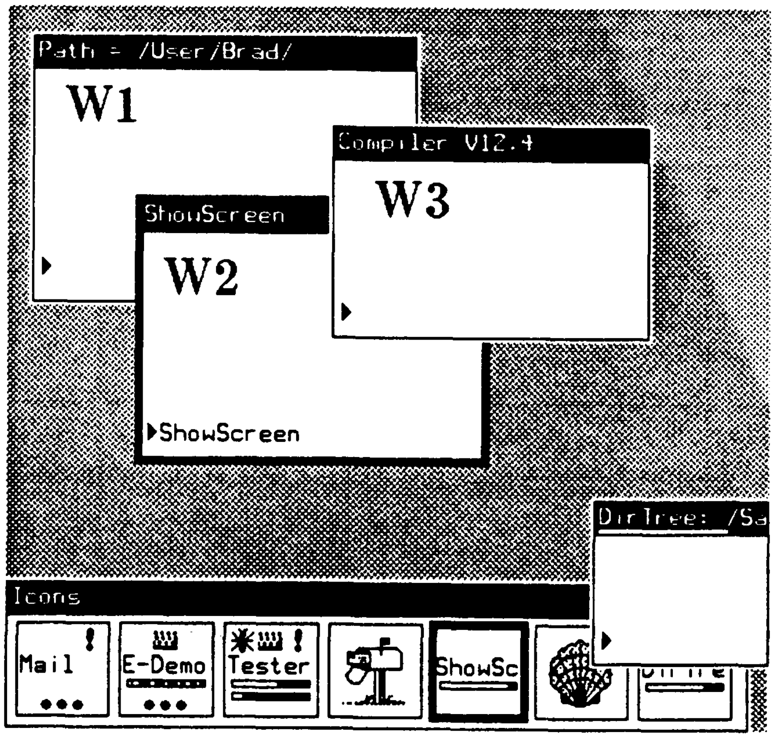
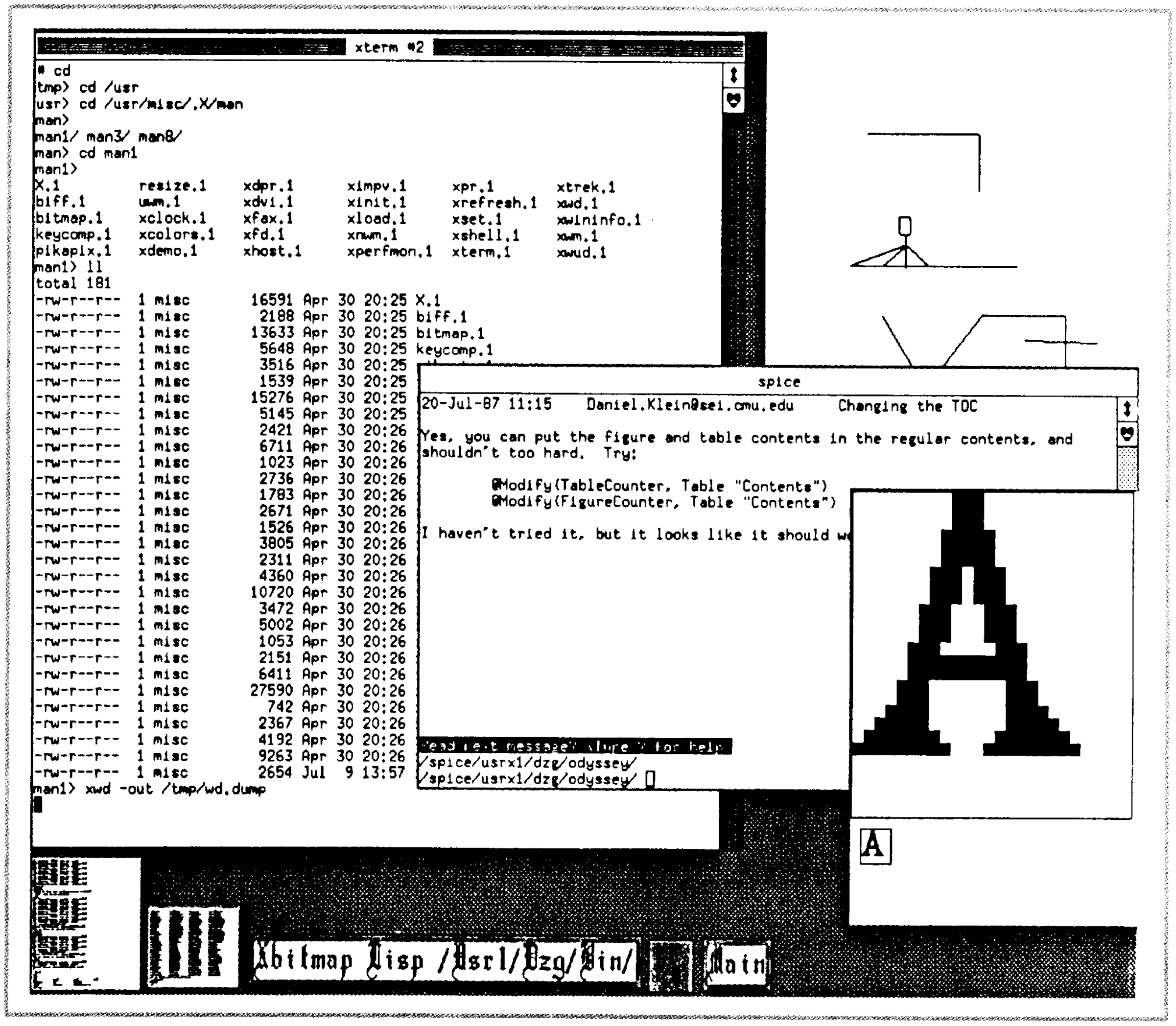
# A Taxonomy of Window Manager User Interfaces

