Karen Hunley
MTPC candidate
Nov. 18, 2008

Reaching maximum potential through minimal design

Unless you are among the lucky few who consider themselves computer-savvy, you've been in this situation. After months of researching, saving, and shopping, you finally bring home a shiny new computer complete with a state-of-the-art software kit only to have your excitement wane when you see the 500-page instruction manual that accompanies it. Frustration builds as you flip through the thick paperback and see a convoluted step-by-step process to even begin using the computer. What do some of the terms within the manual even mean? And how many of those 500 pages do you need to read just to learn how to perform basic functions, like typing letters and memos? However, it seems like that basic information is buried among instructions about how to create charts, graphics, and illustrations – stuff you couldn't care less about, at least right now. You've had some experience with computers and hoped to capitalize on what you already know, but you suddenly feel like you should add *Computers for Dummies* to this bulk of reading material. You wanted to get started on a project as soon as possible, not spend hours reading an instruction manual. At the very least, why can't the manual could be more clearly organized and separated so you could quickly locate your specific area of interest?

One novice computer user may have summed up the issue best when talking with the founder of minimalism, John Carroll, who in 1980 began researching the minimalist design approach as it relates to computer manuals and other types of user support, such as tutorials. The user said, "I want to learn how to do something, not learn how to do everything" (Carroll et al. 126). This indicates users' desire for action-oriented support materials that allow them to jump into a project almost immediately, or "learn by doing." It also indicates their aversion to reading unnecessary instructional material, that which deals with complexities of the program users do not need or care to grasp. Less *is* more, in this case. Even if users do not have prior experience with computers and software, instructions and tutorials can still be more relevant to them, concise, and, therefore, shorter. As someone who shoves most instructional booklets into drawers, preferring to wing it rather than get bogged down in pages of jargon and unnecessary information, I advocate for minimalism in this paper because minimalism advocates for users like me. In doing so, I will discuss some of the interesting history behind this approach, its main principles, and common misconceptions about this user-centered approach to computer documentation design.

John Carroll and the History of Minimalism

When Carroll began diagnosing problems the general public – that is, people who weren't computer programmers and the like – was having with learning to use computers and software in 1980, he essentially found that the problems weren't necessarily due to the complexity of this new technology but rather the complexity of its instructional material (Carroll et al. 124). In his 2003 article "Revisiting Minimalism," Hans van der Meji also recognizes this, stating, "These manuals did not address the needs and desires of that [lay] audience very well, and many complaints were raised" (214). And this is certainly still an issue, even at a time when computer access is so essential that portable, pocket-friendly devices are commonplace. Yet, many lay

users likely still struggle with “preparing manuscripts, letters, memos, and other sorts of documents” because of the overly comprehensive materials that “may exhaust the patience and technical backgrounds of users” (Carroll et. al 124). A minimalist approach to instructional material, Carroll believed and later proved through experiments, would help alleviate some of this frustration. He addressed three specific areas he felt were lacking in standard self-instruction (SS) manuals when developing the first minimalist training models, areas that are still foregrounded to at least some degree in today’s minimal manuals: 1) users’ desire to get started immediately on “real tasks, 2) their preference to skip around to reading sections they find most useful and relevant to them, and 3) the inevitability of errors and the need to recover from them (Carroll et al. 125). He and his colleagues accommodated these three issues by “facilitating the tasks the learners understand and are motivated to work on, slashing the instructional verbiage” and supporting error recognition and recovery (Carroll et al. 150). The latter, especially, has continued to be a primary focus of minimalistic research and design, while other principles have been either further developed or refocused.

