CREATING MINIMALIST INSTRUCTION

John M. Carroll, *The Pennsylvania State University*

The early 1980s were disruptive for computing. In many countries, the set of persons who regularly used computing expanded rapidly from a tiny and homogeneous set of data processing professionals to include nearly everyone. The dawn of the “new user” entrained challenges for training and instructional design: the economics of smaller systems ruled out vendor-provided, face-to-face training, which was succeeded by the paradigm of self-instruction. Our group at IBM’s Watson Research Center was in the cross hairs of this challenge: IBM wanted to expand office systems toward microcomputer platforms, and to enter the personal computing business. Our initial studies focused on office workers using standard self-instruction materials to learn these new systems. We identified key problems in self-instruction, and created alternative designs, such as the Minimal Manual and the Training Wheels Displaywriter, to address them. Our “minimalist” designs sought to leverage user initiative and prior knowledge, instead of controlling it through warnings and ordered steps. It emphasized that users typically bring much expertise and insight to this learning, for example, knowledge about the task domain, and that such knowledge could be a resource to instructional designers. Minimalism leveraged episodes of error recognition, diagnosis, and recovery, instead of attempting to merely forestall error. It framed troubleshooting and recovery as learning opportunities instead of as aberrations.

John M. Carroll is Distinguished Professor of Information Sciences and Technology at the Pennsylvania State University. His research interests include methods and theory in human-computer interaction, particularly as applied to Internet tools for collaborative learning and problem solving, and the design of interactive information systems. His work on minimalist design was recognized with the ACM Joseph T. Rigo Award in 1994 and the IEEE Alfred N. Goldsmith Award in 2004. He has received many other honors and awards including an honorary doctorate from Universidad Carlos III de Madrid in 2012.

Copyright © 2014 by the International Journal of Designs for Learning, a publication of the Association of Educational Communications and Technology. (AECT). Permission to make digital or hard copies of portions of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page in print or the first screen in digital media. Copyrights for components of this work owned by others than IJDL or AECT must be honored. Abstracting with credit is permitted.

INTRODUCTION

As our society charges into an age of notebook and smart-phone computers, it is interesting to revisit the microcomputer revolution that occurred in the decade surrounding the year 1980, and its consequences for the design of user interfaces and of user training and instruction.

In 1976, I joined IBM’s Research Division in Yorktown Heights, New York. As a new PhD in psycholinguistics, I became involved in studies of software design understood as human problem solving, and of names and naming in the context of command languages and personal files and directories (they were not called “folders” yet!). In 1979, I served on a corporate task force to identify emerging needs and opportunities with respect to users and usability.

This was eye opening; 1979 was the year that the Cognitive Science Society was founded and had its first meeting. One original theme differentiating cognitive science from cognitive psychology was its commitment to complex application domains, and not merely to simplified laboratory models of human behavior and experience. The articulation of this commitment in the context of early graphical user interface work at the Xerox Palo Alto Research Center (PARC) led directly to the formation of human-computer interaction.

At IBM, new projects and areas began to form. I joined Clayton Lewis and Robert Mack to begin an investigation of learning through self-instruction. Through the first decades of computing, training was typically quite personalized. Computers were huge and expensive, operated by teams of expensive professionals. Face-to-face training courses, and other field support, were typically bundled into the pricing of such systems. However, as IBM and other companies moved toward much smaller machines, and ultimately to personal computers, training and other field support could not be delivered face-to-face; such services would cost much more than the systems themselves. This was the original motivation for self-instruction designs.

Our investigation continued through the 1980s, though many of our key insights and our most influential designs were developed in the first few years. Thus, in this paper, I am looking back more than 30 years. Our design cases were

directly inspired by empirical studies of self-instruction, and by a particular interpretation of what those studies meant; I turn to those studies and their interpretation in the next section. I will then briefly present two of our early designs: the Minimal Manual and the Training Wheels Displaywriter.

