9. J. Richardson, A. Dillon, and C. McKnight, The Effect of Window Size on Reading and Manipulating Electronic Text, *in Contemporary Ergonomics 1989*, E. Megaw (ed.), Taylor & Francis, London, 1989.
10. A. Dillon, J. Richardson, and C. McKnight, The Effect of Display Size and Paragraph Splitting on Reading Lengthy Text from Screen, *Behaviour and Information Technology, 9*, pp. 215-227, 1990.
11. S. de Mul, H. van Oostendorp, and T. White, Meervoudige Documentverwerking in Meervoudige Vensters (Multiple Processing of Documents in Multiple Windows), in *Informatiewetenschap 1991*, G. Kempen and P. de Vroomen (eds.), Stinfon, Nijmegen, 1991.
12. S. E. Davies, K. F. Bury, and M. J. Darnell, An Experimental Comparison of a Windowed vs. a Non-Windowed Operating System Environment, *Proceedings of the Human Factors Society 29th Annual Meeting*, The Human Factors Society, Santa Monica, California, pp. 250-254, 1985.
13. H. Hoeken, M. Mom, and A. Maes, Translating Hierarchical Instructions into Linear Text: Depth-First versus Breadth-First Approaches, in *Quality of Technical Documentation*, M. Steehouder, C. Jansen, P. van der Poort, and R. Verheijen (eds.), Rodopi, Amsterdam/Atlanta, Georgia, pp. 99-113, 1994.
14. P. Wright, Quality or Usability? Quality Writing Provokes Quality Reading, in *Quality of Technical Documentation*, M. Steehouder, C. Jansen, P. van der Poort, and R. Verheijen (eds.), Rodopi, Amsterdam/Atlanta, Georgia, 1994.

Other Publications On Communication By These Authors

A. Maes, S. Goutier, and E.-J. van der Linden, Online Reading and Offline Tradition. Adapting Online Help Facilities to Offline Reading Strategies, *Proceedings of SIGDOC92, the 10th Annual International Conference on Systems Documentation*, Ottawa, Canada, pp. 175-182, 1992.

A. Maes, Language at Work, Instructive Communication and Advanced Writing, in *Proceedings of Colloquium on Business and Communication*, E. Bogaert, M. van de Velde, and A. Vermeire (eds.), ABLA, Gent, pp. 69-77, 1993.

H. Hoeken, M. Mom, and A. Maes, Translating Hierarchical Instructions into Linear Text: Depth-First versus Breadth-First Approaches, in *Quality of Technical Documentation*, M. Steehouder, C. Jansen, P. van der Poort, and R. Verheijen (eds.), Rodopi, Amsterdam/Atlanta, Georgia, pp. 99-113, 1994.

Direct reprint requests to:

Dr. A. A. Maes
Letterenfaculteit KUB
PO Box 90 153
5000 LE Tilburg
The Netherlands

J. TECHNICAL WRITING AND COMMUNICATION, Vol. 26(4) 453-473, 1996

THE SEQUENTIAL ORDER OF PROCEDURAL INSTRUCTIONS: SOME FORMAL METHODS FOR DESIGNERS OF FLOW CHARTS

CAREL J. M. JANSEN
Eindhoven University of Technology
Utrecht University

MICHAËL F. STEEHOUDER
University of Twente, Enschede

ABSTRACT

Document designers who present procedural instructions can choose several formats: prose, table, logical tree, or flow chart. In all cases, however, it is essential that the instructions are ordered in a way that allows users to reach the outcome in as little time as possible. In this article two formal methods are discussed that help determine which order is most efficient. The first method is based on the selection principle. The second method is based on the principle of the average least effort.

Most current literature on instructional texts focuses on "global" design issues, such as minimalism versus elaboratism, procedural information versus declarative information, and words versus graphics. Nevertheless, concerns about narrower questions are still relevant. In spite of the broad research already done in these smaller issues, many questions still have to be answered. Take for instance questions about syntactical issues such as the effect of using passives [1], terminology (when to use certain kinds of metaphors [2]), typeface, and so on. In this article we focus on a topic that so far has drawn surprisingly little attention in the professional literature on instructional texts: the sequential ordering of instructions. The issue can nevertheless be regarded as fundamental, since instructional texts typically refer to actions that have to be performed sequentially.[1]

It seems obvious that instructions should be presented in the order in which they must be carried out. Instructions for a frame tent should not end with an added note that tells the camper he should have "put the plastic caps on the poles before putting up the roof." Such an instruction comes too late in the day and frustrates the reader. Perhaps, this (real-life) example may seem like an incident, but the fact that warnings often appear after the stepwise instructions instead of before, illustrates that our point is not far-fetched.

It is not *always* obvious, though, what exactly the optimal order of instructions should be: In this article we will discuss procedures where actions may be performed in any order without affecting the *effectiveness*: the extent to which users will be able to complete their tasks successfully. But, as we will show, sometimes the order of instructions can have a dramatic effect on the *efficiency* of such a procedure: the time users have to take to complete their tasks.

SOME DESIGN ALTERNATIVES

Let us explore some of the alternatives text designers can opt for in a given situation. We use the example of an "alarm screen" that informs the user that the system has halted (Figure 1). The instructions tell the users what to do when this screen appears. A conventional way to draw up such an instruction is to use "stepwise prose" (Figure 2). Another option is to use a decision table (Figure 3). Such tables enable the users to decide what they have to do more quickly and more accurately [5].

Many text designers might prefer the format of a logical tree or decision tree, which *shows* the relationships between the possible states of the windows and the actions to be performed (Figure 4). Logical trees were already advocated by several authors in the sixties and seventies [6-8]. Wright and Reid showed that users work better and more quickly with logical trees if they must solve relatively complex problems, but that tables are to be preferred if users must learn how to complete tasks [5].

Another alternative format is a *flow chart* (Figure 5). Several experiments showed that flow charts generally are a more effective and efficient means to solve problems than prose [9-13]. In these experiments, no sharp distinction was made between a logical tree and a flow chart. However, there is an essential difference between the representations in Figures 4 and 5. Figure 4 shows all possible states of the three windows, and users must *identify* which state is

[1] The issue of optimizing the sequential order is extensively discussed in [3] with respect to the ordering of questions on (government) forms. Those who are interested in more detail, can order an English translation of the relevant chapter from the authors. A short introduction to the topic is given in [4].

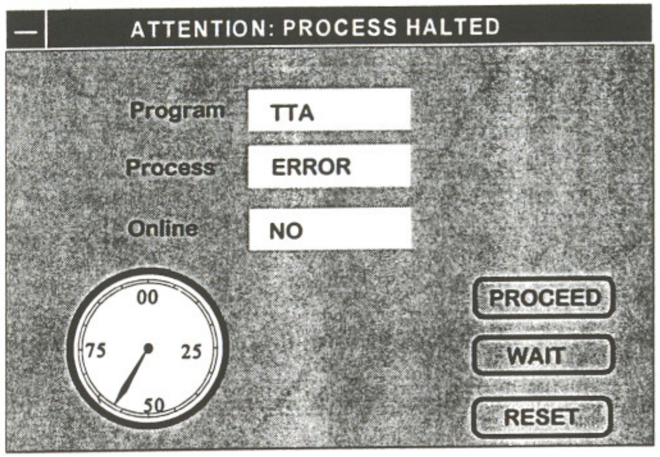


Figure 1. Alarm screen.