The first “Minimalist Training Model” Carroll and company developed that focused on these specific areas was a set of “Guided Exploration” cards, or GE cards. Among other characteristics, they contained “about an eighth as much verbiage” as SS manuals (Carroll et al. 128). The results of test runs with these cards were promising, but GE learners still voiced a desire for instructions in the form of a manual. So that’s what they gave them – an “iteratively” designed Minimal Manual (MM), meaning it was “designed, empirically evaluated, and then redesigned” according to the results of those evaluations. (Carroll et al. 128). For instance, they noticed and documented “principal errors of new users” and, in the redesign phase, included recovery information for these specific errors (131). van der Meij states that the audience-testing facet of an iterative design approach “always yields new insights and unforeseen uses” (219). Next came two experiments, or laboratory studies, to test their final product. The first studied a group of people in a simulated office environment over three days, giving about half an SS manual and the other half an MM. The subjects, who were all experienced in office work but none in the particular software system being used, were asked to learn the system on their own with the support of their respective manuals and “perform periodic performance tasks” (Carroll et al. 135). To gauge just how supportive each manual was, participants were scored on 1) the time it took them to complete their self-initiated training and word-processing tasks, and 2) their performance on those tasks. MM participants, in the end, were significantly faster and more successful, “requiring 40 percent less learning time than the SS subjects … and accomplishing 2.7 times as many tasks as the SS subjects.” (Carroll et al. 140).

The primary goal of the second experiment was to find out what specifically about the MM design was better than that of SS. Again, users were timed and asked to either use an SS manual or MM to learn and work with an unfamiliar software system. Each group then had to demonstrate the quality of their learning by performing timed tasks with the system. As with the first experiment, the results were promising – “MM subjects performed better and more efficiently than their SS counterparts” – and also indicative of just why they were more efficient– “(MM) users got started faster; they coordinated attention better; they made fewer errors, in particular, errors the Minimal Manual targeted and trained against; they made better use of recovery methods; and they made better use of the training manual for better reference” (Carroll et al. 148).

Carroll and his cohorts did not stop with the GE card set and Minimal Manual, however. They also developed the minimalistic Training Wheels tutorial, which reflected its name by making it practically impossible for users to fall off the proverbial bike. It blocked beginning users' access to certain program functions that "had often confused and sometimes trapped subjects of previous studies" (Farkas and Williams 184). In experiments in which users worked with both Training Wheels and a standard system interface, the results again showed the benefit of a minimalistic approach to design – Training Wheels reduced error recovery time and "apparently helped learners form a better mental model of the system" (Farkas and Williams 184). Janice Redish, who has worked at the American Institutes for Research (AIR) to also create more user-friendly tutorials, advocated for Training Wheels, saying new users could work on tasks relevant to them "without getting hopelessly lost when they made a mistake" (291). These initial experiments with GE Cards, the original Minimal Manual, and the Training Wheels tutorial were conducted some time ago, to be sure, but it's important to note that "many of the user propensities noted by Carroll and his colleagues are still valid today" (van der Meij 214).

Minimalism: More about users, less about page numbers

The term "minimalism" is somewhat misleading, as the main goal of this design approach is not simply to whittle down computer-related user support as much as possible. A headstrong battle for brevity – at the expense of omitting important information – has been a perceived flaw of minimalism, say Carroll and van der Meij in their 1996 article "Ten Misconceptions about Minimalism." They say it has never been the "totality of the approach," but rather "one of several principals [sic] that work together" or "a derivative property of minimalist designs, caused by other more central principles" (Carroll and van der Meij 73). And those principles tend to revolve around one focal point, or the "heart" of minimalism – the user. In other words, minimalism aims to be as user-centered and, thus, "action-centered" as possible (van der Meij 213). Everyday users want to get started quickly, not learn all the ins and outs of a software program – at least not at first. They no more want to be immediately confronted with all available information about a program anymore than a beginning jogger would want to try for five miles her first day on the track. As the user in one of Carroll's early observations bluntly stated, novice users want to "learn how to do something … not everything" (Carroll et al. 126). So while "less is more" is not the ultimate goal, it's only natural that minimization would result from an action-centered approach. One specific design measure that prevents these manuals and other documentation from being completely minimized, van der Meij says, is the extensive coverage of how to recognize errors and how to recover from them. This information is presented through minimalism "more often than in any other design approach" (van der Meij 213). And as previously discussed, from the beginning Carroll attributed much of the importance of minimalism to the error recovery component.