Brad A. Myers
Carnegie Mellon University



This article presents a taxonomy for the user-visible parts of window managers. It is interesting that there are actually very few significant differences, and the differences can be classified in a taxonomy with fairly limited branching. This taxonomy should be useful in evaluating the similarities and differences of various window managers, and it will also serve as a guide for the issues that need to be addressed by designers of future window manager user interfaces. The advantages and disadvantages of the various options are also presented. Since many modern window managers allow the user interface to be customized to a large degree, it is important to study the choices available.

A window manager is a software package that helps the user monitor and control different contexts by separating them physically onto different parts of one or more display screens. At its simplest, a window manager provides many separate terminals on the same screen, each with its own connection to a time-sharing computer. At its most advanced, a window manager supports many different activities, each of which uses many windows, and each window, in turn, can contain many different kinds of information including text, graphics, and even video. Window managers are sometimes implemented as part of a computer's operating system and sometimes as a server that can be used if desired. They



```
xterm #2
# cd
tmp> cd /usr
usr> cd /usr/misc/.X/man
man>
man1/ man3/ man8/
man> cd man1
man1>
X.1          resize.1    xdpr.1      ximpv.1     xpr.1       xtrek.1
biff.1       uwm.1       xdvi.1      xinit.1     xrefresh.1  xwd.1
bitmap.1     xclock.1    xfax.1      xload.1     xset.1      xwininfo.1
keycomp.1    xcolors.1   xfd.1       xnwm.1      xshell.1    xwm.1
pikapix.1    xdemo.1     xhost.1     xperfmon.1  xterm.1     xwud.1
man1> ll
total 181
-rw-r--r--  1 misc        16591 Apr 30 20:25 X.1
-rw-r--r--  1 misc         2188 Apr 30 20:25 biff.1
-rw-r--r--  1 misc        13633 Apr 30 20:25 bitmap.1
-rw-r--r--  1 misc         5648 Apr 30 20:25 keycomp.1
-rw-r--r--  1 misc         3516 Apr 30 20:25
-rw-r--r--  1 misc         1539 Apr 30 20:25
-rw-r--r--  1 misc        15276 Apr 30 20:25
-rw-r--r--  1 misc         5145 Apr 30 20:25
-rw-r--r--  1 misc         2421 Apr 30 20:26
-rw-r--r--  1 misc         6711 Apr 30 20:26
-rw-r--r--  1 misc         1023 Apr 30 20:26
-rw-r--r--  1 misc         2736 Apr 30 20:26
-rw-r--r--  1 misc         1783 Apr 30 20:26
-rw-r--r--  1 misc         2671 Apr 30 20:26
-rw-r--r--  1 misc         1526 Apr 30 20:26
-rw-r--r--  1 misc         3805 Apr 30 20:26
-rw-r--r--  1 misc         2311 Apr 30 20:26
-rw-r--r--  1 misc         4360 Apr 30 20:26
-rw-r--r--  1 misc        10720 Apr 30 20:26
-rw-r--r--  1 misc         3472 Apr 30 20:26
-rw-r--r--  1 misc         5002 Apr 30 20:26
-rw-r--r--  1 misc         1053 Apr 30 20:26
-rw-r--r--  1 misc         2151 Apr 30 20:26
-rw-r--r--  1 misc         6411 Apr 30 20:26
-rw-r--r--  1 misc        27590 Apr 30 20:26
-rw-r--r--  1 misc          742 Apr 30 20:26
-rw-r--r--  1 misc         2367 Apr 30 20:26
-rw-r--r--  1 misc         4192 Apr 30 20:26
-rw-r--r--  1 misc         9263 Apr 30 20:26
-rw-r--r--  1 misc         2654 Jul  9 13:57
man1> xwd -out /tmp/wd.dump
```

```
spice
20-Jul-87 11:15    Daniel.Klein@sei.cmu.edu     Changing the TOC

Yes, you can put the figure and table contents in the regular contents, and
shouldn't too hard.  Try:

        @Modify(TableCounter, Table "Contents")
        @Modify(FigureCounter, Table "Contents")

I haven't tried it, but it looks like it should w

Read next message? (type ? for help)
/spice/usrx1/dzg/odyssey/
/spice/usrx1/dzg/odyssey/
```