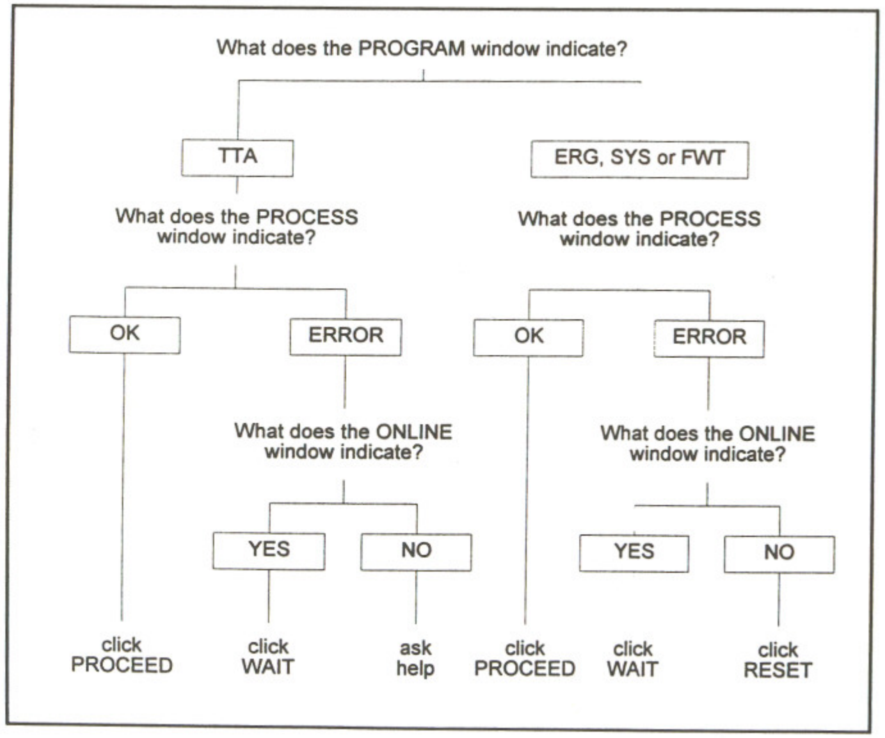
If the PROGRAM window indicates TTA, look at the PROCESS window and the ONLINE window
- If the PROCESS window indicates OK: click PROCEED
- If the PROCESS window indicates ERROR, and the ONLINE window indicates YES: click WAIT
- If the PROCESS window indicates ERROR, and the ONLINE window indicates NO: ask your system manager for help

If the program window indicates ERG, SYS or FTW, look at the PROCESS window and the ONLINE window
- If the PROCESS window indicates OK: click PROCEED
- If the PROCESS window indicates ERROR, and the ONLINE window indicates YES: click WAIT
- If the PROCESS window indicates ERROR, and the ONLINE window indicates NO: click RESET

Figure 2. Stepwise prose format.

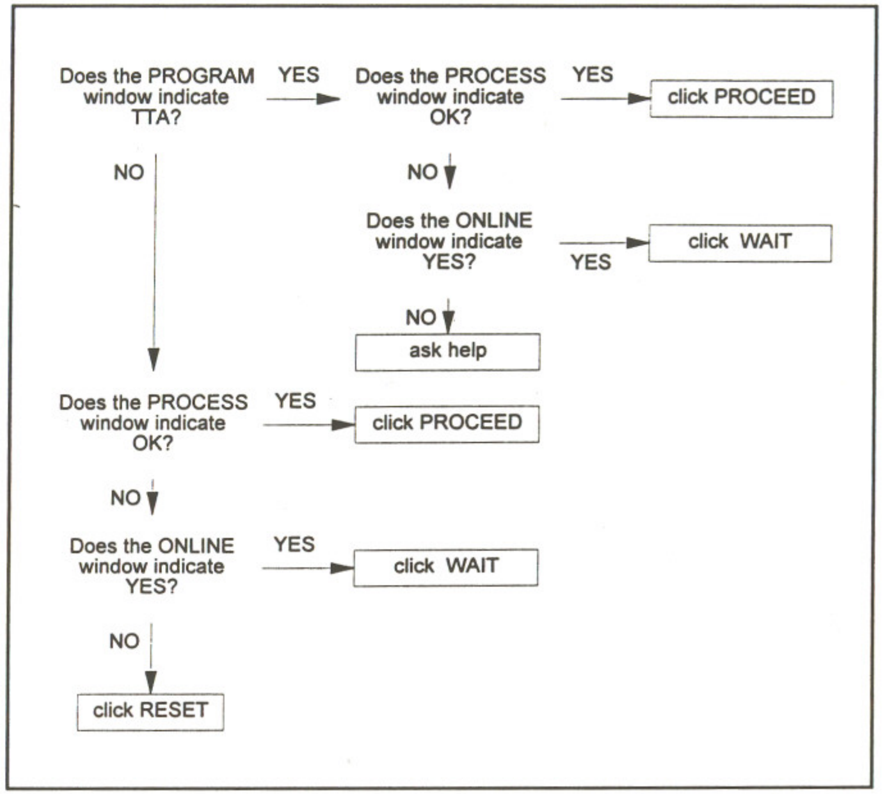
If the PROGRAM *window indicates*	*and the* PROCESS *window indicates*	*and the* ONLINE *window indicates*	*then*
TTA	OK	YES or NO	click PROCEED
TTA	ERROR	YES	click WAIT
TTA	ERROR	NO	ask your system manager for help
ERG, SYS or FWT	OK	YES or NO	click PROCEED
ERG, SYS or FWT	ERROR	YES	click WAIT
ERG, SYS or FWT	ERROR	NO	click RESET

Figure 3. Decision table format.



What does the PROGRAM window indicate?
TTA
ERG, SYS or FWT
What does the PROCESS window indicate?
What does the PROCESS window indicate?
OK
ERROR
OK
ERROR
What does the ONLINE window indicate?
What does the ONLINE window indicate?
YES
NO
YES
NO
click PROCEED
click WAIT
ask help
click PROCEED
click WAIT
click RESET

Figure 4. Logical tree format.



Does the PROGRAM window indicate TTA?
YES
Does the PROCESS window indicate OK?
YES
click PROCEED
NO
NO
Does the ONLINE window indicate YES?
YES
click WAIT
NO
ask help
Does the PROCESS window indicate OK?
YES
click PROCEED
NO
Does the ONLINE window indicate YES?
YES
click WAIT
NO
click RESET

Figure 5. Flow chart format.

indicated in each window. Figure 5 shows only one state, requiring the reader to *verify* whether this is the one on the screen or not. It is not completely clear which option is the best. Although Barnard, Wright, and Wilcox found that people completing forms answered questions more quickly and accurately if they had alternatives in a sentence frame (*I am single/married*)—cf. the logical tree format—than if they consisted of yes/no questions (*Are you married?*)—cf. the flow chart format—it is not clear whether this effect can be generalized to instructions like those discussed here, where there are more than two alternatives that are less familiar to the user.

Fow now, we will assume that there are no strong arguments in favor of either the decision tree format or the flow chart format. For the sake of clarity, the discussion below will be focused on flow charts, but the principles involved can also be applied to other formats.

If the PROGRAM *window indicates*	*and the* PROCESS *window indicates*	*and the* ONLINE *window indicates*	*then*
TTA	OK	YES or NO	click PROCEED
TTA	ERROR	YES	click WAIT
TTA	ERROR	NO	ask your system manager for help
ERG, SYS or FWT	OK	YES or NO	click PROCEED
ERG, SYS or FWT	ERROR	YES	click WAIT
ERG, SYS or FWT	ERROR	NO	click RESET

Figure 3. Decision table format.

