

Role of User Studies in Design of OpenDoc®

Elizabeth Dykstra-Erickson
Apple Computer, Inc.
Human Interface Design Center
1 Infinite Loop
Cupertino, CA 95014 USA
+1 408 974 6462
eade@apple.com

Dave Curbow
Apple Computer, Inc.
Human Interface Design Center
1 Infinite Loop
Cupertino, CA 95014 USA
+1 408 974 4823
curbow@apple.com

ABSTRACT

This paper reviews a number of design decisions that have been made in the development of OpenDoc, CI Lab's component software technology platform, as a result of ten user tests conducted over the life of the project. We take as a specific example the history of the design decisions surrounding the activation/selection model of OpenDoc, from its conceptual beginning to its eventual release as a component of end-user products.

Keywords

OpenDoc, component software, objects, object technology, conceptual model, usability, learnability, design

INTRODUCTION

OpenDoc is a component software system developed by a consortium of companies including at various times Apple Computer, Inc., IBM, WordPerfect, Novell, SunSoft, XSoft, and Taligent. The system is a development platform that was designed to be deployed on multiple operating systems with a consistent user experience, insofar as possible.

As a member of a new paradigm in desktop computing, the user interface for OpenDoc presented several opportunities for design decisions that are, for the most part, unprecedented in traditional application development (see the companion paper, "Designing the OpenDoc Human Interface" in this volume for more detail on the OpenDoc view of designing). This paper will review design decisions in light of the user testing conducted over the life of the project. As such, this paper serves as a meta-analysis of design decisions.

THE OPENDOC PARADIGM

OpenDoc is an object-oriented development platform, initially intended to support a document-centric domain. It allows a user to create and edit "compound documents" — that is, a document serves as a container of components, each of which has its own functionality and user interface. These documents can contain virtually any kind of content, e.g., text, graphics, tables, spreadsheets, sound, video, virtual reality, and network and internet connections, to name a few, all of which may be edited in place (thus

requiring no importing/exporting of data, or application switching to handle multiple data types). OpenDoc runs on different platforms and interacts with other compound document architectures such as Microsoft's OLE 2.0.

The document-centric OpenDoc domain concentrates on the content the user is manipulating, and makes the launching and quitting of applications transparent. In fact, the closest corollary to the application is the editor (or viewer). However, unlike applications, once the user installs editors, he no longer interacts with them directly. Thus, OpenDoc's document-centric metaphor contrasts sharply with the application-centric world of today, where users must specifically launch and quit applications. In addition, the OpenDoc constructionist model allows users to edit content in-place, in contrast to today's applications where they must work with content in its native application and export and import it into the final document.

Three user experience factors differentiate the OpenDoc model from the application model: 1) edit-in-place, 2) inside-out activation, and 3) modality introduced by selection (working on content as a participant in a container document, e.g., moving an object around) vs. activation (working within a component, e.g., editing text or graphics).

Much of the OpenDoc user experience is supplied by OpenDoc APIs. However, there is still much that a developer must provide in their editor. To guide developers, the OpenDoc Human Interface team established guidelines in the OpenDoc Programmer's Guide and other publications [1, 2].

USER TESTING OVER TIME

Since the beginning of this project, we have conducted ten user tests of OpenDoc. Early tests used paper and pencil prototypes; as we accomplished more design and discovered more issues, we used Macromedia® Director® and SuperCard prototypes. We characterize the first nine tests as pretests; largely *qualitative* in nature, we used the data gathered from these tests *inductively* to guide design. The last two tests used OpenDoc 1.0 Developer Release 3 (DR3) and OpenDoc 1.1 Developer Release 6 (DR6). These two tests were conducted as *quantitative* tests. Their results were used *deductively* to gauge the success of the interface and determine specific problems that could be fixed in future releases. It is important to note that OpenDoc has only recently been included in the latest Macintosh system release. This means that it is still primarily a developer platform, and few end-user components are commercially available today. Testing a developer platform in the