**Figure 1. An example of a typical screen using the X window manager[4] with overlapping windows. Some windows have title lines (the top-left window's says "xterm #2"). The background, where there are no windows, is gray. The small windows at the bottom are icons.**

can even be implemented by individual application programs or programming environments.

Window managers have become popular primarily because they allow separate activities to be put in physically separate parts of the computer screen. The user of a computer is frequently shifting focus from one activity to another, including such small shifts as changing from editing one file in a text editor to editing another, and such large context shifts as changing from compiling a program to reading mail.

Before window managers, people had to remember their various activities and how to switch back and forth. Window managers allow each activity to have its own separate area of the screen (its own "window"). Switching from one window to another is usually very simple. This physical separation is even more important when the operating system allows multiple activities to operate at the same time ("multiprocessing"). For example, in Unix, the user can compile one file at the same time a different file is being edited. On a conventional terminal, if the compiler process outputs any data, it is confusingly interspersed with the editor's display. If the two processes request input at the same time, the user may give the input to the wrong program. Window managers help with these problems by providing separate areas in which each process can perform input and output.

Another advantage of window managers is that they provide a higher level interface to the mouse, keyboard, and screen, and therefore can support much higher quality user interfaces. For example, the window managers on the Star[1,2] and Macintosh[3] help support the metaphor that using the computer is like doing operations on a physical desk. This higher level interface can also make application code more portable from one machine to another, since the same window manager procedural interface can be provided on different machines. This was an important motivation for the development of the X window manager.[4]

Today there are a large number of window managers in existence from many companies and research groups, and more are being created all the time. In surveying these window managers, it became clear that there are many similarities between all of them, and the differences can be characterized on a small number of different axes. (This survey was started at the Alvey MMI Workshop on Window Management.[5]) Most of the ideas seem to have originated at the Xerox Palo Alto Research Center, including windows in general (Smalltalk[6]), pop-up menus,[7] icons (Tajo[8,9] and Star[1,2]), and tiled windows (Cedar[10,11]). Of course every window manager has its own original aspects, but most of the important features of the user interfaces of window managers do not seem to vary markedly.

With the advent of the X window manager,[4] which is rapidly becoming a de facto standard, the study of the user interface component is becoming more critical. This is because X and many other modern window managers allow the user interface to be changed, while still maintaining the same application interface. User interface designers therefore are faced with not only a choice for the user interface of their application, but also for that of the window manager. It is therefore important to focus on the different choices in the user interface component of window managers. This article presents a taxonomy of the choices used in existing window manager user interfaces, along with some advantages and disadvantages of each choice.

