CDE Infrastructure

By George Kraft IV

The programming infrastructure, not its productivity tools, is a major strength of the Common Desktop Environment. This article discusses the APIs and desktop services that are benefiting developers and ISVs.

When Project Athena at Massachusetts Institute of Technology® (MIT®) introduced the X Window System® (X), they established the mechanism for their distributive "open" windowing protocol. Yet, they intentionally left out the policy for others to develop. Later, based on technology from Hewlett-Packard® (HP™) and Digital Equipment Corporation, the Open Software Foundation® (OSF®) developed the policies of the Motif® Graphical User Interface (GUI) for X. The Motif style guide established strict guidelines for its human-centric windowing behaviors; however, the UNIX desktop was still left without services.

The UNIX System Laboratory licensed the Destiny Desktop with System V.4, HP developed HP Vue for HP-UX, and IBM shipped IXI's X.desktop for AIX. Although these desktops provided services not offered in X and Motif GUI, they caused a divergence in the open systems market. The cost of integration and the need to remain portable kept many developers from migrating to these desktops.

The Common Desktop Environment

The Common Desktop Environment (CDE) was the result of synergy between HP, IBM, Sun, and Novell® to establish a unified open system desktop. They took the "best of breed" technology from their existing environments, then designed, implemented, and delivered a new level of UNIX desktop services on which developers could rely.

The Desktop's Programming Infrastructure

The desktop's Application Programming Interfaces (APIs) go beyond the basic Motif GUI. CDE provides data-typing, object methods, printing, drag-and-drop, additional widgets, plug-and-play messaging, and a hypertext help system. This new set of system services at the desktop level is guaranteed for all CDE-compliant systems. Developers can now produce more consistent applications and reduce the number of developer-specific core solutions.

Motif GUI

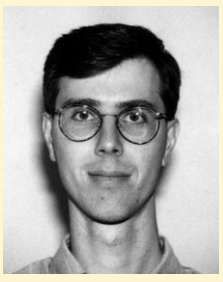
Motif won the open system GUI wars; however, vendors have serviced and enhanced various releases of Motif. Although Motif became the GUI standard, portability became risky if any of the vendors' embellishments were used. Fortunately, the creators of CDE helped eliminate those problems by merging their Motif source bases to re-establish a stock GUI.

Desktop Services

CDE goes beyond the GUI behavior to provide desktop-wide objects and methods for applications to use and call upon. Applications no longer need to depend on old and limited databases like the mime-types and mailcap that are used by applications such as mailers and Web browsers.

Figure 1 shows a message dialogue program that initializes itself with the desktop and loads the desktop's data-type and method database. Then, it simply creates a message dialogue and prompts the user.

When you select the E-mail button, the application calls the desktop's Compose method on the file object. The desktop spawns the mailer with the file object, where it is eventually displayed and ready to be addressed.



George Kraft IV

```
#include <stdlib.h>
#include <Xm/MessageB.h>
#include <Dt/Dts.h>
#include <Dt/Action.h>

void emailCB(Widget, XtPointer, XtPointer);

main(argc, argv)
        int     argc;
        char    **argv;
{
        Widget  topLevelShell, send, help, cancel;
        Arg     xargs[10];
        int     n;
        XmString title, greet, email;
        char *file = 2 == argc ? argv[1] : "/etc/motd";
        char *description;

        topLevelShell = XtInitialize(argv[0], "DtSend",
                NULL, 0, &argc, argv);

        /* CDE Initialization */

        DtInitialize(XtDisplay(topLevelShell), topLevelShell,
                argv[0], "DtSend");

        DtDbLoad();

        send = XmCreateMessageDialog(topLevelShell, "send", NULL, 0);
         /*Get CDE's "DESCRIPTION" of the file*/
        description = DtDtsFileToAttributeValue(file, "DESCRIPTION");

        title = XmStringCreateSimple("DtSend");
        greet = XmStringCreateLtoR(description);
        email  = XmStringCreateSimple("Email");

        XtVaSetValues(send,
                XmNdialogTitle,       title,
                XmNmessageString,     greet,
                XmNhelpLabelString,   email,
                NULL);

        XmStringFree(title);
        XmStringFree(greet);
        XmStringFree(email);

        cancel = XmMessageBoxGetChild(send, XmDIALOG_CANCEL_BUTTON);
        XtUnmanageChild(cancel);

        XtAddCallback(send, XmNokCallback, exit, NULL);
        XtAddCallback(send, XmNhelpCallback, emailCB, file);

        XtManageChild(send);

        XtMainLoop();
}
```