Another facet of the user-centered approach, which again may not always result in the fewest pages possible, is providing both procedural and – when needed – conceptual information. Not all concepts can be grasped with a single phrase such as "left-click the mouse to bring up a menu." In addition to letting users get started quickly, minimal manuals and tutorials are also obligated to "support knowledge and skills development" by methods such as presenting tasks in a logical sequence "that optimizes the possibility of learning" and offering "invitations to explore" at places where users are most likely to take advantage and benefit from them (van der

Meij 217). A user-centered manual will also suit the needs of more than one type of user and learning style. Is the person like me – one that flips to any random page in a manual, hoping that information will lead them in the right direction? Or does he or she prefer to read the manual cover to cover? Certain design strategies accommodate both of these learning styles; for example, having "self-contained" chapters and sections helps spontaneous users "immediately start processing the information" (van der Meij 217). In this case, more thorough readers are still getting everything they need and can skip over information they learned in previous chapters.

You may be asking yourself, "Isn't this what standard self-instruction manuals hope to achieve – instructions that build users' knowledge and recognize their different strategies?" And they may be successful at it, too. What really sets apart minimalism from standard design, however, are "four main design principles" that advocate for users in more specific ways (van der Meij 222). These fundamentals have been derived from 20 years of research – primarily that performed and published by Carroll and his colleagues – and have no doubt changed since Carroll's first experiments. They are most notably emphasized in his 1998 book *Minimalism Beyond the Nurnberg Funnel*, a sequel to his 1990 *The Nurnberg Funnel*, and also in van der Meij's "Minimalism Revisited." In between the publication of Carroll's two books, "key principles of design had been further tested and validated empirically in the meantime." I touched on a couple of these earlier, but each is an equally integral part of achieving minimalistic – or user-centered – design, thus, their ideas and characteristics should all be discussed in some detail. These design principles are "1) choose an action-oriented approach, 2) anchor the tool in the task domain, 3) support error recognition and recovery, and 4) support reading to do, study, and locate" (van der Meij 222).

Principle 1: Choose an action-oriented approach

We now understand that users are most often concerned with learning how to *use* software, not learning *about* software. "They are not hardware- or software-oriented, but ask- and action-oriented" (Van der Meij 222). This is not to say, however, that they don't need more than just simple instructions in order to truly comprehend how to use new software and eventually sit down in front of their computer without their minimal manual in hand. They need to be able to internalize the information, or achieve "knowledge transfer," so they can tackle new tasks. For instance, a user may only be interested in creating one type of file initially, but chances are they will want create another type later. As they get more familiar with a program, users will be more apt to try new and more complicated tasks, but this will be difficult if they've not developed "deeper insights" into preceding ones (van der Meij 222).

To achieve a well-rounded, action-oriented design that supports users in learning new tasks, van der Meij says two concepts should emerge: screen captures and invitations for user exploration. A screen capture is essentially like taking a picture of your computer screen during a specific task "in order to show and explain important screen elements" involved in that task. Blocks of text, or "procedural information" often accompany these screen captures, with lines or arrows indicating to which part of the "captured" screen the information is referring. "[Screen captures] can give a visual impression of how screens follow one after the other as the user acts on the software" (van der Meij 222).

Invitations to explore help develop users' insights into the program in a different way, by encouraging investigation of tasks and elements that aren't integral to understanding the program – at least not initially – but may help broaden users' knowledge about what's possible. And most users are curious enough to explore an application anyway, van der Meij states (224). User-exploration opportunities in minimal manuals usually either take the form of "on-your-own" sections or exercises that give specific instructions, both usually at the end of chapters. In a 2000 study van der Meij cites in "Minimalism Revisted," researchers found that users were more prone to engage in exercises than "on-your-own" invitations and "completed 88 percent of the exercises successfully" (224). Perhaps this is because the goal of exercises is user-oriented, "capitalizing on users' prior knowledge by stating the user's goals rather than the information needed to achieve these (goals)," whereas "on-your-own" sections are more geared toward teaching the "options and possibilities of an application." (van der Meij 225).

