Posix: A Model for Future Computing

Charles Severance, Michigan State University

When I first attended the Posix P1003 working group meetings in 1990, they noticed that I was wearing a tie and sent me packing to the P1003.0 working group. This group was working on a broad guide to the Posix Open System Environment, a generic model for making sense out of computing systems. Most people outside this working group thought the primary purpose of the P1003.0 was to contain tie-wearers in a safe place insulated away from the actually relevant work.

Those of us inside the group set out to make sense of all of the disparate computer standards, such as TCP/IP, IEEE 1994 (Firewire), SCSI, C++, and Posix. Because each standard has a significantly different scope and purpose, we wanted to find the "superset" of all of the scopes and then organize that scope and place each standard within our model. We hoped to help people understand where each standard fit into a big picture.

THE SMOKE CLEARS, A MODEL IS BORN

Most standards people, when faced with a daunting problem, look for a model to help them understand. We were no exception. One person and then another would propose a graphic model, attempting to organize and make sense of the universe of computer standards in a universe. In most of these efforts, we started with several different, yet understandable, models and ended up with a single abstract model. Some members of the group wanted a layered model (after all, OSI networking used a layered model) and others wanted a model that looked like a real computer (CPUs, busses, disks, network cards, and so on). Hand-drawn sketches of these models flew back and forth in flurries at each quarterly meeting. At one point, these "cartoon wars" became so intense that I made a computer animation poking fun at all of the different cartoons under consideration. (You can see the animation at http://www.egr.msu.edu/~crs/stds/pasc/ with RealPlayer.)

Editor: Charles Severance, Michigan State University, Department of Computer Science, 1338 Engineering Bldg., East Lansing, MI 48824; voice (517) 353-2268; fax (517) 355-7516; crs@egr.msu.edu; http://www.egr.msu.edu/~crs

Are standards becoming obsolete as fast as they are churned out? Maybe, but the Posix reference model remains accurate.

We finally agreed on the model shown in Figure 1. The model does not have any layers, and it is very simple. The reference model identifies four main interfaces at which standards really matter:

- *Application programming interface.* Systems—whether hardware, software, or some combination—need a way to accept new instructions in the form of program code. Sample standards at this interface would be C++ or Posix P1003.1.
- *User interface.* This interface allows a user to interact with a system; common examples are keyboards, GUIs, and mice. Sample specifications at this interface would be Windows-98 look and feel, Macintosh look and feel, or Motif look and feel.
- *Data interface.* Most systems exchange data with other systems; this interface usually specifies the formats for that data. Sample specifications at this interface would be PostScript, SGML, or JPEG.
- *Communications interface.* At some point, many systems must interact with standard telecommunications equipment—modems or networks, for example. Sample specifications at this interface would be TCP/IP, ISDN, or RFC-822 (SMTP: Simple Mail Transfer Protocol).

The *application platform* was an abstract combination of hardware and software that presented these interfaces to applications, users, media, and communication networks. We also identified a fifth interface, the platform internal interface—our catchall for the standards that assembles components to build an application platform. There was some contention as to whether or not we should cover these standards in our document. Some felt that if we ignored standards such as SCSI or PCI, we would not be capturing reality. Others thought that

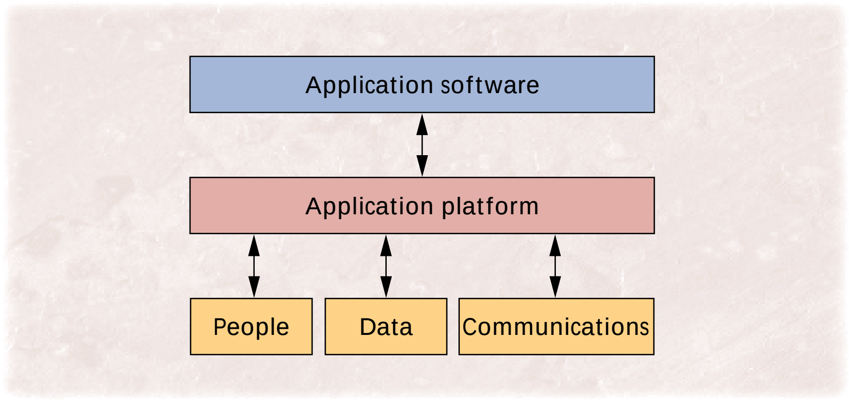

Figure 1. The Posix Open System Reference model, developed in 1990, still holds true today.

Figure 2. PDA compliant with a nearly 10-year-old model.

these internal decompositions were temporary architectural artifacts that would change over time and didn't belong in a broad model intended to last. After significant discussion, the platform internal interface was declared beyond the scope of our document. The resulting abstract model seemed a little foreign to newcomers, but the working group learned to love it.

OBSOLETE IN 1995?

Even though we settled the model issue in the 1990-to-1991 time frame, much work remained. We identified pertinent standards at each of the interfaces and identified gaps in the available standards. We completed the document in 1994, and the IEEE Standards Board approved it in 1995. At that time many new OSs (such as Windows NT) and networking architectures (such as CORBA) emerged. These technologies were touted as "object oriented." To some of us, distributed, communicating objects appeared to be the next fashion, and our model—into which we had poured our hearts and souls—would become hopelessly dated, surpassed by a new wave of "integrated" models.

After the standard's completion, the working group, which over five years had become like a large family, gradually dispersed and many of us assumed that our model would fade into a nice historical footnote.

VINDICATION IN 1998

Recently, I was playing Solitaire on my PDA, similar to the one in Figure 2, and it occurred to me that I was holding a piece of equipment that embodied the Posix Open System Reference Model.

It had a user interface (a stylus), a communications interface (modem), and an API (I could program it in a "standard" language: Visual Basic). My PDA also supported data in several formats. Most importantly, I did not need to care about any of the nasty internal details. It did not matter whether my PDA was based on a client-server, layered, object, or agent model. I didn't need to know whether it used active messages internally. It was an appliance.

I started to wonder why no one has developed a widely accepted object-oriented model similar to the Posix Open System Reference Model. To me, that these models are proposed and then change like the seasons is a huge problem.

WHEN STANDARDS WORK

To set forth the internal decomposition among hardware and the various OS elements in a prescriptive standard would unnecessarily hamper innovation. Over time, these internal decompositions morph into new forms. The only standards that we can hope to maintain over time are those for the exposed interfaces that truly affect the broad user population.

Generally speaking, the application platform grows over time and expands its capabilities. When a company adds new capabilities to the platform, the capabilities initially retain their distinct shape. We can understand their position relative to the platform's other parts. Over time, however, the boundaries blur and eventually melt away completely, and we are back to a single monolithic application platform. The only real changes are the new capabilities.

Assimilation is necessary, because left alone, layers and identifiable components would just keep stacking up until they eventually toppled. To make a point by exaggeration, consider a system with 100 layers of software components, each perfectly specified and from a different vendor. A pacemaker application interacts with the top layer and each layer works with the one below. Would you want a pacemaker with that architecture installed in your body? Another way to ask the question is, "How happy will you be when your plug-ins start needing plug-ins?" (which is happening by the way in some areas).

So it seems that although we are certainly developing object-oriented and distributed operating systems, all of their components are (at best) made out of ice cubes, which are always slowly melting and losing their identity. However, it seems that no matter how computer technology changes, computers continue to look a lot like the Posix Open System Reference Model. That's not a bad product for a bunch of folks wearing ties. ❖