(At the time of this writing, Xerox, AT&T, and Sun had just announced a portable window-manager user interface called "Open Look," which apparently will be implemented on multiple-window managers, including X[4] and NeWS.[12] Open Look, which is based partially on the user interface of the Xerox Star, is designed to match the ease of use of the Macintosh, and thereby make Unix systems more user friendly.)

## Definition of terms

The previous section defined "window managers" and discussed the reason they are so popular. This section defines some related terms that are important for understanding how window managers work.

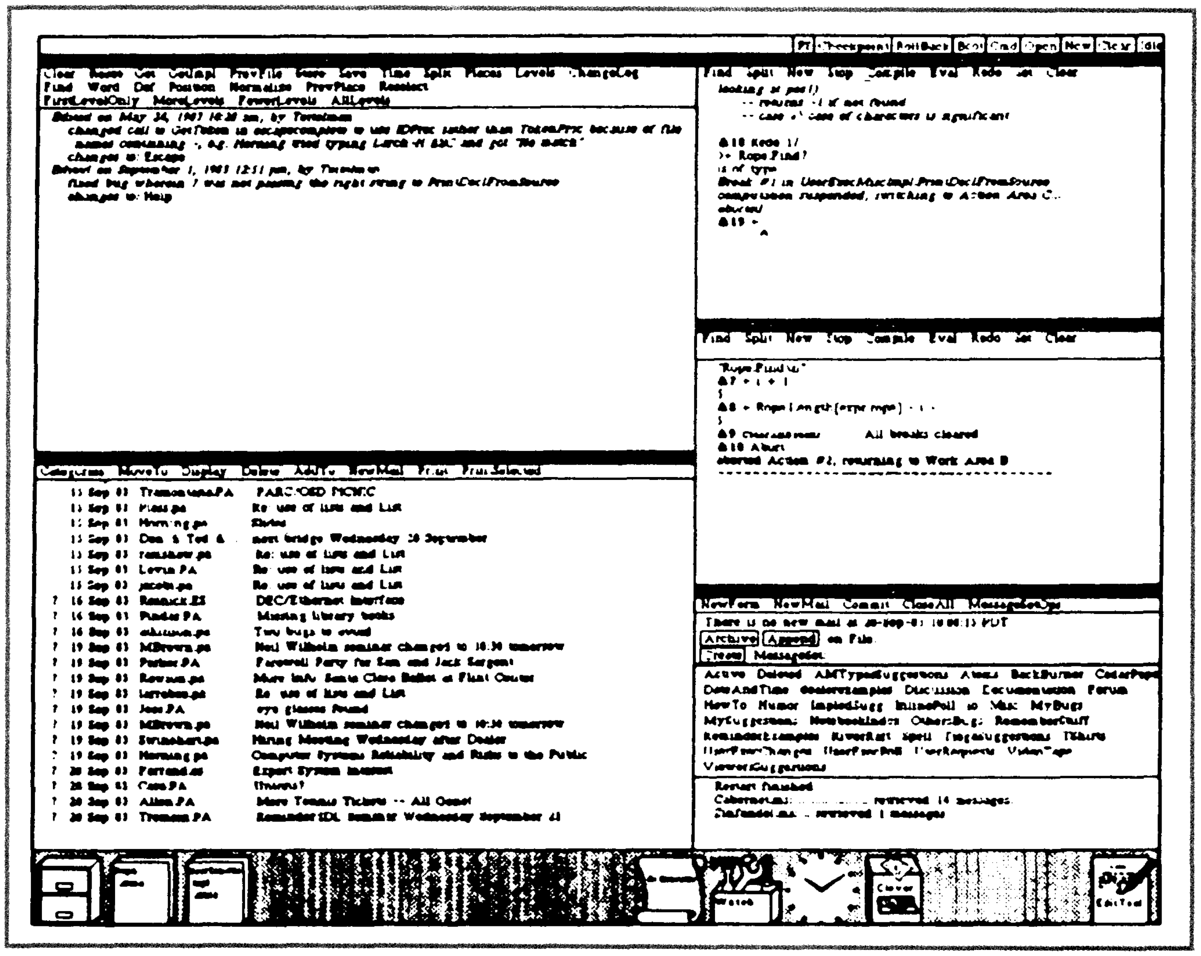
A window manager can be logically divided into two layers, each of which has two parts. The base layer implements the basic functionality of the window manager. The two parts of this layer handle the display of graphics in windows and access to the various input devices (usually a keyboard and a pointing device such as a mouse). The primary interface of this layer is to other programs, and it is called the window manager's *application* or *program interface*. The base layer is not discussed further in this article. The other layer of window managers is the *user interface*. This includes all aspects that are visible to the user. Sometimes the base layer is called a *window system*, reserving the name "window manager" for the user interface layer. Since this article deals only with the user interface layer, the term "window manager" is used here.