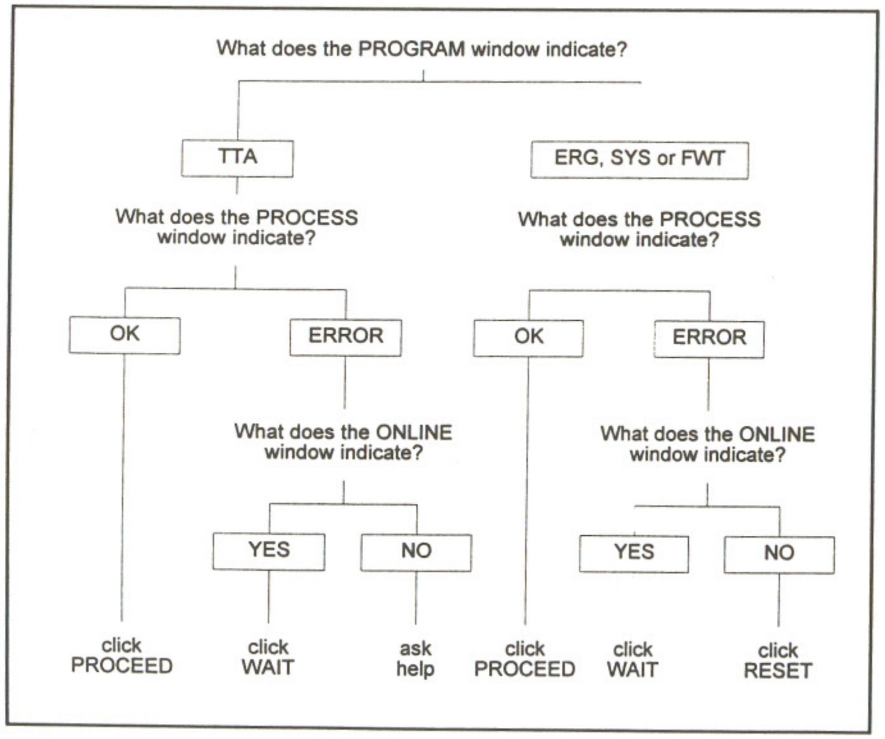

What does the PROGRAM window indicate?
TTA
ERG, SYS or FWT
What does the PROCESS window indicate?
What does the PROCESS window indicate?
OK
ERROR
OK
ERROR
What does the ONLINE window indicate?
What does the ONLINE window indicate?
YES
NO
YES
NO
click PROCEED
click WAIT
ask help
click PROCEED
click WAIT
click RESET

Figure 4. Logical tree format.

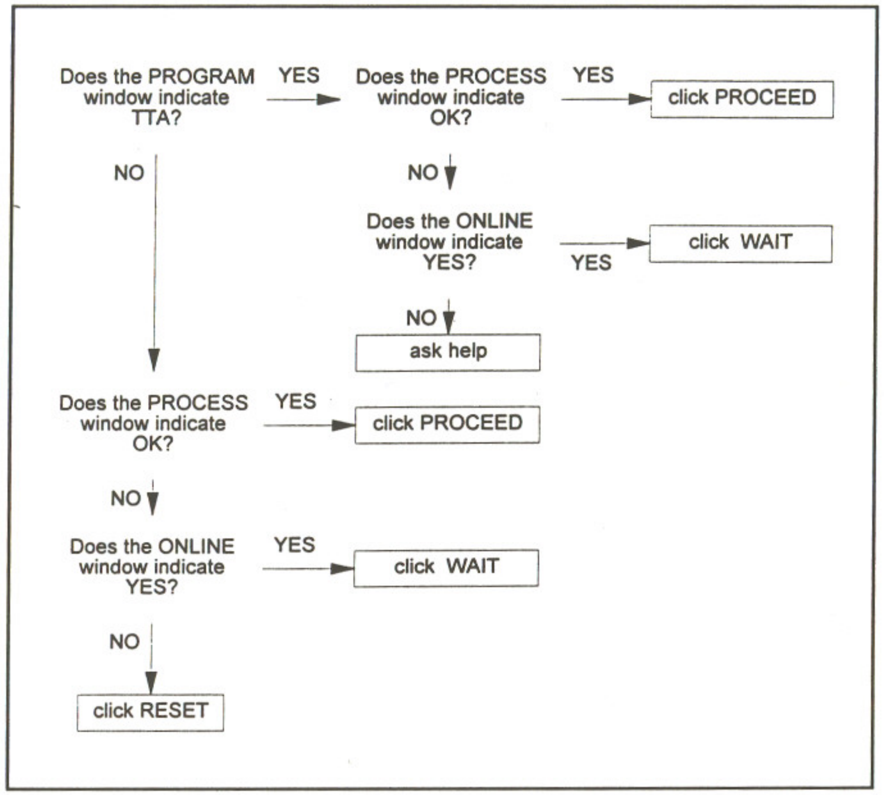

Does the PROGRAM window indicate TTA?
YES
Does the PROCESS window indicate OK?
YES
click PROCEED
NO
NO
Does the ONLINE window indicate YES?
YES
click WAIT
NO
ask help
Does the PROCESS window indicate OK?
YES
click PROCEED
NO
Does the ONLINE window indicate YES?
YES
click WAIT
NO
click RESET

Figure 5. Flow chart format.

indicated in each window. Figure 5 shows only one state, requiring the reader to *verify* whether this is the one on the screen or not. It is not completely clear which option is the best. Although Barnard, Wright, and Wilcox found that people completing forms answered questions more quickly and accurately if they had alternatives in a sentence frame (*I am single/married*)—cf. the logical tree format—than if they consisted of yes/no questions (*Are you married?*)—cf. the flow chart format—it is not clear whether this effect can be generalized to instructions like those discussed here, where there are more than two alternatives that are less familiar to the user.

Fow now, we will assume that there are no strong arguments in favor of either the decision tree format or the flow chart format. For the sake of clarity, the discussion below will be focused on flow charts, but the principles involved can also be applied to other formats.

Experiments by Holland and Rose revealed that the time readers need to find an outcome in a flow chart is proportionally dependent on the number of questions they must answer to get to the outcome [10]. In our example this leads to the question whether there might be an alternative flow chart that lets the users reach the outcome in fewer steps than required by the flow chart in Figure 5. Such a flow chart can indeed be designed (Figure 6). Users who have to click on PROCEED have to answer only one question in Figure 6, against two questions in Figure 5. And users who have to click WAIT, have to pass only two questions in Figure 6, while they have to answer three questions in Figure 5.

THE SELECTION PRINCIPLE

Even in a simple case such as our example, finding the most efficient flow chart could be a laborious puzzle for a technical writer. A helpful method, though, has been developed by Wheatley and Unwin [15]. The basic idea is that a flow chart is more efficient if the user only has to answer questions that are absolutely necessary in deciding what to do in a given situation. We will use our example to demonstrate the method.

