Practical problems and proposed solutions in designing action-centered documentation

by
Stephen W. Draper
Department of Psychology
University of Glasgow
Glasgow G12 8QQ U.K.
email: steve@psy.gla.ac.uk
WWW URL: http://www.psy.gla.ac.uk/~steve

Preface

Based on a contribution to Carroll's workshop "Minimalism since the Nurnberg funnel" 17-19 Nov 1995 at Virginia Tech.

To appear as Draper, S.W. (1997?) "Practical problems and proposed solutions in designing action-centered documentation" Beyond the Nurnberg funnel (ed.) J.M.Carroll ch.13 pp.xx (MIT Press: Cambridge, Mass.)

Introduction

This paper examines and attempts to develop current conceptions of minimalist instruction in four related ways, by looking back to our experience so far, looking down to discern and articulate its foundations, looking forward at the problems of complex domains that seem beyond the reach of the original simple minimalist techniques, and finally proposing an organization for the design of manuals in terms of three levels of structure and four key types of design decision.

In looking back, I draw on my own experience with minimal manuals, based on lecturing on them to classes in an HCI (Human Computer Interaction) course, on supervizing a series of student projects that developed and tested such manuals, on writing one or two myself for introducing classes of students to various bits of software, and on writing a paper with Keith Oatley about them (Draper & Oatley; 1992). This experience convinced me of the great power and robustness of the technique, since it was usually easy to show superiority over the manufacturer's manual even though each student author probably did not follow the minimalist technique exactly and was far from optimizing the manual. It also made me aware of difficulties, described below, that my students, and I myself, would come up against when in the middle of writing a minimal manual.

The rest of this paper attempts to improve and extend the minimalist approach by analysing theoretical and practical problems with it, and then suggesting applications of the analysis. But what do we know about it? Firstly, it is one of the very few techniques in HCI that directly and demonstrably benefit users: you can take a piece of software that gives users trouble, write a minimal manual for it, and see them have a much better experience. Secondly, both task gains in time and errors and learning gains have been repeatedly demonstrated: minimal manuals are good both for immediate use and for supporting learning. Thirdly, the technique is transferable: I have repeatedly taken Carroll's writings and got students to produce measurably improved manuals. Thus however vague the minimalist "principles" may seem to a theorist, stated as they are in informal English in phrases that vary from paper to paper, they have proved to be sufficient to transfer a successful practice. Fourthly, the effect is robust: the manuals produced by my students have varied widely in appearance, and I am sure have not been optimized, and yet they have been sufficient to show gains for users relative to existing manuals. Thus the effect must be big, and insensitive to sub-optimal implementation.

Outline

This paper is organized as follows. First some common practical problems that arise for authors of minimal manuals are discussed, including an analysis of why writing minimalist material at first seems so unnatural to most authors. Other problems, such as whether to give explanations, or when to reject "slash the verbiage" in favour of "support error recognition and recovery", are due to apparently conflicting applications of the minimalist principles. As well as being helpful to practitioners as common problems to watch out for, their existence suggests that we might benefit from deeper principles to use in choosing between the working principles.

Secondly a derivation of practical minimalist techniques from deeper underlying principles is developed, ultimately from the rhetorical approach of designing manuals to have a specific effect on their readers, as opposed to being a neutral description of things as they are. The derivation raises the question of whether the approach is correctly described as "minimalist" or would be better described as "action centered", and identifies the pervasive interaction and conflict between the aims of getting users to learn and helping them achieve practical tasks as quickly as possible. The derived foundations should allow us to resolve apparent conflicts between practical rules, and to help us extend minimalism into new and more complex applications.

Thirdly the paper discusses the issues involved in extending minimalism in the light of how software and the problems it presents for documentation authors has changed since minimalism was pioneered. As lower level interface issues have become easier, it seems vital to extend its application into instructing and supporting users in learning "higher", job-related aspects of the tasks the machine is being used for.

Finally another angle on the design of minimalist documentation, informed by the earlier derivation from principles and hopefully applicable to complex domains, is constructed by reviewing the issues of what types of information to include and how much, the access mechanisms by which the user/reader must find the information they need, how it can be delivered (bearing in mind minimalist approaches to this such as promoting coordination of screen and manual), and how success should be judged.

The relationship of software and documentation

In what follows I assume that, whatever the practical realities may be of writing manuals after the software is fixed, the success and quality of the resulting user experience depends on the joint effect of manual and user interface. To a considerable extent, we can think of writing a minimal manual as supplementing the deficiencies of the interface's display: a perfectly designed interface does not require a manual at all (in an increasing number of cases users succeed without ever consulting a manual). On the other hand, a sufficiently bad interface not only makes the manual designer's job more difficult, but may leave the final user experience poor. However we should not admit this as an established fact: we may just not have developed manual design sufficiently. Claiming that the user interface design prevents a good manual from being written is close to admitting that manual writers are useless: if good interfaces do not require manuals and bad interfaces cannot have good manuals, why bother with manuals at all? The rational approach is to consider the design of interface and manual together, in order to minimize the total effort and allow rational tradeoffs e.g. save screen space at the cost of providing permanent "cheat sheet" quick reference cards for all users. In designing a manual we are really producing a new part of the overall interface (with a printed manual acting as a second parallel display for the user).

Practical problems

This section lists some of the commonly occurring problems from the viewpoint of a manual writer. The first subsection sets the scene by sketching classic problems of pre-minimalism writing. The second details some of the common problems facing practitioners who try to adopt minimalism to overcome the first set of problems.

Pre-minimalism problems

This subsection sketches classic problems of pre-minimalism writing, which also represent issues of continuing importance in minimalism.