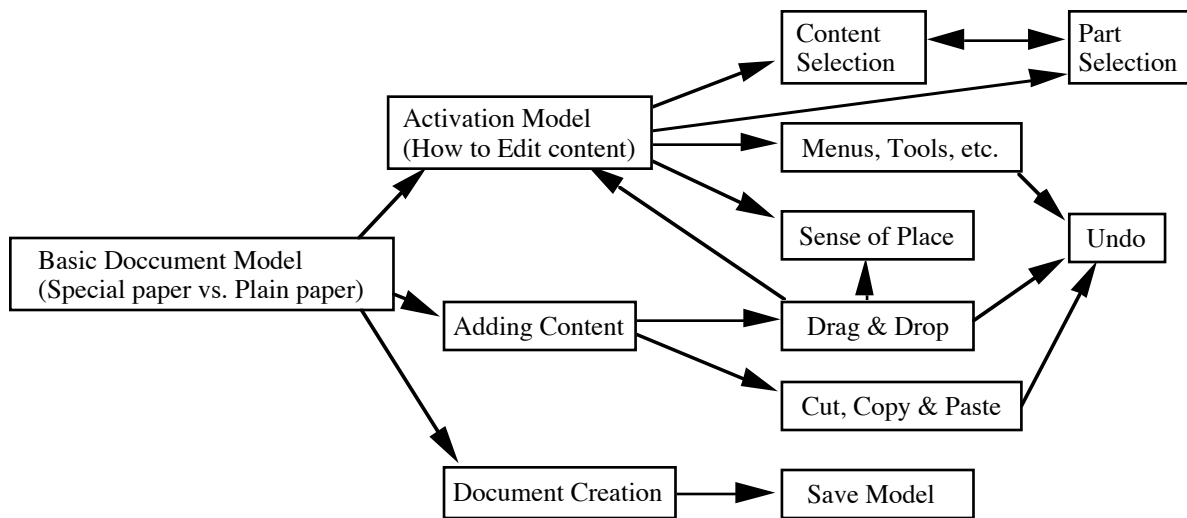


Figure 1. Problem Interrelationships

absence of end user software is tricky. The last two of our tests used Apple internal parts and beta quality commercial parts; we are fairly confident that we were able to provide, in these tests, a close approximation of an OpenDoc user environment. Further complicating testing is the fact that while OpenDoc is a new computing paradigm, we expect that for some time users will continue to operate in the application-centric world. Testing user capabilities to switch contexts between applications, OpenDoc-enabled container applications, and true OpenDoc components remains to be done.

DESIGN ISSUES

Developing OpenDoc presented several interesting design issues that were difficult to prototype and test in isolation. Because OpenDoc is a development platform and not an application, we were only able to test portions of the user experience without having had to create both OpenDoc and third-party components that used it. This was clearly a chicken-and-egg problem!

Usually, we can test specific elements of a user interface because test participants can balance the evaluative experience with their own experiences working with similar applications. In the case of OpenDoc, few subjects had such experience. While there were some systems that provided similar functionality, notably Xerox® Star, there were relatively few users of these systems.

Additionally, most of the design issues were highly interrelated (see Figure 1), requiring a systemic view of both the problem and the solutions. As a result, we could rarely design a user study that focused on one problem alone. Lessons about one problem often affected other related issues.

Among the design problems faced by the OpenDoc human interface team were how to present the document-centered concept of compound documents; how best to create new documents; how to clarify the potential "sense of place" problem when working with one component surrounded by other components; whether to merge or switch menus; how

drag and drop should work; how to implement undo; and how to activate and select component parts.

Over the last four years, we have conducted 10 user studies to help us produce answers to these and other design problems. We describe some of these design issues to help establish the complexity of designing OpenDoc. We then trace the design history of one of the most important design decisions, the activation/selection model, in detail.

Document-centric and the document model

OpenDoc was designed to be document-centric, rather than application-centric. This meant that users shouldn't have to manipulate applications to deal with content. This decision allowed us to move the editors (the OpenDoc analog to applications) into a "hidden" place inside the System Folder, and so hide some complexity from the user.

Although OpenDoc is initially intended to support a document-centered domain, other non-document models are possible (for example, Web browsers are based on navigating rather than on creating documents). We want to be clear that document-centric does not mean just documents that "look like paper". Our definition of "document" is extremely broad, encompassing folders, files, control panels, etc. We were looking for a single coherent model of computing and decided that a broad definition of "document" was the most familiar and extensible metaphor.

Stationery and Creating new documents

There were several interacting factors why we chose stationery as the way to create new content. First, and most importantly, we designed OpenDoc to be document-centric. Therefore, the generally accepted model for creating new content, by double-clicking an application icon (or using the New command from within the application) had to be changed.

We decided to use an existing, if under-used feature, in MacOS. Stationery, and subsequently templates on OS/2, provided the functionality needed. However, our testing revealed that most users don't recognize stationery much

less know how to use it. If we adopted this solution, we would have to educate users about stationery.

The second factor that interacts with the use of stationery is OpenDoc's persistence model. We knew that users were "scared" of losing their work with today's model. That is, even though the user has created a new document and made changes to it, all of their work can disappear if they close the document window without saving. It is easy to see this by simply watching a few users — they tend to Save their documents every few minutes. To combat this fear, OpenDoc actually creates a document (and associated icon) immediately after the user uses New or double clicks on stationery.

The use of stationery helps to support the persistence model by using existing stationery to create real instances of a document. Stationery may contain initial data, for example a letterhead, so that users gain additional value by using stationery instead of a new "blank" document. In OpenDoc, unlike most applications, there is not a one-to-one relationship between a part editor and the documents it creates; stationery can be built using many part editors and content types. For example, stationery can be created that gives the user a form containing a business logo (PICT), text, and a button to submit the form.

