

As seen in Start – July/August, 1998

OPEN SYSTEMS IN SOFTWARE – WHAT REALLY MATTERS

Rich Ryan
President, Rockwell Software
Rockwell Automation

There's an old saying – "I may not know art, but I know what I like."

Whoever said it first might have been an automation software user, because the same could be said for open systems. Ask ten people for their definition of "open," and you'll certainly get ten different answers. Even that old standby, the Webster's New Collegiate Dictionary, has more than 30 different entries. It's joined religion, politics, global warming and the return of Michael Jordan as a topic that's sure to spark endless debate.

In our dealings with automation software users around the world, we constantly ask them if open systems are important to them. The answer is almost always a hearty "Yes!" But that's where the consensus stops. To some, open systems means "plug and play". To others it's integration with higher-level IT applications. To others it's strict adherence to a standard. To many it simply means running on the latest version of Windows.

Opinions vary within the enterprise as well. A Company CEO may define it as sharing financial information throughout the business, while a plant supervisor defines it as interchangeable equipment. A Vice President of Marketing could see it as being able to access data from any regional office, any

time, while a controls engineer may view openness as a way to easily integrate another device into his plant architecture.

The debate on open systems from a hardware and networking perspective seems to be waning with the popularity of Intel-class computers and more interoperable, multi-vendor networks. But think about it – neither of these two is of any use without the software. In fact, you can't have an open system without it. Does "open" mean "integrated"? Probably not, because nearly everything can be integrated in some way, shape or form – usually at a steep price.

Mark Gietz of Programmable Control Services, a system integrator in Spokane, WA in the metals and utilities industry, has a somewhat universal definition of "open". He says that for software to be open, "no matter what application you are using – on the plant floor or in the office – there needs to be an accepted method for accessing data so that we as end users can make it work with other software in the application."

To really understand openness from a software perspective, you need to take a look at the underlying technology enablers that create the environment for open applications and tools.

A few years ago, Microsoft set out to develop open products – products that exposed their properties to each other. The company's earliest "open" development began with OLE and ActiveX controls. Most of us associate OLE with the ability to embed an Excel spreadsheet into a Word document. Excel had to "reveal" itself to Word in order for Word to recognize it. Little did we know at the time how important this first step would be. ActiveX became a standard for

integrated technologies that enables software components written in different languages to work together in networked environments. ActiveX helped make it easier to integrate a system by using different components from different vendors. Since then, we've seen several additional enablers for open systems.

Microsoft introduced the successor to DDE called the Component Object Model (COM), and then further enhanced that by providing a distributed version (DCOM) that allows for better sharing of resources in a distributed computing environment.

Through Microsoft's vertical market initiatives, The OLE for Process Control (OPC) Foundation, a group of the industry's leading manufacturing software suppliers, developed a set of specifications designed to provide seamless, open, enterprise-wide communications between systems and devices, from plant floor to MIS and beyond. The widespread adoption of the OPC specification will reduce the cost and quality hurdles of multiple proprietary servers, drivers and interfaces needed in the past.

While device integration is important, database connectivity will be a more significant reason to push for open systems. Instead of applying software that only allows for connection to proprietary databases, many users are utilizing tools that allow connections to any database. Rockwell Software RSSql, for instance, allows a user to integrate his or her MMI software with virtually any ODBC-compliant database, allowing for more flexibility and ease of use.

Microsoft took its integrated development environment, Visual Basic for Applications (VBA), and made it available to other software vendors. This

standard development environment (which, by the way, looks just like Visual Basic when you open it up), allows not only for customization of the off-the-shelf application you're using, but also better integration with other software using VBA. Rather than jury-rig a spreadsheet in your MMI software, now you can simply invoke an Excel spreadsheet and integrate it into your application. Need a unique drawing? Simply invoke a few commands and that picture you drew in Visio can be dropped right into your MMI.

Do you need to know every last detail about how each of these technologies works? Probably not. But you DO need to make sure that your automation software suppliers are leveraging these technologies for your benefit. What's under the hood really does make a difference, and having a Windows "look and feel" might have been classified as "open" in the past, but today, you've got to go a lot further than that.

When you boil it all down, the definition of "open" from a software technology perspective isn't nearly as important as what it does for your application. Several factors go into choosing the right software for your unique needs – performance, ease of use, life cycle cost, flexibility, reliability, ability to upgrade. If the "openness" of the software you're evaluating helps you achieve these goals, then by all means use it. If not, stay away. One fairly soft-spoken customer of ours truly put open systems in perspective for me. "Open systems isn't about fancy acronyms and bureaucratic committees," he said, "it's about making the stuff that I've got work with the stuff that I want."

I couldn't have said it better myself.