1. <u>Telling users nothing</u>

An early attitude to (or rather against) documentation was that truly educated and skilled computer users can discover things for themselves; only the incompetent need manuals. Originally documentation was often omitted because users', particularly unskilled users', needs were little regarded. Now that is much less acceptable as a principle, but as user interfaces have become much easier to use, omitting documentation can in fact have some success. For instance the recent explosion of use of the World Wide Web since visual interfaces to it have spread has occurred almost entirely independently of user documentation. The half truth behind the tactic of omitting documentation is that learning by exploration is often a successful alternative to documenting information (it is a large part of what makes minimalism successful). The fallacies behind the idea of omitting all documentation are that often a user has to know something about how to learn by exploration from a given system, and that usually some information cannot easily be obtained in that mode. Thus the central problem for users posed by this approach is that of missing information. This problem lingers on, as Thimbleby (e.g. Thimbleby & Ladkin, 1995) emphasizes, in manuals that are incomplete or incorrect in their information. Minimal manuals that choose to cover only a subset of commands are also open to this criticism. Omitting information is an important tactic, but can fail the user if it is unobtainable by any other route: the issue is ensuring that all functionally necessary information is available by some route (not necessarily by describing it in the manual).

2. <u>Documentation as a brain dump from the designer</u>
A first reaction against telling the user nothing is to tell them everything. This results in the classic pre-minimalism manual that tries to record everything, but results in an unusable manual. It is seen in discursive introductory manuals, that seem to have been written as if a new user had just said "Tell me about this software" and settled down for a good read, as one might spend an evening reading a travel book. Its central problem is that it is not organized for users who are actively engaged in using the software, and makes it hard for them to access the information needed. The information is present, but inaccessible. Information presence is not enough: accessibility in practice is equally crucial.

3. <u>Documentation organized by feature</u>
Documentation organized by software function or feature as one kind of reference manual forms another common pre-minimalism format. This is actually of considerable use, but the organisation still reflects designers' not users' main preoccupations. Thus it is useful if but only if the user already knows the existence and name of a function (the same problem as dictionaries have: you can only verify spellings you already know, or find meanings for terms you have come across elsewhere — you cannot look up a word given its meaning). Its central problem is that access is limited to one method (from name to task), and one little needed by new users or users driven by a goal of getting a job done, who need to go from task meaning to name. The information is present, but accessible to only a minority of users who already have the right pre-requisite partial knowledge. Access by feature does not serve all, or even most, user needs.

Common problems with minimalism
This subsection describes some of the most commonly occurring problems from the viewpoint of a writer of minimal manuals. They represent frequently asked questions by students trying to write manuals, but equally problems that I find hard to decide upon when I am writing a manual. These are mainly problems because of apparent clashes between minimalist principles.

1. <u>Maximize learning or maximize getting work done?</u>
Are we trying to get users to succeed at their immediate tasks, or to learn the maximum? If the former, then we generally want to provide every bit of information they need, removing the motivation for learning, while if the latter then we want to set them puzzles that may slow them up but will result in greater retention (a well known learning principle).

The strength of the minimalist approach is that at either extreme there is no conflict: if the puzzle is too difficult the user neither succeeds at the task nor learns anything, while if the instructions are too detailed and complete then users lose more time and make more errors in interpreting them than they gain in extra information. Nevertheless in the central region in which a minimalist approach places a documentation author, there is a real tension. For instance consider sitting in a car and directing the driver where to go. Telling her where the next turn is in terms of immediate local marks (e.g. the second intersection from here) generally works the best, but the driver is less likely to be able to find the place again than if she had been given more global instructions and made to work out their meaning on the ground. Similarly users who are given fairly detailed recipes (e.g. "**Edit** menu; **Select All** command") are less likely to remember them later. It may be true that people do not follow all dictated recipes accurately, but if our goal is to maximize task success then we will develop a skill at giving the recipes in ways that minimize those deviations, comparable to choices of how to tell the driver which turn to make. On the other hand if we want to maximize learning, then we must first decide if it is retention of items (e.g. memorizing command names) or discovering generalisations we are aiming for; and then devise techniques for that pedagogic goal (e.g. "find the menu command that will highlight all the text at once"). The measures we would use to check on our success as authors would also be different: time to first successful task completion vs. success at performing without the minimal manual on later occasions.

2. Tutorial activities or efficient job support

Similarly, is a "minimal manual" aiming to be a reference manual to support work or a tutorial that can direct the user to do tasks to achieve an optimal training sequence? People hired as experimental subjects for research labs, and users on a training course for software, may have some patience for tutorial instructions; many others do not and expect to get useful work done from the start.

This matters, because an issue for authors is whether they can instruct users what task to do, just for the learning experience. In many interfaces, it is very hard to explain something without reference to some other task: if the user encounters tasks in one order it can all seem easy and natural, but in another order it can seem hopelessly obscure e.g. explaining "paste" before "cut", or the point of those little triangles before the user has used the tab key, or the effect of right margin controls before the user has typed in enough text to fill a line. This argument really applies mainly to learning by exploration, but that possibility is one of the most important enablers of minimalism: if the user can learn by experiment, the manual need describe nothing. However in many contemporary interfaces, learning by exploration depends in practice on the sequence of user actions, so a crucial issue for authors is whether they allow themselves the presumption of directing the sequence of a user's exploratory actions.