Because these are “historical” design cases, I have had a lot of time to ponder what they mean, and honestly, to reconstruct what they might mean. These reflections are separate from the design work itself, which had its own validity and impact at a particular time, but which is now “just” history. Here I will address three reflections; first, I think our work on minimalist design succeeded to the extent it did because we adopted a “positive design” perspective with respect to human learning; we tried to strengthen evident strengths of human learning as much as to mitigate problems. Second, this work ultimately convinced me that designs are a key medium for codifying and developing applied understanding, that they are more than just products of knowledge, and should be regarded as expressions of knowledge. Finally, somewhat painfully, designs disappear almost immediately; by the time I wrote my first book on minimalist instruction, none of our original designs could even be run; indeed, the original Displaywriter, a focus for much of our early work, was re-embodied as a program for the IBM PC in 1984. Traces of designs can live on; IBM’s Displaywrite PC software incorporated many of our ideas. At the remove of 30 years, the traces get faint; I am asked regularly how to design minimalist information for the Web! I call these three points reconstructions of minimalism because at the time that we did the original analysis and design work, I did not understand our work in this way.

INSTRUCTION AND THE ACTIVE USER

Our work on minimalist instruction did not start out as a design project. Our initial goal was to understand how self-instruction actually worked so that we could inform and guide IBM designers. We were interested in the characteristics-in-use of the self-instruction model, and examples of it in IBM instructional products. This first generation of self-study tutorials employed hierarchical decomposition and drill-and-practice regimes (Gagne & Briggs, 1979). A basic skill like “typing a short document” was decomposed into subskills (such as navigating to the Typing Menu), which were then successively decomposed (selecting, pressing Enter). Users were expected to recognize, understand, and execute lower level skills, subskills, and basic skills until they were fluent.

We focused our study on clerical workers, people who were expert at typing and other office tasks. We configured our laboratory to look like office space, and asked our participants to use the self-instruction materials, to verbalize their thoughts and experiences as possible (to “think aloud”), and to try to ignore us as we watched them (Mack, Lewis, & Carroll, 1983).

It was not pretty. For the first few moments, it often did look like Figure 1, an image from IBM’s public archive depicting a user of the Displaywriter. Our participants were excited; after all, they were office professionals, and we were paying them to learn new and relevant skills. But quite quickly, people tended to get off track. They noticed and worried about small details that were not addressed in their manuals, and whose primary consequence was to confuse them, frustrate them, and often lead them draw incorrect conclusions and make errors. Conversely, they missed noticing equally tiny details in the manual and in the user interface that were absolutely critical to what they were trying to do, or worse, were critical to something they later would do, often at such a remove that they were never able to disentangle the problem and make further progress.

FIGURE 1. Model using the IBM Displaywriter (courtesy of IBM Archives, IBM Displaywriter, n.d.).

For example, in the Displaywriter, successfully queuing a document for printing caused the system to display the system’s root menu. The design intent was to convey to the user that the system is ready for another task. However, some users struggled to make sense of the root menu as feedback specific to printing; they looked for clues. Others interpreted this menu transition as evidence that they had failed to print, and repeated the procedure, queuing a second print job. Engaged and creative interpretations and actions often led to frustrating and confusing problems.

Displaywriter stored documents on floppy disks, called work diskettes (Figure 2). In creating documents, users were prompted to specify the work diskette on which the document would be stored. If the user mistyped the work diskette name and pressed Enter, the system would prompt the user to insert the diskette with the mistyped name. This was a common error, but there was no simple or obvious remedy for it. From the users’ point of view this is a tiny detail

that is also a showstopper. Understandable and mundane errors nonetheless piled up, error upon error.

In an episode I cannot forget, an expert clerical worker became snarled in a tangle of confusions and actual errors. She was generating plausible ideas and interpretations at a ferocious rate, but they were mostly all wrong. Suddenly, she started to flush. Stammering apologies to a colleague and I, she collected her coat and handbag, and began to move quickly toward the door. We had planned a protocol for this situation. We expressed our thanks, apologized for the difficulties with the technology, emphasized that she would still be paid fully for her participation, and escorted her out of the building. I can't forget her because I have never seen another person so defeated. Unfortunately, I have seen many close seconds, and many of them were in this study.

GUIDED EXPLORATION

For several years, my colleagues and I informed and guided IBM designers and others. We were popular speakers about user miseries, but we wanted to do more than describe what was happening. We wanted to change it. We were inspired by user interface design work at PARC and later at Apple, though none of it fundamentally addressed the specific problems of self-instruction (Carroll & Mazur, 1986).

