OpenCDE Manifesto: Guiding Principles and Project Goals

Contributors

Aaron W. Hsu
arcfide@sacrideo

Karsten Pedersen
kpedersen@opencde.org

Purpose of this Document

OpenCDE has the potential to scale and grow as a large and comprehensive project. However, many projects loose sight and focus as they gain contributors and different bodies push with their own agendas. This document aims to provide a cohesive vision of the direction and intentions of OpenCDE developers with respect to where and how OpenCDE should move forward. In terms of scope, we intend to omit nothing of import, and this document has no requirements of being short. However, it ought to remain self consistent and broad enough to enable developer freedom where important without losing the focus of the project.

A History of CDE

Before the advent of CDE, UNIX vendors had no consistent desktop standard by which they could develop and share applications. There were many different toolkits and styles of programming, and no consistency among desktop interfaces. This lead to numerous problems. The OpenGroup describes CDE in the following terms:

> *The Common Desktop Environment (CDE) is an integrated graphical user interface for open systems desktop computing. It delivers a single, standard graphical interface for the management of data and files (the graphical desktop) and applications. CDE's primary benefits–deriving from ease-of-use, consistency, configurability, portability, distributed design, and protection of investment in today's applications–make open systems desktop computers as easy to use as PCs, but with the added power of local and network resources available at the click of a mouse.*[1]

Indeed, CDE was the standard among desktop environments in UNIX Workstations for some time, especially during the peak of commercial UNIX development. CDE had a reputation as the standard desktop, though it did not necessarily have a reputation as the best, most efficient, fastest, or the like. And especially, in the era of thousand dollar per seat licensing costs, it was not cheap. It was, in a sense, open. The standards on which it were based were open and unencumbered, and the basic protocols that it used were also fairly standard. However, in terms of the code itself, it was highly restricted. Gaining access to the source code was expensive, and it was even more expensive to obtain distribution rights.

Since many companies made their living on CDE contracts, this did not change even though the world changed around CDE. Indeed, the OpenGroup was able to open source Motif, the core toolkit upon which CDE is based, but it was not able to open source CDE. A petition is still outstanding to accomplish this, but the current expectation is the need for a large one time payment to buy the sources. To many, this is too little, too late. The world has moved on, in many ways, and users' expectations have also changed over the years, but CDE development has remained largely stagnant. It should be noted, however, that CDE was never at the cutting edge of desktop features. The point of CDE was stability, predictability, and consistency. Leveraging investment in existing knowledge and training was important. Leveraging existing application investments also played into the design decisions surrounding CDE.

[1] OpenGroup, *CDE*, http://www.opengroup.org/cde/.

In the modern era, CDE has largely grown irrelevant because it simply costs too much. Desktop environments like XFCE may have started as CDE lookalikes, but they have changed so dramatically that they no longer represent a close approximation to CDE, and they do not share the same goals as the original CDE project. Gnome and KDE were never designed with the goals of CDE in mind, and some would argue that this is a good thing.

Nonetheless, the OpenCDE project was started by Karsten Pedersen in an effort to recreate the glory of CDE, its niceties, without the associated drawbacks.

The Principles

The OpenCDE design principles are summarized below, roughly in order of descending importance. Note that while these principles summarize the situation, they do not represent ironclad commandments etched into stone, never to be altered, deviated from, or disregarded entirely.

