

Application of Theory: Minimalism and User Centered Design

Mary Lou Mazzara

In the discipline of software and information development, minimalist design is not just doing with less (less features, words, widgets). It is selectively choosing what to include or eliminate with the purpose of making it easier for the user to quickly learn about a product in a natural and painless way and to start using it to do real work. User centered design fits well with minimalist theory because it incorporates user feedback throughout the development cycle. It is the best way to find out what customers actually do with your product and learn firsthand how you can help them with their goals. My team applied both these theories to our task of designing and building a set of samples for a Web development product. This paper shares our struggles and successes.

ROUND ONE

It is a known fact, almost a mantra within the industry: software development is an iterative process. The standard model is to get user input, design, develop, test, ship, rinse, repeat. Variations for user centered design, as defined by many authorities in the human-computer interaction field, involve getting user input throughout the cycle and making adjustments based on this feedback. (Fleming, 1998, Redish, 1998, Coe, 1996, Barnum and Carliner, 1993, et alia) But, regardless of which model you use, at the end of each cycle there is always the opportunity to take what you have learned and apply it to improve your product in the next iteration. We were at this point in my development group. We had delivered the first version of our product and were grinding through the 'design-develop-test' portion of the second version.

My role in this group was to lead the team that develops the external materials that we ship with the product - materials that show, guide, teach, and explain how to use the product to do real work, quickly and efficiently. Because our product was an integrated environment for Web site development, our samples included Web pages, Web applications, and complete Web sites. At this point in the cycle, we were in the middle of Round One. We delivered the product samples for the first release and were now assessing feedback from users and devising ways to improve them.

From the feedback we received¹, our customers were satisfied with the actual samples themselves: they liked the way they demonstrated our programming model and product features; they agreed they provided helpful templates, ideas, designs, and code that they could copy and reuse. That is, they were satisfied with them once they found them, figured out which ones were meaningful to their work, and ran them! It seems this was not as easy as we expected it to be. Despite the fact that we consulted our customers at every step in the cycle, despite all the thought and effort we put into our design, our customers still struggled with these initial tasks. This paper examines what the product samples expected to accomplish, how they succeeded, where they failed, and how my team followed respected theories of minimalism and user centered design to improve them.

Applying the principles

Our intention for the product was to provide *minimalist* information, as defined by John Carroll in *The Nurnberg Funnel* and revisited by him and Hans van der Meij. (Carroll and van der Meij, 1998) Our Web site development product was complex; to document every possible feature and function would have been overwhelming to the customers and prohibitive to write and maintain in our short delivery cycles. The main source of product documentation was a task-oriented online Users Guide. It had links to both reference and problem-solving information whose focus was to accommodate the customers' "need for knowledge with which to plan and evaluate action." (Carroll, 1998). The very basis of minimalist design is to support the user tasks and goals, not document everything there is to know about the product. (Redish, 1999) The main goal of the product samples was to augment the Users Guide and provide customers with an immediate opportunity to act. The samples offered an entire set of Web pages, from the simplest to the fairly complex, that customers could use as they explored, tested, and tried the product's functions - at their own pace and in their own way. The product we worked on was a suite of tools for creating dynamic interactive Web sites that incorporate server-side Java programs, specifically to run on the companion application server. While Web site development is not a domain with high-risk environments like aviation or medicine (Eiler, 1997), each user feels their work and

¹The feedback came from various sources: product integration teams at IBM, Beta and Early-Ship programs, usability testing, customer newsgroups on external web sites, and technical support personnel.

their time is important. The job market is competitive. We know no one wants to take too long to get something done, especially if the boss is looking or promotion time is rolling around. It is with this respect for our customers' domain, that we began the project.

Where we began

We began with an audience analysis, in part provided by our marketing department. Our suite of Web site development tools was not for someone just starting out with HTML. The audience for this product was expected to have the basics of Web design and development under its belt. The typical novice user was new to the product, not new to the Web and Web concepts. Our current and potential customers already had large Web sites, had made some use of programming techniques such as CGI and Perl, and were now ready to really bring their business to the Web in a big way.