The two parts of the user interface layer are the *presentation*, which is composed of the pictures that the window manager displays, and the *operations*, which are the commands the user can give to manipulate the windows and their contents. Figure 1 shows windows that demonstrate different aspects of the presentation, including patterns or pictures for the area where there are no windows, title lines and borders for windows, etc. Examples of the operations that may be provided for windows include moving them around on the screen and specifying their size.

One very important aspect of the presentation of windows is whether they can *overlap* or not. Overlapping windows, sometimes called *covered* windows, are a feature allowing a window to be partially or totally on top of another window, as shown in Figure 1. This is also sometimes called the *desktop metaphor*, since windows can cover each other like pieces of paper can cover each other on a desk. (There are usually other aspects to the desktop metaphor, however, such as presenting file operations in a way that mimics office operations, as in the Star office workstation.[1,2]) The other alternative is called *tiled* windows, which means that windows are not allowed to cover each other. Figure 2 shows an example of tiled windows. The advantages and disadvantages of each are discussed below. Obviously, a window manager that supports covered windows can also allow them to be side by side, but not vice versa. Therefore, a window manager is classified as covered if it allows windows to overlap.

Another important aspect of the presentation of windows is the use of *icons*. These are small pictures that represent windows. They are used because there would otherwise be too many windows to conveniently fit on the screen and manage easily. When a window is not in use, it can be removed and replaced with its icon, and later conveniently retrieved when needed. Figure 3 shows examples of icons from some different window managers. The section on icons discusses the options available for icons in more detail.

An important aspect of window managers is how the user changes which window is connected to the key-


PT Checkpoint RollBack Boot Cmd Open New Clear Idle
Clear Reset Get GetImpl PrevFile Store Save Time Split Places Levels ChangeLog
Find Word Def Position Normalize PrevPlace Reselect
FirstLevelOnly MoreLevels FewerLevels AllLevels
Find Split New Stop Compile Eval Redo Set Clear
Find Split New Stop Compile Eval Redo Set Clear
Categories MoveTo Display Delete AddTo NewMail Print PrintSelected
NewForm NewMail Commit CloseAll MessageSetOps
Archive Append on File
Create MessageSet
Active Deleted AMTypeSuggestions Atoms BackBurner CedarPages
DateAndTime Discussion Documentation Forum
HowTo Humor ImplediSugg InlinePoll io Misc MyBugs
MySuggestions OtherBugs RememberStuff
ReminderExamples Spell TiogaSuggestions
ViewerSuggestions UserRequests VideoTape
Watch
Clover
EditTool

```
Edited on May 24, 1983 10:28 am, by Teitelman
   changed call to GetToken in escapecomplete to use IDProc rather than TokenProc because of file
    names containing -, e.g. Horning tried typing Larch-H.BAL and got "No match"
   changes to: Escape
Edited on September 1, 1983 12:51 pm, by Teitelman
   fixed bug whereby ? was not passing the right string to PrintDeclFromSource
   changes to: Help
```

```
looking at pos!)
     -- returns -1 if not found
     -- case => case of characters is significant

&18 Redo 17
>+ Rope.Find?
is of type
Break #1 in UserExecMiscImpl.PrintDeclFromSource
computation suspended, switching to Action Area C...
aborted
&19 +
```

```
"Rope.Find\n"
&7 + i + 1
5
&8 + Rope.Length[rope: rope] - i -
5
&9 ClearAllBreaks        All breaks cleared
&10 Abort
aborted Action #2, returning to Work Area B
---------------------------------------------
```

```
    15 Sep 83  Tramontana.PA     PARC-OSD PICNIC
    15 Sep 83  Plass.pa          Re: use of lists and List
    15 Sep 83  Horning.pa        Slides
    15 Sep 83  Dan & Ted & ..    next bridge Wednesday 28 September
    15 Sep 83  ramshaw.pa        Re: use of lists and List
    15 Sep 83  Levin.PA          Re: use of lists and List
    15 Sep 83  jacobs.pa         Re: use of lists and List
?   16 Sep 83  Rennick.ES        DEC/Ethernet interface
?   16 Sep 83  Pundar.PA         Missing library books
?   16 Sep 83  atkinson.pa       Two bugs to avoid
?   19 Sep 83  MBrown.pa         Neil Wilhelm seminar changed to 10:30 tomorrow
?   19 Sep 83  Parker.PA         Farewell Party for Sam and Jack Sargent
?   19 Sep 83  Rowan.pa          More info: Santa Clara Ballet at Flint Center
?   19 Sep 83  larrabee.pa       Re: use of lists and List
?   19 Sep 83  Jean.PA           eye glasses found
?   19 Sep 83  MBrown.pa         Neil Wilhelm seminar changed to 10:30 tomorrow
?   19 Sep 83  Swinehart.pa      Hiring Meeting Wednesday after Dealer
?   19 Sep 83  Horning.pa        Computer Systems Reliability and Risks to the Public
?   20 Sep 83  Ferrand.es        Expert System interest
?   20 Sep 83  Cain.PA           Tharsis?
?   20 Sep 83  Allen.PA          More Tennis Tickets -- All Gone!
?   20 Sep 83  Truman.PA         Reminder: IDL seminar Wednesday September 21
```

```
There is no new mail at 20-Sep-83 10:00:15 PDT
```

```
Restart finished
Cabernet.ms: ............ retrieved 14 messages.
Zinfandel.ms: .. retrieved 1 messages
```

