The Sequential Order of Instructions: Impact on Text Quality

Michaël F. Steehouder and Carel J.M. Jansen

In written instructions, the sequential order of procedural steps is crucial for effective and efficient performance. In this paper we demonstrate several "rules" for optimizing instructions in this respect:

- *First things first: put instructions in an order that prevents users from neglecting important steps.*
- *Minimize cognitive load: put instructions in an order that allows readers to forget what they read.*
- *Save time and effort: put instructions in an order that "on average" requires as little time as possible of the readers.*

Although much interest of technical communicators nowadays focuses on new technologies and management of processes, textual issues still demand full attention. After all, in order to be effective, texts have to be understood and applied as well and easily as possible. Our text-analytical research focuses on instructions: texts aimed to support people in completing a task such as programming a VCR, recovering form errors while using computer software, or putting up a new frame tent.

A crucial aspect of the quality of instructions is the concise wording of all steps that the user should take. It is also crucial that the instructions are presented in the exact order in which they have to be carried out. Looking at instructions, it strikes that even this self-evident demand is not always met. Sometimes instructions are simply given in the wrong order, possibly due to carelessness of the writer.

However, this is not the only issue ahead. Even if the order of instructions is acceptable (at first sight), there are several instances of not-optimal ordering. Some of these we will focus on in this paper.

First things first

Before a step of a procedure can be performed, all conditions for that step have to be fulfilled. Data can be changed only after they have been added to a file, a file can be printed only after the printer has been turned on, and one has to put a diskette in drive A: before files can be copied to A:.

Special attention should be given to warnings. By convention, warnings appear often *after* the procedure. That may be too late. Warnings may require actions that have to be performed in advance, as can be seen in example 1 and 2.

How to insert a column?

1 Put the cursor on a random cell of the column left of which the new column has to be inserted.
2 Choose menu options COLUMN and INSERT. On the status line appears: NUMBER:...
3 Type the number of columns to be inserted and press ENTER.

Warning: The spreadsheet cannot contain more than 64 columns. If you insert too many columns, the utmost right columns will be deleted.

Example 1: Instruction with warning afterwards

How to insert a column?

1 The spreadsheet cannot contain more than 64 columns. If you insert too many columns, the utmost right columns will be deleted. Therefore you should first check how many columns can be inserted.
2 Put the cursor on a random cell of the column left of which the new column has to be inserted.
3 Choose menu options COLUMN and INSERT. On the status line appears: NUMBER:...
4 Type the number of columns to be inserted and press ENTER.

Example 2: The procedure starts with the warning, and a extra step has added.

Why do writers give often warnings afterwards? We believe that it has to do with their orientation on technique. Checking the number of columns is not a "technical" step in the procedure (it does not involve actions with the mouse or the keyboard). It is a "psychological" step, often neglected in a task analysis.

Minimize cognitive load

Sometimes readers have to remember information while performing a procedure. This causes "cognitive load" and may lead to errors. By putting steps in the right order, the cognitive load may be minimized, as examples 5 and 6 show.

1	Calculate your income
2	Calculate you partner's income
3	Add 4.5% to your income
4	Add your partner's income to yours (with 4.5% added)

Example 5: The result of (1) must be remembered until (3) has been completed; likewise the result of (2) must be remembered until (4) has been completed.

1	Calculate your income
2	Add 4.5% to your income
3	Calculate you partner's income
4	Add your partner's income to yours (with 4.5% added)

Example 6: The result of (1) can be forgotten after (2); only the result of (2) has been remembered until (4) has been completed.

The cognitive load does not only depend on the number of results that must be remembered before other instructions can be performed. There are two other relevant factors:

- the difficulty of the intermediate actions: a complicated calculation will be more of an obstacle to correctly remembering the information than answering a simple question;
- the nature of the information to be remembered: a round figure such as 2,000 will be easier to remember than for instance 2,163.85.

Saving time and efforts by the right sequential order

To introduce this principle, we start with an example from a Dutch governmental document. In this document the Ministry of Economic Affairs explained under what conditions companies were entitled to a so called "Innovation Investment Subsidy". The brochure started with a detailed explanation of how the Ministry defined the concept "Innovation Investment". After three pages the text was ended with some minor conditions, one of which was that a company should have at least 500 employees to be entitled for the subsidy. Many managers of small companies who had hopefully started reading the brochure, would have preferred to know this condition *before* they read the long and complex section on Innovative Investments.

The example illustrates the importance of careful considering the sequence of instructions. Two rules in particular are important to prevent unnecessary efforts.

1 If possible, difficult and complex steps should be put at the *end* of a procedure. Then only those users who really have to perform that step, have to read it (see example 3).

2 Information relevant for *many* users or readers, should be put at the *beginning*; information relevant for only a *few*, should be punt at the *end* (see example 4).

Sometimes, the two rules do not lead to the same order. The most preferred action might also be the most complicated one. In such cases a compromise has to be found. If we know exactly how many users prefer each of the options and how much time each option takes, it is possible to calculate exactly which order is optimal by formulas derives form operation research. If those figures are not available, it is possible to make a reasonable guess.

If you get the message "Divide overflow", several problems might have occurred.