In 1982, we started a project on guided exploration; our idea was to replace comprehensive self-instruction with modular and deliberately incomplete resources to encourage and support self-initiated learning. For example, the card in Figure 3 addresses the user concern with typing. It presents four brief hints focused on things to do toward the concern: name a document, display the typing page, keys to try to move the cursor, and end the document. It presents a simple checkpoint (text appears), and error recognition and recovery information.

The guided explorations cards were more a design probe than a design. We wanted to confront the primary design commitments of self-instruction as a way of further studying them, and of beginning to imagine alternatives. The fatuous instruction DO NOT DO ANYTHING UNTIL AFTER YOU HAVE READ EVERYTHING was our touchstone. Despite the caps, we had never seen anyone follow this instruction. In contrast,

FIGURE 2. IBM Displaywriter (dual diskette drive on right, printer partially visible on left). Image courtesy of We Look 4 Things (n.d.).

FIGURE 3. Guided Exploration card addressing a basic user concern, with a set of hints, a checkpoint, and error information.

the cards conveyed to users that they could get started immediately, engage their own reasoning and improvising, integrate planning and acting, and make progress in goals meaningful to them. Importantly, the cards, in their form and emphasis, conveyed that errors and error recovery are standard and routine, not failures or crises.

We designed 25 guided exploration cards addressing the content of about 100 manual pages. We carried out a study contrasting clerical works learning with the manual or the cards. We found that participants made more progress in less time, and that participants spent a smaller proportion of their time reading cards versus working with the system. We also looked in detail at specific characteristics of the learning experience. Cards learners engaged in exploration far more than manual learners, and with far greater success. They also recognized errors better and recovered from them more often and more rapidly (Carroll, Mack, Lewis, Grischkowsky, & Robertson, 1985).

THE MINIMAL MANUAL

Although the guided exploration project was inspiring and successful, the cards also raised new issues. For example, learners free to pursue their own meaningful goals occasionally pursued goals that were unwise or impossible to achieve. Some learners were uncomfortable with the extreme lack of structure in a “deck” of cards. We felt that an appropriate challenge would be to investigate whether our ideas could be directly incorporated into the genre of self-instruction manuals, and not just embodied as a design probe to deconstruct that genre.

We created the Minimal Manual as a self-instruction manual, incorporating many of the design concepts of the cards. Thus, in Figure 4, the user is given rather general directions for exploring cursor keys and deletion, emphasizing doing things independently, and seeing their effect on the display. The objective, as for the cards, was to encourage and support learners to reason, improvise, plan, act, and think about goals. Instead of telling them exactly where to move the cursor or what to delete, we emphasized that that was their choice to make. Relying on user initiative and sense making, rather than extensive step-by-step instructions, allowed our Minimal Manual to be 45 pages long; less than a quarter of the official manual.

As with typical self-instruction manuals, prior topics and skills are leveraged in subsequent presentations; thus, the deletion exercise in Figure 4 utilizes the cursor movement skill

introduced immediately before. Unlike standard manuals of that time, error recognition, diagnosis and recovery was emphasized. Also unlike standard manuals of that time, each topic in our manual included open-ended suggestions of small projects users could try on their own. Apparently, our manual was a convincing alternative version for the original IBM manuals; the manual we wrote is the one most easily accessed on the Web as the IBM Displaywriter training manual (IBM Displaywriter System Operator Training, n.d.).

As was the case the with the cards, our Minimal Manual produced better instructional outcomes; participants got started faster, spent more time coordinating the manual with user interface interaction, were more successful in referring back to earlier sections of the manual as they progressed through the training, made fewer errors overall, and used error recovery methods more and more effectively. Overall, they learned more in less time. They were also more successful in applying their learning of the basics in further learning (where all participants used the name manual; a so-called

FIGURE 4. The Minimal Manual incorporated design concepts from Guided Exploration cards but embodied them in a standard self-instruction manual design.

transfer effect). See Carroll, Smith-Kerker, Ford, & Mazur-Rimet (1987).