One difference with our suite of tools and others on the market was our Java servlet and JSP programming model. This was the learning curve that our 'newbies' had to master. They had to learn what it was, how it could benefit them, and how to implement it. In *Designing the User Interface*, Ben Shneiderman cites John Carroll's work and supports the use of examples (in our case, interactive samples) as a way to help users understand a predictive model of the system. He found the users often mimic the examples during their first tries using the software and claims it "can improve performance dramatically by giving users access to fundamental structures." (Shneiderman, 1998) We felt that a variety of samples that show our programming model by example and allow users to experiment and 'try it themselves' would ease our users along that learning curve.

With all this in mind, our team expected that, in principle and in concept, the product samples and their accompanying documentation would succeed at supporting the four main principles of minimalist instruction for computer software (Carroll, 1998):

1. *Choose an action-oriented approach.* The samples were a fairly extensive piece of work. They included HTML pages, dynamic JavaServer Pages, Java code that runs on the server, JavaScript that runs on the client, cascading style sheets, and XML files. The samples themselves were presented in a three-frame format that encouraged exploration. It had a navigation area on the left side and context area on the right. Users could browse the menu in the navigation area, and then point and click to find out a little more about each sample. If their interest

was piqued, they could run it, test it, and look at its source code - all opportunities to act.

2. *Anchor the tool in the task domain.* From our usability studies and surveys we knew that the product users wanted to learn about our programming model. They wanted to learn how to convert their CGI programs to Java servlets and they wanted to see how this new model fit with their current Web site. The samples showed some of the most common (and simplest) CGI programs implemented as Java servlets. They offered a variety of techniques that users could apply. They included source code for the Java classes and the JSP files.
3. *Support error recognition and recovery.* The product samples were shipped as a complete set of archived source files (similar to a zip file). If customers experimented with them and messed them up, they could always start over from the archived files. The samples came with their own instructions about configuring and loading the database. The accompanying Users Guide had a section on problem solving that was always available while the users were working with the sample code.
4. *Support reading to do, study and locate.* All samples ran in separate browser windows, enabling new users to maintain focus and easily return to explore the rest of the sample offerings. The introductory information with each sample included links to further reading where appropriate. It also included suggestion for ways that the customers could adapt the samples for their own Web sites.

We tried to avoid some of the practical problems with minimalism as offered by Steven Draper. (Draper, 1998) We did not depend on the product's interface to be so intuitive that documentation was not needed. Nor did we do the 'brain dump' and just tell everything there was to know about the product. We also did not organize the information according to the system model and document the all features and functions. For instance, instead of the pseudo-task of 'How to use the wizards', we described a true user task such as "How to get database information onto your Web pages" which incidentally was done with a wizard.

The results

Our various sources of customer feedback showed us that our well-intentioned ideas did not always succeed.

Finding the samples. The samples were not readily available when you installed the product. In fact, it was often not apparent that there even *were* samples with the product. You had to read somewhere in the accompanying material that samples were included as archived source files in one of the product directories. And once you knew about them, you had to open (unarchive), configure, and publish them before you could really do anything with them. (Unfortunately, there were true technical impediments to other forms of delivery and the possible solutions to that problem are beyond the scope of this paper.)

The samples, however, were mentioned in just about every piece of written information that accompanied the product. The Users Guide pointed out the samples in its introduction to the product, and it had a high-level task in its table of contents called “How to run the samples”. The Getting Started book said it was a good place to begin and pointed the customers to this topic in the Users Guide, and the lowly ReadMe (do people ever read these?) did the same in its plain-text way.