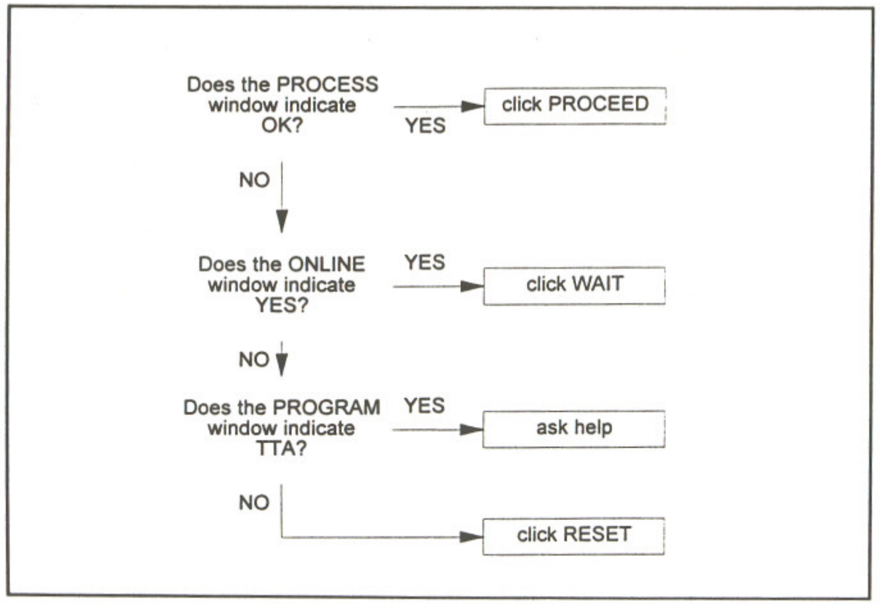

Does the PROCESS window indicate OK?
YES
click PROCEED
NO
Does the ONLINE window indicate YES?
YES
click WAIT
NO
Does the PROGRAM window indicate TTA?
YES
ask help
NO
click RESET

Figure 6. Flow chart format: an alternative.

We start by looking at the actions that the users have to perform in the end. There are four possibilities:

1. click PROCEED,
2. click WAIT,
3. click RESET, and
4. ask for help.

We call these the *outcomes* of the procedure. Which outcome applies to a given situation depends on three *conditions*: the status of the PROGRAM, PROCESS, and ONLINE windows. Each condition can have the value *true* (yes, +) or *false* (no, –). A bit of mathematics teaches us that three conditions yield $2^3 = 8$ possible combinations of values. These can be represented in a *logic table*. Each column represents one combination of values, associated with the appropriate outcome (Figure 7).

If we consider this table, we can easily see that columns 1 and 2 share the same outcome and differ in *only one row*, that of the ONLINE window. This means that the status of the ONLINE window turns out to be irrelevant for the outcome in the situations presented by columns 1 and 2. If the PROGRAM window indicates TTA and the PROCESS window indicates OK, the outcome always is *click PROCEED*, whatever the status of the ONLINE window might be. When two columns are exactly the same, except for one row, as in this case, we call them a *pair of columns*.

	STATES							
	1	2	3	4	5	6	7	8
CONDITIONS: PROGRAM window TTA?	+	+	+	+	–	–	–	–
PROCESS window OK?	+	+	–	–	+	+	–	–
ONLINE window YES?	+	–	+	–	+	–	+	–
ACTIONS	click PROCEED	click PROCEED	click WAIT	ask help	click PROCEED	click PROCEED	click WAIT	click RESET

Figure 7. Logical table representing conditions, states, and actions.

Wherever we find such a pair, we can simplify the table by crossing out the irrelevant plus and minus. Two identical columns are left now (1 and 2) which makes the table redundant: one of the two can be crossed out. This operation leads us to the table shown in Figure 8.

There is still another pair of columns, however: 5 and 6. Again the columns are exactly the same, except for one row. When the PROGRAM window does not indicate TTA, and the PROCESS window does indicate OK, the outcome is *click PROCEED*, no matter what the ONLINE window tells us. So within these columns the ONLINE window is irrelevant and can be crossed out (Figure 9).

The trimming of the table has not been completed yet. Figure 9 still contains a pair of columns, this time with some distance between them: column 3 and column 7. We apply the same procedure to simplify the table once more (Figure 10).

We still have not reached the end: there is still a pair of columns that enables us to simplify the column. Columns 1 and 5 show that for the outcome *click PROCEED* the status of the PROGRAM window is not relevant.

Figure 11 shows the final boundary. There are no more pairs of columns and so, further simplification is not possible. Figure 11 shows that the eight situations we started with can be reduced to no more than four different situations without loss of information. This table will be the starting point for the second part of our operation: constructing the most efficient flow chart.

In our table only one row is completely filled with pluses and minuses: the second one. This means that the corresponding condition (PROCESS window OK) is the only one that must be verified in *all* situations. The other two conditions need to be verified only if the PROCESS window is *not* OK. Since all users must

	1	2	3	4	5	6	7	8
PROGRAM window TTA?	+	+	+	+	−	−	−	−
PROCESS window OK?	+	+	−	−	+	+	−	−
ONLINE window YES?	+	−	+	−	+	−	+	−
	click PROCEED	click PROCEED	click WAIT	ask help	click PROCEED	click PROCEED	click WAIT	click RESET

	1	3	4	5	6	7	8
PROGRAM window TTA?	+	+	+	−	−	−	−
PROCESS window OK?	+	−	−	+	+	−	−
ONLINE window YES?		+	−	−	−	+	−
	click PROCEED	click WAIT	ask help	click PROCEED	click PROCEED	click WAIT	click RESET

Figure 8. Columns 1 and 2 make up a 'pair': the ONLINE row is irrelevant.

	1	3	4	5	6	7	8
PROGRAM window TTA?	+	+	+	−	−	−	−
PROCESS window OK?	+	−	−	+	+	−	−
ONLINE window YES?		+	−	+	−	+	−
	click PROCEED	click WAIT	ask help	click PROCEED	click PROCEED	click WAIT	click RESET

	1	3	4	5	7	8
PROGRAM window TTA?	+	+	+	−	−	−
PROCESS window OK?	+	−	−	+	−	−
ONLINE window YES?		+	−		+	−
	click PROCEED	click WAIT	ask help	click PROCEED	click WAIT	click RESET

Figure 9. Columns 5 and 6 make up a 'pair': the ONLINE row is irrelevant.

	1	3	4	5	7	8
PROGRAM window TTA?	+	+	+	−	−	−
PROCESS window OK?	+	−	−	+	−	−
ONLINE window YES?		+	−		+	−
	click PROCEED	click WAIT	ask help	click PROCEED	click WAIT	click RESET

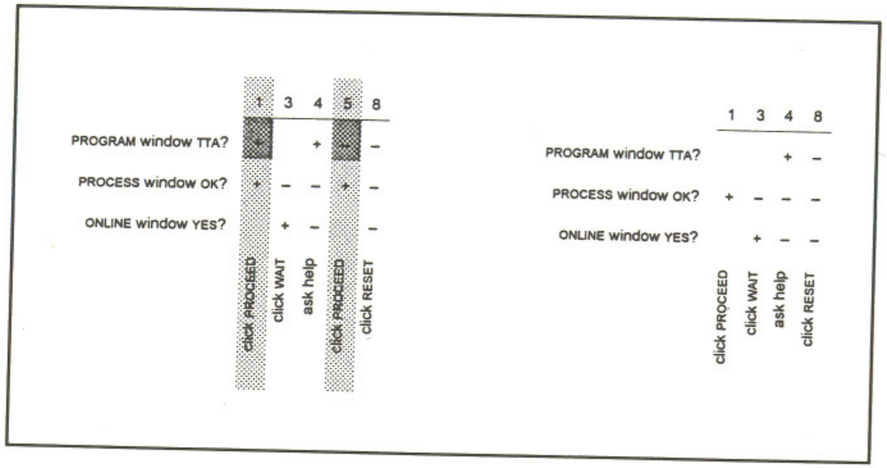
	1	3	4	5	8
PROGRAM window TTA?	+		+	−	−
PROCESS window OK?	+	−	−	+	−
ONLINE window YES?		+	−		−
	click PROCEED	click WAIT	ask help	click PROCEED	click RESET

Figure 10. Columns 3 and 7 make up a 'pair': the PROGRAM row is irrelevant.

verify the second condition, it is logical to start the flow chart with that one (Figure 12).

