

Paper #: 3050
Title: Revolutionizing the Way Business Interacts
Colin Browning
Interactive Marketing Manager
Pilot Software
One Canal Park
Cambridge, MA 02141
(617) 374-6690

The Internet, Intranet and Extranet are revolutionizing the way that businesses interact with their customers, partners, prospective clients and employees. These interactive technologies have removed the obstacles of platforms, time and distance to create unprecedented efficiencies of communications. In order to leverage these improvements in communications, organizations are pushing to use the Internet for more practical business applications to derive more value from their investment in Web technologies.

New business applications designed for the Internet are being developed every day. Not only are desktop programs, such as spreadsheets and word processors, built with Internet functionality, but also large client-server systems, such as data warehouses and decision support systems, are using the Web to distribute vital information throughout organizations.

As a baseline for how the Web-enablement of desktop programs and client-server systems will impact future business communities, it is important to understand the nature of dynamic interaction of Web-based applications from the perspectives of the developers, end-users and the Information Technology Managers that have to purchase and support them.

Developers

For software developers everywhere, the imperative has been to add Web-based functionality as a basic program requirement needed by customers. Within the past 24 months "Web-enabled" technology has moved from an optional component highlighted by the press to a required product feature, without which sales will erode. Many organizations were slow to recognize the significance of the medium and were driven largely by external factors such as customers or competition. The example of Microsoft failing to develop a comprehensive Web strategy until early 1996 is just one of hundreds of companies slow to realize the rapid acceptance of the Internet by the public. Complicating matters for developers who need to integrate Internet functionality into their products has been the wide variety of technology options available at both the desktop program and client-server system levels.

The desktop programs, such as word processors, spreadsheets, and presentation packages have two basic directions to incorporate Internet functionality. The first, followed by most organizations is to embrace the Internet through added functionality to converting information into Internet compatible formats. Many word processors, spreadsheets programs and presentation packages now have a "save as HTML" option. The second option is attempting to create an entirely web based application through Java applications, ActiveX or Plug-In components (i.e. Shockwave) embedded within websites. To date, these more ambitious attempts of useful Internet applications have yet to really deliver results that rival the traditional commercially available products. At the current pace of development advances it is simply a matter of time before this rivalry becomes more significant. (see Cooper & Peters for a great example of the future of Java applications at **Error! Bookmark not defined.**

More powerful server-based solutions such as databases and decision support systems are faced with essentially three solutions in bringing their functionality to the Internet. These choices range from creating a Webserver specific to the product functionality, Common Gateway Interface (CGI) scripts or writing to a Webserver Application-Programming Interface (API). Depending on the application, each of these has advantages and disadvantages.

At present there are approximately 120 different organizations that produce Webservers. These range in price from \$0.00 to \$55,000 (an overview of all the available Webservers and costs can be found at

<http://webcompare.internet.com/compare/chart.html>). This is a dynamic and fast-paced market where it is a considerable effort to maintain comparable features and a tremendous task to maintain a competitive lead. Several organizations have successfully entered into this market in order to bring Internet functionality to their core product. Notably, Microsoft, Oracle, Lotus and Information Builders have all developed capable and competitive Webserver. Although certainly this is a valid development approach, it certainly requires the resources that only the larger software companies can afford. Smaller software companies, looking for a more efficient allocation of resources look to the alternate approaches to Web enable their products.

Alternate approaches to developing Web functionality for larger client-server applications have been to develop CGI scripts or to write directly to a Webserver API.

Of these two approaches, CGI scripts can be the most rapidly deployed for an organization desperate to meet growing client demands for Web-enablement. These scripts typically do not require the deep understanding of the code beneath the Webserver program or even of the program that is being "Web-enabled." Additionally, the CGI scripts, typically written in PERL (Program Extraction and Reporting Language) have an advantage of platform independence, which can also save untold development hours. However, it should be noted that this platform independence is a function of program complexity and as the complexity of the features increases the degree of platform independence diminishes.

As large client/server applications move towards developing their features and benefits for the Internet, overall performance becomes a critical development component of the development solution. With each request made through an application developed with a CGI script, a separate session is started on the Webserver. For CGI applications supporting a large number of users making complex queries and analyses, server performance will quickly become a critical issue. This performance issue has lead many developers to work directly with the Webserver API to gain processing efficiencies.

For interactive Web applications that use the Webserver API to extend functionality there are tremendous processing efficiencies. These API requests are processed within the same session that the Webserver software is running, eliminating the multiple session start-ups that hamper server response times with CGI scripts.

By utilizing the Webserver API, developers create a direct link from a program to a Webserver. Typically this is accomplished through a dynamic link library file (dll).

To illustrate the power of the Webserver API advantage, Shiloh Consulting ran performance tests on various Webserver in February of 1996. In the case of the Microsoft Information Server, applications running on the API were five times faster than comparable applications using CGI scripts. Similar tests performed using the Netscape Netsite server provided performance increases of 100% on API vs. CGI implementations (for the complete study please see www.tedhaynes.com).

However there are challenges to the more efficient API approach. API's are proprietary for each specific brand of Webserver. For an organization that wants to ensure that all possible customers will be able to utilize Internet functionality, there is a minimum of 120 that need to be programmed for. This number increases exponentially with the number of versions available and for all the possible platforms. For practical reasons, few, if any companies actually proceed with this task and instead work with the API's of the more popular Webserver or the Webserver that meets the of their client base.

