



HMIs: How Web Access Technologies Affect Visualization Efforts

Inadequate software performance slows data sharing and remote HMI use on the factory floor, but it doesn't have to be that way.

By Andy Balderson and Brent Meranda

Sometimes, it seems the revolutions of the “information age” have skipped the factory floor altogether. Don’t get us wrong: there have been vast improvements in HMIs (human machine interface) over the years, and we’ve come a long way since the computer first started replacing hard wired push buttons and relays. But the fact remains that the majority of today’s HMI and SCADA (supervisory control and data acquisition) systems are standalone applications. When it comes to sharing information, the factory floor has simply not kept pace with the rest of the business world.

For example: When was the last time you saw someone using a clipboard in an office? Quite a while I suspect, but I bet you’d be hard pressed to walk through a factory floor without seeing multiple clipboards, pens, and paper being used to record production schedules and down time. You’re also likely to see operator cheat sheets taped to workstations along with printed work instructions. You might notice an electronic data collector or two, but in many cases they are faithfully recording data no one will ever act upon as information any time soon.

This is in sharp contrast to the office environment, which has fully embraced data sharing technologies like email, instant messaging, MRP/ERP systems, office productivity suites, and electronic project tracking. So why the difference? It’s all about performance and stability.

There has been a host of HMI packages over the last few years offering Web access or Web publishing. In most cases, however, the underlying technologies of these packages do not meet the needs of the factory floor. In truth, their design simply does not allow them to do so.

The most straightforward way of publishing Web data is using standard HTML (hypertext markup language). This is the original language of the World Wide Web and designed to display static pages using a common language that any Web browser supports. While this technology works great for viewing static pages of data, it fails miserably when trying to view changing data or provide real-time control of a machine.

Means: Client driven, screen scraping, browsers plug-in

The reason is that HTML is designed to be client driven. In other words, the client browser is responsible for requesting new data, and there is no mechanism for the server to “push” new data to the client. To get real-time updates, the client must “poll” the server by requesting refreshes on a regular basis, and since this means resending an entire page of data, update frequency rates are

severely limited. This makes it impractical for applications requiring instant alarm notification, real-time data monitoring, or interactive animations.

Another common option for remote monitoring and control is a class of technologies known as “screen scraping.” This is the approach taken by most remote HMI products today. As the name implies, these technologies essentially scrape off the image on the server’s screen and transfer it to the client. The client machine then displays the server’s screen image and transfers its own user input to the server, allowing the operator to interact with the machine as if he or she was in front of it.

This technological approach is used in many of today’s “Web Published” HMIs, and, while it does allow for remote access, it is not real-time. In fact, the main problem with this approach is performance. Because so much data are being transferred, screen scraping requires high-speed connections to be useful at all, and it severely limits the number of clients. Imagine a simple pushbutton changing lens color from red to green on activation. Screen scraping requires the entire screen to be re-transmitted to the client application just to update the one lens color change. In addition, the server and all clients must display the same screen data. Different operators cannot look at different data or interact using a different language.



Parker’s Xpress HMI software uses Adobe Flash technology for unprecedented runtime performance.

Another common approach is to create a custom browser plug-in that is responsible for communicating with the server and transferring information to a remote thin client. If the program is designed well, this approach can solve the bandwidth problems by eliminating the need to poll for data while transferring only information that has changed. This approach also can allow remote operators to view different screens than what the server shows. In practice, however, these solutions are often overly complicated to set up and administer. They also require installing custom, and usually expensive, applications on every client system.

Flash installed on their systems, and its installed base ensures wide acceptance and continuing support. Using this, we can bring the same productivity improvements that have swept through the rest of the modern economy to the factory floor. In the future, the HMI systems that will succeed will be those that support the sharing, analysis, and distribution of information.

There are off-the-shelf browser technologies such as Adobe Flash that can solve these issues. More than 98 percent of all desktop Web browsers already have

Andy Balderson is a product sales manager and Brent Meranda is an HMI software engineering manager with Parker’s Electromechanical Automation Division.

Reprinted from Control Engineering, December 2008