

# Designing the OpenDoc® Human Interface

**Dave Curbow**

Human Interface Design Center  
Apple Computer, Inc.  
1 Infinite Loop MS 302-1HI  
Cupertino, CA 95014 USA  
curbow@best.com

**Elizabeth Dykstra-Erickson**

Human Interface Design Center  
Apple Computer, Inc.  
1 Infinite Loop MS 302-1HI  
Cupertino, CA 95014 USA  
+1 408 974 6462  
eade@apple.com

## ABSTRACT

This paper tells the story of the development of the human interface for OpenDoc, a large-scale, complex, cross-platform commercial development project at Apple Computer. OpenDoc was an ambitious four year design and development effort by Apple with IBM and other partners.

The OpenDoc HI is a departure from traditional applications. This historical review highlights how we designed OpenDoc and the lessons we learned.

## Keywords

OpenDoc, design process, user-centered design, human interface specification, collaboration

## INTRODUCTION

OpenDoc is a development platform, and thus is very complex — designing it was more like designing an operating system than designing a new application. It was an ambitious and visionary project for Apple, involving a massive effort with complex logistics and politics. Its design posed a number of very unusual HI challenges.

Like most projects of this size, OpenDoc had a multi-year incubation period; it involved cross-disciplinary teams; and it followed user-centered design principles. Unlike most projects at Apple, it has had a cross-company collaborative development team. It continues to exhibit one of the best models of team behavior. Also, it is one of Apple's most thoroughly documented new technologies, due to a rigorous and finely-grained HI Specification. The success of this project influenced the HI design process at Apple.

Briefly defined, OpenDoc is an object-oriented software development architecture that enables software developers to create component software used in compound documents. It allows a user to create and edit documents containing virtually any kind of content — e.g., text, graphics, tables, spreadsheets, sound, video, virtual reality, and network and internet connections. These are referred to as *parts* and manipulated via *part editors* (see Glossary at the end of this paper). Most compound document systems have a similar appearance. OpenDoc's advantage is its

support for the creation and manipulation of compound, collaborative, and customizable documents on multiple computing platforms. It replaces data import and export functions with edit-in-place and drag-and-drop for adding and changing content.

Our goal was to develop an open standard for distributed component software which would interoperate with other architectures, such as Microsoft's OLE. A standards body, CI Labs<sup>1</sup>, was established to own the technology being developed.

The majority of this paper will consist of a description of the development of OpenDoc from the initial concept to its implementation. We'll break our discussion into three phases -- investigation, design, and implementation — that correspond to the internal development process. As we give this account, we'll occasionally pause to reflect on the utility of various decisions. After a description of the process, we'll step back and reflect on the lessons of the OpenDoc project for HI Design.

Because this is a vast project, we can't provide all the details; instead we'll focus on four themes that will be of value to other HI designers: leveraging existing research, the role of documentation in development, the politics of development and inter- and intra-company collaboration.

## PROCESS

First we'll give a quick sketch of the development process followed by OpenDoc. Next, we'll discuss the make up of the HI team over the course of the project. Finally, we'll discuss our work during each phase of development.

### *Design Process Sketch*

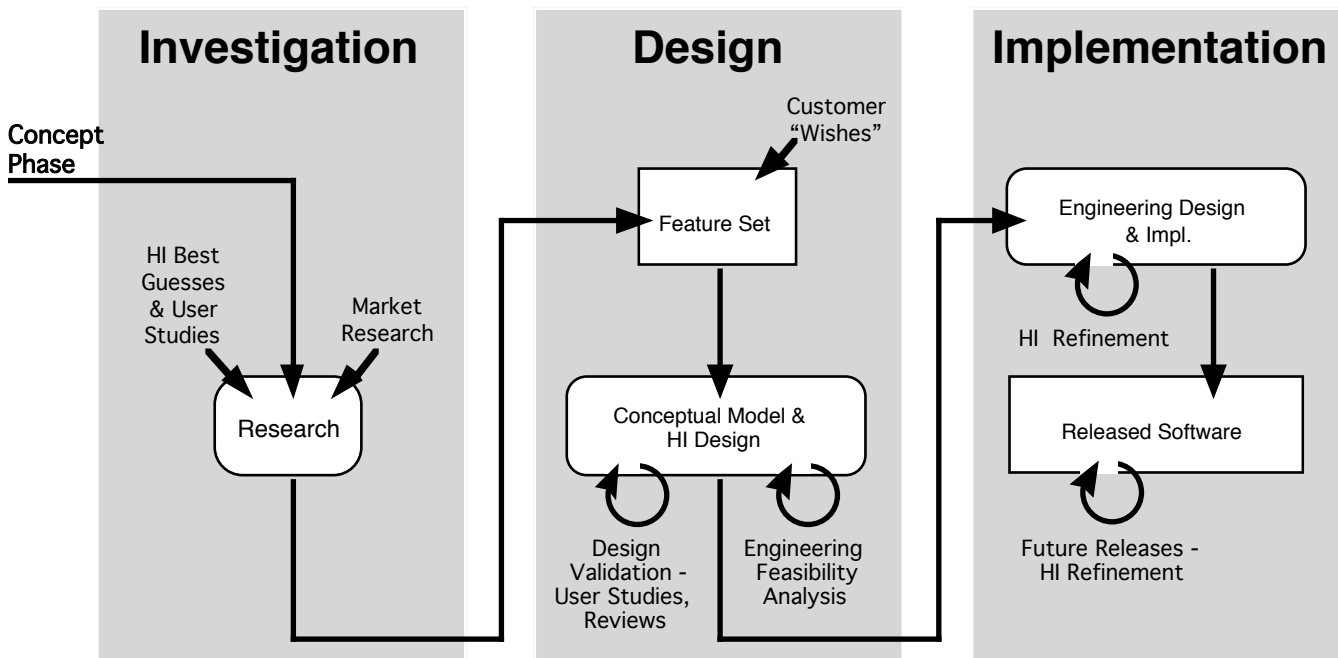
The typical Apple development process begins with the Concept phase — that's when an idea for a product is briefly outlined for management approval. Often, the real genesis of an idea occurs long before this presentation. For OpenDoc, the genesis occurred in the mid '80s when Jed Harris, a researcher in Apple's Advanced Technology

---

<sup>1</sup>Component Integration Laboratories, Inc., is a nonprofit association dedicated to promoting the adoption of OpenDoc as a vendor-neutral industry standard for software integration and component software. See <http://www.cilabs.org>.