The third factor that interacts with the use of stationery is the drag verb default. In trying to reinforce the document-centric model, we originally decided that the default operation for a drag of some content would move it, rather than copy it. While we were worried about people losing their originals accidentally, we didn't want to automatically make a copy that was unwanted. Nor did we want to make drag behavior in OpenDoc inconsistent with general Finder behavior (Move within volumes, Copy across volumes). Thus, given that the default drag verb would be Move, we needed a way to bring new content into a document that didn't simply move it from elsewhere. The standard stationery model provided a clear way to allow users to create new content via double-clicking the Stationery icon, or by using drag and drop, to drag a sheet of stationery off the icon — virtually "tearing off" a new copy of the content as the user drags a stationery item into another document.

Sense of place

Sense of place is important in a number of ways; it relates to identifying parts (what is the user working in at the moment?), to visual feedback for making a selection (e.g., how large a selection can the user make without going outside the boundaries of the part?) and to drag and drop (where are the boundaries of the drop location?). Menu switching as well as tool bar and tool palette content also help reinforce what the user can do and what part they are currently working in.

Drag verb

When content is dragged from one location to another, there is an implicit verb. For example, if the user drags a file to the trash can, the verb is delete, even though under the covers it's more complex than that. In today's Finder, there is some ambiguity in the drag verb — sometimes a file dragged to a folder is copied, other times it is moved. The

verb used depends on whether the source file is on a different volume than the destination. We were concerned about this ambiguity, and its possible effects on users.

Thus, we considered alternative designs. The most promising design was based on the way the Xerox Star handled drags: a Move was always a Move, a Copy always a Copy. Specifically, on the Star the user makes a selection, presses a dedicated key for Move or a different key for Copy, and the selection attaches to the cursor; then a click at a destination drops the selection there [3]. This reduced the mental load for the user and gave some visual feedback for which operation was being used.

However, we could not use this model on the Macintosh for two reasons. First, our existing models of keyboards didn't have dedicated keys for Move and Copy, and it was not feasible to ask users to buy a new keyboard. Second, the modal nature of the Star drag operation was at odds with the model long used in the Macintosh.

Thus, we decided to make the unmodified drag a Move operation. We allowed the user to be explicit about the verb and force the drag to be a Copy operation by holding down a modifier key (e.g., option). However, during extensive user studies on this model, we repeatedly saw users accidentally move content from one document to another and not realize the loss until some time later. Their recovery mechanism, to close the document(s) without saving, often meant that they lost other work. We concluded that this model was unworkable.

We decided that the model used by the Finder, which originally seemed incorrect, was better than our model. By adopting the same model, with slightly better feedback, we created a model that works well and meshes better with the one used by the Finder.

By coincidence, a chance dinner conversation with a colleague at Sun revealed that his team had just gone through this same design and test process and had arrived at the same conclusions.

Undo

When we began to design Undo, we had a lot of evidence that users accidentally wipe out the undo buffer, for example by hitting Command-C instead of Command-Z, and thus lose their ability to Undo. We decided to fix this problem, and give users a better recovery mechanism than the single level of Undo that exists today.

At the time we initially designed Undo, the default operation for drag and drop was Move. Because of this, we wanted users to be able to recover from an apparent loss of data as they moved content across documents. Therefore, the Undo model works across documents (specifically, the Undo stack captures events within a single process). For example, if a user drags some content from one document to another and then invokes Undo, the content will be removed from the destination document and returned to the source document.

We also wanted design symmetry: one operation to Move, and one operation to Undo the Move. However, now that the drag model now defaults to Copy between documents,

we should also have reexamined the Undo model, perhaps to establish an Undo stack per document. Consulting our interrelationship diagram should make this apparent. We had sufficient evidence to conclude that multiple-level Undo is attractive and useful; it is the boundaries of the stack that are in question.

Selection and Activation model

Editing is based on selection (an insertion point is a null selection), and selection is based on whatever contains the selected object. Selection of embedded content is based on activation: a part must be active to be edited; it is in that activated state that the part can put up its own menus and palettes. Therefore, active and selected states are fundamental distinctions that need to be easy to see and remember.

Notice in Figure 1 that Selection / Activation has the most related issues. As a result, nearly every user study addressed some issue related to the Selection / Activation model. Table 1 reviews the tests we conducted, the media we used, and the various foci of the tests. For example, the first 3 studies directly addressed it. The 5th study was focused on Drag & Drop, yet it also gave us information on the Selection / Activation model. Selection / Activation has been bolded in the table to show that it was, indeed, a part of every test we conducted.

TO ACTIVATE OR SELECT, THAT IS THE QUESTION

The remainder of this paper presents a meta-analysis of ten separate user studies' findings as they relate to the topic of activation and selection. The user studies are summarized in Table 1 below.

The design problem

OpenDoc components have three states: activated or selected, or neither. The active state allows the component's content to be selected and manipulated, while the selected state treats the component as content in its containing part. This is a necessary distinction in OpenDoc, and has occupied a great deal of our time and attention in ensuring that the user interface makes these states visually apparent and easy to work with. This is also known as the "use/mention problem" [4].