3. Explanations

Should manuals contain explanations, as opposed to direct support for actions? e.g. should they explain what the clipboard is (or other hidden buffers). A truly minimalist approach has no explanations in at all (and boasts of avoiding long descriptive introductions); but sometimes learning (although perhaps not doing) is likely to be greatly aided by explanation. Note that none of Carroll's summary statements of minimalist principles (e.g. van der Meij & Carroll, this volume) have anything that would promote the inclusion of explanations, and his experimental results tell against them (Carroll; 1990 p.178-180), but elsewhere he reports that users complained about their absence. As an author my intuition is that there are a few cases where they are vital to allow the user to understand some hidden pattern in action procedures (e.g. the hidden storage buffers linking the effects of cut and paste i.e. delete and re-insert commands); and the Task Action Grammar work on consistency and learning time might support this (Payne & Green;1986), but there seems to be no direct evidence.

4. Addressing misconceptions as an additional requirement

Barbara Mirel (this volume) suggests that preventing misconceptions is an important additional requirement. This is in the basic tradition of observing actual problems users have, and trying to address them; and it is furthermore recognisable as compatible with experiences in the education field, where it has become clear that it is not enough to present information for learning but instead at times educators must explicitly address misconceptions known to recur. The tension for minimalism is the same as that for explanatory material: it will add more material, and unlike in education, users may not recognize that they need to use it.

5. Necessary detail for whom?

How can an author decide on the level of detail? A big saving in verbiage is to omit all details that the user does not need, but omitting any detail that a user does need is disastrous (for that user). There is a big tension in practice between supporting novice users and achieving minimal text (and so supporting users who do not need the greater detail). In general the principle is that a manual must be targeted for a particular user, and there can be no such thing as a single manual for all users, but ultimately a library full of different manuals: minimalism in reading for an individual user but an explosion of material per product.

The immediate problem is that you can write much slimmer manuals for experienced users than for novices, but for complete novices you have to include material on how to use the mouse, where to find the menus, etc. One solution I have tried is to write a graded series of manuals, omitting more and more detail but including more and more functions as the series goes from introductory to expert in its target audience. For the first (getting started) manual, the important thing is to include a complete set of functionality but no more than the minimum e.g. in a word processor, a way of adding and removing text, and some very basic formatting. Later manuals can introduce faster but logically redundant commands e.g. a way of moving text instead of deleting it and retyping it in a different place. For the users, this grades series is minimalist since each manual is by itself usefully minimal (either few functions or few procedural details), although for the author and manager it is not, as there are now several documents instead of one.

6. Organizing by task vs. avoiding repetition

In some cases at least there is material, typically error recovery information, that apparently cannot be put in a task-organized manual, because the problems could arise in any context. For instance in the Unix editor **vi**, observations of problems users have shows that there is a set of baffling error states any of which may arise at any time. These are due both to the editor itself and to general features of the Unix environment interacting with it. For instance the editor allows the number of lines displayed to be set: if it is accidentally set to one then it is hard to understand what has happened as the screen is mostly blank. Similarly, if the user ever presses <control>S then all screen output is suppressed, making it appear that the keyboard has gone dead. There are at least nine different such cases where an accidental keystroke or two can throw the machine into a baffling state, requiring special error recovery actions. Duplication of this material for every normal user task seems unthinkable, but putting it in a separate place is a dangerous lapse from having the material in the exact place in the text that a user needs it. That is, minimizing verbiage can conflict with organizing the manual by task.

It is actually a consequence of "consistency" that makes such error recovery actions (and the accidental actions that cause the problems) valid in all or most states. This general applicability also makes it apparently sensible to put such universal error recovery task procedures in a single place in a manual. However, as the minimalism literature tells us, this approach has a poor record of serving users in practice. This is probably because when an apparent problem occurs users naturally tend to suspect it is to do with what they just did and the particular task they were in the middle of. It is not obvious to them that this is a general problem, and might be addressed in another part of the manual, under a general heading of problems and error recovery. One solution is to instruct beginners to induce such a problem state deliberately and then to use the error recovery procedures. This solves this problem only by trading it for the problem already described of persuading all users to go through a tutorial before they experience the need for it.

7. The unnaturalness of the writing task.

Why is it so hard for people to learn to write minimal manuals? Because it goes against the basic approach to writing that we have all (painfully) learned from childhood: that in most writing, the difficulty is in describing everything that you (the writer) knows and can see to someone who isn't there. The easy use of language is in human-human dialogue. However in most writing we face the essential difficulty of achieving clarity and non-ambiguity for a piece of paper standing alone: this is difficult for children but eventually writers develop the skill of including enough explanations and background context in their writing so that readers can understand descriptions without being there and without asking questions.

With a manual for supporting a practical task, however, we face a third situation where what we are describing is there, physically present. The writing is like speaking to someone present with both you and the machine, but who cannot question you — monologue but with copresence of what is discussed. You don't need to describe what is obvious and visible, and anyway you probably can't very well.

People don't read but try things out, partly because the text isn't comprehensible without looking at the machine. The minimalist approach notes that you can't fight this, but that actually it is also an advantage, allowing great brevity. Thus this writing is NOT like most writing — letters, novels, etc. — that are for making present something remote from the reader. It isn't an encyclopaedia for recording everything known about a device (although technicians may need this kind of manual). In most writing the main problem is being explicit enough to overcome differences in the prior knowledge and perspectives of writer and reader, but here we can and should use the actual device as much as possible to show what we mean: this is more like a conversation about something we are standing beside than it is like most other types of writing. Could we devise special training techniques for authors?

8. Counterexamples to minimalism

How are we to understand very successful cases which are the opposite of minimalist? For instance a printed phone directory at home is very usable: you can often find the entry you want in seconds, yet it has an extraordinarily low ratio of useful (to you) entries to useless ones. Perhaps, in the two year life of a directory, you might only refer to 10 numbers, yet a fairly small directory has hundreds of thousands of numbers. This is extreme anti-minimalism, yet very successful.