Group (ATG) formed an informal team to examine component software. One of his team's key ideas was the transition from an application-centered paradigm to a content-centered paradigm (automatically invoking

whatever components are required to handle the content). This research continued in ATG until Jed moved to product development where he put together a small team that made a Concept proposal to management in mid-1992.



**Figure 1.** Software Design Process

Following management approval of the Concept proposal, a product team was formed for subsequent stages of product development. Figure 1 shows a schematic of these phases, and associated tasks from an HI perspective.

The goal of the Investigation phase was to determine if OpenDoc could be successful, if it would make a good product for Apple. It was during this phase that a small number of HI designers and engineers joined the team. Our two primary tasks were to perform the initial research and create a design sketch of what the product might look like. This work is represented in Figure 1 as the Research task – discussed later in detail.

Table 1 summarizes the major phases and deliverables from the HI team during the development process.

| Phase          | HI Deliverables                              | Duration  |
|----------------|--|-----------|
| Concept        | Management Presentation                      | N/A       |
| Investigation  | 1) Research Requirements<br>2) Design Sketch | 3 months  |
| Design         | 1) Conceptual Model<br>2) HI Specification   | 11 months |
| Implementation | 1) Revised HI Spec<br>2) HI Guidelines       | 15 months |
| Post Release   | 1) User Studies<br>2) Revised HI Spec        | Ongoing   |

**Table 1.** Phases and Deliverables

At the end of the Investigation phase, management agreed that we should continue to the Design phase and provided funding for additional HI designers, engineers, marketing and support staff.

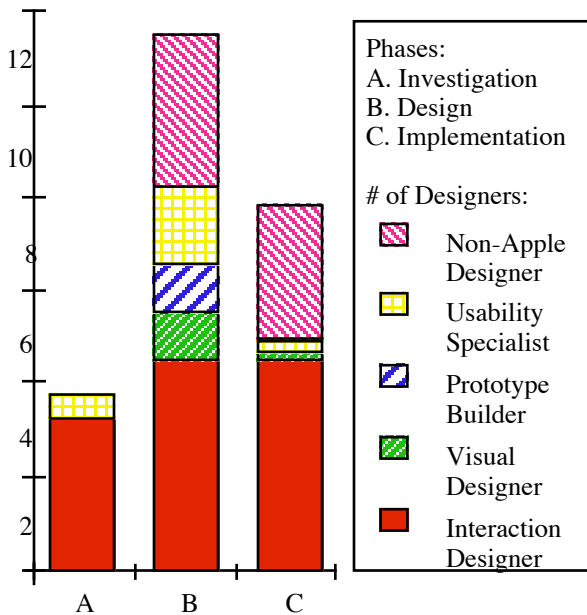
During the Design phase, our goal was to create a complete design, one that Engineering could begin to implement. We began by reexamining our prior research, looking for loopholes and filling in the gaps. Also, it was during the Design phase that we began working with designers from our partners. This is discussed in detail in the section, *Design Review*.

When we were able to deliver the HI Specification to Engineering, the project entered the Implementation phase. During this phase, Engineering built the product according to the specification, while Quality Assurance checked for deviations between the product and the specification. We collaborated with these other teams to design solutions to the problems and shortcomings found – and we continually revised the specification.

Because OpenDoc is a technology intended for *developers* (see Glossary), we targeted our releases for developer-specific milestones, for example Apple’s World Wide Developers Conference held each April and the Macworld Expo/San Francisco held each January.

### HI Team Composition

Now we'll describe the various kinds of HI specialists working on this project. Interaction designers were responsible for creating the conceptual model and appropriate behaviors. The HI Architect was an interaction designer. Visual designers provided visual designs and feedback on interaction design. Usability specialists created plans and studies to provide supporting and confirming data from user studies. Our Prototype Builder was skilled at quickly building prototypes which could be used to try out design ideas or during early user studies. Figure 2 shows how the balance of specialties shifted during the course of the project. The designers labeled "Non-Apple Designers" were from our partner companies.



**Figure 2.** Number of HI Designers Over Time

For example, during the Design phase, we needed skills such as a Prototype Builder and Usability Specialists. These additional people also helped when brainstorming possible design ideas. However, during the longer Implementation phase we reduced the number of HI designers because our needs changed — we no longer needed the prototype builder and we needed fewer usability specialists. During this phase we continued to revise the specification and run user tests. Our team had four interaction designers plus occasional support from Visual Designers and Usability Specialists. The number of designers among our partners also decreased slightly, for the same reason. However, the number of people we worked with increased. For example, a large number of engineers, product marketing specialists and *evangelists* (see Glossary) were added during this phase.

### Research

Now we will discuss in detail the research work done initially during the Investigation phase and refined during the Design phase. This research was subdivided into several tasks and deliverables, as summarized in Table 2.

Although some of these tasks had dependencies on prior tasks, we were able to do some work in parallel. Each task represents major milestones for the HI team in the development process and is discussed in more detail below.

#### Identify Target Users

One of the first tasks was to accurately identify the target user population for OpenDoc. During the Investigation phase we made some educated guesses. But, during the Design phase we had to do better. To get an accurate reading of our target user population we worked with the OpenDoc marketing specialist. He identified desktop publishing, K-12, higher education, and small business/home office as primary target market segments. He then brought us together with several Apple market specialists, for example, someone responsible for Apple's K-12 business. These specialists, through their ongoing research, had a wealth of information available about users in their particular market segment.

| Research Tasks                      | Deliverables  |
|-------------------------------------|---|
| Identify target users               | User Matrix   |
| Identify user tasks                 | Scenarios   |
| Identify user needs                 | 1) User Needs document<br>2) User Needs matrix                          |
| Identify issues in meeting features | Issues Database   |
| Survey Apple technology             | 1) Issues Database<br>2) User Needs document                            |
| Perform competitive analysis        | 1) Reports on competing products<br>2) User Needs<br>3) Issues Database |

**Table 2.** Research Tasks

They did not gather this data explicitly for HI design. Thus, it was important for us to work closely with the marketing specialists for their interpretation of this data. They helped us to better understand the user population, their tasks and needs. For example, the publishing market specialist told us about the need to provide control over color separation for that market. From the data we collected from marketing specialists, we were able to characterize first-time, novice, and experienced users within each market segment. These characteristics contributed to user profiles that we used to write task scenarios, identifying typical user tasks. We used the profiles and scenarios to generate product features and to support design decisions.

