-line documents a "daisy-wheel" terminal (currently the Trendata 4000), and have no trouble producing good-looking documents in that format. The daisy-wheel printer is slow (30 characters/second) and for review it is preferable to sacrifice quality for speed and use a nearby printer. But the remote line printer types 10 columns per horizontal inch and 6 lines per vertical inch, so the appearance of the output is much different than intended. When editing a document at a video terminal, the overstruck characters are usually fouled up, either by omission of one of the overstruck graphics or by sequential rather than overstruck display. Our video terminals display only 24 lines, so it is not possible to review a whole page in the form it will appear on paper. If we send the memo to

a reader at another site, that reader may have an upper-case-only video terminal, a model 33 teletype, or an APL terminal available, any of which destroy any intended graphical impact of the document. There is a Xerox Graphic Printer (XGP) nearby, but it is located on another floor, its use requires knowing how to transmit files over the ARPANET, and the XGP itself is not a tool that can be quickly used by an amateur.

6) Hyphenation. A trustworthy automatic word hyphenation scheme is not currently available, and a good human engineered interactive hyphenation strategy isn't available either. Thus all on-line documents are unhyphenated, which leads to ragged right margins (elite type helps reduce the impact of this problem, but does not eliminate it) or else to vertical rivers of white space when automatic right-justification is used. Thus, it is hard to get an on-line document that looks as good as one typed the old-fashioned way.

All of the problems described here seem relatively easy to overcome, and probably do not require any very deep research. But their cumulative impact is very great. Decisions to use the on-line document preparation system are made with hesitation, and only when there is some clear benefit to be derived, such as quick communication with a distant reader, avoidance of a duplicate keyboarding, or a plan for many versions of the document with small differences one from another.

High-quality human engineering, though not commonplace, does exist, and in my opinion is the primary missing ingredient in current hopes for a revolution in office automation. Certainly the traditional computer engineering approach, which usually leaves the customer with a barely tolerable interface, is not going to work.