Presumably it works because people can easily skip text they don't need, provided they know they don't need it. When can we use this in our manuals? In fact minimal manuals already do use this in a small way by structuring the parts of each entry to separate task description, procedural instructions, and error recovery material (say), so users don't have to read the error recovery entries unless and until they need them.

Another example is that of command reference manuals (e.g. the Unix online manual). These are not successes for new users, but they are successful some of the time as you can observe by watching system experts at work: these experts do use such manuals frequently and often successfully. On the occasions they succeed, their lack of minimalism (their sheer bulk) is clearly not a drawback.

Identifying underlying principles for "minimalism"

The central feature of the minimalist technique for designing manuals is that it is empirical not theoretical. The sequence of studies that culminated in minimalism (Carroll; 1990) were, unusually for both HCI and experimental psychology, driven not by prior theories but by responding to the observations at each phase by creating another version of the technique intended to overcome problems observed with the earlier versions. The bottom up empiricism of the research programme is echoed in the method for developing each new manual, where first designs are tested on real users and modified in the light of observed problems. In order to communicate the method to new practitioners, about six heuristic principles (such as "slash the verbiage") have been extracted from the successful applications and experiments. These can be handed in written form to new practitioners, who are then generally able to create new manuals that show the expected advantages. I myself have been able to do this with a sequence of students, so I know first hand that this is a technique transferable through these principles written in English.

The principles have been presented in various forms. For instance Carroll (1989), in relating minimalist principles to features of human learning, listed:

1. Allow the user to get started fast
2. Rely on the user to think and improvize
3. Direct training to real tasks
4. Exploit what people already know
5. Support error recognition and recovery

Earlier versions had, in combination with some or all of those:

a) Slash the verbiage
b) Guided exploration
c) Promote coordination of display and manual.

In Carroll (1990) the section headings reveal this variation on the list:

1. Training on real tasks
2. Getting started fast
3. Reasoning and improvizing
4. Reading in any order
5. Coordinating system and training
6. Supporting error recognition and recovery
7. Exploiting prior knowledge

The most recent presentation (van der Meij & Carroll; this volume) has 11 heuristic principles, structured under 4 main principles thus:

1. Choose an action-oriented approach
    1.1 Provide an immediate opportunity to act
    1.2 Encourage and support exploration and innovation
    1.3 Respect the integrity of the user's activity

2. Anchor the tool in the task domain
    2.1 Select or design instructional activities that are real tasks
    2.2 The components of the instruction should reflect the task structure
3. Support error recognition and recovery
    3.1 Prevent mistakes whenever possible
    3.2 Provide error information when actions are error-prone or when correction is difficult
    3.3 Provide error information that supports detection, diagnosis and recovery
    3.4 Provide on the spot error information
4. Support reading to do, study and locate
    4.1 Be brief; don't spell out everything
    4.2 Provide closure for chapters

It is clear that the exact statement of the principles is important for communicating enough to new practitioners for successful manuals with demonstrably high performance to be created. Conversely, there must be a very powerful and robust effect underlying the minimalist approach as some statements of principles must be better than others, and almost certainly some practitioners are better than others: this means that many minimal manuals actually produced must be less than optimal and yet show good performance. Nevertheless it might be useful to reanalyze the principles, both to attempt a better statement of them for new practitioners and also to resolve some of the practical problems and apparent contradictions that arise, some of which were described above. Draper & Oatley (1992) offered one attempt at this. Here I offer a revised version that incorporates the connection Johnson (this volume) draws between the study of rhetoric, user centeredness, and minimal manuals.

This analysis is presented from proposed underlying principles through a series of derivations towards the practical prescriptive principles usually listed. It covers five or six levels, each derived from or generated by the previous one, the last two being the levels of practical heuristics. My claim is that these minimalist prescriptive principles can be derived solely from the notion of action-centeredness, which in turn derives from the user-centered or usability testing approach of focussing on optimizing support for observed user performance at their work.

A derivation

<u>Level 1: "User centeredness" i.e. taking a usability approach</u> (Fundamental principle)
The most basic principle is that of user centeredness in the sense that the overriding criterion for manuals (in the "minimalist" approach) is how well the user performs with them: their length, their use of language and format, their consistency, whether they look nice, whether they use correct English — none of these matters except to the extent that observations show they affect the user's ability to succeed in this particular case. Everything else follows from that.

The corollary associated with this principle is the use of usability methods and testing in designing documentation, in particular iterative design based on testing versions on real users and modifying the design as a result.

In the context of writing this corresponds to being reader-centered rather than text-centered; to writing to achieve a specific (rhetorical) effect on the reader and to getting them to act as a result, rather than on producing a text whose interpretation and effect depend as much on the context and the reader; to an emphasis on the process of writing and still more on the net effect of the process of reading, rather than on the text as product. This is related to a speech act view of language (speaking to produce an instrumental effect), as opposed to the use of texts for bills of lading, legislation, rules of a game, or novels in all of which applications the effect of the text and indeed its meaning depend a lot on the reader and is relatively independent of the intentions of the writer. (See Johnson (this volume) for a related, fuller discussion of this point.)

<u>Level 2: Action centered manual (not "minimal" except as a consequence)</u> (A first consequence of the fundamental principle.)
It follows from taking a usability approach that manuals are designed to support successful user action and judged by how well they do that. Hence their essential characteristic is to be action centered, and only "minimal" if and when that best supports action.

Level 3: Two aspects of "action centeredness"
3.1 Organize the manual to support doing tasks with the software
3.2 Organize the manual to be used during action, as a resource — not to be read like a textbook. (I.e. aim at learning by doing, not learning by study of an exposition.)

Level 4: Three "super" principles about what to include and omit
The above leads to one principle of what to cover, and two principles of what to leave out, given the additional observation that users act in a context of knowledge and information from sources other than the manual — they do not suspend disbelief and read manuals like fairy tales.