1) Provide a modern integrated graphical desktop experience providing modern equivalents to traditional CDE end-user benefits including the following:
   - Standard graphical interface with consistent look and feel
   - Single graphical interface for the management of data and files
   - Single graphical interface for the management of applications
   - Easy-to-use[2]
   - Consistency
   - Configurability
   - Portability
   - Distributed Design
   - Protection of Investment (A.K.A. Stability, Compatibility)
   - Easy access to powerful resources
2) Provide high-performance, low-latency, low footprint computing without the associated loss in functionality.
3) Integrate with standards and best practices, rather than reinventing the wheel.
4) Integrate rather than segregate the desktop from the rest of UNIX.
5) Provide best-anywhere documentation, both at an user and at a developer's level.
6) Appeal to developers and power users
7) Emphasize quality, good design, and stability over rapid progress, feature creep, and desktop fashion.
8) Favor function and user efficiency over eye-candy

Clarifying Traditional CDE End-user Benefits

Standard Graphical Interface. At the time CDE was being designed, UNIX had many different graphical existed. Applications built on one toolkit would behave entirely differently to others, and there were little standards

[2] This does not necessarily mean intuitive.

among applications, either command line or graphical. One might be written in straight Xlib, another in OpenLook, and another in something else entirely. Command line options were not standard or consistent, and the style of each application's visual interface usually differed from another application significantly. Just look at some common X applications to see the situation even among applications written for a single Xlib toolchain. Xman(1), Xfontsel(1), Xmore(1), or Xfig(1). All of these have slightly different behaviors and it's not clear or intuitive what actions will do what on what systems.

CDE provided a solution to this by standardizing the look and feel of applications inside of a single standard: Motif and CDE style guides. The style guides of these two systems (CDE was built using Motif) converged in the 2.1 version of CDE, further improving CDE's ability to offer a consistent look and feel. It achieved this with the help of a layering of technologies. The Xlib libraries provided the basic connections to the X server, but on top of that, the Xt toolkit provided the basic infrastructure for object-oriented widget design, enabling sub-classing and attribute sharing among widgets. The Motif toolkit took advantage of Xt to build a standard set of widgets that emulated the interface styles and features that people expected coming from Personal computers.

Beyond just a look and feel, though, CDE provides access to common elements that the user would likely want to use. It enabled administrator's to more easily control the general look and feel of workstations across a computing grid, and enabled a consistent theme across international boundaries with internationalization.

Single interface for the management of data and files. CDE provided a common interface, both programmatic and user-level for managing files and data as it crossed application and network boundaries. At its simplest, it provided a graphical interface for the management of an user's file system, through a simple file manager. However, this file manager also spoke a communication protocol that enabled users to graphically send data from files and directories to other applications with a simple mouse action, rather than requiring complicated shell pipes and the like. Applications that spoke the data communication language of CDE could accept drag and drop elements from the file manager, and vice versa.

CDE's network services enabled this same easy transparency across applications and computers, so that server client code could communicate more easily, and file sharing became something that the user could accomplish with the mouse and pointer, rather than requiring the editing of complex programs and daemons under the hood.

This consistent programming interface meant that user's on the UNIX desktop now had access to drag and drop functionality that was akin to or more powerful than what was available on systems like the Apple Macintosh. Using a single interface and common, consistent desktop motions that an user could remember, data could easily be transmitted from one application to another through things like cut and paste, drag and drop, and automatic interprocess communication.

Management of applications. In the same way that CDE enabled consistent, predicatable data interchange between applications, it also enabled the user to have a consistent way of launching, accessing, and managing applications, both in terms of look as well as in functionality.

The most obvious element to this was the inclusion of the front panel. This was a configurable launch pad that enabled the user to set up different

application actions, launch the application of their choice, and even embed applications into the panel itself.

Further adding to this, CDE introduced the ability to create custom actions for applications, which was basically a way of starting up an application the way you wanted it, with the appropriate icons and other behaviors all in place. These actions where encapsulated into icons or objects in the file system that were displayed in the desktop or in the front panel. This prevented the user from needing to deal with things like aliases and internal shell scripts for simple things like launching graphical applications.

For system administrator's, it made it easier to install and manage applications at a multi-system level, so that applications would be available on multiple workstations over the network, and that they would have the same look and feel across those workstations.

CDE's integration technologies made it possible to easily integrate existing software into the CDE desktop, so that it would play nicely with the user's expectations, even if the application itself was not necessarily written with CDE in mind. OpenLOOK applications come to mind here.