The action-oriented idea of user exploration, or "discovery learning," is sometimes confused with learning by trial and error, state Carroll and van der Meij in "Misconceptions," because users are encouraged to work independently with a program shortly after they begin learning it. In their critique of minimalist documentation, David Farkas and Thomas Williams contend that exploration actually inhibits learning by encouraging users to induce, through trial and error, the correct procedures needed to accomplish that work before they've reached full comprehension of an idea or task (182). Carroll and van der Meij make it clear, however, that minimal manuals do not disregard "curriculum and support" and rely only on discovery learning to teach concepts, rather, minimalism "entails fundamentally different kinds of curriculum and support than passive and rote-structure approaches" (Carroll and van der Meij 75). As stated earlier, invitations to explore are placed at the end of chapters so users have the prerequisite information to take the wheel. It should also be noted that while these invitations are not always absent in standard instructions, they are not as explicit and easily accessible as in minimal design material (van der Meij 225).

Principle 2: Anchor the tool in the task domain

Have you ever opened up an instruction manual and not even read past the introductory section before feeling confused, maybe even intimidated, by the material? It's likely because the manual offered you too many possibilities too soon – jargon and activities outside your initial realm of comprehension. Van der Meij offers the example of a standard self-instruction manual introducing the software program HyperCard "by arguing that card stacks are more flexible than relational databases" (226). Two questions immediately come to mind: what does this even mean, and why is it placed in the introduction? "Special options" such as this, Van der Meij says, "can only be realized after considerable practice … there is a risk of de-motivating the user by promising more than people can achieve after an introductory training period" (226). Minimal manuals, on the other hand, are designed to prevent feelings of frustration by initially offering tasks that are realistic and meaningful to the user. Through their positive experiences in a realistic domain, users may then be more inclined to take on more complicated tasks and broaden their knowledge of the program (Van der Meij 226).

While "anchoring the tool in the task domain" indeed capitalizes on users' desires to get started on relevant tasks, as well as the motivation that results from completing tasks successfully, the intent is not to "merely satisfy every preference of the user" (Carroll, van der Meij 82). This is

yet another obstructed view about minimalism. Thus, it's not about adhering to a user's every whim – it's about "respecting the integrity of the user's activity" by designing a manual that helps users meet their goals and gain the confidence to tackle new tasks (82). In fact, users may not love the manual at first – possibly finding the minimal instructions challenging – but later find they learned more through working for the answer rather than having it handed to them immediately (82).

Principle 3: Support error recognition and recovery

Mistakes are inevitable when traversing an unfamiliar technological terrain, but they don't have to ruin your day if you know what kind of mistakes to expect and how to handle them. This is another way minimalism advocates for users – by supporting error recognition and recovery. After all, previous studies show that mistakes cost users as much as 25 to 50 percent of their time, so users should have the tools to deal with them (Carroll and van der Meij 78). This is not to say minimalism "advocates error-prone programs so users can learn from making mistakes," another misconception of this design approach. (van der Meij 227). Rather, minimalism does not ignore the fact that user errors frequently occur and, thus, devotes much design effort to ensure users can overcome and capitalize on their errors. These efforts can include incorporating detailed explanations, or critiques, of errors within a users' "programming and design style;" checkpoints, or "explicit error recovery suggestions" (Carroll and van der Meij 79).