4.1 Organize the design around user activity.

4.2 Design around the effect of users' prior knowledge (and beliefs and desires).

4.3 Design around the effect of users' (future) knowledge during action. (I.e. Exploit the information available from the interface at the relevant moment.)

Levels 5 and 6: Specific practical principles
All the specific practical principles in their various versions (level 6) and their (level 5) groupings in the van der Meij & Carroll form (this volume) can be derived from the above: only a couple of examples are discussed here. For example "provide on the spot error information" follows because errors occur during user activity and recovering from them is part of that activity, and error information needs to be supplied on the spot (as part of the task information in the manual) since reading the manual is itself part of the task the user is executing, but placing the information elsewhere will demand a new task of searching for it. Similarly the older "slash the verbiage" heuristic follows from avoiding creating extra user tasks of reading a lot by designing around the user's prior and action-supplied knowledge. The higher level principles can suggest when to apply this heuristic (when the user has other sources of information) and when not (e.g. only leave out error information when the user can recover just by responding to the software's error messages). Similarly in considering the question (discussed earlier) of whether to duplicate material so as to ensure it is present where needed, we can see that that the underlying principles imply that the true rule is to minimize reading for the user. One way to do this is to minimize the printed material, but another is organize the material (cf. phone directories) so that users do not have to search through what they do not need.

Is the technique essentially "minimalist" or "action centered"?
Two important issues can be brought out by the above analysis. The first is that, although the approach is called "minimal manuals" or "minimalist instruction", it might more accurately be called "action centered manuals". The usual names are good for selling the idea to managers, but I have found them distinctly counter-productive in teaching the technique to would-be manual writers. This shows up most acutely in that one of the most important aspects of the technique is the inclusion of error detection and recovery information, which usually involves adding not deleting material relative to conventional manuals. The above analysis gives priority to action support over verbiage reduction, as is required for successful manual design, and it would save time in training practicioners if the technique's name reflected this.

Is the goal of manuals to support learning or to support doing?
The above change of analysis can also be seen as a claim that minimal manuals are about supporting user doing, not user learning, which implies that we can look to theories of doing such as Norman's (1986) "theory of action", rather than to theories of learning. In reality however users do both, inextricably combined, and manuals when successful support this mixture. Hence we should not ignore learning, but it is essential to realize that doing is at least as important an outcome to support as learning, and furthermore that the indissoluble mixture raises issues of its own that are central to understanding how to design manuals.

Commonsense, much of design, and much of psychology all presuppose that learning and doing are separate: done by different people (novices vs. experts) at different times (e.g. training vs. doing the job) by different parts of the mind (learning vs. problem solving) and supported by different parts of the user interface (meaningful menu labels vs. learned menu positions or keystrokes). A little reflection shows that this tacit view is deeply mistaken. A more nearly correct view is that all human activity involves both doing and learning. Such a statement however does not capture the entailed key problems for the design of both user interfaces and manuals, problems that make this both a theoretical puzzle and also an important and difficult practical tension. The problems are: what knowledge is actually used in operating an interface, what knowledge is learned by experts, and the large effect on learning of a person's decisions about what to learn.

To begin with, we know rather little about how much knowledge (and so learning) in fact underpins successful human action. If you shut your eyes, you will not be able to travel to work or even to move round your own home with anything like the skill you normally show. Thus you do not know your way over these familiar routes in the sense of having a self-sufficient knowledge of the route in your head: instead you have some partial traces which are sufficient to complement perceptual input during the task. Similarly people have been shown not to be able either to recall or recognize many of the features of the coins and banknotes they use every day (Nickerson; 1979). This also applies to users' memory for the menus of the programs they use (Mayes et al.; 1988). Prolonged skilled use does not lead to anything approaching complete knowledge of the objects being used. Doing, even skilled doing, does not require learning in the most obvious sense of knowing the main attributes. Thus even if we were to take learning as the goal of manual design, we know little of what users need to learn. In minimalism this appears as exploiting coordination of display and manual to omit text: but this principle labels the existence of an opportunity, rather than telling us what people either use or learn.

A tempting hypothesis is that people learn only those attributes necessary for the tasks they are doing. Although this is a good first attempt to counteract the assumption that people just remember every obvious feature, it is turning out to be a poor approximation to the facts (Kapitelinin, 1993; Moyes,1995). People do do some "incidental learning" and accumulate knowledge of some apparently non-functional attributes, but on the other hand they do not always learn apparently useful ones. For instance, mouse driven menu systems require the use of position information as part of users' actions, and most users do start to learn some positions and can move to some menu commands without using their eyes, but they do not learn the positions of most of the menu commands they use.

All people then, whether in computer use or in other domains, from first time use to world class expertise, employ some mixture of prior knowledge and reference to external information sources. Librarians do not know all the books in their library, but instead are characterized by special skills for getting the information they need. The same applies to Smalltalk programmers who do not know thousands of Smalltalk classes by heart, but have skills at finding them. The use of pre-flight checklists by aircraft pilots is another example of the use of reference material in expert practice to avoid reliance on human memory. Expert users of large systems depend on reference manuals, and the aim of those manuals must be to promote not user learning but user doing. Effective use of reference manuals is part of what constitutes expertise in many domains (cf. Draper, 1985.)

In summary then, all human activity comprizes a mixture of learning and doing, drawing on a mixture of internal memory and external information. Remarkably little is known about these mixtures although the success of information sources such as manuals must depend on it. Learning never stops (even after thousands of trials, people are still improving a skill), yet many "obvious" features apparently go unlearned despite daily exposure or even use.