(continued on next page)

Figure 1. Simple CDE GUI application

(continued from previous page)

```
void
emailCB(w, client_data, call_data)
        Widget w;
        XtPointer client_data;
        XtPointer call_data;
{
        DtActionInvocationID actionId;
        DtActionArg actionArgs[] = { DtACTION_FILE, (char *)client_data };

        actionId = DtActionInvoke(XtParent(w),
                "Compose",              /* action */
                actionArgs, 1,          /* action arguments & count */
                (char *) NULL,          /* terminal options */
                (char *) NULL,          /* execution host */
                (char *) NULL,          /* context directory */
                True,                   /* "use indicator" */
                NULL, NULL);            /* action callback & client data */
}
```

Figure 1. Simple CDE GUI application

Drag-and-Drop

The desktop provides a convenient and consistent drag-and-drop API for interpreting data transfer across the desktop. Text, file names, and buffers can be transferred from the dragged icon to the drop zone. The type of data being dragged determines the drag icon's appearance and configuration. Since Motif can distinguish the different types of data, applications have a more robust drag-and-drop behavior.

Desktop GUI

The desktop's `DtWidget` library helps bridge the current gap between CDE's converged Motif and Motif 2.0. Developers do not need to wait for Motif 2.0 because the spin box, menu button, and editor widgets are in CDE. As always, to ensure binary compatibility, developers should take special care to use `XmResolvePartOffset` when subclassing from one widget to make another; otherwise, an updated shared library could cause unpredictable results.

Help

CDE provides standard help APIs and GUI dialogues, but it also delivers a feature-rich SGML DTD compared with the HTML used on the World Wide Web. CDE's help links support hypertext, definition, man page, execution, and application-define links.

CDE documents are pre-processed for quicker loading. Since these binary formatted documents cannot be read by a person, one of their added benefits is that they cannot be reverse engineered when copied, which prevents copyright infringements. Publishers who provide documents in HTML format are at a disadvantage, because complete unabridged duplicates can be made from most browsers.

ToolTalk Plug-and-Play

CDE's ToolTalk® is a message brokering system that enables applications to communicate with each other without having direct knowledge of one another. Application clients and servers can be developed independently, mixed and matched, and upgraded independently through plug-and-play.

Applications registered to handle message requests act as servers for applications that broadcast their requests. Message brokering is an evolutionary step beyond file sharing, peer-to-peer, and ICCCM inter-client communication.

Graphical Korn Shell

The Graphical Desktop Korn Shell provides much of the desktop's Motif GUI, services, help, workspace management, session management, and ToolTalk plug-and-play. Developers can prototype and deploy with the standard `ksh93`

scripting language. This means that small to moderate-sized programs can be written, then interpreted on any CDE-compliant system without any additional work.

The dtksh shell script in Figure 2 is a conversion of the C program that was illustrated earlier. Unlike the popular Tcl/Tk shell and GUI, dtksh has nearly a one-to-one migration path to native Motif for performance. With dtksh, code can be easily migrated and developers find that their knowledge transfers easily between C and dtksh.

The Desktop's Infrastructure

CDE not only provides a new set of Motif, Drag-and-Drop, Desktop Widget, Help, ToolTalk, and DtKsh APIs, but it also provides system services in which applications can participate and follow. The desktop services provide the login manager, session manager, color server, workspace manager, and ToolTalk server. It is tempting to provide these features in large software suites; however, if developers try to mimic these desktop services, precious development time and energy is taken away from creating the actual products.

Login Manager

The desktop login manager provides the basic X Display Manager Protocol (XDMCP) to manage login sessions for X terminals on the network and workstations on the desktop. The login manager starts up the X server on the bitmap display; it also initiates the session manager.