We developed a user matrix to document and contrast user characteristics for *first time*, *novice*, and *experienced users* according to the tasks we could expect them to be able to accomplish using OpenDoc. Here are some examples of those user characteristics:

#### First-Time Computer User

- Can learn how to create compound documents
- Doesn't want advanced capabilities to get in the way of getting started with compound documents
- Doesn't expect to do advanced operations such as *linking* and *scripting* (see Glossary)

#### Novice Macintosh User

- Can use multiple parts in a document, but only those originally installed
- Can use stationery
- Does not know about the concept of "part editors;" only sees and uses parts
- May be using "At Ease" or other non-Finder environments

#### Experienced Macintosh User

- Can install new part types onto the system.
- Can use multiple editors and switch between editors in a document
- Can use library of parts and stationery
- Familiar with scripting and interpart communications capabilities
- Very familiar with all basic Mac and most advanced features

#### Developer

- Can deliver new part types and associated editors to users

Today, four years since the beginning of the OpenDoc project, we've found this early documentation still very useful. Consulting it has helped us understand why some users have problems using OpenDoc — there are occasional mismatches between our user definitions and actual users. For example, we believed that novice Macintosh users would be familiar with and able to use stationery, but we've been surprised to discover through user testing that this is not true. This limitation does not, however, hold for OpenDoc on all platforms. In contrast, OS/2 users are more familiar with stationery, although they use the term *templates*. We also learned that several features of the MacOS we characterize as fundamental are not used by novices, such as marquee selection and opening files via the Standard File interface. After recognizing our mistakes, we worked with Apple Instructional designers who created materials to educate novice users.

In retrospect, we believed that users were somewhat more sophisticated than they proved to be. We see two solutions to this problem — 1. Refine our user profiles. 2. Provide

additional instructional materials to users for use during user studies and actual use.

Our companion paper, *Role of User Studies in the Design of OpenDoc* (this volume), provides additional information on this topic.

#### *Identify User Tasks*

Given a model of our target user population, we set about to identify user tasks. We did this by way of first building scenarios of use and then deriving user tasks from them. We identify user tasks by determining exactly how a user performs a job. Explicitly describing the steps taken by users when performing a task provides details of the information and process flows. Knowing these steps helps us to better determine users' needs so that the system we design can effectively address those needs.

#### Generate Scenarios

The OpenDoc Human Interface team generated ten sets of user task scenarios to develop a better understanding of what an OpenDoc user might need. In the ideal situation, we would write up user task scenarios from field observations. We would collect empirical data to obtain a description of the task and develop an understanding of user needs. However, we could not have conducted a field study without having had to create both OpenDoc and third-party components which used it. This was clearly a chicken-and-egg problem!<sup>2</sup> Ultimately, we used generated (rather than observed) scenarios in the OpenDoc project. We were comfortable using such an approach because we were able to supplement actual test results with competitive analyses of similar existing systems, such as the Xerox Star. For example, we were able to determine that users liked compound documents and edit-in-place. Also, we learned that the *inside-out activation model* (see Glossary) worked well, but users had some problems selecting embedded objects. Based on this, we determined that we would make selection of embedded objects easier in OpenDoc than in the Xerox Star.

#### Derive User Tasks

We identified user tasks independently from system constraints and focused instead on the users' needs. This allowed us to think of new design solutions to users' problems. In particular, the task descriptions did not presume a system design (except to describe a system the user may currently be using), but instead uncovered the *underlying needs* the user has, independent of how he or she may be satisfying that need today (e.g., with a current system).

---

<sup>2</sup>OpenDoc 1.0 was released December 1995. Field studies were designed early but are only just recently in planning, made possible by the availability of mature end-user components.

### Identify User Needs

By analyzing the user tasks, we were able to identify specific user needs. We documented these needs and used them to derive the OpenDoc feature set. The following is an example of the user needs identified via user task scenarios.

- Coexist with today's applications
- Transfer existing skills
- Able to select data
- Natural way to deal with and edit content of different kinds in a document
- Transfer data from non-OpenDoc to OpenDoc documents (and vice versa)
- Easy management of different parts of a document (e.g., *in situ* editing, drag and drop)
- Need a clear input focus (i.e., feedback)
- Clear feedback to user of boundaries that affect content's behavior or appearance
- Have all parts of compound document in one file and be able to modify each part
- Easy to add new kinds of data to the document, including kinds not yet invented
- Need to transfer data from and to documents, irrespective of kind
- Able to change form, appearance, shape and other attributes of parts and content

Because we characterized users as we did, we were able to design for different categories of users. For example, experienced users may need to check to see what OpenDoc editor is being used to edit a particular part, while novice users would not have this need. The design implication here is that we could use progressive disclosure to reveal advanced features of simple tasks, thus moving complexity away from the novice user but still making it available to experienced users. Likewise, one class of users may need capabilities not needed by any other class. For example, users in the desktop publishing market segment may need color separation capabilities while few other user will. The design implication here is that we needed to be able to extend controls for certain markets, resulting in OpenDoc's very flexible plug-in architecture.

### Identify Issues in Supplying Features

In the course of many meetings and discussions to work with the data we collected and build the scenarios, we were continually identifying design issues that needed to be resolved. As we began designing the conceptual model, we had identified approximately 100 issues. Some were clearly related, and we needed a mechanism that allowed us to make sense of the relationships. We chose to group related issues. To do this, we used the "post-its on the wall" method. That is, we wrote each issue on a sticky note and posted it on the wall, looking for obvious groupings. This

method allowed us to easily move issues around and regroup.

We used this set of sticky notes to develop a simple issues database to track issues, their design implications, and their resolutions. This repository, similar to those maintained by certain design rationale support tools such as IBIS or DRL [1], was structured but informal. Recognizing that some issues were clearly more important than others, we instituted a simple priority system. This way we were able to make certain that we factored high priority issues into the design before lower priority ones. We continued adding issues until the Design phase was complete. Table 3 shows an example of HI issues.

### Survey Apple Technology

We knew that various groups within Apple had previously investigated many aspects of component software and compound documents. In addition, we knew that Apple had delivered, or was about to deliver, several technologies that we could use to support OpenDoc. The engineering and HI teams examined this work for relevance to OpenDoc and, when appropriate, we influenced the design of these products. For example, the DragManager (supports drag & drop between applications) was being developed in the early days of OpenDoc. In our third user study, we received strong feedback that users felt drag & drop was very powerful and easy to use. Based on this, we decided that it would be an important technology to include in OpenDoc. Thus, we worked to influence its HI design and implementation.