However there is a further important issue here: the mixture of doing and learning is strongly affected by the intentions of the person. Learning is not an automatic side effect of doing (as some theories seem to assume), except possibly at the lowest level. This is because what a person does is determined by their goals: if their goal is to learn they do different things e.g. deliberate exploration, while if their goal is to get job done they act differently and so learn differently and much less. This is most obvious in higher education, where almost all learning depends upon student effort which in turn depends on their decisions about where to expend it. Sitting through a lecture or video without wanting to learn results in very low retention as in most evening TV watching, whereas the same external experiences coupled with a goal of learning (whether derived from personal interest or institutional coercion through exams) results in quite high retention. The ratio of material retained in these two cases may perhaps be of the order 100:1. This is a big effect — bigger than almost all effects ever reported in the literature from varying learning materials or methods.

When users confront an interface, they will apply their own goal priorities. On a training course this goal may be to learn; in their office it may be to do a job and they may have as little patience for activities justified only by learning as I would have if I was trying to use a new toaster for my kitchen, or to drive an automobile I had just hired at an airport. It is not obvious, and probably not true, that we can assume that one kind of manual will do for all. Taking the goal of minimal manuals to be learning by users would be tantamount to saying that minimalism has no contribution to make to manuals for most workplaces other than training courses; but in actual fact minimal manuals seem to do quite well for users who only want to get tasks done. Thus I believe that minimal manuals are more than just learning aids, and our conscious design objectives should be to support work as much as or more than to support learning.

In summary, the interaction of learning and doing leads to three issues in the design of manuals, and in fact in HCI in general.

1. What information do users actually use in operating an interface? And, for manuals, what information in addition to that supplied by the interface and common knowledge? For instance, if a command is actually recognized as having the longest name on the menu or being at the top, should that be how it is described in our manuals?
2. What information will expert users normally learn? It is only this subset that we might be justified in writing tutorial material for.
3. Learning is strongly controlled by the user's intention: to learn or just to achieve a material task. This is not under the designer's control: both alternatives must usually be served. One way to develop this idea is to add learning to the set of tasks used in analysing what manuals (and user interfaces) must be designed to support: see below.

Extending minimalism to more difficult cases

Given the many successes in designing minimalist documentation for applications like word processors, and the various analyses of the principles underlying these successes, it becomes natural to ask how far we could extend its application. In the years since minimalist work began, changes in technology have affected the challenge of designing action-centered documentation. The two main changes are that on the one hand user interfaces have become easier to use, and so providing documentation is easier or even unnecessary, and on the other hand that instructing the user in the task domain as opposed to merely the user interface has become more important, which makes documentation more challenging to design. This has tended to move the focus of design for manuals up from the lowest to the highest level in terms of the three levels of structure implicit in most minimalist designs.

The three levels within minimal manuals

Minimal manuals typically have three levels of structure.

1. The level of individual actions is the lowest e.g. "press <control>V — the selected text will disappear".

2. The level of a single user task is the middle level and the chief unit in minimal manuals. Each such unit describes a multi-step procedure for a single user task, complete with associated error recovery and success recognition information (e.g. select the text, delete it, select the new position, do "put").

3. The set of user tasks forms the top level, and the main design difficulty is how to organize the complete set of alternative tasks so that users can find the one they need and recognize that they have found it. Part of this problem is how to name or describe each task (e.g. do you describe the task as "moving text" or "cut and paste"), the other part is how to support the user's search over the set. One approach is to add an index, or contents list, or a hierarchical structure of user tasks e.g. group tasks under topic headings.

This three level structure was clear in the Guided Exploration cards approach (Carroll, 1990, ch.5). Here the set of cards formed level 3 (there was no index, users just flipped through the cards), a single card corresponded to level 2 with the different aspects (e.g. actions vs. error recovery) labelled separately within a card, and level 1 corresponded to one instruction, typically represented by one or two lines.

Minimalism (that is, the approach of minimizing the amount of printed material) operates separately at each level. At the lowest level (of individual actions) it is about being sparse in the amount of detail for each instruction step by coordinating display and manual i.e. exploiting all the clues in the screen display, and the observation is that this not only reduces problems in finding and following printed instructions, it also seems to reduce many comprehension problems. At the middle level of the entry for a single task, there is the structural system of subdividing entries into standard parts (e.g. description of the outcome and purpose of the task, procedure, error recognition and recovery) with standard headers or graphic symbols. This allows users to skip parts they don't need at any given moment, and so achieves minimal reading by the user, although not minimal printing of words. Presumably to the extent that this works, we could add more parts to an entry without harm (e.g. extra optional explanations).

At the top level is the issue of how a user is to find the right entry. The problem here is that manual writers and users do not in general have a pre-agreed language in common for naming or describing tasks, so in practice users have to scan the whole list of entries looking for a best match to their need. This is what is so different from phone directories, where if you know the person's name you can skip right to the entry you need by exploiting the alphabetic ordering and without having to read and consider every name. Minimalism is more problematic here. One approach is to remove all lists (contents and index), which works if the whole manual can be fitted on to one sheet so that the user can scan it all without turning a page. When this is possible this works well (one reason that one-sheet reference cards are so popular). Because the user must scan all entry titles to find the task they want, there is great pressure to apply the other minimalist tactic at this level, and to reduce the number of tasks described as radically as possible. This leads to one aspect of a "training wheels" approach, where only a specially selected subset of tasks and machine features is covered. This top level is often the hardest to to do well, and some manuals only manage to apply minimalism to the lower two levels, while leaving the top level organized essentially like a conventional manual with one entry per software command or feature and indexed by command name.

