Mac OS X: Welcome to the jungle

A look inside the Mac OS X software ecology

Chris J. Karr

If software platforms are habitats, the Mac OS X platform is surely the jungle. Mac OS X is a modern Unix-based operating system that combines the classic Unix/X11 environment, a modern Java toolset and runtime, the classic Mac OS Carbon framework, and the NextStep-derivative Cocoa framework in an elegant and user-friendly operating environment. This diversity of strongly supported programming options, combined with Apple's modern hardware and operating system, presents developers and users with a compelling platform for producing and using software packages.

The continued success of the Macintosh platform is due in no small part to the different ways that developers from other environments can apply knowledge and experience from other platforms to produce Mac OS X applications. These different development platforms can be separated into a few large groups – Unix-based, Java-based, and the derivatives of classic Macintosh and NextStep platforms. Since developers targeting each of these groups come from different backgrounds and development philosophies, developers of each platform tend to produce significantly different types of applications. For example, developers targeting the BSD Unix portions of Mac OS X are more likely to develop and produce programs found in other Unix environments, such as command-line text tools and interpreters. Java-based developers bring cross-platform applications such as the Apache Tomcat server and IBM's Eclipse to OS X users. Developers specializing in Carbon are responsible for modern incarnations of applications from Mac OS 9 and before, such as Microsoft Office and the Adobe multimedia applications, while developers targeting the Cocoa framework have applied object-oriented principles to create unique types of applications found only on the Mac OS X platform.

To understand the diversity of Mac OS X's programming options, it helps to be aware of the operating system's history. Prior to the acquisition of Next, engineers at Apple were busy working on the next-generation successor to Mac OS 9 codenamed Copland. When this effort, along with others (such as the Pink partnership with IBM and Motorola) failed, Apple looked outside the company to acquire a successor to Mac OS. Be, with its modern multimedia-oriented BeOS, was a favored choice, but Apple ultimately chose Next, with its more mature NextStep technologies as the next Apple operating system. The NextStep operating system had a number of traits in its favor. It was a modern and mature cross-platform operating system with solid underpinnings and a strong developer community.

It had modern and robust networking capabilities. (The World Wide Web was originally designed and implemented on a Next machine.) In a time when Apple was floundering in the computer market and approaching irrelevance, the Next acquisition also returned the visionary (and not uncontroversial) Steve Jobs back to the helm of the company he co-founded years before.

While Jobs' vision and drive are often credited with Apple's resurgence, the Unix and NextStep technology, combined with Apple's new focus on Java and open standards, created an environment where developers combined skills acquired when working on other platforms with Mac OS X's native feature set to create new applications and libraries. In order to bridge the legacy developers' transitions from the classic Mac OS platform, Apple provided the C-based Carbon framework to ease the porting process and minimize transition costs. The effort to accommodate and provide familiar environments for programmers from the classic Mac environment and elsewhere is one of the key factors in Mac OS

Fig. 1: Safari (Apple) is a Cocoa browser using the brushed-metal theme. It uses standard Cocoa widgets and styles, but does not make use of customizable toolbars or other advanced Cocoa UI elements

X's success as a development platform.

One of the interesting results of this integration of various development environments is that different types of software developers brought their different development processes to the Mac. This diversity of processes is directly responsible for the different types of modern Mac software. Larger developers who have produced software since the classic Mac OS era tend to use Carbon-based technologies. Smaller developers writing new applications exclusively for Mac OS X tend to use Cocoa-based technologies. Migrant developers from the Linux and Unix community continue to program to the Unix interfaces and use the BSD and X11-based technologies, while business and open-source developers of cross-platform tools and applications have adopted Apple's version of Java.