Each of the ten user studies we've conducted since the inception of the OpenDoc project has in some way addressed the user's response to the visual feedback and the behavior of activation and selection (see Table 1). Conducted with varying media, the tests have had various results, and have influenced design decisions as reported below.

1st and 2nd tests, July 1992

This first user study was conducted with a paper and pencil prototype and 12 users. Users were asked to explain how they would interact with sketches of different kinds of media; specifically, they were asked to describe the operations they would use to make specific changes to a document.

We followed this up with a second paper and pencil study with 6 experienced Macintosh users. For each task, the participant wrote or sketched the steps she would take to complete that task. Before and After sample documents were mounted on either side of a large sheet of blank paper, and the participant used markers to show the actions to modify the document and thereby complete the task.

Both tests provided the same results: users were split on how many mouse clicks they "expected" were necessary before they could edit deeply embedded content. But, universally they said they "wanted" to use only one click. This would require that a single click both activates a part *and* sets the insertion point (or selection) inside it, so the user can immediately begin to edit content.

Observations

Users who are less experienced at working with multiple data types in a single document expect simple, straightforward, Macintosh-consistent application behavior.

More experienced users were aware of changing modes between tasks, used a tools palette, and generally reported more steps to complete each task than the less experienced users. That is to say, less experienced users tended to imagine tasks as less complex than they actually were.

Design Implications

Based on our observations and user expectations, we tentatively decided to use an inside-out model. We realized that we didn't have a lot of data to support this decision, so we were willing to reverse this decision should the results of future tests indicate an alternative. The inside-out model refers to clicking once on content you wish to edit, no matter how deeply embedded it is, rather than the outside-in model, wherein the user traverses via clicks the hierarchy of containers until the desired level is reached.

3rd Test, November '92

This test was conducted with a paper and pencil prototype and 6 users. With tentative support for the inside-out model of activation, we wanted to determine what users expected to have to do to be able to interact with parts. Should a single click select an object, or allow editing of its contents? Is it appropriate for the model to be different, based on the type of content being worked on?

Observations

Users tended to use different strategies for moving and copying different types of content. For example, they used drag and drop for graphics and spreadsheets, but used cut and paste for text. Half of the users did not differentiate between selecting an object in order to edit it, and selecting an object in order to move it. If the task called for them to move an object, they would simply click to select it, then drag it. If the task called for them to edit the object, they would simply click on the part they wished to edit. They were not aware of modes until they were taught the distinction. The other half of the users DID recognize the need to differentiate between selecting to edit and selecting to move.

Test #	Date	Features Tested	Target System	# Users
1	92.07	Selection & Activation Edit in place	Paper & Pencil	12
2	92.07	Selection & Activation Edit in place	Paper & Pencil	6
3	92.11	Selection & Activation Document-centric model Create/edit/move embedded content Create/edit intrinsic content Link content	Paper & Pencil	6
4	93.01	Test overall user model Test document-centered paradigm Create embedded content Edit intrinsic content Move content within/between document(s) Selecting content Saving changes Link content	Paper & Pencil Director Demo	8
5	93.03	Drag Verb: drag/move vs. drag/copy	Prototype	33
6	93.06	Frame Appearance Move & Resize Frames Active vs. Selected States	Prototype	11
7	93.10	Locating Get Info/Preferences Properties vs. Preferences Selection & Activation Finder Get Info vs. OpenDoc Get Info Icon in Titlebar	Interface Theatre	12
8	93.12	Frame Appearance Insert, Open commands Locating Properties/ Preferences Icon in Titlebar Selection & Activation	SuperCard & Director Prototypes	3
9	95.09	Selection & Activation Resize parts Create & Edit intrinsic content Create & Edit embedded content Delete parts Use part controls	OpenDoc 1.0 (DR3)	15
10	96.11	Learnability Document/content creation Create/Edit intrinsic and embedded content Selection & Activation Part & Document Info	OpenDoc 1.1 (DR6)	10

Table 1. User Study History

Two of them accomplished this by using a single-click to select an object to move it, and a double-click to edit that object. Another user wanted a “move” tool (with up, down, left, and right arrows) which would be used to move parts around. Users seemed to realize they must switch modes in order to be consistent. For example, after editing a graphic users almost always remembered that they were using a line tool and needed other “tools” to edit text. For those that had forgotten, a little prompting reminded them that they were using a line tool and needed to switch to a text tool.

Design Implications

This test did not provide conclusive results suggesting regarding the preferability of either the inside-out or outside-in model. It did suggest that we needed to provide clear feedback regarding which part was currently active; this resulted in continuing work on improving an active frame border design.