An extremely important aspect of the Minimal Manual project was that most of the testing work was carried out at IBM's Austin Development Laboratory, the home of the Displaywriter product. There is always a concern, of course, whenever design and evaluation are carried out by the same group, and this collaboration allowed us to bring in disinterested partners, and to replicate and extend our findings. Our small group could never have afforded a study involving transfer effects for advanced Displaywriter functions; the Austin study involved participants in a real office work environment for three full days. Perhaps even more importantly, this collaboration helped our colleagues in Austin to become familiar with our design and its consequences for users. This kind of connection is invaluable since it facilitates the most difficult step in design innovation: getting someone else to understand and adopt a new design.

TRAINING WHEELS

A complementary design direction for realizing minimalist instruction was the user interface itself. Online self-instruction also emerged in 1980s, and some argued that the combination of new graphical user interfaces and online self-instruction would address various problems of procedure following, coordinating the system and instruction, error and error recovery, etc. We studied the Apple Lisa in 1983, and found no basis for this hope. Indeed, graphical user interfaces were still novel and less intuitive than their designers expected, and online tutorials often required manuals and added new coordination problems (Carroll & Mazur, 1986).

Our approach to online self-instruction was again inspired by guided exploration: we started by asking what complexities, and therefore what risks for error and misinterpretation, we could eliminate or mitigate while still providing an instructional experience anchored in the system software. Our training wheels concept is simple and general: we edited the Displaywriter system software to block the execution of functions unnecessary to beginners, and functions we had found to trigger difficult or distracting errors. Here again, our relationship with the IBM Austin Lab was critical: a Displaywriter system programmer created the Training Wheels Displaywriter for us, modifying the actual Displaywriter system code.

FIGURE 5. The Training Wheels Displaywriter blocked the execution of advanced and problematic functions, producing a feedback message without changing system state.

Figure 5 illustrates our approach. A common error was mistyping the Work Diskette name. This should have been a trivial slip, but recovering from it required system functions that were not introduced at the beginner level. In the Training Wheels Displaywriter, mistyping the name of the Work Diskette (specified as TRAIN in the self-instruction manual), elicits a message rejecting the name, and leaving the user in exactly the same system state as before the typo. This is the key design idea: when the user requests advanced functions, or makes a recognizable error, the Training Wheel software rejects the command, and explains to the user that that function is not available in the training system.

Our rationale for this design was that this intervention would allow users to engage in reasoning and improvising, but suffer fewer consequences with respect to error recovery. We indeed found that users were able to progress more rapidly and successfully through initial self-instruction. They did better on final performance and comprehension tests, and they felt better about the experience. They spent far less time in error recovery.

Although error recovery activity can itself be a powerful occasion for learning, tangles of cascading error could lead to confusion, as we had seen in our original studies. The Training Wheels design did not eliminate error recovery; only certain errors were intercepted. Indeed we found that participants using the Training Wheels and the full system made comparable numbers of non-blocked errors. We found, however, that for non-blocked errors, the Training Wheel participants improved their recovery times dramatically

as they progressed relative to those using the full system (Carroll & Carrithers, 1984).

We hypothesized that initial self-instruction with Training Wheels would not penalize users when they subsequently transferred to the full system (near transfer). We also hypothesized that seeing how the advanced and problematic functions were accessed, and receiving confirmation about how they were accessed, would be useful to users later when they were able to articulate goals that really required these functions (that is, in far-transfer to the full Displaywriter system). We found that, in both cases, Training Wheels participants did better than those whose original self-instruction experiences were in the full system (Catrambone & Carroll, 1987).

ADOPTION AND GENERALIZATION

The success of our minimalist instruction design work in the early 1980s required significant immersion in the word processing systems of that time, Displaywriter especially. This is unavoidable in that we could not have understood how to reshape Displaywriter manuals and software without such immersion. But it is also self-limiting; by the time our project concluded, Displaywriter was making way for PC-word processor programs, including Displaywrite. Our work had some immediate impact on these successors, but we wanted broader and more lasting impact than that.

Our original goal was to inform and guide IBM designers. Our collaborations with the IBM Austin Lab, and other interactions with product development organizations in IBM were critical to advancing this goal. However, in retrospect, an equally significant path to adoption was publication. Because our group was located in the Research Division, external publication was regarded as important evidence of the soundness and impact of contributions. Our early publications on minimalist instruction were appropriated by many other research groups, and by product development groups outside IBM.