Interfaces are often easier to use

A lot of user interfaces nowadays need no manual e.g. bank cash machines, many public information systems. Many users of the world wide web have never consulted documentation for their web browser. This is essentially because the two other sources of what the user knows (on screen information delivery, and previous experience of similar interfaces) are often sufficient so that a manual has nothing to add. This affects all levels, but has had most effect on the lowest level of individual actions.

As user interfaces have become broadly easier to use in the years since minimalist work began, so has writing manuals. We should perhaps therefore use that spare capacity somewhere else. If we could create successful minimalist manuals for keyboard command interfaces, then perhaps we could succeed in cases with a simple graphical interface but a more complex kind of software.

The increasing need for task domain instruction

Partly because the surface of user interfaces has become easier to use, the greatest limitation on user performance — the goal by which we judge our documentation's performance, just as we judge a user interface — becomes the user's knowledge of the domain and its work tasks. Direct evidence of this can be found in an experiment by Baxter & Oatley (1991). They measured the time to learn two spreadsheets, and found no difference between two rather different commercial designs, but a large difference in learning time between subjects who had used some other spreadsheet before, and those who had not. Clearly a large part of the learning burden concerned knowledge about the kind of thing a spreadsheet is and what it is useful for. Although it may be that the original users of spreadsheets had long experience of adding up columns of figures, many learners nowadays do not, and have to be prompted into both thinking of a task that is meaningful to them and into using a column or matrix layout, and into typing a formula to calculate. Pocket calculators, now familiar to all, do simple arithmetic in a rather different way (with no separation of numbers and formulae).

Domain knowledge does now frequently seem to be the main barrier to wider user success (and so wider use of the software). Why don't more people use databases (now a mature technology) routinely in their work? — mainly because relatively few people understand how a database would fit in with their work, or rather how to describe their work in terms of a consistent data model that could be represented by a database.

This issue is part of a broader trend for computer systems to be seen, not as a central piece of equipment around which the work is structured, but as just part of an overall work pattern. It is that overall work pattern that is the real topic to be supported by documentation (because it is what matters to users), just as more and more computer systems if they are to succeed must be designed, not as devices in themselves, but as one component among many in a wider pattern of work.

Whether documentation authors are prepared to write a manual to instruct users in a new external task domain rather than in how to operate a user interface depends upon their job remit. But if they do take it on, then this is mandated by minimalist principles of supporting user action in any way required, being action centered, and matching the documentation to user's real tasks; and they will face problems that were always present in minimalist approaches, even if with a different balance. Supporting users' learning about new task domains means focussing on the top structural level of the set of user tasks, and addressing it successfully. Obviously, the more complex the domain the harder this becomes. (E.g. "Balancing your accounts" vs. "pasting a special function". See Mirel (this volume) for a discussion of complex task domains.)

The key design decisions for action-centered documentation

This chapter has reviewed the main problems that have arisen in minimalist documentation so far, looked at the direction that future work is likely to have to tackle, and discussed the relative priorities of the various principles underpinning "minimalism": not least of which is that designing around user action takes priority over minimizing words. It has also introduced the notion that such documentation is structured into three levels, each of which has its own problems and tried solution techniques. In this section, I suggest that the design of a manual consists of four main types of decision, and that future progress depends upon improving solutions for these problems.

1. What information to include?

The earliest question for documentation is what information to include, and minimalism offered a novel compromise answer between the first two obvious approaches of supplying no information or all possible information. However deciding the exact balance remains an active issue, different for each case.

A lot of manual design is driven by compiling a list of commands or features, although omissions are by no means unknown. However users need things to be organized by task. This leads to further issues of how to decide what tasks to include. "Below" the level of the software's main commands are all the sub-procedures or articulatory procedures of how to execute those commands: how to open a menu, how to operate the mouse, etc. Should these be included? Do you have to include how to use a mouse in every section? Too much information undermines delivery (making it even harder for users to follow instructions), but too little undermines the whole enterprise. My conclusion is that I have to design a different manual for each type of user as defined by their prior knowledge. Online manuals delivered by software however could try dynamically adjusted amounts of detail (cf. Wood et al.s' (1978) contingent tutoring technique).

"Beside" the main commands are other issues, not the responsibility of the software package designer, but which an effective manual writer must deal with, to do with the environment the software operates in e.g. switching on, firing up the application, dealing with switching between applications. For instance stray mouse clicks just outside a window often cause a switch to another application, and cause trouble for new users.

"Above" the main commands is an indefinite hierarchy of larger tasks for which the software is a means to an end. Should a manual for a word processor deal with how to manage the writing of a book, interfaces to software for managing bibliographies, etc.? If these are the tasks that users are engaged in and which motivated them to use the software, then it seems hard to avoid, but there seems no obvious boundary, so this threatens an explosion of content and the defeat of minimalism. New work is perhaps needed most in this area. There are also other grounds for expanding the tasks covered. For instance, offering learning as a possible explicit user task might be one way of handling the problem of whether to offer pedagogic activities or only immediate work tasks.

2. Access mechanisms

A crucial aspect of manuals as with any information resource is its index or access mechanisms: as noted in the opening section, having the information present but not in practice accessible by the user is as bad as omitting it. The bottom level has none below it, and so no access design issue. At the middle level of an entry for a task, the access issue is about how the user can move between the parts of an entry, e.g. skipping the error recovery information unless it is needed. Standard headings or other page layout mechanisms have been made to work repeatedly to solve this problem. The top level of providing efficient access to the right entry from within the whole set of tasks is the most problematic case, and one which seems to require new work to support an extension into supporting work domain instruction.