4th Test, January '93

This study was conducted with 8 users and a combination of explanation and a software prototype. First, we showed users how compound documents are created in today’s application architecture. The prototype covered drag and drop, and edit-in-place. We asked the users questions as we went through a demo, such as: what do you expect and how would you like to accomplish certain tasks? Do you understand what you’ve seen? After explaining the approach a second time, users were asked again whether the approach made sense and how we might do things differently.

Observations

At this time, OpenDoc had handles on the active frame border. Users correctly expected to use these handles to resize the frame, but they also expected to be able to move the frame by dragging on the handles. They were confused by the fact that the handles were sometimes visible, and other times not visible — essentially, users were not recognizing the active vs. selected states.

While editing the content of a part, several users indicated that in order to switch to another part, they would need to “close out” of the current part first. Users seemed to need a sense of closure on something before moving elsewhere. When pushed to articulate how they would prefer to be able to switch, the users indicated that they would like to simply click elsewhere.

Design Implications

This test seemed to confirm our choice of the inside-out activation model.

5th Test, March '93

This test was conducted with 33 participants (11 novice users, 9 intermediate users, and 13 advanced users) and a code prototype of OpenDoc.

The primary focus of this test was drag and drop, with the emphasis placed on discovering the appropriate drag verb default. While Move was expected to be the default, using a modified drag (drag + Option) to Copy, the test results showed that the default Move behavior caused participants to lose data. Conversely, Copy as a default drag verb also introduced difficulties, but those were mainly the result of

poor dragging skills. The results from this test concluded that “having a few unwanted files or taking an extra step to delete the source is certainly less severe a problem than lost or unrecoverable data.” Thus, the default drag verb in the current OpenDoc release is Copy whenever data might be lost. For example, when content is dragged within a document, it will be moved, but when it is dragged between documents it will be copied. At present, a drag verb pop-up is being considered to assist users in making an explicit choice before dropping.

This test also uncovered naming difficulties for parts dragged to the desktop, and difficulty dragging frames in general. Participants were confused when they could not select content in inactive windows. This difficulty was most likely an artifact of the prototype design (see Figure 2), in which only the handles of the frame were displayed, without a visible frame border.

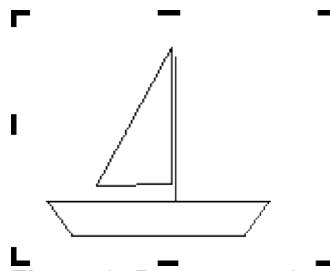


Figure 2. Prototype selected frame appearance

Design Implications

To improve the visual feedback, we changed the selected frame appearance from Figure 2 to Figure 3, using a dotted-line frame to connect the frame handles by which the part can be dragged, and adding a cursor change to cross-hairs when moving over the frame.



Figure 3. Selected frame appearance today

An interesting note in this test is that only four of the 33 participants used marquee select, and only seven knew how to shift-select—and three of those used incorrect key combinations for the multiple select keyboard shortcut. These findings reappeared in the most recent OpenDoc tests, where we found that users do not, in general, have alternative selection strategies to whichever method they normally use.

6th Test, June '93

This test was conducted with 11 participants (3 novices, 4 intermediates, 4 experienced) and a code prototype of OpenDoc.

The primary focus of this test was frame appearance preferences, and move/resize mechanisms, with an emphasis

on activation, selection, and manipulation of frames. We used the two following selected frame appearances (see Figure 4).

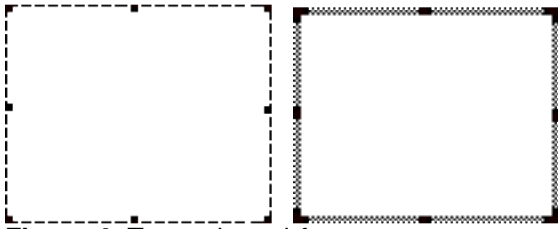


Figure 4. Two selected frame appearances

For this round of user testing, our prototype did not support embedded parts or manipulating intrinsic content within a part. For this reason, users were unlikely to develop an accurate frame behavior model. Therefore, activation and selection findings from this test can not be considered conclusive.

Key findings revealed that

- Participants showed no preference for either appearance.
- All participants wanted to move parts by clicking in, then dragging content.
- Participants had difficulty discovering that clicking on an active border allowed them to change from the active to the selected frame.
- Participants tended to use control points when control points were present, even when the function of the control point was not the same as the user's intent.
- Cursor feedback greatly assisted participants when manipulating parts.

Recommendations from this user test indicated we should use a heavier selected frame appearance (note the light and heavy appearances in Figure 4) to increase the visual difference between the active and selected frame borders. We also decided to allow users to move content from a part in the selected frame state. We provided a control on the active frame that, when clicked, switches from the active to selected frame. We also decided to take advantage of users' tendency towards using control points by providing them for unclear interactions. Finally, we provided positive cursor feedback wherever appropriate to more closely match user expectations to the defined function (see Figure 5). The open was chosen as a good means of notifying the user that the cursor is over a clickable border. The cursor changes to a closed hand when the user has successfully grabbed the frame border.