Figure 6 shows a guided exploration card developed and shipped by Northern Telecom in the mid-1980s. Xerox,

Hewlett-Packard, and many other companies adopted ideas from minimalist instruction (Carroll, 1990). During 1985–1993 (I left IBM in January, 1994), I received many calls from IBM product managers who wanted to consult with us on instructional designs after encountering implementations of minimalist instruction in competitive products. This sounds ironic, and I am not in a position to assess the business case, but I do know that valuable connections among IBM professionals were created that were originally mediated by the external research literature.

Adoption greatly facilitates generalization. We did not think our ideas were specific to word processing, or to office work, but the only meaningful test of generality is through diverse

Introduction	
Your Norstar M7310 Phone offers the simplicity of a standard telephone plus the versatility of many special features. This card will help you to use your new Norstar phone and to customize some of its features. When using your phone refer to the detachable Feature List and the Receiver Card under the receiver for quick reference.	
Telephone Buttons	Rls To release a call or a feature press this button.
	Hold To put a call on hold press this button.
	Feature To use a feature press this button and then enter a feature code.
	Press the dialpad to make a call or to enter feature codes.
	When using the receiver, handsfree speaker or headset, press this button to adjust the volume.
Adjusting Ring Volume	
Try using some of these buttons now to set the volume of your phone's ring.	
1 To make your phone ring press Feature * 8 0 .	
2 To reach desired volume press repeatedly	
Display and Display buttons	
Display The top line shows you the time and date, and call information. The display helps you use Norstar features. Follow the instructions.	
Display buttons Function-names for the three display buttons appear on the second line of the display.

Selecting Ring Type	
Now select a distinctive ring to help you distinguish between your phone and others nearby.	
1 Press Feature * 6 .	
2 Press the Next display button until you hear the ring type you want.	
3 Press OK display button to finish.	

FIGURE 6. Guided exploration cards for a telephone system (used with permission of Northern Telecom)

research and implementation. The many product adoptions inside and outside IBM were critical to making this case. In 1996, I organized a workshop for the Society for Technical Communication on minimalism beyond the Nurnberg Funnel, trying to clearly signal that I knew much more could be done than I could ever do (Carroll, 1998).

In our own technical work, we deliberately moved on to domains of value at IBM different from document preparation, for example database systems and object-oriented programming environments. We designed a programming environment called View Matcher that displayed multiple coordinated views of a Smalltalk program as it executed. We focused on this problem because in the late 1980s IBM had many experienced procedural programmers who wanted to adopt the object-oriented programming paradigm, and IBM had entered into a relationship with Digitalk, with regard to its Smalltalk/V, and later Smalltalk/VPM, products.

Figure 7 shows a View Matcher analysis of a program that plays blackjack: In the upper right is the game; in the upper left is the stack of Smalltalk methods currently executing—the program has been halted at the execution of the `takeCard:` method; in the lower left are panes reporting variable values (the card the player has just drawn is the four of clubs) and below those the commentary pane presenting documentation about what the program is doing at this point; in the lower right are the several panes composing the class hierarchy browser, a key system tool, displaying the code that implements the `takeCard:` method—including the halt message. The user can step through the program execution by selecting methods in the execution stack and manually analyze how the program carries out its computations (Carroll, Singer, Bellamy, & Alpert, 1990).

Our users in this project were again experts. Where before we had worked with expert clerical workers, in this project we worked with experts in procedural programming languages like PL/I. As we had with the word processing work, we tried to leverage user expertise in our design. We expected that expert programmers would be able to leverage this expertise to analyze communication among Smalltalk objects through the execution stack, even though the software design concepts of object-oriented programming might be somewhat new to them.

FIGURE 7. The View Matcher presenting coordinated views of the halted execution of a blackjack game.

We designed the blackjack game to have an intriguing quirk; it occasionally cheated by taking two turns. We hoped that this sort of software misbehavior would appeal to people who were professional programmers and had debugged many programs. We developed a minimalist self-instruction package around the View Matcher (Rosson, Carroll, & Bellamy, 1990); the package guided users to encounter basic software structures of the game at different levels of analysis; again, we hoped, building upon their professional capabilities to help them quickly see the key paradigmatic differences between procedural programming and object-oriented programming. The View Matcher provided a customized bundling of standard Smalltalk tools with documentation we created (the commentary pane, and the halts we placed in the code), but it was completely integrated with Smalltalk, and users were free to break out whenever they wanted, and to continue learning Smalltalk on their own.