It was a modern and mature cross-platform operating system with solid underpinnings and a strong developer community. It had modern and robust networking capabilities

Commercial developers

The most visible Mac software developers tend to be larger developers. Microsoft, Adobe, Macromedia, and (of course) Apple. They all design and market large software packages for the Macintosh. Microsoft is known in the Mac world for its Office and Internet Explorer products. Adobe has been active in the Mac community for years with its digital image and multimedia creation tools. Macromedia continues to develop and market its web authoring applications. Apple develops and distributes its iLife applications for casual users in addition to its more professional line of media tools such as Final Cut Pro. Because of the high overhead of marketing and distributing these products via traditional channels and distributors, mostly larger companies occupy the brick-and-mortar shelf spaces. Furthermore, many of these types of applications predate the Mac OS X operating system and consist of significant amounts of code created during the classic Mac OS era.

Fig. 2: Internet Explorer (Microsoft) is a Carbon application. Note the continued use of the heavy pin-stripe theme that was the style of MacOS X prior to 10.3

Since the amount of working legacy code in these products is non-trivial, the producers of larger Mac software packages continue to develop and maintain these products using the Carbon framework. In contrast to the heavily object-oriented Cocoa technologies, the Carbon framework consists of low-level C-based functions and libraries. The use of this framework allows Carbon developers to control basic underlying features, such as Quartz and Quicktime. However, this control comes at a price; the large amount of source code and increased complexity creates an inertia that is hard to overcome when implementing new features or re-targeting the applications to new markets. The primary outcome is that these applications are more complex and full-featured (due to longevity of the product), but these applications evolve slowly and are updated much less often than their Java and Cocoa counterparts.

> The continued success of the Macintosh platform is due in no small part to the different ways that developers from other environments can apply knowledge and experience from other platforms to produce Mac OS X applications

While larger developers tend to use Carbon, small independent software developers tend to use Cocoa. Because of the object-oriented nature of the Cocoa framework (previously known as NextStep or OpenStep) and the rapid application development possible with Xcode, smaller developers use Cocoa as a quick route from creating an idea to implementing that idea and making that idea available to interested users. Furthermore, because of the exclusion from traditional channels of distribution due to the overhead involved, smaller developers use the web as the primary means to market and distribute their applications. Since these applications tend to be smaller than their larger commercial counterparts, the market for these applications consists of many users willing to purchase these applications for less money than the larger general applications. Finally, the robust shareware community that the classic Mac platform was renowned for has adapted to this new market configuration.

Because of the smaller codebases, smaller development teams, and lower price points, a rigorous competitive market has emerged where developers compete for paying users. Given that the primary distribution of these products is online - typically in the form of downloadable disk image files - communication between developers and users is conducted online via e-mail, weblogs, and discussion forums. These factors result in a market where developers are in closer touch with their users. Furthermore, rigorous competition spurs continual development and updates, and new applications are produced daily that attempt to establish new markets. The RSS reader market emerged from such an environment. Although Ranchero's NetNewsWire established the RSS aggregator market, it is currently in constant competition with many similar competitors. This is in stark contrast to markets for products such as Microsoft Office or Adobe Photoshop, which face significantly less competition in their respective markets.

> . . . the Mac OS X platform sports a healthy and growing commercial developer population, it also hosts an equally healthy free software community

An interesting aspect of the small commercial development community is the cooperation between applications in different markets. While applications targeting the same markets compete rather than cooperate, developers will reach out across market boundaries to establish interoperability with other applications. This strategy provides a competitive advantage as sophisticated users can combine different applications to accomplish tasks that a single application would be unable to address. This type of cooperation is evident in the interoperability between RSS aggregators and weblog-authoring tools. In some cases, a developer will offer both types of applications, but still provide compatibility with competitors' products in other markets.

The Java community

Commercial and free software developers creating software with cross-platform compatibility as a feature find Mac OS X to be another viable deployment platform. Commercial software applications have been successfully ported and open-source Java applications have met similar success. In corporate environments where custom Java applications are used to interface with various custom server applications, Mac OS X has proven itself capable of providing an environment that is consistent with the application's goals and requirements, while providing the advantages of Mac OS X.