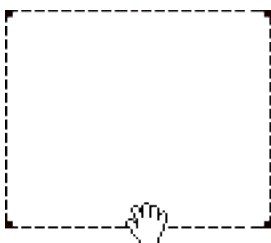


Figure 5. Cursor feedback for selected frame

Ultimately, it was not possible nor was it desirable to put affordances such as state-switchers on the border for

tooggling between the active and selected states. We revisited this issue several more times.

Design Implications

We decided to use the appearance shown in Figure 6 for the active frame border. The target size of the frame border was increased from 3 pixels to 5 pixels (2 opaque pixels on either side), and the design does not depend on a highlight color to be visible.



Figure 6. Active frame appearance today

7th Test, October '93

This test was conducted with 12 participants and a prototype of OpenDoc using a variant of the Interface Theater method [5]. The interlocutor ran a demo and asked for participants to tell him what to do and explain what they think *should* happen.

The primary focus of this test was part properties and part preferences, and frame appearance and behavior. Participants were also asked their opinions of a window design which included a draggable icon in the titlebar.

Observations

Predictably, users had some difficulty discriminating between part preferences and part properties. Because properties and preferences depend on whether a part is in an active or selected state, users may have difficulty viewing the desired information for a part.

It was at this point that the active and selected appearances became less of a debate and more guideline-oriented. Because part editors will create parts with many different content models, this test helped us realize that a single selected appearance would not work for all kinds of content. For example, in a text frame, selecting a graphic will reverse highlight it the same way the text in the parent frame would be selected. In a drawing frame, selecting embedded text by marquee dragging would place four handles (or more, depending on the way the drawing part sets handles) surrounding the text object. The active border previously specified had used the system-defined highlight color; we learned that this color may be too light to see.

Design Implications

This test confirmed for us that the border redesigned as a result of test 6 was an improvement. The active border was respecified to 3 pixels wide and painted in gray, rather than dotted. Today, the active frame border can be seen in either the dotted or solid version. However, we have since learned that a border that we expect users to click should be no less than 5 pixels wide. The selected border, as described above, is completely dependent upon the container. Selection of an embedded part may be inverted, appear with handles, or

appear with marquee selection, depending upon the container.

8th Test, December '93

This test was conducted with 3 participants using two Macromedia Director prototypes and a SuperCard prototype.

The primary focus of this test was frame appearance and behavior, menus, and again, the icon in the titlebar. Users were shown the Director prototype to view interactions and were asked how they expected to be able to accomplish tasks. They then worked hands-on with a SuperCard prototype to assemble and manipulate OpenDoc content. Finally, they watched a Director demo of the icon in the titlebar.

Observations

Despite confusion over frame states and menu items, users felt that it would be easier to create documents with different content types using OpenDoc than it is today using applications. We realized that users' perception that there is no difference between working on a part's content and manipulating the part as a whole (moving it or resizing it) is the source of the confusion over frame states. OpenDoc does not differentiate between the two states; because the availability of menus and commands depends on the frame state, the user must learn what each frame border means and how to make them appear. In this test, users could easily switch between states but would often arbitrarily change the frame state, indicating that they had not really learned the distinction. Clues to frame state such as menu switching, appearance/disappearance of toolbars and palettes, and frame border seemed to be too subtle.

We observed users having difficulty remembering to select objects before trying to work on them; this we attribute to a general forgetfulness about the very basic noun-verb interaction model in the MacOS: first pick something to do something to, then do something to it. We saw this behavior repeated in the later quantitative tests.

Design Implications

This was the last of the qualitative studies. Rather than point us in a specific direction, the findings from this study confirmed that we still had a difficult task ahead of us — educating users about the modality of OpenDoc, and reinforcing the standard MacOS noun-verb interaction that is critical to OpenDoc.

9th Test, September '95

After almost two years of design, redesign, and engineering, the first release of OpenDoc and Apple-internal parts made it possible to do user testing in a real OpenDoc environment. This within-subjects test was conducted with 15 participants using OpenDoc Developer Release 1.0, Apple Parts, and tutorial material in the form of QuickTime movies of screen activity using OpenDoc. The entire focus of this test was activation and selection. The test was designed to test for participants' ability to distinguish between the two states both visually and conceptually.

We set up this test to see if users could see and make use of the distinct selected vs. active states on their first exposure to OpenDoc. Test participants completed 9 tasks using a

drawing container part in which they could embed text, 2D graphics, bitmap graphics, QuickTime movies, and a ticking clock part. They were then given some training and asked to complete another set of 17 tasks.

Observations

The novice group success rate improved after training from an average of 71% for the first set of tasks to 86% for the second set of tasks; for experienced participants those rates improved from 88% successful completion to 96%. We saw a significant increase in the success rate over nine repeated measures tasks: the overall success rate goes from 80% in the first set of tasks to 91% ($t=-3.16$, $p<.05$) in the second set of tasks.