I have never stopped generalizing and further exploring minimalist instruction. In 1994, I became a professor at Virginia Tech. Among other things, professors teach and write textbooks. Mary Beth Rosson and I (2002) wrote a text on scenario-based usability engineering, and created a set of online cases (Figure 8) illustrating the use of the concepts and techniques in our book. Students can browse these cases in any order; they can examine how specific designs emerged from analysis and fieldwork; they pose their own inquiries, such as why did one project used prototyping differently than another (Carroll & Rosson, 2005, 2006).

Vygotsky) learning theories (Carroll 1990, 1998), but it is critical to note that the hierarchical decomposition and drill-and-practice designs we reacted to were also inspired by these theories. Even for the simple guided exploration card, one can perhaps never capture all the relevant nuance and consequence of experience. This reflection led me to investigate heuristic design representations and design rationale (Carroll, 1995; 2000; Burge, Carroll, McCall, & Mistrik, 2008). I think the project of curating design cases, in the sense of IJDL, is another constructive reaction to the centrality of artifacts in practical inquiry.

DESIGNS DISAPPEAR

My colleagues and I were drawn to create minimalist instruction in order to transform the awful instructional experiences we studied in the early 1980s. Changing the world sounds a bit preposterous, but of course people do it all the time. As I have explained, working toward this objective required us to immerse ourselves in the design milieu of self-instruction, and ultimately in our own designs. But there is a painful consequence of immersion in the world and in design: to the extent you succeed, you may very well disappear.

As I mentioned earlier in alluding to my own current work on mobility and social computing, many of the main ideas we discovered and developed in our designs in the early 1980s are now not only obvious, they are invisible. I think this just comes with the territory. We people of 2014 do not really “see” graphical user interfaces, how could user interfaces be otherwise? But in 1980, user interfaces in general and graphical user interfaces in particular were very visible; they were shocking.

This overarching phenomenon of disappearance by being transformed into a cultural norm is complemented by a lower-level of disappearance: historical design work depends on infrastructures of various sorts, and these also are constantly being transformed, or just abandoned. Many historical designs can no longer be used or experienced at all; they no longer exist in the sense that the infrastructures they depended upon are now gone. For example, the Displaywriter, once an office system innovation, is now a museum piece. I kept one in my lab at the IBM Watson Research Center until 1994, when I moved to Virginia Tech. I believe I still have the program diskettes with the system code for the Training Wheels Displaywriter in the back of a file cabinet somewhere, but I should paste them into a scrapbook: I don’t have access to a machine that could run that code.

As I look back, I understand better now that part of what was going on in our investigation of self-instruction designs was a major transformation of the technological infrastructure for office work. In 1981, I watched a participant balk at the instruction “Backspace to erase.” She knew what backspace was (on a typewriter), and she knew very well that it did not

erase anything. Today, there may well be no one alive who would have this problem, no expert typists to be confused by design metaphors that only partially match. The transformation of office work has left us with no typists in that sense; today there are only users. We—my colleagues and I, and our hapless participants—were too close to what was going on in 1981 to fully realize what we were part of.

ACKNOWLEDGMENTS

Our original work on minimalist instruction was supported by the IBM Corporation. Subsequent work has been supported by the Society for Technical Communication, the US National Science Foundation, and the US Department of Education. I am most grateful to Clayton Lewis and Robert Mack, my close collaborators at the start, and to Sandra Mazur and Mary Beth Rosson who helped to develop this work subsequently. I thank the editors and reviewers for organizing this special issue, and for their feedback and encouragement.

