CLASSIC
SYSTEM SOLUTIONS, INC.

Five Mistakes to Avoid When Implementing a Thin Client GUI Strategy

May 2000

By James Hobart, President, Classic System Solutions

The Myth: A thin-client solution is just a simple change to our existing user interface.

The Reality: A thin-client solution can be much harder to develop than a traditional client/server application.

Could it be possible? Absolutely. Here are three important reasons why and five mistakes that we often make.

Three Important Reasons

1. **Scalability and timing –** Today's thin-client application is often on a four to six month timeline, yet is expected to scale to thousands of users immediately.
2. **Technical challenges –** You must get a thin-client solution to perform both fast and reliably across a myriad of network firewalls, gateways, and servers. This can be a challenge even for the brightest network engineer.
3. **High user expectations –** Users have a 'browser mentality' that says that the software should be 'instantly usable'. This is a new usability goal for developers to attain, which often results in a reengineering of the existing client/server user interface.

Mistake #1: What's My Deployment Platform?

Choosing a deployment platform can be fairly difficult. As we migrate our existing applications from the client/server environment, users become accustomed to a 'robust' user interface provided by a GUI, with code deployed on the client. As we attempt to maintain this functionality, we often opt for deployment platforms that provide increased interaction at the expense of limiting our potential audience. For instance, a development team may choose a '100% Java' approach for their next generation application, only to find out that the decision may have been valid for the back-end and middle-tier pieces of the application, but not the front-end client. This may occur due to inherent flaws in forcing a Java client due to security, performance, and stability issues. There is no doubt that Java has come a long way in a very short time, however, many of our clients are still experiencing these issues when attempting to deploy a Java client in an enterprise application. They often end up deploying the Java application on each local machine to avoid the download performance issue, but instead create a new software distribution issue, which the original thin-client initiative was supposed to solve. This problem is not just associated with Java. We've seen

101B Sand Creek Road • Suite 209 • Brentwood, CA 94513
925.250.2550 • 925.516.9658 • www.classicsys.com

similar client deployment issues with projects implementing Microsoft®'s new ATL (Active Template Libraries) technologies or even with excessive JavaScript code for handling client-side business logic.

Choosing a 'browser-only' approach can also present a series of obstacles to success. Remember that the browser is a SDI (Single Document Interface) implementation that is not well suited for applications where multiple views, direct user manipulation, or multiple windows are required. Trying to force 'power' users, who have become accustomed to the flexibility of modeless windows, drag and drop, and other traditional GUI features to accept a new and improved browser-based web interface may be met with a great deal of resistance.

Mistake #2: Who Are My Users?

Creating a thin-client application opens up the possibility of deploying our application to a much wider base of users including our business partners, employees, and, of course, our customers. This dramatically changes the level of expectations that must be met with our software and creates a series of usability challenges that we must overcome. Gone are the days when you can roll out your application to a few 'friendly' users, get some feedback, and then make adjustments for both usability and performance. The inherent nature of web-based applications is that they are immediately available to everyone and normally installed without any training. The result of this is often skyrocketing support desk calls and user frustration. I have heard comments from developers such as "If only I could go out and train those customers" or "Put an IQ test on the home page to filter out the dumb users" as ways to overcome this issue. Good user interface design and an effective usability testing process are much more likely to lead to success. When we do not know, at a detailed level, who our users are and how they work, it becomes nearly impossible to develop effective user profiles that could be used to tailor our thin-client interfaces for the users' specific needs and interaction requirements.

Mistake #3: Just Web-Enable Our Client/Server Application...

We often assume that the success of a current application can easily be re-deployed with a new 'web front end', requiring very little work on the middle and back-end tiers. This approach can work successfully only if you take into careful consideration a number of factors including visual interaction, existing navigational model, and existing client desktop platforms. With an HTML-based interface, the standard 'rich' graphical controls such as tabs, tree, and list views are much more difficult to implement effectively. In addition, direct user manipulation techniques, such as drag and drop, are nearly impossible to implement even with Java and DHTML.

Navigation models are very different for web-based applications as compared to traditional client/server GUI's. This is a result of the limited scope of the typical navigation controls available and the hierarchical nature of most browser-based solutions. Normal methods of 'flattening' out navigation, such as multiple modeless windows, also pose challenges since most users do not like having windows launched from their existing browser window. In fact, the navigation models of most mainframe systems are more similar to web-based applications, with their typically deep hierarchical navigation design.

101B Sand Creek Road • Suite 209 • Brentwood, CA 94513
925.250.2550 • 925.516.9658 • www.classicsys.com

Mistake #4: Let's try these cool, new tools!

The reality is that development tools are changing constantly and that keeping abreast of the latest technologies is challenging for even the most advanced software engineer. As a team, you will need to set some limits on what will be supported for your project implementation. This will require technology decisions, regarding such issues as browser support, XML, style sheets, and the use of technologies like DHTML and Java, to be made early. We try to avoid options that automatically generate all of the client-side code, as these solutions often limit your deployment platform choices or your usage and placement of the GUI controls. Your technology decisions will often affect the user experience. Developer training is also an issue. Object-oriented languages, such as Java, often take 6 to 12 months to learn and master. Many leading edge companies are creating hybrid designs that take the best aspects of a browser and the flatter navigation of the traditional client/server application to create a new breed of 'weblications'. This approach can often be implemented with traditional client/server tools that generate HTML and browser-friendly code, while letting the developer work in a Windows®-friendly environment such as Visual Basic™ or Powerbuilder™, which are now enhanced for web-based development with libraries such as Microsoft's ATL (Active Template Libraries).

Mistake #5: Inconsistent Look and Feel

One of the great advantages of a thin-client solution is the ability to inexpensively deploy your software to a much larger user base. Novice users often learn in a cognitive mode with little to no formal training. This type of learning relies on past experience and behavioral consistency to be effective. Software that has an inconsistent look and feel often results in significant support costs and rejection by the users to which it was intended.

To complicate matters, the browser is not very friendly to the concept of 'applications'. When in a browser, we don't select from a 'Start' menu to launch another thin-client solution. Instead, we simply click on a link and the content is displayed. Since all of the application content is only 'one click away', your design must provide a unified workspace where content, rather then the traditional menu structure is the main navigation method. To solve this problem, developers need design standards and patterns for different classes of users (novice, expert, etc.) and different deployment platforms.

The guidelines need to address implementations with both 'rich GUI' interfaces and more constrained web-based interfaces. Collecting and sharing this design knowledge should be accomplished via a web-based knowledge repository that allows for easy update, access, and dissemination of the design patterns developed within the organization.

Implementing a thin-client application can result in a highly scaleable solution that dramatically reduces deployment costs for your users and provides a consistent, easy to use interface for even the most novice users. Just realize that the path to success has been traveled by others who have made these mistakes. If you learn from them, you will have a much better chance of achieving success with your thin-client migration efforts.

101B Sand Creek Road • Suite 209 • Brentwood, CA 94513
925.250.2550 • 925.516.9658 • www.classicsys.com

About the author:

James Hobart is an internationally recognized user interface design consultant based in California, USA. He specializes in the design and development of large-scale, high-volume client/server and web applications. He is an expert in GUI design for transaction processing systems and strategies for migration to thin-client graphical user interfaces. He can be reached at jimh@classicsys.com

Find more articles on Usability at www.classicsys.com

101B Sand Creek Road • Suite 209 • Brentwood, CA 94513
925.250.2550 • 925.516.9658 • www.classicsys.com