few CEOs ago Apple had a grand vision for the future of computing. It was called OpenDoc. We even ran a cover story on it in ATPM 2.09. OpenDoc promised to end the age of bloated do-everything applications by providing an architecture where users could mix and match collections of highly focused “containers” or “parts” to suit their needs.

Apple’s Publish and Subscribe, which debuted in System 7.0, let you “publish” pieces of documents into “edition” files, which other documents could then “subscribe” to and embed within themselves. With OpenDoc, all the data pieces were stored in the same file, and each could be edited in place. When you clicked on a chart, for example, the menu bar would show all the commands that your chart editor supported. OpenDoc promised to completely free users from the application software paradigm, leaving them to focus on their documents, rather than on the tools they used to make their documents.

Cyberdog, originally intended merely as a demonstration of OpenDoc’s potential, suddenly became Apple’s “Internet Strategy.” Although Cyberdog could never compete with stand-alone Web, mail, or FTP clients, it showed enormous potential and is still used by a handful of OpenDoc loyalists. Sadly, an e-mail signature file saying “This message sent using the Cyberdog mail system” or “On the Internet no one knows you’re an OpenDoc part” is all most Mac users are likely to hear about OpenDoc now. It seems that OpenDoc was yet another Apple technology that was released before it was ready (too slow and resource hungry) and before the market was ready for it. While it probably won’t be called OpenDoc or even made by Apple, I think that someday we will be using software that is remarkably similar to OpenDoc.

The Unix Paradigm

In Unix, applications are small and specialized, but general. (Each command-line command is itself a small application.) Multitasking and memory protection aside, the power of Unix comes from the ability to use all these applications in tandem to accomplish tasks. Nearly all documents are stored in plain text format. Thus, each program can operate on another’s data. Whereas OpenDoc Editors could operate on different “parts” of a document, in Unix one can cascade (or “pipe”) the output of one program into the input of another.

The rise of the Internet during OpenDoc’s fall brought Unix’s paradigm for applications and documents to the Mac (and Windows) because most Internet standards are Unix-based. Everyday Mac users are increasingly using text or HTML for tasks when they would formerly have used specialized applications to store documents in proprietary formats. Even Apple is doing this. Sherlock plugins are essentially just text files, and the new Mac OS Help in OS 8.5 is in HTML format. Apple is learning that while custom solutions like AppleGuide may have

advantages over standard formats like HTML, the number and variety of tools (and people skilled in using them) available for standard formats are often more attractive.

XML: The Great Equalizer?

The ultimate in standards-based formats is the eXtensible Markup Language, XML. Like HTML, XML is an extension of SGML, the Standardized General Markup Langauage. In HTML, there is a standard set of tags you can use to format text, include graphics, and create links to other Web pages. In XML, however, you can define your own tags (hence the "eXtensible"). Each XML file is prefaced by a link to Document Type Definition that describes the tags it contains and how they may be used. (Thus, an application reading an XML file can tell whether the tag syntax is correct, even if the application was written before the particular tags were invented!)

Most Macs users will never need to read or understand XML, but it may soon affect the way they work with their Macs. Since XML allows custom tags, online publishers will be able to make their databases return data in XML format. The tags will describe the meaning behind each part of the data, so applications on the user's Mac will be able to understand what it means. For instance, an invoice from Amazon.com in XML format could easily be imported into an XML-savvy version of Quicken for record keeping.

The next version of Microsoft Office will use XML as its standard file format. This might seem like an odd move for Microsoft, which has long used proprietary "standards" maintain its dominance. This indicates one of two things: either Microsoft thinks XML will be big and doesn't want to be left behind, or they feel that they can make Office a more attractive package by storing its data in a format that other applications can understand. Either way, users win.

XML's goal is to facilitate the exchange of data between applications (and platforms). It will make it easy to edit the same file in different applications, each one excelling at a specific task. This is the opposite of OpenDoc, which placed the different applications (visually) within the same file. It's not a question of XML or OpenDoc being better than the other—each was designed for a different purpose.

The problem is that right now the Mac is seriously lagging behind Windows and Unix in XML tools. The XML-savvy Internet Explorer 5 for Windows is already in beta, while the Mac version won't be out until sometime this summer. Probably the brightest XML product for Mac users is the commercial version of UserLand's cross-platform scripting and content management system, Frontier. (ATPM used to use the freeware version of Frontier for Website management, but has since switched to a custom BBEdit and AppleScript solution.) On the editor front, I have high hopes that a future version of BBEdit will be XML-savvy. After all, it already includes an SGML parser.

Java and AppleScript

Apple just released version 2.1 of the Macintosh Runtime for Java (MRJ). It's faster and more stable than MRJ 2.0, and offers much improved compatibility. Unfortunately, Sun's specification for the Java Development Kit 1.2 has been out for several months, and Apple's MRJ only just gained support JDK 1.1.6. (JDK is a much bigger deal than the 1.2 version number implies, hence Sun refers to it as the "Java 2" platform.)

Most Mac users' only experience with Java is "applets" that load inside Web browsers. In fact, although Java may not have lived up to all its hype (Remember the talk of whole office suites written in Java?), it is becoming a very important tool. Improvements in virtual machines are making it faster, and I'm told that theoretically it could eventually run faster than conventional applications written in languages like C or C++. Much interesting work with XML is being done in Java, and it is becoming increasingly popular in academia and as a tool for developing custom solutions. In short, much work in emerging technologies is being done with Java, so it should be very important to Apple. Steve Jobs has said that his goal is to make Macintosh the leading Java platform, and I think he's serious.

Macintosh has the potential to be a great Java platform because Apple's MRJ knows how to talk with AppleScript. AppleScript has been around since System 7 Pro, but it really started gaining steam last year. In my opinion, it's one of Mac OS's greatest advantages over other platforms—and it will only get better with OS X. The more I learn about AppleScript, the more I am amazed at how much power was lurking in the applications I use each day.

So what does Java have to do with XML and AppleScript? Well, lots of "glue" applications for integrating XML with existing software solutions are being written in Java. MRJ knows how to receive AppleEvents (the messages that AppleScript uses to control applications), and many Mac applications are now scriptable. To take full advantage of XML, applications must be updated to support it. I think there is serious potential here, for the Macintosh to take advantage of some of what XML has to offer by using AppleScript to interface existing applications with Java-based XML tools.

The power of the Macintosh has always lain in the way different applications could work together smoothly. In the mid-eighties, this meant that applications provided a consistent user experience and could share files in common Mac formats like text and PICT. In the early nineties, it meant that they could share dynamically updated data via Publish and Subscribe. Today it means that most applications can exchange data directly via drag and drop and automagically via AppleScript. Tomorrow, data exchange will be freer than ever, removing another layer of the seemingly arbitrary restrictions that technology places on us. Isn't that what Macintosh is all about?

Links

UserLand Frontier: <http://www.userland.com/frontier5>
A good XML resource: <http://www.oasis-open.org/cover/xmlIntro.html>
AppleScript Sourcebook: <http://www.applescriptsourcebook.com>
ScriptWeb: <http://www.scriptweb.org>

"The Personal Computing Paradigm" is copyright © 1999 by Michael Tsai, <mtsai@atpm.com>.