REFERENCES

- Avital, M., Boland, R. J. & Cooperrider, D., Eds. (2008). *Designing information and organizations with a positive lens: Advances in appreciative inquiry* (Vol. 2). Oxford, UK: Elsevier Science.
- Burge, J., Carroll, J. M., McCall, R. & Mistrik, I. (2008). *Rationale-based software engineering*. Berlin, Germany: Springer.
- Carroll, J. M. (1990). *The Nurnberg funnel: Designing minimalist instruction for practical computer skill*. Cambridge, MA: MIT Press.
- Carroll, J. M., Ed. (1995). *Scenario-based design: Envisioning work and technology in system development*. New York, NY: John Wiley & Sons.
- Carroll, J. M., Ed. (1998). *Minimalism beyond the Nurnberg funnel*. Cambridge, MA: MIT Press.
- Carroll, J. M. (2000). *Making use: Scenario-based design of human-computer interactions*. Cambridge, MA: MIT Press.
- Carroll, J. M., Ed. (2014). *Innovative practices in teaching information sciences and technology: Experience reports and reflections*. New York, NY: Springer.
- Carroll, J. M. & Campbell, R. L. (1989). Artifacts as psychological theories: The case of human-computer interaction. *Behaviour and Information Technology*, 8(4), 247-256.
- Carroll, J. M. & Carrithers, C. (1984). Training wheels in a user interface. *Communications of the Association for Computing Machinery*, 27(8), 800-806.
- Carroll, J. M. & Mazur, S. A. (1986). LisaLearning. *IEEE Computer*, 19(11), 35-49.
- Carroll, J. M., Mack, R. L., Lewis, C. H., Grischowsky, N. L., & Robertson, S. R. (1985). Exploring exploring a word processor. *Human-Computer Interaction*, 1(3), 283-307.
- Carroll, J. M. & Rosson, M. B. (2005). A case library for teaching usability engineering: Design rationale, development, and classroom experience. *ACM Journal of Educational Resources in Computing*, 5(1), Article 3, 1-22.

- Carroll, J. M. & Rosson, M. B. (2006). Cases as minimalist information. *IEEE Transactions on Professional Communication*, 49(4), 297-310.
- Carroll, J. M., Rosson, M. B., Farooq, U., & Xiao, L. (2009). Beyond being aware. *Information and Organizations*, 19(3), 162-185.
- Carroll, J. M., Singer, J. A., Bellamy, R. K. E., & Alpert, S. R. (1990). A view matcher for learning Smalltalk. In *Proceedings of CHI90: Human Factors in Computing Systems* (pp. 431-437). New York, NY: ACM Press.
- Carroll, J. M., Smith-Kerker, P. S., Ford, J. R., & Mazur-Rimet, S. A. (1987). The minimal manual. *Human-Computer Interaction*, 3(2), 123-153.
- Catrabone, R. & Carroll, J. M. (1987). Learning a word processing system with guided exploration and training wheels. In *Proceedings of CHI+GI87 Human Factors in Computing Systems and Graphics Interface* (pp. 169-174). New York, NY: ACM Press.
- Gagne, R.L. & Briggs, L.J. (1979). *Principles of instructional design*. New York, NY: Holt, Rinehart and Winston.
- IBM Displaywriter. (n.d.). *IBM Archives*. Retrieved June 7, 2014 from http://www.ibm.com/ibm/history/exhibits/pc/pc_8.html
- IBM Displaywriter System Operator Training (n.d.). *Bitsavers' PDF Document Archive*. Retrieved June 16, 2014 from http://bitsavers.trailing-edge.com/pdf/ibm/6580_Displaywriter/Displaywriter_Operator_Training_Book_1.pdf
- Mack, R. L., Lewis, C. H., & Carroll, J. M. (1983). Learning to use office systems: Problems and prospects. *ACM Transactions on Office Information Systems*, 1, 254-271.
- Rosson, M. B. & Carroll, J. M. (2002). *Usability engineering: Scenario-based development of human-computer interaction*. San Francisco, CA: Morgan-Kaufmann.
- Rosson, M. B., Carroll, J. M., & Bellamy, R. K. E. (1990). Smalltalk scaffolding: A minimalist curriculum. In *Proceedings of CHI90: Human Factors in Computing Systems* (pp. 423-437). New York, NY: ACM Press.
- We Look 4 Things. (n.d.). Retrieved June 7, 2014 from http://www.welook4things.com/sub_displaywriter_imgs.html
- Winner, L. (1980). Do artifacts have politics?. *Daedalus*, 109(1), 121-136.