Figure 2. A screen from the Cedar[10,11] window manager. Windows are "tiled" into two columns. There is a row of icons along the bottom. Each window has a fixed menu of commands below the title line.

board. Although there will typically be multiple windows, there is usually only one keyboard for each user. Therefore, only one window at a time can be attached to the keyboard. This window is termed the *listener*, since it is listening to the user's typing. Another term for this window is the *input* (or *keyboard*) *focus*. Older systems called the listener the "active window" or "current window," but these are poor terms, since in a multiprocessing system, many windows can be actively outputting information at the same time. Window managers provide various ways to specify and show which window is the listener.

Most window managers use some form of pointer, which is an input device that returns a 2D value used to identify locations on the screen. Pointing devices are typically used for specifying window size and position, for selecting characters in an editor, for drawing lines in a graphics program, and for transferring a picture (such as a map) into the computer by specifying points (this last use is called *digitizing*). Examples of pointing devices are light pens, electromagnetic tablets with pucks or pen-like styli, touch-sensitive surfaces (touch tablets or touch screens), trackballs, and mechanical or optical mice.[14] Since the most popular pointing device for window managers is a mouse, the term *mouse* will often be used in this article to mean *pointing device*.

Light pens and touch screens are used for pointing directly at the screen, but with the other types the user moves a device on the desk or on a special surface, and a small picture, called the *tracking symbol* or *cursor*, follows the movement on the screen. In many window managers the picture for the cursor can be changed, but a common picture is an arrow pointing to the upper left.

Pointing devices usually have one or more buttons. For example, there are typically one to three buttons on the top of a mouse. Some window managers allow the user