Looking again at the PROCESS line in Figure 11, we see that there is only one action for a true (+) condition: click PROCEED. We can put this action at the end of the YES arrow.

If the PROCESS window does *not* show OK, there still are three possible outcomes, depending on the two conditions that are left (Figure 13). Again we



	1	3	4	5	8
PROGRAM window TTA?	+		+	+	–
PROCESS window OK?	+	–	–	+	–
ONLINE window YES?		+	–		–
	click PROCEED	click WAIT	ask help	click PROCEED	click RESET

	1	3	4	8
PROGRAM window TTA?			+	–
PROCESS window OK?	+	–	–	–
ONLINE window YES?		+	–	–
	click PROCEED	click WAIT	ask help	click RESET

Figure 11. Columns 1 and 5 make up a 'pair': the PROGRAM row is irrelevant.

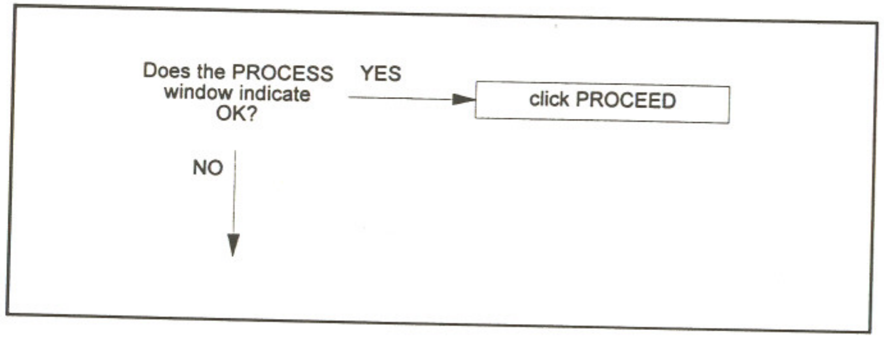
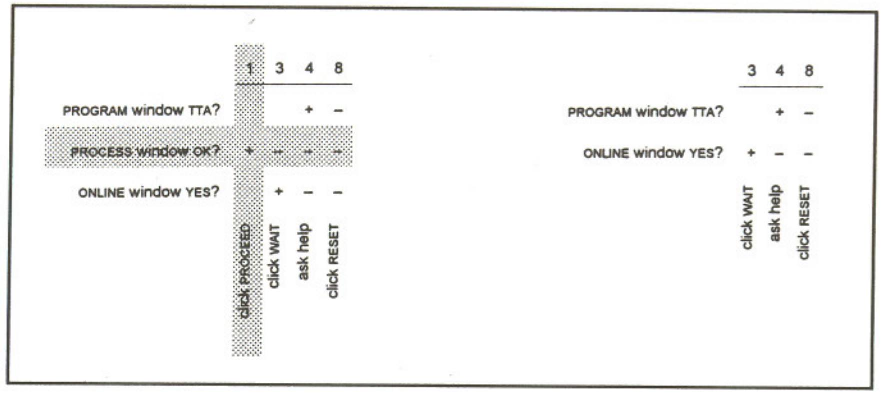


Figure 12. First step in the flow chart.

start with the condition that has most pluses and minuses in its row. In this case, it is the condition in the second row: *ONLINE window YES*. This is the condition that we place under to the *no*-arrow of the flow chart. After that, it easy to complete the flow chart in accordance with the table. The result is shown in Figure 14, which is the same as Figure 6 on page 458.

DRAWBACKS OF THE SELECTION PRINCIPLE

The selection principle results in the most efficient flow chart for our instructional text: the number of questions that need to be answered to find the correct outcome, is minimal. The principle's relevance is not restricted to constructing



	1	3	4	8
PROGRAM window TTA?			+	–
PROCESS window OK?	+	–	–	–
ONLINE window YES?		+	–	–
	click PROCEED	click WAIT	ask help	click RESET

	3	4	8
PROGRAM window TTA?		+	–
ONLINE window YES?	+	–	–
	click WAIT	ask help	click RESET

Figure 13. The PROCESS row and column 1 have been recorded in the flow chart. After removing them, the right hand table is left over.

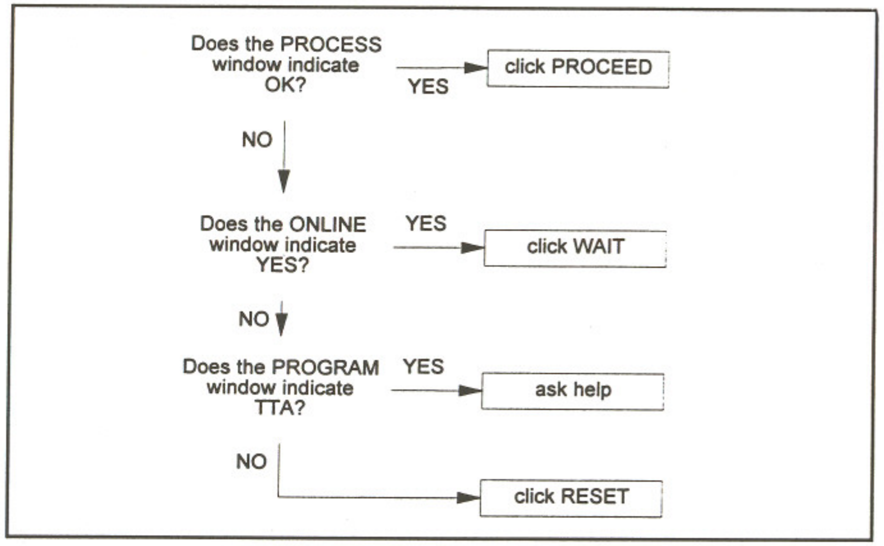

Does the PROCESS window indicate OK?
YES
click PROCEED
NO
Does the ONLINE window indicate YES?
YES
click WAIT
NO
Does the PROGRAM window indicate TTA?
YES
ask help
NO
click RESET

Figure 14. Most efficient flow chart.

flow charts. Instructions in the stepwise prose format or the decision tree format can also be simplified by putting the elements in the most efficient order.

One objection to the final flow chart might be that the order of conditions no longer matches the order of the three windows on the screen (*program-process-online*). This might be a dilemma for the technical writer who has to design the manual: Should the order be adjusted to the interface, or should a more efficient order be preferred? As a matter of fact, it turns out to be a false dilemma. The solution is to redesign the interface according to the most efficient order. The selection principle should not be applied by designers of instructions only, but also by interface designers.

Another drawback might be that the method becomes very cumbersome as the number of conditions increases. In fact, if we have *n* conditions, the number of columns in our table would be 2^n. What we need is a computer program that applies the procedure automatically.

The most serious disadvantage of the selection principle, however, is that it is not always decisive. Let us have a look at another example. Suppose that the alarm screen of Figure 1 requires the following instruction:

If the program window indicates TTA, and the pointer of the tester indicates more than 50, then click RESET. Otherwise click WAIT.

Following the procedure sketched above, we start with the logical table shown in Figure 15, which can be simplified as in Figure 16. It tells us that the procedure should start by verifying the program window (Figure 16a).