We considered these to be fairly high successful completion rates. Only some operations were particularly troublesome for participants: resize handles were difficult targets to hit, and selecting parts became visually confusing when there were multiple overlapping frames.

We tested whether users' mental models were correct by showing them the QuickTime movies of actual OpenDoc operations, stopping the movie periodically to ask the participant what was active or selected. We found little difference between the novice and experienced users' percentage of correct answers: novices averaged 66% correct answers, and experienced participants averaged 69% correct answers. We then looked to see if participants understood one of the concepts (active or selected) better than the other, and saw no difference between the novice or experienced users. When asked to describe the differences between the two states, experienced participants were better able than novice participants to describe what the borders meant, although both groups were able to work with them successfully. We conclude that even though users often couldn't describe what they were doing, they were quite competent at doing it. We further conclude that selecting content is a more important concept to understand than activation, and is really no different in importance than it is with applications today.

Design Implications

We determined in this test that activation is a by-product of the object/action model, and selection is still the primary concept to learn. The object/action, or noun-verb interaction, model is really no different in OpenDoc than it is in today's applications, yet users have difficulty remembering to select before operating. This finding is consistent with the findings from our very first user tests, where users were given simple documents and asked how they would like to work on it. Their answers caused us to decide on the inside-out, click once to activate model. While this conclusion did not present us with new design tasks, it did tell us that what we face is primarily a learnability, rather than a usability, challenge in presenting a new interaction paradigm to users. With a little training, users can use OpenDoc easily. This test did not, however, gauge how easy or difficult it would be for users to retain that learning.

10th Test, October '96

In preparation for an end-user release of OpenDoc and the availability of robust parts, we conducted this tenth user test to help us understand how easy or difficult it is for users to learn the OpenDoc paradigm, and how they feel about shifting to the new paradigm. Therefore, the primary objective of this user study was to evaluate the learnability of the OpenDoc paradigm and to identify learnability issues. A secondary objective was to evaluate the usability of the Apple parts used in the test.

This test was conducted with OpenDoc 1.1 DR6, a suite of parts which included Apple 3D Viewer, Apple Audio, Apple Button, Apple Draw, Apple Image Viewer, and Apple QuickTime Viewer, and a hands-on tutorial. Ten intermediate and experienced users worked through 39 tasks. We used no novice users for two reasons. First, we anticipated that intermediate and experienced users will have the most difficulty shifting to the OpenDoc paradigm since they will have to learn OpenDoc and *unlearn* application behaviors. Second, we needed to filter out difficulties with standard MacOS behaviors such as noun-verb interaction, drag and drop, marquee selection, and the use of Find File, the Standard File dialog, and stationery.

Participants were randomly assigned to one of two test conditions: with tutorial, and without tutorial. We analyzed data from the first 17 of the 39 tasks. An analysis of variance (ANOVA) was calculated for all 39 tasks with factors *tutorial use* and *task time* (repeated measure). Because the ANOVA showed a significant interaction between tutorial use and task time ($F(1,38) = 4.3725$, $p < 0.0001$), we ran a Newman-Kuels test of multiple comparisons. This showed that specific tasks presented significant differences between the with and without tutorial conditions, but these particular tasks are tangential to the activation/selection issue. (We found significant interaction for the with tutorial group on three tasks: creating an OpenDoc document ($p < 0.0$), adding embedded content to an OpenDoc document ($p < 0.01$), and changing the label of an embedded Apple Button ($p < 0.05$). Only the last of these tasks dealt specifically with activation and selection.) In general, these summary findings are at too high a level to measure the study participants' conceptual grasp of the activation and selection model, although we did find some specific improvements could be made.

Observations

Our assumption in the design of the tasks was that users would *have* to activate or select a minimum number of times, and from timing data we would be able to analyze the learnability of these two major concepts. However, we had not anticipated that users would employ their own diverse strategies that did not necessarily exercise activation and selection to accomplish tasks. For example, in a task that required interaction with a sound part, we expected that users would first put it in the document and then use it, e.g., by first activating and then recording a sound into it. However, they tended instead to launch the sound part as its own OpenDoc document and experimented with it there, which obviated the need to activate the part as embedded content in a containing part. Users figure out their own way

of doing things and don't adhere to our expectations for how they will work with OpenDoc.

What is interesting about this observation with respect to the activation and selection model, is that users with many different strategies for accomplishing tasks are likely to be successful with OpenDoc without closely examining how to use it. Successful operation is not, however, coincident with actually learning the paradigm. Recognizing a useful strategy and employing it *without* a firm grasp of the underlying mental model requires the user to rethink strategies every time they are faced with a problem. We have yet to determine 1) how to increase user comprehension of the model; 2) how complete the user's mental model of OpenDoc needs to be in order to use it and feel good about it; and 3) what instrument is best to measure learnability.