### Perform Competitive Analysis

We analyzed competing products, to see if there were user needs that we had overlooked. Products we analyzed included the Xerox Star, Microsoft OLE and the OLE 2.0 prototype, Taligent's "PINK" project, ClarisWorks and Claris Impact, as well as several early Apple research projects.

| User Model of Parts                                     |
|---|
| <b>High Priority</b>                                    |
| part metaphor (tool, container, accessory?)             |
| multiple views of same part                             |
| can any part be opened anywhere? what happens?          |
| where can part frames appear (on desktop, in document)? |
| where can part icons appear (on desktop, in document)?  |
| interaction with desktop model                          |
| <b>Medium Priority</b>                                  |
| part editors (a user visible concept?)                  |
| direct manipulation of parts                            |

**Table 3.** HI Issues

One of our conclusions was that the Xerox Star dealt with a large percentage, but not all, of the same issues that OpenDoc was facing. It had been widely used for several years by several thousand users and so had been refined extensively. We were able to follow up our competitive analysis by interviewing two of the principle designers of the Xerox Star. We asked questions like, “Were there things in the design that now appear to be mistakes?” We were also able to identify a few users of some of the other products and interview them. The data gathered was recorded in papers and the Issues database.

**Design**

Now we will talk about creating a design, based on the data gathered. Table 4 summarizes the steps in this process. Completing these tasks required several months.

*Map Needs To Features*

The first step in the design process was to define a set of features that satisfied the user needs identified previously. Sometimes simply looking at a set of needs suggested a set of features; for example, the need for users to transfer skills from today’s applications, as well as for OpenDoc to coexist with today’s applications, suggested that we reuse as many of today’s designs as would work in OpenDoc. Thus, we kept Cut, Copy and Paste, but modified them to work seamlessly with content containing embedded parts. Other features were suggested by examining our survey of previous Apple research and our competitive analysis reports.

In an effort to keep the feature set as small as possible, yet still cover all the user needs, we created a 35x99 matrix of *User Needs by Features*. A portion is shown in Table 5.

This matrix allowed us to:

- Understand the relationship between user needs and features
- Effectively modify or design new features by easily referring to the user need

| Design Tasks                          | Deliverables   |
|---------------------------------------|--|
| Map Needs to Features                 | Input to Feature Set   |
| Create Conceptual Model               | 1) Conceptual Model document<br>2) Prototypes                          |
| Develop Usability Goals               | Usability Goals document   |
| Develop Specific Performance Criteria | 1) Usability Test Plan<br>2) Performance Requirements document         |
| Document the Design                   | 1) HI Spec<br>2) HI Notes<br>3) HI Guidelines<br>4) Platform Checklist |

| Document the Rationale Behind the Design  | Design Rationale document   |
|---|---|
| User Tests  | 1) Test Plan<br>2) Test Prototypes<br>3) Test Results & Recommendations |
| Review Design with:<br>1) Engineers & Marketing<br>2) Apple HI teams<br>3) External Partners<br>4) 3rd-Party Developers | 1) Revised HI Notes<br>2) Revised HI Spec<br>3) Issues Database         |
| Work with CI Labs partners to make OpenDoc cross-platform   | Consultation to<br>1) IBM<br>2) WordPerfect/Novell                      |

**Table 4.** Design Tasks

**Editing Parts**

|          |                      |   |   |   |   |  |  |
|----------|----------------------|---|---|---|---|--|--|
| Features | Copy                 |   |   |   |   |  |  |
|          | Copy via Drag        |   |   |   |   |  |  |
|          | Delete               |   |   |   |   |  |  |
|          | Undo                 |   |   |   |   |  |  |
|          | Redo                 |   |   |   |   |  |  |
|          | Show Properties      |   |   |   |   |  |  |
|          | Change Properties    |   |   |   |   |  |  |
| Needs    | 1. Select data       |   |   |   |   |  |  |
|          | 2. Edit naturally    | x |   |   |   |  |  |
|          | 3. One compound file |   |   |   |   |  |  |
|          | 4. Easy part mgmt    | x | x | x | x |  |  |
|          | 5. Easy to add parts |   |   |   |   |  |  |

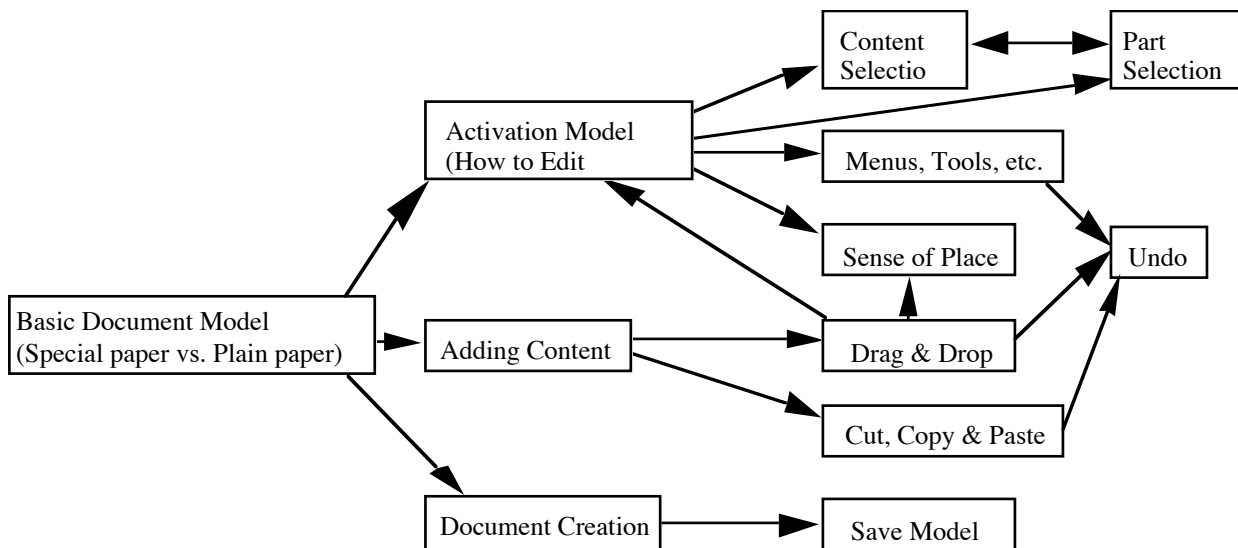
**Table 5 .** User Needs by Features Matrix

- Identify "holes" in the feature set (when a need has no feature to satisfy it)
- Identify unnecessary features (when a feature has no corresponding need)
- Identify redundant features (when two or more features satisfy the same needs)
- Effectively design and focus our user tests

Features in this matrix were grouped to make it easier to find related features, as above where we show a portion of the “Editing Parts” category. The features and their categories map directly onto the description of the features later written in the OpenDoc Human Interface Specification.