What to do first. This was the next stumbling block on the users’ learning curve. When the users found the samples, (typically from the introductory material in the Users Guide), followed the instructions to open the archived files, published them, and then pointed their browser to the host where they were running...they didn’t know what to do or where to start. When they opened the samples in a browser, the ‘Introduction’ was already displayed. After the ‘Introduction’, all the samples were listed in the navigation area in alphabetical order. If a user followed this from top to bottom, it presented a confusing sequence. The first item was ‘Banner’, a one-of-a-kind sample for the optional applet building tool. It didn’t show the programming model but instead covered advanced bean and applet concepts. It really was not a good place to start. There were other simpler samples in this list that could provide building blocks to learning. ‘Database setup’ was the next alphabetically and it actually was a reasonable place to start; it provided necessary configuration instructions to set up the relational database that many of the other samples need. Unfortunately, once users started randomly poking around, they often mistook ‘Database setup’ for another sample and skipped over it.

Running the samples. We found many new users made errors. When inexperienced product users tried to run the samples, many things went wrong. Publishing the samples required learning new techniques and ideas. The Users Guide had a section devoted to both the task

(‘how to’) and concept information behind the publishing scenario but the companion application server was a complicated product. It often did not work as expected, especially for new users, for a variety of reasons. The database might not be doing what it is supposed to either. Maybe it wasn’t set up right or maybe it just wasn’t started. If the samples just didn’t run - or ran and then stopped, customers had to go the Users Guide Problem Determination section and rummage around for clues. Unfortunately, it did not have a section devoted to the samples. The product service and support team received many calls from new users who struggled to get the samples running.

ROUND TWO

Our team came up with several improvements to the samples for the next release of the product that we felt better applied the theory of minimalism.

Seeing before trying. The problem of finding the samples was the most difficult to address. We did not think we could make it more obvious in our own documentation but we did get marketing to support an external Web site that customers could go to where we knew the samples would be running 24/7. This gave us a guaranteed direct link we could include in all our publications and advertising. Customers would no longer have to open an archive, configure, and then publish to see what these samples would look like. They could go to the site, take a look around, and then when they were ready, try it themselves - in the privacy of their own shop - after they had some familiarity with the scenario.

Meaningful categories. Minimalism emphasizes getting learners started quickly. Because the tasks of complex domains tend to incorporate many subtasks...(it) becomes more a matter of sustaining interest and activity than of quickly dispatching a task.” (Carroll, 1998) Barbara Mirel proposes that we can evoke and sustain user’s motivations by starting users immediately on work that is meaningful to them, meaning work that is tied to their pragmatic considerations. (Mirel, 1998) We thought we could more easily sustain user motivation and interest by grouping the samples according to their level of difficulty (Basic, Intermediate, Advanced) rather than according to the first letter of their name. Alphabetical listing was out, categorizing by difficulty was in. Users could then start at a point they felt was appropriate to their own level of experience and the work they were trying to accomplish, and when ready, continue with an incremental approach. Beginners could find the simplest samples and the more advanced users could move up to the complex ones. The new

introductory text explained the meaning of the terms: Basic, Intermediate, and Advanced. It placed the database configuration in the context of a 'starting task' and one necessary for the Intermediate level samples. We thought this new grouping would encourage exploration by indicating, but not enforcing, a sequential order.

Better support for errors. The third improvement we decided to make was to provide better support for error recovery. If exploring the product 'by way of the samples', is an approach that we wanted to encourage, specific information for common sample problems and their solutions was needed. From customers' experiences as reported to the service and support department, we had a good idea of the common pitfalls. We decided to write special Problem Determination information and include it with the samples as well as with the Users Guide. This would be redundant information in some aspects, but because we do not know where the customers will install the product or where they will be publishing and running their copy of the samples, we could not link from the samples to the installed copy of the Users Guide. We thought it was important enough to give immediate support for error recovery to justify the duplication.

Layered information. Some team members still felt that the sample names did not indicate enough about what the sample is demonstrating. For instance, the Quote sample demonstrates how to read a flat file and call a servlet directly from an HTML page, so that it is called each time the page is refreshed. The word 'Quote' does not indicate any of this. Each sample had an introductory page that explained what it does and how you can use it. We did not want to make these pages overly long and wordy and considered adding simple explanatory text in a window that pops-up when the mouse is held over the name of the sample in the introductory window. We decided to test it with internal product users if we could not get it into the next round of customer usability tests being conducted by our Human Factors team.