Session Manager

The session manager uses a set of conventions and protocols that enable the desktop to save and restore a user's session from one login session to the next. Using the session manager, users can also configure a set of sessionetc and sessionexit scripts to be called when logging

```
#!/usr/dt/bin/dtksh

editCB()
{
        dtaction Compose $FILE
}

main()
{
        XtInitialize TOPLEVEL dtSend DtSend "$@"

        DtDbLoad

        DtDtsFileToAttributeValue DESC $FILE "DESCRIPTION"

        XmCreateMessageDialog SEND $TOPLEVEL motd \
                dialogTitle:"DtSend" \
                helpLabelString:"Email" \
                messageString:"${DESC}"

        XmMessageBoxGetChild CANCEL $SEND DIALOG_CANCEL_BUTTON
        XtUnmanageChild $CANCEL

        XtAddCallback $SEND okCallback exit
        XtAddCallback $SEND helpCallback editCB

        XtManageChild $SEND

        XtMainLoop
}

if [ $# -eq 1 ]; then
        FILE=$1
else
        FILE="/etc/motd"
fi
```

Figure 2. Simple DtKsh GUI application

```
dpy  = XtDisplay(topLevel);

save = XmInternAtom(dpy, "WM_SAVE_YOURSELF", False);

XmAddWMProtocols(topLevel, &save, 1);

XmAddWMProtocolCallback(topLevel, &save, saveProc, topLevel);
```

Figure 3. WM_SAVE_YOURSELF setup

in and exiting respectively. This enables user-defined tasks to be performed at login and logout.

The key responsibility of the application is to acknowledge the WM_SAVE_YOURSELF message when the desktop is being shut down by the user, as shown in Figure 3.

The WM_SAVE_YOURSELF saveProc routine tidies up for the application, then sets the WM_COMMAND property to be saved and reused later by the session manager to restart the terminated applications.

Color Server

The session manager acts as the color server that controls foreground and shadow colors, limits color use, and restricts the creation of colors in the colormap. Using applications that conform to the color server reduces the depletion of the colormap and coordinates the color scheme of the desktop. If the colormap does become depleted, an "unsocial" application is usually lurking somewhere.

Workspace Manager

The desktop window manager (dtwm) serves as the default window manager for the desktop and extends the capabilities of the Motif window manager. It provides a control panel for the desktop to launch applications. Its multiple screens, or workspaces, allow users to switch between screens.

Desktops are often considered mutually exclusive end-point solutions, such as Web browsers or collaborative software suites. However, CDE views them as application groups or workspaces being managed and serviced by the desktop infrastructure. The desktop window manager is ideal for managing workspaces for Internet suites and collaborative tools.

ToolTalk Server

The ToolTalk server can be relied upon to broker client and server messages for inter-client plug-and-play. The Common Object Request Broker Architecture (CORBA) movement has a great deal of strength behind it. But ToolTalk is lightweight, does the job, and forms an integral part of CDE, which is deployed across a gamut of UNIX platforms.

ToolTalk takes message-handling registrations from method servers, then holds that information until client applications broadcast for those registered services. Platform gateways are not needed, because ToolTalk interoperates between heterogeneous systems.

Go For It...

There is much more to CDE than meets the eye. Application and workspace developers alike have a rich feature-filled infrastructure upon which to build. Just like in the past, we still rely on the tried and true X and Motif; however, now we can count on the Common Desktop Environment for its development libraries and desktop management infrastructure.

If you are getting ready to write a new application or just thinking about sprucing up something that you have been working on, consider how your application can become more feature rich and desktop friendly for less code.

For a list of CDE reference materials, visit IBM's CDE Web page at URL http://www.austin.ibm.com/software/CDE/TOC/Further/further.html.

George Kraft IV, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Kraft is an advisory programmer for IBM's RS/6000 Division working in the AIX Desktop group. He is currently working on the Common Desktop Environment Version 2.1 and a version of the network computer. He has a BS in Computer Science and Mathematics from Purdue University.