But there is another possibility. Since not only columns 3 and 4 make up a pair, but also columns 2 and 4, we can come to another simplified table which tells us exactly the opposite: the procedure should start with verifying the pointer! (see Figures 17 and 17a).

It is clear that the selection principle does not lead to a decision here. But this does not mean that the order of instructions is irrelevant in such cases. Suppose we know that when our alarm screen occurs, 80 percent of the cases show TTA on the ONLINE window, while in 10 percent of the cases the pointer indicates more than 50. Then it would make sense to start the procedure by looking at the pointer. That would lead the users directly to the outcome in 90 percent of the cases (when the pointer does *not* indicate over 50), while only in 10 percent will they still have to look at the program window.

	1	2	3	4
PROGRAM window indicates TTA	+	+	–	–
pointer indicates over 50	+	–	+	–
	click RESET	click WAIT	click WAIT	click WAIT

Figure 15. Decision table representing the new example.

	1	2	3	4
PROGRAM window indicates TTA	+	+	–	–
pointer indicates over 50	+	–	+	–
	click RESET	click WAIT	click WAIT	click WAIT

	1	2	3
PROGRAM window indicates TTA	+	+	–
pointer indicates over 50	+	–	
	click RESET	click WAIT	click WAIT

Figure 16. Columns 3 and 4 make up a 'pair': pointer row is irrelevant.

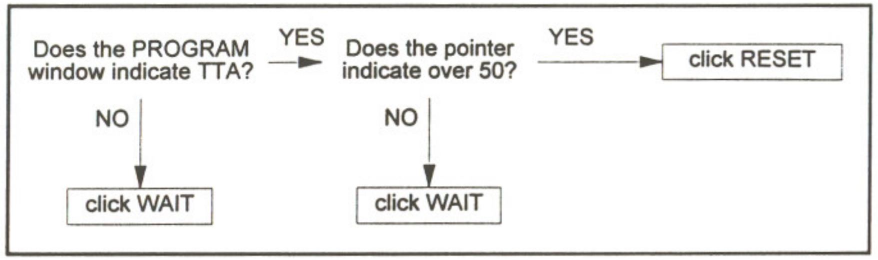

Does the PROGRAM window indicate TTA?
YES
Does the pointer indicate over 50?
YES
click RESET
NO
click WAIT
NO
click WAIT

Figure 16a. Flow chart based upon Figure 16.

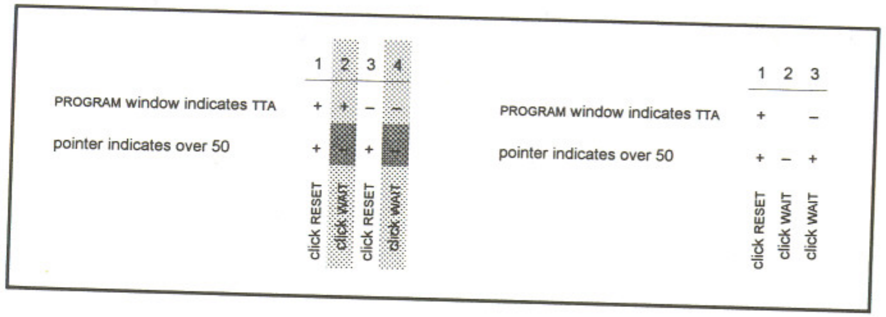
Now suppose that the instruction was a bit different: *If the program PROCESS window indicates TTA, and the pointer indicates over 50 for more than thirty seconds, then click RESET; otherwise click WAIT.*

In this case it would probably be more efficient to start by looking at the ONLINE window, because if this does not indicate TTA, users do not have to wait thirty seconds to verify the other condition. Even if there is only 20 percent chance that the online window does not indicate TTA, this order would probably save time "on average."

Below we will present a method to determine with *certainty* in cases like this what the most efficient order of instructions would be.

THE PRINCIPLE OF AVERAGE LEAST EFFORT

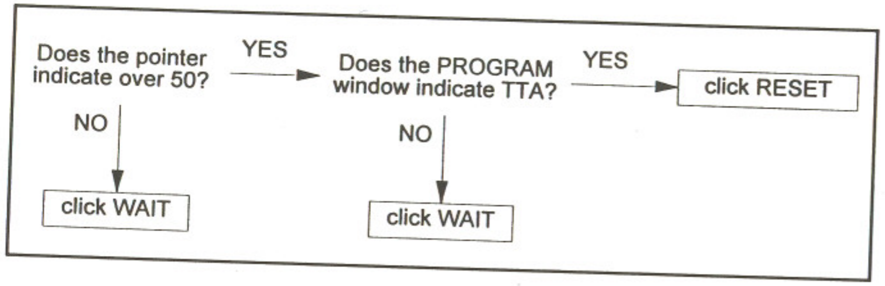
To determine the optimal order of instructions, if the selection principle does not work, we can benefit from the work by the educational psychologist Landa [16]. While striving to improve the teaching of Russian grammar in secondary schools, Landa applied the *principle of the average least effort.* As we will show, this principle applies to more than just grammar teaching.



	1	2	3	4
PROGRAM window indicates TTA	+	+	−	−
pointer indicates over 50	+	−	+	−
	click RESET	click WAIT	click RESET	click WAIT

	1	2	3
PROGRAM window indicates TTA	+		−
pointer indicates over 50	+	−	+
	click RESET	click WAIT	click WAIT

Figure 17. Columns 2 and 4 make up a 'pair': pointer row is irrelevant.



Does the pointer indicate over 50?
YES
Does the PROGRAM window indicate TTA?
YES
click RESET
NO
click WAIT
NO
click WAIT

Figure 17a. Flow chart based upon Figure 17.

The starting assumption is that the flow chart that requires the least average time to complete, is the most efficient. Landa deals at length with the way in which the principle of average least effort can be applied to determine the most efficient flow chart with two possible outcomes. We will briefly summarize his approach here. A certain degree of abstraction in doing so is inevitable.

First of all, in applying the principle of average least effort, two key figures are needed: *distribution* and *time*. The *distribution* of a condition indicates the proportion of cases in which the condition is true. Distribution is expressed as a number between 0 and 1. The *time* required for a condition is the figure indicating how much time on average users need to implement a verification. This can be expressed in seconds. For distribution and time, we will use the following symbols: if C is a condition, then dC is the distribution of the verification and tC is the time of the verification.

If the values of dC and tC are known for all conditions in a flow chart, a calculation can be made of the average time spent on implementing the flow chart as a whole. If there is more than one possible version of the flow chart, the time required for each version can be determined. Comparing the results will indicate which version is the most efficient.

This procedure can be illustrated with a simple example. Consider an instruction of the form: *If and only if A and B then do P; otherwise do Q*. Two versions of the flow chart are possible to determine whether P or Q applies (see Figure 18).

Suppose the following data are available:

$$dA = 0.5 \qquad tA = 3$$
$$dB = 0.2 \qquad tB = 2$$

The average time spent on version 1 can now be calculated as follows. All users have to verify A. This takes an average of three seconds. In cases where A is negative, the outcome (Q) is known and no more time is required for B. In cases where A is positive, B has also to be verified. For 50 percent of the users an extra two seconds must be added on. On average, therefore, an extra *0.5 * 2* seconds is added. The total average time spent therefore is four seconds. In the same way we can calculate that the total average time spent on version (2) is equal to 2 + *(0.2 * 3) = 2.6* seconds. According to the given values of *d* and *t*, version 2 would be more efficient than version 1.