SUMMARY

The theory of minimalism is often simplified with the axiom 'less is more'. And, while this is commonly true in computer documentation, institutional instruction, and user interface design (Cooper, 1998), it is not merely because there is summarily less (words, steps, widgets) but because there is selectively less. What we choose to eliminate, keep, rewrite, or redesign, is done with a keen eye to making it easier for our users to quickly learn about a product in a natural and painless way and to

actually start using it to do real work. In *Reconstructing Minimalism*, John Carroll attests that "Design iteration was always the touchstone of minimalism". My team thinks that we are on the right track, but we know our work is not done. We will follow the model of good user centered design - implement as many of these improvements as time and budget allow, test them on cooperative users, fine tune them, and ship the next version of our product. And then...as in 'test, ship, rinse, repeat'... we will revisit it again for ROUND THREE.

REFERENCES

- (1) Barnum, Carol M. and Saul Carliner. *Techniques for Technical Communicators*, pp. 305-336. Boston, MA: Allyn and Bacon, 1993.
- (2) Carroll, John M. *The Nurnberg Funnel*. Cambridge, MA: MIT Press, 1990.
- (2) Carroll, John M. "Reconstructing Minimalism." In *Minimalism Beyond the Nurnberg Funnel*, ed. John M Carroll, pp.1-17. Cambridge, MA: MIT Press, 1998.
- (3) Carroll, J.M., and H. van der Meij. "Principles and Heuristics for Designing Minimalist Instruction." In *Minimalism Beyond the Nurnberg Funnel*, ed. John. H. Carroll, pp.20-53. Cambridge, MA: MIT Press, 1998.
- (4) Coe, Marlana. *Human Factors for Technical Communicators*, pp. 177-206. New York, NY: John Wiley & Sons, Inc., 1996.
- (5) Cooper, Alan. *The Inmates are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity*. Indianapolis, IN: SAMS, 1999.
- (6) Draper, Stephen W. "Practical Problems and Proposed Solutions in Designing Action-Centered Documentation." In *Minimalism Beyond the Nurnberg Funnel*, ed. John M. Carroll, pp. 349-374. Cambridge, MA: MIT Press, 1998.
- 7) Eiler, Mary Ann. "Minimalism and Documentation Downsizing: The Issues and the Debate." In *the Newsletter of the Chicago Chapter of the Society of Technical Communication*. 39.4 [WWW document] <http://english.ttu.edu/kairos/3.1/reviews/eiler/minimal.html>

(8) Hackos, JoAnn T. and Janice C. Redish. *User and Task Analysis for Interface Design*. New York, NY: John Wiley & Sons, Inc., 1998.

(9) Hargis, Gretchen, *Developing Quality Technical Information: A Handbook for Writers and Editors*, pp. 49-70. Upper Saddle River, NJ: Prentice Hall PTR, 1998.

(10) Fleming, Jennifer. *Web Navigation: Designing the User Experience*, pp. 29-44. Cambridge, MA: O'Reilly & Associates, 1998.

(11) Mirel, Barbara. "Minimalism for Complex Tasks." *In Minimalism Beyond the Nurnberg Funnel*, ed. John M. Carroll, pp. 179-218. Cambridge, MA: MIT Press, 1998.

(12) Redish, Janice. "Minimalism in Technical Communication: Some Issues to Consider." *In Minimalism Beyond the Nurnberg Funnel*, ed. John M. Carroll, pp. 219-246. Cambridge, MA: MIT Press, 1998.

(13) Shneiderman, Ben. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Third Edition. Reading, MA: Addison-Wesley, 1998.

Mary Lou Mazzara
Managing Editor, WebSphere Developer Domain
<http://www.ibm.com/websphere/developer>
IBM
4400 Silicon Drive
RTP, NC 27709
(919) 254-7203

Mary Lou Mazzara is an Advisory Software Engineer at IBM in Research Triangle Park, NC and recently received her Masters of Science degree from Rensselaer Polytechnic Institute in Troy, NY. (Ask her about the commute!)