When there is no productive knowledge to be gained from working through errors, or when errors could cause serious consequences, minimalism "provides a safety net" (Carroll and van der Meij 79). Automatically blocking certain user actions is one way to do this – remember Training Wheels? – keeping users from losing important work or spending valuable time recovering from a mistake. Acknowledging error and error recovery has not been shown to be a priority in most standard software user support, according to studies by Carroll and van der Meij. One survey showed that six out of eight conventional word-processor manuals "failed to help users deal with a set of specific problems." That same survey showed – this time through analysis of 60 conventional manuals – that when help was present, it was difficult to locate. (Carroll and van der Meij 79). Van der Meij also conducted research that quantifies the benefit of error support and recognition; users in one study increased their software program training speed considerably with a minimal manual because they spent 38 percent less time dealing with errors. Interestingly, he also found that "15 percent of all presentations of error information was not used to remedy 'regular' mistakes but served exploratory purposes" (Van der Meij 229).

Principle 4: Support reading to do, study, and locate

Not every user of a new software program or computer comes to it with the mindset of getting in and out as quickly as possible. But most instruction manual reading is in fact action-oriented, especially when users are on the job. In her article "Reading to learn to do," Janice Redish cites a study that found only 15 percent of workplace reading is "reading to learn," the majority of reading being geared toward specific tasks (289). However, minimal manuals and tutorials are not made only for the workplace, and users do "sometimes read to study and sometimes read to locate information." That's why it's important for minimalism to support all these situations; the iterative design testing discussed earlier is one way to do this, and a software technique called "fading" is another (Van der Meij 230).

Fading is a way for users to internalize and remember information so that they do not have to continue consulting the user support material. As Van der Meij says, “without fading, there is a serious risk that the user will never really learn a procedure” (230). In this process, uses are gradually given fewer direct instructions (e.g., “move the arrow on top of the file icon and double-click to open”) and more indirect yet “conceptual and meaningful” information (e.g., simply, “open the file”) (230). The level and speed of fading can vary, but experimental studies have shown users benefit most from slow fading – “detailed instructions at the start followed by a slow decline in the support that they receive” (231). This process can be related to teaching a child to walk; over time they need less physical support and more words of praise when they get it right. In the world of software, praise comes in form of a successfully completed task.

Sometimes iterative testing shows that instructions in a manual, tutorial, or software help system neither support reading to learn nor reading to do – they simply aren’t needed at all. For example, Apple Macintosh, which applies minimalistic design concepts within its software help system, found that users did not need specific instructions on how to play a CD. So, in keeping with another design principle, Apple “replaced this and other instructions with a format that focused on error handling” in its next software model (van der Meij 232).

The concept of reading both to do and to learn may come into play more with tutorials than instruction manuals. While obviously similar to instruction manuals, tutorials should help readers genuinely grasp concepts of a program and become comfortable with it, according to Redish (290). These users may need to know a little more about why and how a program works than minimal manual readers. While minimal manuals generally do not contain previews or elaborations of instructional material, tutorials should still provide this additional information, Redish says (291). Other concepts of minimalism, however, align neatly with Redish’s findings about tutorial users: “[They] will not read long prose passages … although we want them to understand the concepts and structures of the program, we have to build that knowledge through their use of the product, not by giving them pages and pages to read” (290). Like Carroll, she also found that tutorial users work best when they can focus on tasks that are relevant to them (293).

Minimizing doubt: other minimalism misconceptions

We’ve already addressed – and debunked – some of the most common misconceptions about the minimalistic approach to software user support: minimalism always means brevity and trial-and-error learning, minimalism overemphasizes the importance of error recognition and recovery, and minimalism “merely reflects the preconceptions of users” (Carroll and van der Meij 72). There are more, to be sure, and every one should be addressed to provide a thorough analysis of minimalism and demonstrate that it is clearly the most user-centered design strategy. It is important to note that these misconceptions are “understandable” and best described as “simplifications of what (is) intended by minimalism” (Carroll and van der Meij 72).

Closely related to the misconception of brevity, “incomplete instructional analyses” are often thought to characterize minimalism (Carroll and van der Meij 73). These manuals may contain incomplete instructions in order to help users become autonomous with the program and guide beneficial exploration, but this not mean designers fail to properly analyze documentation (73). If anything, a minimalistic approach “requires more, not less analysis” to determine what

information can be left out of an instruction document. Designers most thoroughly investigate users' "knowledge and skill against tasks they will be carrying out" (74). A similar misconception is that minimalism is meant to offer a "complete documentation solution" (82). If this were the intent, minimal manuals would in fact appear incomplete when compared to more comprehensive manuals, such as tutorials. But Carroll and Van der Meij do not claim minimal manuals contain every possible piece of instructional verbiage – after all, that is what sets minimal manuals apart from standard manuals.