*Create the Conceptual Model*

The Concept phase is generally short and determines if a project should go forward. We created an initial design and produced a *User Experience Document* and a Macromedia Director® prototype that illustrated the user experience.



**Figure 3 . A Fragment of the Conceptual Model: Issues and their Dependencies**

As we began the Design phase — during which we dug deeper into the requirements and double checked our initial findings and recommendations — we also produced several new documents reflecting the volumes of data we had gathered.

The first step in creating the OpenDoc HI design was to create an appropriate conceptual model. To do this, we first identified the features and related issues that were most important and attempted to create a coherent design that satisfied all of them. We began by brainstorming, sketching on a whiteboard and creating rough prototypes.

As we were creating the conceptual model and later fleshing out the rest of the design, we followed a few design principles:

- Make it possible for users to reuse as much as possible of their knowledge of today’s applications and apply it to using OpenDoc

- Avoid surprising the user
- If two things look very similar (conceptually, visually), strive to MAKE them the same — or very distinct

After writing a document describing a coherent conceptual model, we began to create the HI Specification for all the features, showing how they fit into the conceptual model. To do this, we examined the feature set and issues database again, looking at the lower priority features and related issues.

There are clear dependencies between components of the conceptual model. Figure 3 (below) shows a few components. These dependencies mean that if we modify the design of one component, we must examine the design of dependent components and then possibly modify them to be in alignment with the higher priority component. For

| Goals                                      | Methods                                   | Measures   |
|--|---|--|
| Improved ease of use                       | Drag & drop all over<br>Edit in place     | task timings<br>error rate   |
| Improved learnability                      | Easy user model<br>Progressive disclosure | task completion rate with and without tutorials and/or assistance<br>task timings with and without tutorials and/or assistance |
| Functionality matches user needs and tasks | Field study (in planning)                 | user satisfaction ratings (in planning)  |

**Table 6.** Usability Goals

example, if we made a change to the Activation model, we would need to revisit the design of Content Selection, Part Selection, Menus, Sense of Place, etc. Likewise, if we

made a change to Drag & Drop, we had to double check the Activation Model, Sense of Place and Undo. You’ll notice that most of the dependencies are one way, but two of them

are two way — Content and Part Selection. Modifying the design of either of these components required that we examine and possibly modify the other component.

In fact, we never created a complete diagram, but juggled this information in our heads. At one point, we counted the number of components in the model — it was about 200 items, far too many to easily remember. We now realize that keeping this map in our heads was a mistake, for not all the relationships were obvious, and we spent much time “remembering” these dependencies.

*Develop usability goals*

After developing a conceptual model, we began work on usability goals. An excerpt from the *OpenDoc Usability Goals* dated 11/9/92 is shown in Table 6. We used these goals to direct user studies performed throughout the design process.

These goals are general and were used to construct more specific *Usability Objectives*. Table 7 shows an excerpt of document-centered usability objectives. These objectives led directly to the design of specific user studies.

| Compound Document Objectives                          |
|---|
| 1. Develop more natural application/document paradigm |
| 2. Support natural interaction with parts             |
| 3. Develop effective selection model                  |
| 4. Seamless integration with current system           |
| 5. Effective and natural use of pasting and linking   |

**Table 7.** Specific Usability Objectives

We believed it was important that users be able to quickly use OpenDoc, so we tried to facilitate transfer of knowledge from the current Macintosh UI to the OpenDoc UI. However, we also recognized that there are some problems that are not well handled by the Macintosh UI, and better solutions would require users to learn a new model.

This was reinforced by an early conversation with Apple Fellow, Alan Kay. He reminded us that it is important for users to explore their environment, and to learn about its capabilities. One example of how we took this to heart is our use of progressive disclosure as a means of hiding complexity, relying on users to explore the system.

*Develop specific performance criteria*

Table 8 illustrates the performance criteria determined jointly by the HI and Quality Assurance teams, based on MIL-STD-1472C [2]. The engineering team used these as criteria for deciding between implementation alternatives. When OpenDoc 1.0 shipped, all but the launch time criterion had been met.

| Event                                      | Performance Goal  |
|--|---|
| Part Activation*                           | .2 - .4 sec   |
| Frame highlight when drag over             | .1 sec  |
| Resize/Reshape Frame                       | Reshape frame must follow pointer in real time where mouse velocity $\leq 200$ pixels/second  |
| Open Document                              | $\approx 1$ sec   |
| New Part Creation                          | 1. Via Paste<br><.5 sec (no translation needed)<br><2 sec (simple translation)<br>2. Via Drag-and-Drop<br>.5 - 1 sec  |
| Menu command completion                    | 1. Editing Cmds, Formatting/Layout [ $\approx 1$ sec]<br>2. Close Document (no save dialog) [ $< 3$ sec]<br>3. Link propagation (Doc A $\rightarrow$ B real time update) [ $< 3$ sec] |
| Activate / Deactivate Frame (change focus) | Frame hilight when cursor moves into frame .1 sec   |
| Open Part Property Sheet                   | < 1 sec   |
| Drag / move part within document           | Same as Resize/Reshape frame  |

\* Frame Highlighting, palette & menu switch

**Table 8.** Performance Criteria

*Document the Design*

OpenDoc was not the first object-oriented component software product. Several members of the team were familiar with the Xerox Star and the design process used in its creation, as described in the 1982 Byte article, *Designing the Xerox Star User Interface* [3].

The “bible” of that project was the “Star Functional Spec,” a very detailed specification that spelled out the behavior and appearance of the finished product. Often specifications are vague enough that there is room for interpretation; this was not the case with the Star Functional Specification. During their implementation phase, the Xerox engineering staff discovered that they could not implement some behaviors exactly as specified. New designs were created, but the specification was not revised to reflect the changes. As a result, there wasn’t a single document which specified the human interface for

the product. Later this was recognized as a flaw in the process.

The OpenDoc HI team resolved to deliver a similar HI Specification, but keep it always up to date. This turned out to be a key decision. This document, plus the HI Guidelines and the Design Rationale (both described later), were instrumental in guiding the OpenDoc implementation. Much of the information in these documents also became part of the OpenDoc Programmer's Guide, the publicly available developers' resource book.

#### Writing The HI Specification

Given the Conceptual Model paper, plus information in the feature spec and the issues database, two senior HI designers wrote an initial draft of the HI Specification.

Once we had written this draft of the spec, we realized we had overlooked several issues. As a result, the team spent several more months investigating issues and reworking the design and specification. The spec had many holes to fill. So, each interaction designer was made responsible for completing the design of various sections.