Obviously, in more complicated situations than this example it would take much more effort to calculate what the total time spent by all users would be for every possible flow chart. But, unfortunately, that is not necessary. Some rather simple formulae enable the document designer to determine what the most efficient flow chart will be according to the principle of average least effort. We will show the derivation of these formulae.

The first formula applies to so-called conjunctive structures, the second one to disjunctive structures. The difference between these structures is essentially this. In a *conjunctive* structure, *all* conditions must be true for an outcome to apply,

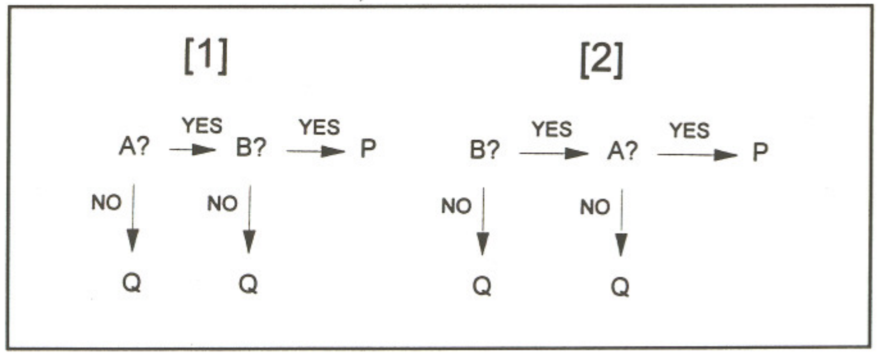


Figure 18. Two versions of a flow chart with two conditions and two possible outcomes.

while in a *disjunctive* structure it suffices if *only one* of the conditions is met. The underlying rules can be shown as follows.

- conjunctive: If A *and* B then P (otherwise not P)
- disjunctive: If A *or* B then P (otherwise not P)

A conjunctive structure can be ordered in two ways (Figure 19). If we start with condition A (left), the average time required to solve the problems is *tA + dA*tB*. If we start with B (right), the average time is *tB + dB*tA*. The most efficient order is that requiring the least average time.

If we use *t(A, B)* as the average time needed to solve the problem when starting with A, and if we use *t(B, A)* as the average time needed when starting with B, then the left version is more efficient than the right one if, and only if:

$$t(A,B) < t(B,A)$$

$$\Leftrightarrow \quad tA + dA*tB < tB + dB*tA$$

$$\Leftrightarrow \quad tA - dB*tA < tB - dA*tB$$

Since tA > 0 and tB > 0, both sides can be divided by *tA*tB:*

$$\Leftrightarrow \quad \frac{tA - dB*tA}{tA*tB} < \frac{tB - dA*tB}{tA*tB}$$

Now the numerators and denominators can be divided by tA, and tB respectively:

$$\Leftrightarrow \quad \frac{1 - dB}{tB} < \frac{1 - dA}{tA}$$

And finally, the direction of the < can be changed:

$$\Leftrightarrow \quad \frac{1 - dA}{tA} > \frac{1 - dB}{tB}$$

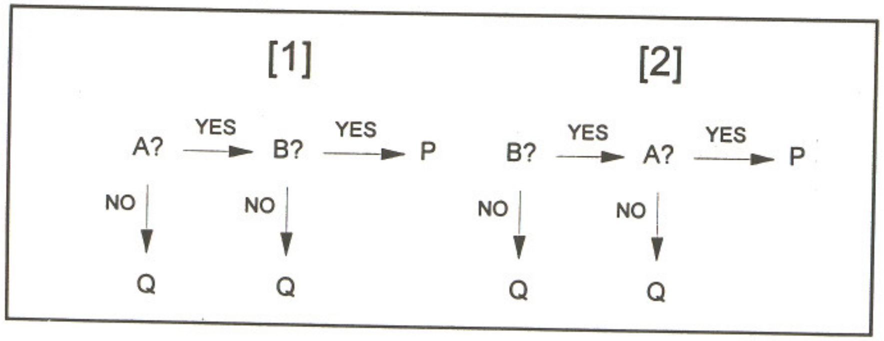


Figure 19. Two versions of a conjunctive structure.

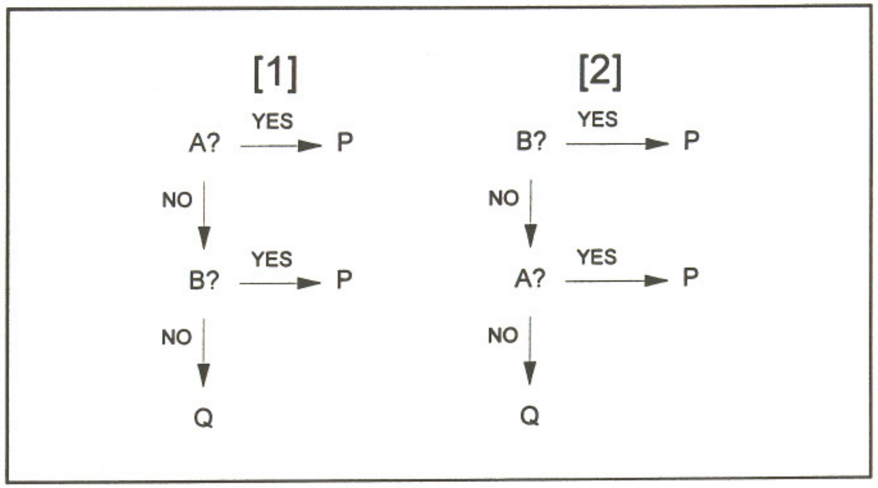


Figure 20. Two versions of a disjunctive structure.

The conclusion is that, if two conditions *both* have to be true to ensure a specific outcome, the most efficient flow chart starts with the condition for which $\frac{1 - d}{t}$ is at its maximum. In other words, if a procedure has a conjunctive structure, it should start by mentioning the condition that probably will apply to the smallest part of the readers (maximum *1-d*), and that will take the average reader the shortest time to verify (minimum *t*). If the condition with the smallest *d* is also the condition with the largest *t*, the quotients $\frac{1 - dA}{tA}$ and $\frac{1 - dB}{tB}$ have to be calculated and compared.

In a disjunctive structure, just as in a conjunctive structure, the conditions can be ordered in two ways (Figure 20). If we start with the condition A (left) the average time required to solve the problem will be *tA + (1 – dA)*tB*. If we start with B (right), the average time will be *tB + (1 – dB)*tA*. If we use *t(A, B)* as the average time needed to solve the problem when starting with A, and if we use *t(B, A)* as the average time needed when starting with B, then the left version will be more efficient than the right one if, and only if:

$$t(A,B) < t(B,A)$$

$$\Leftrightarrow \quad tA + (1 - dA)*tB < tB + (1 - dB)*tA$$

$$\Leftrightarrow \quad tA - (1 - dB)*tA < tB + (1 - dA)*tB$$

$$\Leftrightarrow \quad \frac{tA-(1-dB)*tA}{tA*tB} < \frac{tB-(1-dA)*tB}{tA*tB} \quad (tA > 0 \text{ and } tB > 0)$$

$$\Leftrightarrow \quad \frac{dB}{tB} < \frac{dA}{tA}$$

$$\Leftrightarrow \quad \frac{dA}{tA} > \frac{dB}{tB}$$

The conclusion here is that, if one out of two conditions has to be true to ensure a specific outcome, the most efficient flow chart begins with the condition for which $\frac{d}{t}$ is at its maximum. In other words, if a procedure has a disjunctive structure, the instruction should start by mentioning the condition that probably will apply to the largest number of readers (maximum *d*), and that will take the average reader the shortest time to verify (minimum *t*). If the condition with the largest *d* is also the condition with the largest *t*, the quotients $\frac{dA}{tA}$ and $\frac{dB}{tB}$ have to be calculated and compared.