CDE also provided session management, and a protocol to enable applications to save their own sessions (through standard X protocols) to preserve state between logins. The user would then be presented with the same desktop he logged out with in their previous session. More complex forms of session management were also possible.

As funny as it may sound today, the provision of an icon editor also made managing applications easier, because the user could create or import images and icons that they wanted to use for a given application, especially considering that some applications may not have had an icon associated with them in the first place. Today, this might not be considered such a special idea, because of the pervasive nature of icons and graphical applications, but this was something to be considered back then.

Easy to Use. CDE hailed itself as being easy to use. That is not to say that it was necessarily intuitive. Indeed, it could be said that any desktop environment is unintuitive to the user who is used to another. On the other hand, CDE did design itself so that user's from Personal Computers could easily transition their basic knowledge of computer operations onto the more powerful UNIX system without the learning curve usually associated with other desktop shells, such as the Korn Shell and the command line. The ease of use came in the form of a consistent interface that could be easily taught to people, and which worked reliably across systems. The graphical nature of the system, with its graphical configuration utilities, and other user-friendly aids, made it possible to operate a UNIX workstation in a more comfortable and efficient way.

System administrators also found the benefits to this in terms of easier application management and easier deployment of resources. It reduced training necessary to get users productive on the system, and enabled user's to move from one UNIX operating system to another without having to worry about the underlying quirks fo the new operating system, because CDE provided a common interface for them all.

Beyond just how easy the system was to use, the CDE system provided a single, unified documentation system that was linked into all the applications, written in a single standard language, and enabled users to browse and search for various answers inside of a single information help system. The system provided a one stop shop for help in any of the desktop applications, and its hypertext nature made it easier to maneuvour through.

Consistent. One aim of CDE was to provide consistency. Widgets looked the same and worked the same across multiple operating systems. Back then, an important element was consistency across platform on the UNIX systems, and not just consistency within the desktop itself. The user was expected to be able to learn how CDE worked, and then work effectively on any number of Operating systems without having to be retrained. It also provided a level of consistency on the desktop in the form of the object oriented emphasis of the system. Icons and mouse based actions were familiar, and applications used the same sort of actions across the desktop. A drag and drop was the same in one application as it was in another, and you could configure and change the look and feel of each application using the same Xresources based settings, which could also be set graphically, much in the form of a registry editor of Windows. Other options besides look and feel could also be set through these resources, giving a common interface for managing all sorts of setttings in applications, which could be layered in the form of defaults, system level settings, and user-level overrides.

Further more, documentation was all done using a single standard language and was interfaced through a single standard programming interface. Applications would be able to install their manuals and guides and then the user could browse it with their help system.

Another feature added into the 2.1 version was a common printing interface, that allowed any application to communicate a rendered version of a widget and have it sent to the printer instead of the screen.

Configurability. CDE was very configurable. Whether through configuration files or through the graphical configuration managers, user's could control many of the aspects of their system. It allowed for all sorts of flexibility that users of UNIX systems have come to expect. These configurations meant that whatever the situation, CDE could probably be fit within the existing needs of your organization. It helped that Motif and X itself was extremely flexible. CDE made it possible for users and administrators to access this flexibility in a relatively easy manner compared to the traditional methods.

Portability. CDE was built in collaboration with a number of corporations, each with their own UNIX operating system. Before CDE and its standardization, users would need to adjust to each operating system and learn its quirks before becoming productive. CDE provided a single, easy to use, and graphical interface to all of the Operating systems, so that user's could become productive with their workstations regardless of the underlying operating system. If it could run X, it could run CDE, and you could have the benefits of the same system across many different platforms.