In other words, although the successful task completion rate was fairly high, we can't conclude that these users would be able to come back a week later and be able to perform at the same level, with or without the tutorial. We believe a longitudinal field study is the appropriate test of retention, and may be the best way to measure learnability as well.

Design Implications

This test showed that clicking borders or using the command click shortcut (when instructed) was problematic. Users thought they should be able to both drag and select from the center of a part, rather than by using the borders.

We have no clear design direction from users that we could easily apply. Our options are to make borders more salient, make handles easier to see and drag from, and make cursor changes more usable. In today's OpenDoc, the cursor changes to a hand when it rolls over a clickable border. Users liked this because the change is highly visible. However, since the hand icon (see Figure 7) has traditionally been used to support panning operations in many applications, we should consider using a different icon than the hand when the pointer rolls over the activation border to indicate that the user can click and drag the active border. For example, some applications use the second icon in Figure 7 for grabbing and moving whole selections or windows. And, the third icon is a version of the icon that was going to be used when the cursor rolled over the activation border in Version 1.0 of the OS/2 implementation of OpenDoc.



Figure 7. Alternative cursors

Second, we should consider a "layout mode" for containers to make copying, moving, and deleting embedded parts easier for users. In "layout mode" a single click would select an embedded part, allowing the user to bypass activation for typical layout activities.

THE ACTIVATION / SELECTION SOLUTION

Over the four years of OpenDoc's development, we have revisited the design for activation and selection many times. Our testing concludes that while we could improve the

visual feedback for these two states, it is the introduction of modality and the concomitant learning curve for users that is most problematic. We could conduct a user study to further investigate the activation/selection model. Such a test could include the following aspects of the model that have not yet been studied:

- embedded parts that have their own selection model for their particular intrinsic content, i.e., text, drawing, spreadsheets
- multiple levels of embedding (at least three) and a balance of layout and editing activities

However, it appears that whatever design decision is made, we will still have to grapple with learnability, which is the true matter to study.

CONCLUSION

We learned what user studies can and cannot be expected to accomplish; they can help to support good design decisions, and provide input for changing unsuccessful design decisions, but they can't provide the design solution.

We found it was increasingly important to be able to test OpenDoc in an environment with working parts. Yet we couldn't wait until the product was complete to begin testing. Because so many of the aspects of the design we wished to test were highly interrelated, small studies focusing on single features conducted at different times were likely to give us varying results. Therefore, our early tests used a number of different prototypes and evaluative methods, and we specifically focused on getting study participants to tell us what they thought, how they felt, and what they expected. In other words, qualitative data was extremely important to us as we completed the design. Our strategy of evaluating design inductively until working code was available helped us to see alternatives while the design was still flexible. It was not until DR6 was available with working parts that we were able to do quantitative testing in a realistic environment.

We found that it is very difficult to isolate parts of the design for testing; for example, although we've focused on many different areas for our pre-tests, we've documented findings on selection and activation in almost every test. We recommend others undertaking similar studies to watch for unexpected results! We often were looking for information about one component of the design and learned about unrelated components.

Finally, it is impossible to do an infinite number of studies. So, there will always be some uncertainty about the results. And that's why we have follow-up releases.

ACKNOWLEDGMENTS

We thank our colleagues on the OpenDoc team, especially original members Kurt Piersol and Jed Harris, the architects of OpenDoc; and David C. Smith.

We also thank Kerry Ortega and Katie Candland for their carefully executed user studies and detailed documentation, as well as Mark Stern, Dan Jordan, Michael Emmett Thompson, Per Nielsen and Kristin Bauersfeld for their Jedi and Amber studies.

TRADEMARKS

OpenDoc® and QuickTime® are trademarks of Apple Computer, Inc. registered in the United States and other countries.

Director® and Macromedia® are registered trademarks of Macromedia, Inc.

Supercard® and Allegiant® are registered trademarks of Allegiant Technologies, Inc.

XEROX® is a trademark of XEROX CORPORATION.

Any other named products profiled herein are trademarks of their respective companies.

REFERENCES

1. Apple Computer, Inc. "OpenDoc Programmer's Guide," NY, NY: Addison-Wesley, 1995.
2. Apple Computer, Inc. "Macintosh Human Interface Guidelines," NY, NY: Addison-Wesley, 1992.
3. Smith, D. C., E. F. Harslem, C. H. Irby, R. B. Kimball, and W. L. Verplank. "Designing the Star User Interface" *Byte*, April 1982.
4. Randy Smith, David Ungar and Bei-Wei Chang, "The use-mention perspective on programming for the interface" in B. Myers (Ed.), "Languages for Developing User Interface Software," Boston, MA: Jones and Barlett, pp. 79-89, 1992.
5. White, E.A., Wildman, D.M., and Muller, M.J., "Games and other group exercises for participatory design of systems," Tutorial/workshop in Proceedings of Bellcore/BCC User Centered Design Symposium 1993, Piscataway, NJ: Bellcore, November 1993.