Incomplete documentation, meant to support quick, hands-on learning, does not necessarily equal leaving behind those who learn best by reading. Carroll and Van der Meij address this misconception by saying that they try to balance some users' need to read with design techniques that support learning-by-doing. It's important to first consider, however, that there exists no clear evidence that proves users can learn skills just by reading about them, without hands-on practice. And even most people who claim to "read everything before trying to do anything" show eagerness to engage with the program as they read about it (76). Moreover, users who "learn best by reading" will more likely retain information and stay motivated if the manual does not have such an overwhelming, "massive appearance" (77).

Another falsity is that minimalism is "just another word for job aid" (Carroll and van der Meij 79). While this is not completely out in left field – job aids are also action-oriented – there are considerable differences between the two types of learning material. Because jobs aids are most often "used for reference by people who earlier received instruction … they are not designed to convey a model of the system or of the task domain" (79). They do not require the user to reason, nor do they address errors; they are merely a refresher for steps an employee has already been taught (79).

Taking into account the foundation and main principles of minimalism, it can also be easy to assume that "minimalism works only for simple domains" (Carroll and van der Meij 81). The concept did apply to mostly easy-to-learn word processing programs in its early days "because this domain offered the chance for greater and more rapid practical impact," but it has evolved to apply to more complex domains as well (81). There is no reason why minimalism's simplified terms, couched in an action-oriented design approach, can't be applied to a variety of software user support. The results speak for themselves: in addition to its presence in word-processing software, minimalism has been applied to digital design programs like Pagemaker, desktop publishing software, "time-registration programs," e-mail applications, database programs, and more (81). To confront another misconception, it's hard to imagine such a diverse number of companies adhering to a design concept that has "no theoretical foundation" (83). Carroll and van der Meij say that some writers see minimalism as "essentially consisting of rules of thumb" that, despite the fact that they work well, aren't supported by logical reasons *why* they work well (83). This is clearly not the case, since minimalism "would not be capable of providing a framework for developing new approaches to instruction and documentation … if its foundations could not be identified and explicated" (83). To support the ideas in just his first book, *The Nurnberg Funnel*, Carroll cites more than 200 books and technical documents from fields like educational psychology, cognitive science, and human-computer interaction (83). And researchers' knowledge about users' learning tactics and problem-solving abilities has only broadened and, in turn, strengthened the ideas behind minimalism in the last 20 years.

Conclusion

Users are at the heart of minimalism. They are closely studied and queried during iterative testing of user support, and then changes are made to the documentation that reflect not only users' preferences but what is best for them. It's hard to imagine that all software companies would not want to produce the most user-centered documentation possible, but some have still not adhered to minimalism more than 20 years after its inception. Macintosh, IBM, Lucent Technologies, and Hewlett-Packard are a few that have, in fact, reported positive outcomes as a result of embracing minimalism; Hewlett-Packard reduced its laser printer user manual from 650 pages to 270. (Van der Meji 232). And as Carroll says, the minimalist slogan 'less is more' can extend beyond that which is beneficial to users to companies' time and cost to develop documentation. "The analytic and subskill phases of design for the [original] Minimal Manual together required less than a man-month of effort" (Carroll et al. 150). One satisfied client of Stuart Title Guarantee Company, which created a hypertext application based on minimalism, ordered about 50 copies of the program each month "for a return on (the company's) investment in five months' time. The client stated that "'[the company] had been considering his needs'" (van der Meij 232). Actually, if the Stuart Title remained loyal to the foundation of minimalism – its four main principles – the company not only considered his needs but made them a central focus of the documentation design process.