The first element is to describe each task in a way the user will recognise e.g. "typing in some text" rather than "using the Insert command". The second element is then to allow the user to search through these descriptions to find the best match to their need. This is typically done by an exhaustive scanning either of separate pages (flipping through looking at the titles of each entry) or possibly in a list (an index or contents list). Because the task descriptions are largely unpredictable by the user, they are effectively unordered and the scan must be exhaustive: a technique that does not scale up well to manuals with many tasks. Progress here is needed. For instance online manuals with free text matching from users' free text request to the task descriptions in each manual entry might be one way forward. Another might be graphical maps of task hierarchies to provide more structure to what users scan while searching for a correspondence with their internal mental descriptions of what they want to do.

3. Information delivery

When the user has found the item, how should it best convey what they need to know? Minimalism is striking here in being terse and relying on coordination with the display and in effect forcing a little local learning by exploration. This is probably important for speed and for improving both comprehension and learning, independently of reducing volume to facilitate finding by scanning. The issue is about how to describe an action in a way that works in conjunction with the display. Iterative design and testing of the manual will remain necessary to debug these descriptions, but we are now confident that we will succeed at this.

Hans van der Meij (this volume) has explored using pictures of screen displays for information delivery here. Experience in other applications suggests however that similar issues, and minimalism, may well apply. For instance simplified diagrams may do better than veridical screen snapshots (just as maps are usually better than aerial photographs for many tasks). And in extreme cases, special perceptual training could be needed: just showing medical X-rays does not tell you how to perceive the medically relevant features, so perhaps in some cases users must be trained in how to read screen displays.

4. What should be the success criteria for manuals?

When we test our manuals in order to improve them, what should our criteria be: learning outcomes or success at material tasks? In practice we often react most strongly as designers to neither of these but to evidence of breakdowns and to user complaints. Perhaps we should consider how we imagine our users to be monitoring and controlling their activity. Are they judging by a feeling of understanding? of certainty / confidence in the choice of actions? of confidence in eventually achieving their goals?

Complaints about ultra-minimalist manuals (e.g. the Guided Exploration cards) center on feeling anxious and under-informed. But too much certainty removes the motivation for learning. It is plausible that people only try to understand when confronted by a problem (a "breakdown"), and only when they try to understand is there an opportunity for learning. But giving users problems means raising their anxiety at least for a while, and is likely to lead to complaints. So we must not only decide whether our aim is to maximize learning or job completion, but we must decide during testing whether to use breakdowns or user complaints as our main measure.

Conclusion

The design of action-centered documentation can be seen as revolving around the three levels of structure, and the four main types of design issue. This framework seems to apply to all such manuals, to let us describe which areas seem to be getting easier and which harder, and yet to cover the oldest problems in manual design. Most usefully, though, it identifies the hardest part in extending the approach to complex task domains — designing an effective access mechanism to a large collection of user tasks — and so suggests where innovation is most needed to extend the application of this approach to documentation.

References

Baxter, I. & Oatley, K. (1991) "Measuring the learnability of spreadsheets in inexperienced users and those with previous spreadsheet experience" Behaviour and Information Technology vol.10 pp.475-490.

Carroll J.M. (1989) An overview of minimalist instruction Technical report RC 15109, IBM Yorktown heights.

Carroll J.M. (1990) The Nurnberg funnel: designing minimalist instruction for practical computer skill (Cambridge, Mass.: MIT press).

Draper S.W. (1985) The nature of expertise in UNIX in B. Shackel (Ed.) Interact '84: First IFIP Conference on Human-Computer Interaction. (North-Holland: Amsterdam) pp.465-471.

Draper S.W. & Oatley K. (1992) "Action Centered Manuals or Minimalist Instruction? Alternative theories for Carroll's minimal manuals" in P.O.Holt & N.Williams (eds.) Computers and Writing: state of the art ch.15 pp.222-243 (Intellect Books: Oxford; and Kluwer Academic Publishers: Norwell, MA).

Johnson,R.R. (this volume) "The "art" of minimalism: constructing a rhetorical theory of computer documentation"

Kaptelinin, V. (1993) "Item recognition in menu selection: the effect of practice" pp.183-184 Interchi'93 Adjunct proceedings (eds.) S.Ashlund, K.Mullet, A.Henderson, E.Hollnagel, T.White (ACM)

Mayes J.T., Draper S.W., McGregor A.M., & Oatley K. (1988) "Information flow in a user interface: the effect of experience and context on the recall of MacWrite screens" People and Computers IV (conference proceedings of HCI'88) (eds.) D.M.Jones & R.Winder (Cambridge University Press: Cambridge). pp.275-289.

Mirel,B. (this volume) "Minimalism for complex information processing tasks"

Moyes,J. (1995) Putting icons into context: the influence of contextual cues on the usability of icons PhD dissertation, dept. of Computing Science, University of Glasgow.

Nickerson R.S., & Adams M.J. (1979) Long-term memory for a common object Cogntive Psychology vol.11 287-307.

Norman D.A. (1986) Cognitive Engineering ch.3 pp.31-61 User Centered System Design (Erlbaum: London).

Payne S.J., & Green T.R.G. (1986) Task-Action Grammars: a model of the mental representation of task languages Human-Computer Interaction vol.2 93-133.

Thimbleby,H. & Ladkin,P.B. (1995) "A proper explanation when you need one" pp.107-118 in M.A.R.Kirby, A.J.Dix & J.E.Finlay (eds.) People and Computers X (proc. HCI'95) (Cambridge University Press: Cambridge, UK).

van der Meij, H. (this volume) "On screen captures in manuals"

van der Meij, H. & Carroll, J.M. (this volume / 1995) "Principles and heuristics for designing minimalist instruction" Technical Communication, (1995) vol.42, pp.243-261

Wood, D., Wood, H. & Middleton, D. (1978) "An experimental evaluation of four face-to-face teaching strategies" Int. j. of behavioral development vol.1 pp.131-147.