Console and command-line based applications can often run with few changes. Apple provides a full Java runtime and

software development environment that is accessible via its Bash command line. Graphical Java applications written with Swing run on the Mac using Sun's Steel theme, or Apple's Aqua theme. Apple's Swing-compatible implementation of the Aqua look and feel allows platform-independent applications to run and behave like native applications.

Developers creating software to be run on different Java platforms occasionally discover incompatibilities in areas such as printing or the system clipboard, but these incompatibilities are easily addressed and are not significant enough to deter Java developers from excluding Mac OS X as a target platform.

Free software developers

While the Mac OS X platform sports a healthy and growing commercial developer population, it also hosts an equally healthy free software community. In contrast to the Windows platform where free software development is insignificant compared with commercial development, and the Linux/BSD platforms where the reverse is true, free software developers enjoy parity with commercial developers on Mac OS X.

A large catalyst for seeding free software development on Mac OS X was Apple itself. Mac OS X is built upon a free software BSD Unix variant called Darwin. Apple also provides a native BSD command-line environment, which is accessible via the Terminal application. The release of Mac OS X 10.3 included a Quartz-aware version of the X11 windowing system. Apple continues to support free software by integrating the KHTML web engine in the Safari web browser and uses free software applications such as Samba and Apache to provide network file sharing and web hosting. Apple is a very active member in the free software community, and this is expected to continue.

Unlike the Cocoa, Carbon, and Java environments, the command line and X11 environments are populated almost exclusively with free software. A significant portion of this environment consists of Apple's own free software programs and ports, but other free software projects such as the MySQL database and OpenOffice.org office suite target the BSD foundation and primary platform targets often include Mac OS X. Furthermore, projects such as DarwinPorts, Fink, and Gentoo have aggregated and packaged Unix software management tools for Mac OS X. Using these applications, the Mac can install many programs found on the BSD and Linux platforms.

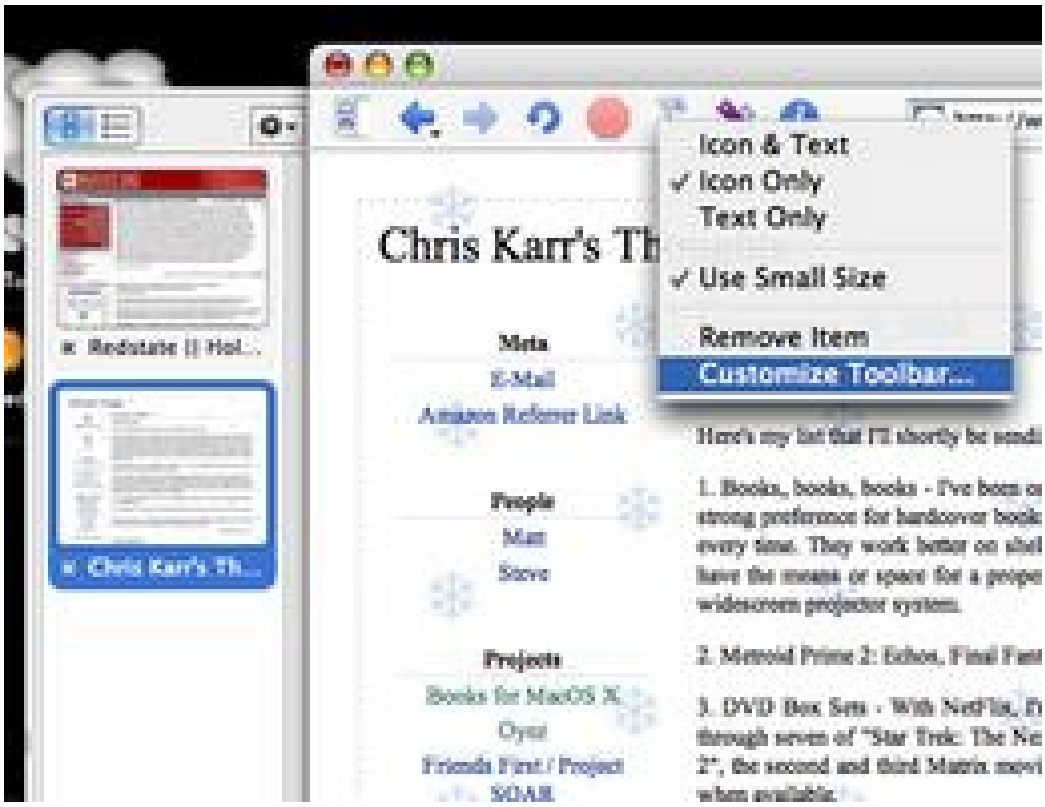
While some developers are content that their applications can run on Mac OS X via the Terminal and X11 environments, other free software developers adapt open-source GUI applications to take full advantage of Mac OS X. Prominent projects such as Mozilla's Firefox were ported from their X11 and Windows origins to Mac OS X's native interfaces. While some of the Gecko widgets in Mozilla applications are not "Aqua-fied", the icons, application windows, and application packaging follows Mac OS X standards. In addition to the Firefox port, the OpenOffice.org and Gimp projects currently pursue similar porting goals.