Today's developers have an array of options available to bring their program's functions to the Internet. Although there are many technological advantages and disadvantages in choosing one of these approaches, developers must also consider the perspective of the people who will actually use their products.

End-Users

While there are numerous client side applications utilizing Java, ActiveX and Plug-In technologies, few of these are designed to help users become more productive. In fact, the Gamelan Java directory lists 209 applications under the Business and Finance heading -- out of a total of 8,593 total Java applications. As more productive applications become readily available to business users, features such as easy installation, reliability and safety will become increasingly more important.

Of the technological choices of Java, ActiveX and Plug-Ins, none meet all the criteria.

While Java and ActiveX components each install without assistance from the user, Plug-In's frequently require finding a Plug-In directory or registering a new file type under the browsers options. While these tasks don't require programming knowledge, they do not provide the seamless operability sought by the majority of users.

Both ActiveX and Plug-In programs provide enhanced reliability because they reside on the users hard-drive after installation. Application functionality specifics are loaded when the Web browser encounters a WebPages with one of these components installed. By comparison, the complete Java applets or applications must be loaded via the Internet each time. With increased complexity, the load time for Java programs lengthens, thereby increasing the likelihood of transmission problems. Any interruption or interference with the loading of any of the classes of a Java program will prevent it from running correctly on the user's computer.

Java's restriction to the virtual machine of the computer memory offers users the safest approach to increasing the Internet interactivity of client side applications. The possibility of rogue programmers creating malicious ActiveX or even Plug-In programs has received a lot of attention in the press. News of intercepting Quicken account information or accessing other files on the computer's hard drive is especially troublesome for users. Although these instances are actually quite rare, the risk does present most individuals with a certain degree of fear and uncertainty in trying new Internet programs.

Client-server applications are currently providing the bulk of the truly useful Internet applications. These range from search engines to dynamic decision support tools. From the end user perspective the developers approach of a separate Webserver, CGI or API is invisible. What does matter is response time. As discussed previously, either a specialized Webserver or programming to a specific Webserver API will provide the best performance advantages.

The Organization

The technology environments for Web-enabled desktop programs present organizations with a number of problems from productivity and resource issues to security concerns.

The bulk of the Java, ActiveX and Plug-In programs used on the Web today are incorporated to enhance the design of the WebPages, improve navigation through a Website or, more likely, to create a fun interactive game. A recent survey of the Gamelan directory found that games outnumbered business and finance applications five to one. While the Internet certainly presents a countless number of distractions for employees, organizations tend to blame external factors when the focus should be internal in keeping their workers motivated and focused on their jobs.

Resource issues present more of an issue as ActiveX and Plug-In components start to invade hard-drives with increased frequency. While this presents less of an issue for Java client applications, as more and more powerful Java applications emerge the emphasis will be more on available RAM.

Finally, while the security issue is a mild concern at the user level, it presents more of a problem at the corporate IT level. Predominately this is the issue for ActiveX and Plug-In components. While the security concerns of ActiveX have been getting a lot of the exposure in the press, the same security issues are possible with Plug-Ins, however, as stated above the danger has yet to reach the proportions that pundits claim is possible.

For large complex client-server applications, organizations should pay very close attention to the

chosen development platform.

Although the creation of a new type of Webserver provides application performance benefits, there may be issues with employee productivity. Specifically, this becomes yet another program for IT to support and maintain. An additional consideration is the specialized Webserver's ability to implement security. Separate Webservers will handle security differently. This creates the potential of numerous logon Ids and passwords that will not only produce problems for end users, but also for system administrators.

Although lacking the better performance of the server or API approaches, the CGI approach does may allow an organization to add functionality to Webservers already existing throughout the organization. Additional drawbacks to CGI approaches in general do not allow for a tremendous amount of flexibility. Each additional piece of functionality that needs to be surfaced through the Internet will require a separate script. While an organization may be presented with a product that is "Web-enabled" they may find that the bulk of the functionality that they want or need is missing because it would involve the programming of additional CGI scripts. For the individuals working on these CGI enabled programs there is an additional learning curve in learning how to use these new scripts. Finally, most CGI scripts allow for executable programs to be run within the Webserver directories. This presents a significant security risk that should be closely examined.

Adding Internet functionality through writing directly to the Webservers API allows an organization to overcome all of the obstacles encountered by the Webserver and CGI approaches. Client-server programs that allow Internet functionality through a Webserver API may fit into existing IT Webserver standards, open most of the functionality of the client-server program and are bound within the existing security structure of the Webservers. The one drawback is that any given software company may not support the API of the Webservers currently being used by a given organization.

Conclusion

The Internet is revolutionizing the ways that businesses interact through enabling desktop programs and larger client server systems with Internet communications capabilities. The pace of programs becoming Web-enabled has accelerated as most programs have leveraged the advantages of the Internet. As developers take advantage of ActiveX, Java, and Plug-In technologies as well as Webservers, CGI scripts and server APIs, there must be an understanding of the impact of adding Internet functionality. Although developers make these technology choices, the end-users and their employers will also feel the impact of this choice. Only through understanding the affects of these technology choices will organizations be able to choose web enabled desktop programs and client-server systems that truly meet their needs.