We sifted through the mountain of data, and talked with other HI designers and the engineers. We reused existing designs where applicable and brainstormed new ones. We recorded everything — even sketches from brainstorm sessions. We used a whiteboard that produced a printed copy, and we signed, dated and filed a copy in case of legal need later.

Each designer's objective was to write a formal *HI Note* for each feature for which they were responsible. In this note, they laid out design alternatives, and their pros and cons. Next, the entire HI team reviewed the work. The team might accept a recommendation, or employ a user study to choose between alternatives. Engineering was responsible for qualifying the implementability of these decisions. Finally, we updated the specification. To date, there are 36 OpenDoc Human Interface Notes each addressing a different set of issues. These HI Notes also continue to serve as repositories for information that needs to be included in future versions of the specification or guidelines.

The HI Specification is 211 pages describing the interaction and visual behavior of OpenDoc -- down to the level of mouse events and pixels. It includes sections on:

- Conceptual Model of Documents and Parts
- Part Icons
- Part Frames
- Editors and Viewers
- Menus
- Interaction with non-OpenDoc documents
- Windows
- Part Categories
- Part Properties and Settings
- Preferences
- Viewing a Part
- Changing a Part's Representation

Selecting Parts vs. content in a part

Activating a Frame

Editing a Frame

Cut, Copy and Paste - additional behaviors for OpenDoc

Drag and Drop

Create, Delete, and Undo

Stationery/Templates

Document and Part Naming

Installing and Removing Editors

Document Versioning (Drafts)

Linking

Scripting

Extending OpenDoc with Plug-Ins and Services

Adapting OpenDoc to other platforms

The specification addresses each of these aspects of the human interface and examines it in terms of its definition, guiding metaphor, behavior and states, visual appearance, and interaction with other interface elements. The specification addresses in particular the user's conceptual model and step-by-step user interactions. In contrast, the HI Guideline omits most user behaviors and concentrates instead on what developers must do to implement the specified system.

#### HI Guidelines

While we were writing the specification, we recognized that we would need to write guidelines for developers. So we began writing that document, and we revised it several times over the next 3 years.

By the time of the publication of the OpenDoc Programmers Guide [4], 107 pages of that book were devoted to new or revised HI Guidelines. We also influenced the guidelines for other Apple projects and a forthcoming edition of the *Macintosh Human Interface Guidelines* [5].

#### *Document Rationale Behind Design*

Near the end of the Design phase we produced a *Design Rationale* document, capturing the information on which we based our decisions. We wrote it without knowledge of the formal method known as Design Rationale, although we later discovered some overlap with this method.

Our purpose in writing our Design Rationale was to support the review process (discussed below) and to provide future team members with answers to questions about design decisions. We intended this document, and the other data in our archive, to provide a rational basis for making changes in the future. The archive, currently over 450 MB, of papers, email, and prototypes is used by team members as a research source. Work continues on the OpenDoc HI design, so periodically a new CD is pressed and distributed to team members and other interested groups.

#### *User Tests*

During the Design phase, we ran four user studies to gather information about user tasks and our initial design ideas. Two of these were paper-and-pencil tests. We also used a

Macromedia Director® prototype to determine if users could understand the basic interaction model.

In the middle of the Design phase we constructed a prototype in C to allow us to test our Drag & Drop model. A couple of months later we were able to begin using prototype code written by the engineers, rather than building our own. This allowed us to perform more complete user studies. Later in the Implementation phase, we ran six more user studies.

#### *Design Review*

During the writing of the HI Specification, there were daily informal reviews between HI, Architecture and Engineering that shaped the design. There were also two kinds of formal reviews:

- Review by engineering for feasibility
- Review of user needs, features and designs

When we released the first draft of the HI Specification, engineering went over it with a fine-toothed comb, looking for features that would be impossible to implement as specified. We had a formal meeting where the engineering team talked about specific difficulties and recommended modifications. We produced a second draft of the spec, incorporating their feedback.

Next we sent the HI Specification and Design Rationale to HI designers and researchers throughout Apple for their comments. At this review we defended the design, answered questions, recorded problems found and suggestions made. Afterwards, we wrote another draft of the specification.

Now that we had created a coherent, written specification, we were ready to go back to external parties for their review. In particular, we requested feedback from 3rd-party developers who had given us input early in the process. These reviewers reported problems such as overlooked user needs, and they suggested design alternatives. We came back to these developers several times over the next couple of years.

We also began meeting with our partners in CI Labs—IBM, WordPerfect, Novell, Sun, etc. They helped us understand other platforms and special requirements. We conducted periodic face-to-face meetings and established an email mailing list to discuss issues and solutions.

When we started talking publicly about OpenDoc (then known by the code name “Amber”) we gave out hundreds of copies of the HI Specification. At that time many Apple developers started asking questions and making suggestions that helped us to further refine the design. Later, after the Windows- and OS/2-specific specifications were released, we received feedback from those developer communities.

#### *Work With Partners on Other Platforms*

The first step in porting the Apple design was taken by HI designers at our partner companies. They reviewed the Macintosh OpenDoc HI spec and identified conflicts and

shortcomings. We then collaborated on a solution — deciding either to make a change to the Apple design, or to add a feature to their platform. We did not use a least common denominator approach to design, but rather a top down design process.

During this review process, we realized that much of the design of OpenDoc for Macintosh could be “ported” to the other GUI platforms with few changes because the Macintosh was largely a superset of other GUIs. However, there were a few mismatches, and we found ways to resolve them. If a feature was missing on one platform, we investigated how we might add it. For example, UNIX does not support the concept of stationery — a key component of OpenDoc. It was added to IBM’s implementation of UNIX, known as AIX.

Occasionally, other platforms provided a key feature in a way that was very different from how it is provided on the Macintosh. For example, where Drag & Drop exists, it behaves very similarly — except that the visual feedback is very different. In cases like this, we abstracted the design and permitted a variance. That is, we specified the feature in sufficient detail that each partner was able to create a design that satisfied that feature for their platform. Another example is the OS/2 standard for menus. This required menu commands to be organized differently from the Macintosh design. Thus, we created a variance for menu command arrangement, but not for the commands themselves.

We were concerned that there should be one identity for OpenDoc, with no large differences between the different platform implementations. To this end, we created a *Platform Checklist*. This checklist specified the basic design of features that make up OpenDoc and must be present on all platforms claiming to support OpenDoc.

#### **Implementation**

The HI Specification became the “bible” for OpenDoc. The Quality Assurance team refused to pass the implementation of a feature unless it matched the specification. As a result, engineers coded closely to the specification. In fact, if they were asked (even by HI) to make a modification, they refused to do so until they saw it in the HI Specification. This forced us to keep the specification up to date.