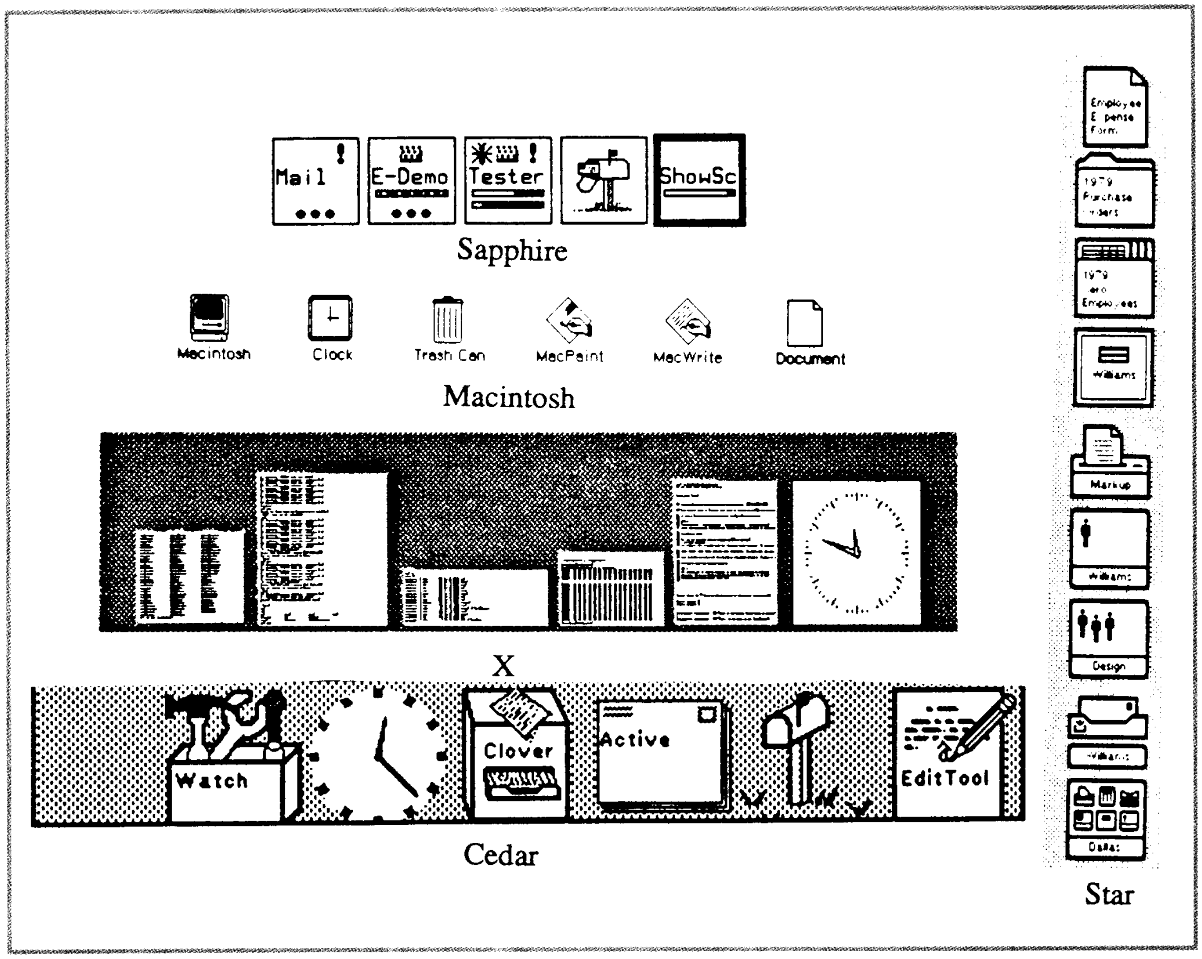


**Figure 3. Examples of icons from different systems: Sapphire,[13] Macintosh,[3] X,[4] Cedar,[10] and Star.[1] Some of the X icons contain the actual text displayed in the window in a tiny (unreadable) font.**

to press two or three times quickly to specify additional commands. This is called *multiclicking* (for example, pressing twice quickly is double-clicking). Window managers may also support holding down keyboard keys (such as the shift key) while pressing a mouse button. This is often used to modify the button's meaning.

## The window manager versus add-ons

To compare window managers, it is first necessary to establish the boundaries of discussion. A window manager provides the basic service of managing different windows on the screen, as defined above. In many systems, however, other services are also provided, and these are often classified as part of the window manager. By providing these services in a central place, the system promotes consistency and makes applications easier. To compare the window managers of these different systems, however, it is important to classify which aspects are being compared and which are considered add-on services. This section discusses some of these add-ons so that the rest of the article can concentrate on the window manager portion itself.

Some common add-ons are the following:

1. a typescript package (handles user typing)
2. entire editors
3. a graphics package for output (also called the *imaging model*
4. menus of various kinds
5. forms (also called *dialogue boxes*)
6. scrolling mechanisms
7. general tool kits (which usually include menus, forms, and scrolling mechanisms)

# Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

## Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

## Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

## Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

## API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

**LAW FIRMS**

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

**FINANCIAL INSTITUTIONS**

Litigation and bankruptcy checks for companies and debtors.

**E-DISCOVERY AND LEGAL VENDORS**

Sync your system to PACER to automate legal marketing.