Possible Problem	How Can You See?
You have loaded too many TSR-programs	1 Save your file 2 Leave the program 3 Use the MEM command to check how much free memory you have. It should be at least 256 KB.
Your file is too large	1 Click on "File" 2 Click on "Info" 3 A window appears that shows among others how large your file is and how much free memory you have. The length of your file should nod exceed the amount of free memory.

Example 3: The order of instructions is not optimal. Since the first procedure is much more complicated and time consuming, user better start with checking if their file is too large. If this is the case, they save time and effort. If this is not the problem, they have to check the other possible problem, but they have not lost much time.

How do you want your figures displayed?

If you want them	Then
in standard letters	click on "OK"
in colored letters	click on "color" and choose your color
in a box	click on "box"
hidden (not visible)	click on "NONE"

Example 4. If the know the proportions of users who prefer each of the option, than we can put the options in the most "ergonomic" order: start with the one preferred by most users. They can stop reading after the first row. Only a small group have to read the last row.

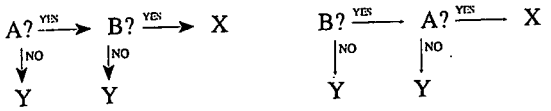
The principle we applied in example 3 and 4 has been developed by the Russian psychologist Lev Landa in his analysis of algorithms in grammar teaching: *That algorithm is more simple and more efficient whose application requires less time on the average* (3, p. 245). To explain this approach, we use a (very) simple example:

If you can use the XYZ Program and if you have at least 256 KB of free memory, press Ctrl-C, otherwise, press <Return>.

Or, symbolically:

If (and only if) A and B, than X, otherwise Y.

The reader has to verify two conditions: A and B. Which order would be the best? There are two possibilities.



To decide between A-first and B-first, two key figures are needed:

- The distribution of each question, i.e. the proportion of readers that will answer "yes". The distribution of a question Q *(dQ)* is expressed as a number between 0 and 1.
- The time the average reader will need to answer the question. This amount of time in seconds *(tQ)* represents the difficulty of the question.

If the *d*- and *t*-values of both questions A and B are known, it is possible to calculate which of the two procedures will take less time on the average. Suppose

$dA = 0.5$ and $tA = 3$
$dB = 0.2$ and $tB = 2$

The mean time needed perform the procedure with the A-first ordering van be calculated as follows. All users have to answer question A. This takes an average of 3 seconds. For those 50% of the readers that answer question A negative, the outcome (Y) is clear; they need no more time. The other 50% have to answer B, which takes them 2 seconds on average. So they need 5 seconds in total to find the outcome (Y or X). The mean time needed for all readers is therefore *(3 + 5) / 2* seconds, which makes 4 seconds.

In the same way, it can be calculated that the mean time needed for the B-first version of the instruction is equal to 2.6 seconds. The conclusion is that, given the values of *t* and *d*, the B-first version takes less time, and thus is more efficient than the A-first version.

As stated, this is only a simple example. In reality, instructions are much more complicated, and finding the optimal order of instructions requires more extensive calculations to find the most efficient order of instructions (1). Moreover, the exact values of *d* and *t* are not always known (usability testing may help to approximate these values). Heuristics have been developed, less exact than the method we demonstrated, but nevertheless lead to reliable decisions (1).

Practical applications

The analysis and calculations demonstrated in this paper can be applied in a variety of instructional materials. We applied them for finding the optimal order of questions in government forms (2, 4) and for optimizing instructions in software manuals. Other applications are educational materials and administrative procedures.

References

(1) Jansen, C.J.M. & Steehouder, M.F. (1989). *Taalverkeersproblemen tussen overheid en burger [Communication problems between government and citizens]*. The Hague: Sdu Uitgeverij.

(2) Jansen, C.J.M. & Steehouder, M.F. (1992). Forms as a source of communication problems. *Journal of Technical Writing and Communication 22*, 179-194.

(3) Landa, L.N. (1974). Algorithmization in learning and instruction. Englewood Cliffs: Education Technology Publishing.

(4) Steehouder, M. & Jansen, C. (1987). From bureaucratic language to instructional texts: how to design an effective problem-solving tool for citizens. *Information Design Journal 5*, 129-139.

Carel J.M. Jansen
Eindhoven University of Technology
P.O. Box 513 TeMa Building 0.25
5600 MB Eindhoven, The Netherlands
Tel. +31 40 247 3746

Carel Jansen teaches language and technical communication at Eindhoven University of technology and Utrecht University (both in The Netherlands). Together with Michaël Steehouder he wrote text books about communication *(Leren communiceren)*, forms design *(Formulierenwijzer)* and computer manuals *(Handleidingenwijzer)*. He is also the (co)-author of many articles on technical communication.

Michaël F. Steehouder
University of Twente
PO Box 217
7500 AE Enschede, The Netherlands
Tel +31 53 489 3315

Michaël Steehouder teaches communication at the University of Twente in The Netherlands. He was co-organizer of several conferences, such as *Quality of Technical Documentation* (Enschede 1992) and *Forum'95* (Dortmund, 1995). He is editor of the Dutch professional writers journal *Tekst[blad]*.