We actively encouraged QA and Engineering to issue bugs against problems and underspecified features in the HI Specification. Often, the problems were edge cases we had failed to consider during the Design phase. Sometimes we could redesign the feature on the spot, but other times we required a week or more. Some problems required us to build prototypes and conducted user studies to choose the correct design alternative. Each time we chose an alternative to the documented design, we updated the HI Spec and marked the bug as FIXED. This was Engineering’s cue to make any needed changes to the implementation and then QA would test and approve the change. This system kept the various teams in tight synchronization.

As a result, the HI Spec is quite large and detailed. When the first release of OpenDoc shipped, the HI Specification was 211 pages and had been through at least 5 major revisions and as many minor ones. It has served as a model for subsequent HI Specifications here at Apple.

### **LESSONS FROM THE DESIGN PROCESS**

HI teams at Apple are typically cross-disciplinary. The OpenDoc project team was interesting because it was cross-functional and it included designers from multiple companies. As a platform, OpenDoc was conceived by Apple and supported by Apple's partners IBM, Novell, and WordPerfect. Early supporters also included Sun, HP, Lotus, Taligent and XSoft.

Members of the project team included engineering, human interface, product marketing, evangelism, and legal teams from Apple, IBM, WordPerfect, Novell and Sun. Specialists on the team included market researchers, interaction designers, visual designers, usability specialists, prototype builders, software architects, coding wizards, quality assurance engineers and technical writers. This brought many different perspectives to our tasks.

#### **Team Process**

The working relationship engendered at the beginning of the project was remarkably positive and effective and remains to this day one of the best examples of team collaboration within Apple. Part of the reason was the initial make up of the Apple project team. We know that HI designers are often brought into a project after it is well underway. Even at Apple, which prides itself on user-centered design, this sometimes occurs. However, the OpenDoc software architects insisted that HI was not their domain and that suitable designers should be assigned to the project. This recognition that HI design was a specialized role, just as engineering or marketing are specialized roles, set the stage for all subsequent interteam interactions.

Each group (architecture, engineering, marketing, management) helped define the product by publicly specifying constraints. For example, marketing specified the class of computer systems on which OpenDoc would run, while HI specified performance criteria for important operations. These constraints were valuable in helping us choose between design alternatives.

Anyone can provide a good idea -- but the HI team has the responsibility to determine if a particular idea works in the overall design. When the team rejects an idea, it is important to document why it was rejected and to communicate this information to the originator of the idea. Any member of the product team was free to contribute ideas and criticism to another team, but in the end, each team was held *responsible* for the excellence of their deliverables. Thus, throughout the first four years of OpenDoc, the HI team was responsible for ensuring that the user interface was the best one possible. We found it very helpful to decree that the HI Architect had the authority to

unilaterally decide an issue. This worked as a forcing function to bring designers together.

#### *Social Control*

The HI team had great influence on the design process because we worked to be in control of it. That may seem like an odd statement; after all, most designers WANT to be in control, but often are not. Because we were brought in at the beginning of the process, and we were expected to be responsible for the design, we decided that our job would be easier if we could be instrumental in driving the design process. Thus, at the first kick-off meeting, we presented a proposal for how *we* recommended doing the design work. Because no else presented a competing proposal at that time, everyone focused on modifying our proposal until it was acceptable.

Another example of how we worked to be at the center of the process is more literally physical. To focus effort, Apple management sequestered the Apple team in a small building away from the main campus. The HI team staked out four cubes in the middle of the group of work cubicles, and tore down the walls to create a large open space. Because of a lack of space in the building, this area became the informal meeting place for everyone. Whenever a hot topic arose, people from the surrounding cubes seemed pulled to our area and joined in the discussion.

#### *Physical Control*

The breadth of the entire multi-company team caused complexity in communicating; only the Apple teams were located in the same city. Other members were scattered through the US and UK. This distance and disciplinary diversity required an open model of sharing documentation across teams. Lots of paper was exchanged and weekly telephone conversations were conducted with an emphasis on engineering (primarily API) negotiations. The HI team conducted its inter-company business through email and periodic face-to-face meetings with our partners from across the US and UK.

#### *Creative Control*

The feature set was "owned" by the project manager, but was largely defined by the Apple Human Interface team. Features were negotiated with the various teams and included input from a broad range of contributors, but the human interface team and the architects had the largest degree of control. We had a shared goal to keep the feature set as small as possible but sufficient to meet user needs.

Apple has a well-known tradition of user-centered design and interface excellence which results, in part, from strong support of human interface guidelines. In the case of OpenDoc, our engineers were well versed in the guidelines and were highly sensitized to the human factors consideration of user needs. The HI team decided when it was necessary to break the guidelines or replace them with more appropriate guidelines. This is particularly important when the technology worked on is new and very different from traditional applications for which the guidelines were

written. We felt comfortable doing this because our HI team members are part-owners of the larger Apple HI Guidelines, and we expected a foundation technology such as OpenDoc to stress and advance these Guidelines.

In many aspects, the Apple HI team was at the center of the design process -- gathering research, influencing the Apple Engineering and Marketing teams, influencing other companies, and creating the official design documentation. It was important to build on Apple's history and learnings and not do design in a vacuum. Therefore, our team leveraged much existing material. For example, there are various Apple research organizations that carefully track users' responses to our products. We used this information, rather than beginning with a "blank page" and then doing primary research to understand users and their needs. Because we were able to do this, we were able to avoid much time and effort that we otherwise would have spent doing primary research on the spot.

The HI team clearly influenced the Apple Engineering team via documentation such as the HI Specification, but also via collegial discussions and negotiations. We also influenced marketing by providing prototypes and input to collateral which were used to sell the ideas in OpenDoc.

Our partners relied on the Macintosh OpenDoc HI design for their own platforms. We proposed the initial design, took their input, reviewed their specifications, wrote the Platform Checklist and made the "final" decisions on contested design decisions -- all of which gave us a great deal of control over the overall design of OpenDoc.

We influenced developers by producing material to showcase OpenDoc's developer opportunities. For example, we did three presentations, each to audiences of more than 1000 developers. We also wrote four major journal articles, a magazine article and a dozen monthly FAQ articles sent to Apple developers. We appeared in an informational video produced for the developer community. We also provided consulting via email. We participated in Apple's World Wide Developers Conference in 1993 through 1996, presenting and offering on-the-spot consulting.