So far, we have only looked at cases that involve two conditions. Often, however, document designers must deal with (many) more than two conditions. If all these conditions are linked conjunctively (*A and B and C and . . . and N*) or disjunctively (*A or B or C or . . . or N*), the formulae set out above still apply. Where there is a mixed structure, with both conjunctions and disjunctions, the most efficient flow chart can be developed from bottom to top (for illustrations and examples, see [3]).

A logic table with two outcomes can always be rewritten as a formula like the ones above. Thus, in principle, it is possible to determine the most efficient flow chart for each table by calculating the formula outcome.

Does the principle of average least effort always lead to the optimal order of instructions? In the procedure outlined above, it was assumed that the values of *tC* and *dC* are known for each condition. In practice, however, document designers not always have the relevant values at hand. It is more often exception than rule that statistics are available from which all *d*-values can be derived. In addition, *t*-values can only be determined by means of empirical testing. Often there is insufficient time and resources for studies of this nature.

Fortunately, it is not always necessary to know the *d*- and *t*-values *exactly*. Often, it is sufficient to know their mutual relationships and to know whether these can have any effect on the determination of the sequential order. In many cases, *t*-values can be considered as equal. For instance, time differences for verifying information on the screen can be neglected in practice. In such cases frequency figures are usually the only criteria to be considered in determining the optimal order.

CONCLUSION

We have tried to show that document designers who want to find the most efficient order of instructions, do not always have to reinvent the wheel. If the instructions at hand lead to two possible outcomes, the principle of average least effort can be a very helpful tool. If there are more than two possible outcomes, applying the selection principle is effective. However, it is also clear that in complex situations, the practical implementation of either principle can be rather time-consuming. It would be regrettable, though, if that were to restrain text designers from using these principles to improve their products. In many technical documents, putting the instructions in the right order means working on the essence of the message. If that takes designers some more time than their other writing tasks, it indicates only that they have gotten their priorities right.

ACKNOWLEDGMENTS

We thank Karen A. Schriver and Michael V. Sharp for providing useful comments on earlier drafts of this article.

REFERENCES

1. L. Cornelis, The Passive Voice in Computer Manuals: A New Perspective, *Journal of Technical Writing and Communication, 25*, pp. 285-301, 1995.
2. M. Mulder, Perception of Anthropomorphistic Expressions in Software Manuals, (*this issue*), pp. 489-506, 1996.
3. C. J. M. Jansen and M. F. Steehouder, *Taalverkeersproblemen tussen overheid en burger. Een onderzoek naar verbeteringsmogelijkheden van voorlichtingsteksten en formulieren*, (Communication Problems between Government and Citizens, A Study into Techniques for Improving the Quality of Instructional Texts and Forms (with a summary in English), SDU Uitgeverij, The Hague, 1989.
4. M. Steehouder and C. Jansen, Optimizing the Quality of Forms, in *Studies of Functional Text Quality*, H. Pander Maat and M. Steehouder (eds.), Editions Rodopi, Amsterdam/Atlanta, Georgia, pp. 159-172, 1992.
5. P. Wright and F. Reid, Written Information: Some Alternatives to Prose for Expressing the Outcomes of Complex Contingencies, *Journal of Applied Linguistics, 57*, pp. 160-166, 1973.
6. B. N. Lewis, I. S. Horabin, and C. P. Gane, *Case Studies in the Use of Algorithms*, Pergamon Press, London, 1967.
7. E. Berry, How to Get Users to Follow Procedures? *IEEE Transactions on Professional Communication, 25*, pp. 22-25, 1982.

8. I. Horabin and B. Lewis, *Algorithms, The Instructional Design Library*, Vol. 2, Englewoods Cliffs, New Jersey, 1978.
9. R. Kamman, The Comprehensibility of Printed Instructions and the Flow Chart Alternative, *Human Factors, 17*, pp. 183-191, 1975.
10. V. M. Holland and A. Rose, *A Comparison of Prose and Algorithms for Presenting Complex Instructions*, American Institutes for Research, Washington, 1981.
11. G. S. Krohn: Flow Charts Used for Procedural Instructions, *Human Factors, 25*, pp. 573-581, 1983.
12. M. Lagendijk-Swartbol and J. Driessens, *Presentatievormen voor handleidingen. Een toepassing voor de Telemix 512*, (Design of User Instructions: An Application for the Telemix 512), master's thesis, Utrecht University, 1987.
13. C. Jansen and M. Steehouder, Improving the Text of a Public Leaflet, *Information Design Journal, 4*, pp. 10-18, 1984.
14. P. Barnard, P. Wright, and P. Wilcox, Effects of Response Instruction and Question Style on the Ease of Completing Forms, *Journal of Occupational Psychology, 52*, pp. 209-226, 1979.
15. D. M. Wheatley and A. W. Unwin, *The Algorithm Writer's Guide*, Longman, London, 1972.
16. L. N. Landa, *Algorithmization in Learning and Instruction*, Educational Technology Publishers, Englewood Cliffs, 1974.

Other Articles On Communication By These Authors

C. Jansen and M. Steehouder, Improving the Text of a Public Leaflet, *Information Design Journal, 4*, pp. 10-18, 1984.

C. J. M. Jansen, M. F. Steehouder, A. Pilot, D. Schrauwen, and P. J. M. Looijmans, ALEXIS: Computer-Assisted Feedback on Written Assignments, *Computer and Composition, 4*, pp. 32-45, 1986.

M. Steehouder and C. Jansen, From Bureaucratic Language to Instructional Texts: How to Design an Effective Problem-Solving Tool for Citizens, *Information Design Journal, 5*, pp. 129-139, 1987.

C. Jansen and M. Steehouder, Forms as a Source of Communication Problems, *Journal of Technical Writing and Communication, 22*, pp. 179-194, 1992.

M. Steehouder, The Quality of Access: Helping Users Find Information in Documentation, in *Quality of Technical Documentation*, M. Steehouder, C. Jansen, P. van der Poort, and R. Verheijen (eds.), Editions Rodopi, Amsterdam/Atlanta, Georgia. Utrecht Studies in Language and Communication, Vol. 3, pp. 131-144, 1994.

C. Jansen, Computerised Writing Aids: Do They Really Help?, in *Quality of Technical Documentation*, M. Steehouder, C. Jansen, P. van der Poort, and R. Verheijen (eds.), Editions Rodopi, Amsterdam/Atlanta, Georgia, Utrecht Studies in Language and Communication, Vol. 3, pp. 239-248, 1994.

M. Steehouder and C. Jansen, Issues in Developing an Online Advisory System for Text Writers, *Journal of Technical Writing and Communication, 24*, pp. 137-146, 1994.

C. Jansen, Research in Technical Communication in the Netherlands, *Technical Communication, 41*, pp. 234-239, 1994.

M. Steehouder and C. Jansen, Optimizing the Quality of Forms, in *Studies of Functional Text Quality*, H. Pander Maat and M. Steehouder (eds.), Editions Rodopi, Amsterdam/Atlanta, Georgia, pp. 159-172, 1992.

Direct reprint requests to:

Michaël F. Steehouder
University of Twente/WMW
P.O. Box 217
7500 AE Enschede
The Netherlands
Tel. ++31 53 4893315
E-mail: M.F.Steehouder@WMW.Utwente.NL