Distributed Design. CDE was designed in an era that was seeing the change from time-shared mainframe systems into distributed client-server systems where the graphical rendering occurred on either lightweight or heavyweight graphical display servers that hosted the display of an application that was hosted on an application server. CDE's design enabled it to ease the lives of user's who wanted to work in a heavily networked, client-server world. Resources available on other machines, such as applications or data could be accessed easily with CDE, even though the other servers may not have been running CDE themselves.

The system of CDE itself was to some extent, distributed. This enabled the pieces of the system to be addressed at the appropriate level by the ap-

propriate person, rather than requiring a single monolithic approach. This can be seen in the form of how layered the configuration is, so that an user can use their own settings to override system ones, and system administraters can make site-wide changes without having to worry about default settings of the application or user-level changes.

This designed enabled the whole CDE system to appear as a unified system to the user, when in reality, many elements might have been distributed across machines all over.

Protection of Investment. When CDE came along, companies had already invested huge sums of money into existing applications. If their existing applications did not work or integrate nicely into the CDE environment, CDE would have been worthless. CDE enabled administrators and users to effectively integrate current applications that they had written easily. This meant that an OpenLOOK application could be wrapped and integrated in look and feel with the rest of CDE.

This protected the investments of companies in existing custom applications and various other workflows; CDE would integrate and merge with existing needs rather than requiring an all or nothing commitment from the company.

Easy access to powerful resources. As mentioned before, CDE was built on UNIX, and this enabled it to access the power of UNIX's networked nature. Something that PC users couldn't yet dream of. This networked capability was sometimes hard to get to, though, and CDE contributed to UNIX by providing a consistent and mouse-oriented means of accessing these powerful networked and local services.

What does CDE do right?

The most important thing CDE did right, historically, is the standard, high-quality level of integration that the system could provide. That is, it provided consistency and regularity to a degree not found in other options. Its support of networking was, of course, a plus. Moreover, as time went on, CDE remained stable, reliable, and did not break compatibility. This was vitally important for companies where sources were usually not available, and systems had to remain ABI as well as API compatible or they would not run. This ABI stability is something that gave CDE a strong endurance.

Ironically, in today's world, CDE's ABI stability is now one of its best features. Other desktops have supplanted CDE as the “standard” among desktop systems, and GTK might arguably have that role now, but Motif is still the standard for mission-critical applications, and CDE is still the only full scale desktop environment targeting this stable, reliable toolkit.

CDE can now also claim to be one of the lightest weight full-featured desktop systems, in large part because it has avoided the feature creep of other desktop environments.[3] It has a remarkable number of features for the level of resource intensity it provides. Being built on the Motif toolkit, it is also one of the only desktop environments that can integrate well with high-performance applications over X Forwarding. Neight GTK or Qt are particularly efficient over the network, making it difficult to distribute application servers in a client-server fashion.

[3] This performance comes at the cost of keeping up with the latest advancements in Desktop technology.

CDE's general development attitude was built not on open-source, but on open standards technologies. These standards have been around for a while, and it also means that CDE does a much better job of not getting in the way, or inneficiently layering over duplicate or redundant technology for desktop use that is not also part of the core UNIX system. CDE is a desktop that integrates better with the traditional UNIX philosphy, while desktop systems like KDE, in particular, have taken an attitude of developing a number of layering, insulating technologies that diverge or duplicate existing UNIX functions. Additionally, KDE tries to hide the underlying mechanisms of the UNIX operating system, preferring to reimplement features rather than to expose those features directly to the user. CDE does not do this.

In summary, CDE is the lightest and most stable full-featured desktop environment conceivable today.

Where does CDE go wrong?

Some of the technologies of CDE requires that the application be aware of CDE and be coded especially for it. At least in the short term, the OpenCDE project, and CDE, cannot expect to have this standard position. Applications are often written with Gnome or KDE in mind, but they will not, as a whole, likely be written with OpenCDE in mind. Thus, CDE's reliance on the applications to provide certain features will not be as plausible in today's desktop market as it was when CDE was the standard desktop.