To expose OpenDoc to HI professionals, we did a presentation to BayCHI (the San Francisco Bay Area branch of the Association for Computing Machinery's SIGCHI) and then at British Computer Society HCI '94; presented posters at ECSCW'95 and British Computer Society HCI '95; and manned a heavily-trafficked demo booth at CHI '96.

## **INSIGHTS**

We view the OpenDoc project as an extremely successful long-term effort in both design process and in the quality of the interface. We'd like to conclude by offering some of our insights on how to make other development projects as rewarding as ours has been.

### **1. Have a Process**

- Established milestones, well-understood deliverables, and a time table is crucial to managing the design development process.
- Clear roles (e.g. HI Architect, Interaction Designer, Visual Designer, Usability Specialist) and clearly assign responsibility or "ownership" for a portion of the design.

### **2. Keep Comprehensive Documentation**

- If the product design isn't clearly specified, it will be open to interpretation.
- The specification can become a living focus for design. If done correctly, it will outlive individuals and product versions.
- It is important to document your rationale for design -- you will need to justify decisions.
- It is as important to document ideas that were rejected and why, as it is to document the ideas you accept.
- Anything you can build as a designer (prototypes, resources, etc.) will help ensure that your design is understood.
- Documentation relieves you of the burden of remembering every detail. For example, had we completed the Conceptual Model Issues diagram (shown in figure 3) we would have spent far less time recalling forgotten details.

### **3. Have a Team Process and a Focus for Control**

- Having a team composed of a broad set of disciplines can bring needed diverse perspectives on design.
- It was helpful that team members had experience on all the target platforms. We made it a goal early in our project to actively recruit members from partner companies to complement our experience.
- Having this expertise on target platforms early in the process helped us avoid single-platform bias in design.
- Distance made it difficult to collaboratively design. One team was responsible for leading/proposing and other teams for ratifying (with changes and suggestions for improvement).
- When working with other members of the project team, e.g. engineering, it was important that we were viewed as members of the team, not as consultants to the team.
- We found that anyone can have a good idea -- our job as designers are to find the ones that will work.
- Apple's overall reputation for HI design skills made it easier for our design team to lead, and kept us accountable for living up to the reputation.
- An important company objective should be to perform ongoing customer research to find requirements we've missed or misunderstood, including opportunities to refine our products.

- Existing research will never provide all the information you need, but it can reduce your time-to-market and your costs.
- There has to be a “user” in user-centered design — taking time to understand your target users and their needs will allow you to build a product that meets those needs without guessing.

#### 4. User Studies

We also learned some things about user studies and what they can and cannot be expected to accomplish. They are:

- Study results cannot be used to say what a design should be, only that a design doesn't work.
- Expect imperfect user studies! It is often impossible to construct a good testbed.
- Expect to make mistakes doing a study, but be prepared to iterate the process to catch mistakes.
- Look for unexpected results! We were often looking for information about one component of the design and learned coincidentally about unrelated components.

Of course, it is impossible to do an infinite number of studies. So, there will always be some uncertainty about the results. (That's why we have follow-up releases.)

#### GLOSSARY

We've used a few terms that may be unfamiliar. Here are a few short definitions:

**Developer:** When we speak of developers, we are primarily referring to third-party software developers who use our technology platform to develop components which could be sold to customers. The secondary developer target is fourth-party developers, who may package components from third-party developers as customized solutions for specific customers.

**Evangelist:** A marketing specialist who serves as a point of contact between developers and Apple. Often they help developers adopt a new technology, such as OpenDoc.

**Inside-Out Activation:** Inside-out activation refers to the model for activation of an embedded part via a single click. For example, consider a table embedded within this text document. The user positions the pointer inside the table and clicks once. If the user can immediately begin editing, this is inside-out activation. By contrast, in the outside-in activation model when the user clicks inside the table the table is selected. Some other gesture, such as a double-click, is necessary to edit within the table. This becomes a major problem when the user cannot see the structure, and therefore cannot know if one, two, or more clicks are required before she can edit the desired content.

**Part:** The fundamental building block in OpenDoc. All parts have content and properties. Parts may be represented on the display screen as either icons or frames.

**Part Editor:** The software needed to display and edit a part. Analogous to an application.

**Linking:** A link is a relationship between two parts or pieces of part contents. A Link operation is a modified Copy in that the copy is updated every time the original changes. A link allows part contents to appear to be in two different places, even though it is only stored once. For instance, a document might link in a picture from another document, or a bar chart may display some data which is linked in from a row of a spreadsheet in the same document.

Links may be within a part, between two parts in the same document, or between two parts in different documents. They may even span networks. Both part contents and entire parts may be linked. Links are one-way only.

**Scripting:** OpenDoc allows users to use scripts to automate tasks. Scripting is one type of end-user programming.

#### ACKNOWLEDGMENTS

We thank our colleagues on the OpenDoc team, especially Kurt Piersol and Jed Harris, the architects of OpenDoc; also, David C. Smith who wrote the first Conceptual Model paper; and Dan Jordan who co-authored the *Design Rationale*, an Apple internal document (some of which was reused in this paper). Both David and Dan were members of the Apple HI team, along with (in chronological order) Mark Stern, Stephanie Guerlain, George Corrick, Jeremy Hewes, Jennifer Chaffee, Sue Bartalo, Kristin Bauersfeld, Michael Thomas, Per Nielsen, Jeff Kreeger, Kerry Ortega, Geoff Schuller and Katie Candland. Principal IBM HI designers included, Scott Isensee, Shirley Martin, Dick Berry and Dave Roberts.

We also acknowledge early OpenDoc-related work at Apple performed by Annette Wagner, Frank Ludolph, Lee Honigberg, Scott Jenson, John Sullivan and Gitta Salomon.

#### 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.

Xerox® is a trademark of Xerox Corporation.

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

#### REFERENCES

1. Moran, T. P., Carroll, J. M. (Eds.) (1996). *Design rationale: concepts, techniques, and use*. Lawrence Erlbaum Associates, Inc.
2. MIL-STD 1472C (1984). *Human engineering design criteria for military systems, equipment, and facilities*. US Dept. of Defense, 2 May 1981 (including Notice 1, 1 September 1983 and Notice 2, 10 May 1984).
3. Smith, D. C., Irby, C., Kimball, R., Verplank, W., & Harslem, E (April 1982). "Designing the Star User Interface." *Byte*, pp. 242-282.

4. Apple Computer, Inc. (1995). OpenDoc Programmer's Guide. NY, NY: Addison-Wesley.

5. Apple Computer, Inc. (1992). Macintosh Human Interface Guidelines. NY, NY: Addison-Wesley.