Fig. 3: FireFox (Mozilla) is a port of the Linux and Windows browser. It uses a theme similar to Aqua, but this can be altered using FireFox's theming capabilities. The widgets used within the web pages are the standard Gecko widgets - not the standard Apple widgets

In addition to free software originating from other platforms, Mac OS X inspires developers to create native applications unique to the platform. One factor motivating developers are the advanced multimedia and network capabilities of Mac OS X that Apple exposes via the Carbon and Cocoa frameworks. The other factor in the motivating developers is the availability of free developer tools and documentation. Unlike Windows development tools, Xcode and related applications are available online via Apple's website and Apple bundles these with Mac OS X install disks. Furthermore, while O'Reilly publishes books like "Learning Cocoa with Objective-C", Apple makes the same content freely available online. Free tools and documentation enable new developers to start developing full-fledged Mac applications quickly.

The free software native Mac OS X community mirrors the small and independent developers in many ways. The same factors that motivate small developers to flock to the Co-

Fig. 4: OmniWeb (Omni Corp.) is a pure Cocoa browser using the standard Aqua theme. Note the use of advanced Cocoa functionality such as drawers, customizable toolbars, and the WebKit rendering engine



coa also attract free software developers. In some cases, free software applications compete directly with commercial counterparts. For example, the open-source Adium instant messenger client competes with iChat and the Proteus clients. Another example is the Camino project that seeks to produce a fully Cocoa implementation of a Gecko-based browser to the Mac. This product competes with browsers from large developers, other free software projects, and independent developers (Internet Explorer, FireFox, and OmniWeb, respectively).

Other native free software applications carve out their own new niches. The Growl notification engine is one such application. Furthermore, free software applications also cooperate with other free software and commercial applications to provide enhanced functionality for their users.

If the number of new applications and the updates reported on sites like MacUpdate and VersionTracker are any indication, software development on Mac OS X continues to grow and progress. As Apple continues to attract new users with their hardware and software, its development options will continue to attract developers with new ideas and energy to produce new software for the Mac. If recent trends continue, there is no reason to think that the diversity, quality, and quantity of Mac OS X applications will decline any time soon.

Welcome to the jungle.

Bibliography

[1] Linzmayer, Owen “Apple Confidential 2.0”, No Starch Press:2004

Copyright information

© 2005 by Chris J. Karr
This article is made available under the ”Attribution-NoDerivs” Creative Commons License 2.0 available from http://creativecommons.org/licenses/by-nd/2.0/.

About the author

Chris is a software developer for Northwestern University and develops the open-source Books library management software in his free time.