The style guides of Motif and CDE were only integrated into a single style guide in the 2.1 versions of both systems. Beyond this, many applications in the CDE package were not designed for usability as we now think of it. The workflows on some of the user-level utilities would not pass today's current standards in usability, though they may have been considered an improvement back then.

CDE is also out-of-date concerning integration. At the time when it was receiving the most active development effort CDE needed to worry about integration and migration from things like XView, OpenWindows, or OpenLOOK. Today, Gnome, XFCE, GTK, KDE, and Qt represent the standard applications and environments that CDE needs to worry itself with.

In other words, there are places and areas where CDE doesn't fulfill its potential in terms of either consistency or usability.

The changing desktop climate

The climate in which CDE thrived was the commercial, workstation, enterprise level UNIX labs. These were often high-end labratories doing things like graphics design or engineering; they might even have been doing defense and geological research. People using these systems underwent intensive training, and CDE provided an important reduction opportunity to those training costs, but the user-base was still largely technical, even though many did not have degrees in computer science.

In the above climate, preservation of long term investments in applications (which were not open-source) was of primary importance, as was ensuring ease of central administration, remote access, and remote application services. These systems were not concerned with fashion or the like, though that sometimes did play a part in things.

Today's desktop climate for UNIX machines continues to grow increasingly complex. Linux still remains a strong server operating system. It is used throughout, and UNIX operating systems like the BSDs also see heavy use in the server fields. Workstations are increasingly becoming Windows or Mac OS machines, and the desktop is seeing an increasing number of Linux/BSD users. This creates an extremely diverse and competitive desktop eco-system. There are five major desktop environments that vie for the attention of users in all sectors ranging from education to workstations to business to home use. Microsoft Windows and Mac OS X are the current leaders among many consumer oriented devices. On Linux, Gnome has a strong holding in many distributions, and has strong support from enterprise Linux distributions. KDE has a strong following among consumer and user-level distributions, but OpenSUSE has also made a strong push for KDE in the enterprise. XFCE is the current lightweight desktop favorite, built on the GTK toolkit.

With so many options, it is no wonder that the emphasis has shifted away from stability in many ways, and on to innovation and rapid progress. The Open-source development model has created a rapid release cycle with many users running alpha or beta quality software. Flash and eye candy are also a significant element in today's desktop systems. Compiz and other graphical features make possible very sophisticated graphical animations as part of the desktop experience. These features also encourage a great deal of experimentation in how intefaces might interact. KDE has Workspaces that each have their own set of desktop widgets, Ubuntu is working in a brand-new unity desktop, and Mac OS X recently released another version with an emphasis on rapidly viewing and accessing all elements of a running desktop with a few mouse gestures. Windows 7 made strong progress forward in terms of how the user on those systems interact with windows and applications.

In a world of rapid development and progress, stability, ABI compatibility, and especially preservation of existing investments is not nearly as important to the average user as it once was. However, it does remain a strong issue for many sectors, and arguably, this need is not being met well by existing developments.

Another trend in desktop development is the eagerness to development integrated systems that are also highly modular. These modular elements often plug in and out of a aystem, and distributions may or may not include them. Gnome has a model like this, and KDE, to a lesser degree, has a model like this. However, Gnome's extreme use of modularity has caused some distributions to stop shipping it, since maintaining the development progress of Gnome became too difficult.

Many desktop environments also provide or host many backend services far above and beyond the normal desktop interactions that the user sees, such as the Akonadi indexing server on KDE. A remarkable number of backend services, many of which are duplicates of existing OS services run in conjunction with window managers and file managers.

What is wrong with the modern desktop?

Where does OpenCDE fit in?

Workalike versus lookalike

The tension between the past and the future

Development Goals

Development Practices, Norms, and Expectations

User expectations and Developer Response

Guidelines for deviating from and improving CDE

Bibliography

OpenGroup. *Desktop Technologies: CDE*. http://www.opengroup.org/cde/. Accessed on 1 August 2011.