

Critical Mission

If you would like to view previous newsletter articles, visit our web site at www.AltoConsulting.com.

FEATURED ARTICLE

By
Craig Yellick, Vice President



Silverlight 3.0 *Ready for Business*

Silverlight version 3.0, released in July 2009, offers a complete set of business-oriented features and capabilities that make it the obvious choice for all web-based software applications. The new offline “out of browser” mode makes Silverlight a direct replacement for traditional Windows applications. Cross platform, cross device, cross browser support for Windows, Macs and Linux – this is THE development tool for 2010 and beyond.

Third Version of the Third Generation

It's a cliché with a ring of truth – it usually takes Microsoft until version 3.0 of a product before it's ready for prime time and Silverlight holds up this tradition. Interestingly, Silverlight 3.0 is the third version of the *third generation* of Microsoft's web development technologies which makes it doubly cliché-ready. The first generation Microsoft web development technology from 1998, called ASP, is a rough first attempt with poor performance, weak developer tool support, lousy debugging and security; you name it. We were all thrilled with the 2002 release of ASP.NET, which solved all of the problems with ASP and then took it to new heights with AJAX in 2005. However, even with AJAX, ASP.NET is still at heart a very large and complex layer on top of the HTTP POST and GET methods in a web browser. ASP.NET is judged “much better” only in comparison to the horror of ASP.

What really sets Silverlight apart is that it is an **entirely new approach** to developing web applications.

A Fresh Start

Silverlight frees developers at all levels, from data entry form programmers to complex control designers, from the tyranny of a transport protocol developed back in 1996 and a markup language devised in 1991. Ask any programmer if they think HTML, Javascript, cookies, query strings, hidden form variables and manual session state management are a good foundation for a million-dollar software project. As solid and productive as ASP.NET can be, it's still a victim of a weak operating architecture.

Silverlight is the result of the decision to start over and apply all the lessons learned from the previous two generations of web development technologies. The result takes advantage of modern browser capabilities and advances in client workstation capabilities.

You can read elsewhere about the evolution of Silverlight from version 1.0 to 2.0. This newsletter is about what's new and better in version 3.0 such that **all businesses both large and small** ought to be planning their new applications around this technology.

We're not Talking About...

If you skipped looking into earlier versions Silverlight it was probably because all of the attention was on the flashy, colorful, textured 3D effects and streaming media that appeared to be central to every demonstration. This impression is based in fact because Silverlight 1.0 was pretty much limited to constructing fancy video players and picture viewers. Version 2.0 added a lot more “business-y” features but was still seemed to be all about the flash. Rest assured that all of that history is still present in version

3.0, with a whole lot more. You can easily make modern, attractive and highly functional applications – we just won't be talking about that, here.

Data Binding

Data binding is decidedly *not* flashy, and in fact totally invisible to the end user, yet this is arguably one of the most important features that make Silverlight 3.0 ready for business.

Data binding concerns itself with 1) getting data out of a data source, 2) on to a form and presented in text boxes, lists and grids, 3) allowing editing and validating changes, and 4) getting the new version back to the data source. All four steps are well-supported in Silverlight 3.0. The first and last step are well understood and have been present for a long time in the .NET Framework.

The middle two steps are where Silverlight 3.0 really shines. Element-to-element binding allows one user interface element, for example a list, to be declaratively bound to the state of another element, let's say a checkbox, along with a comprehensive *declarative* validation mechanism where you define business rules in one place and have them applied wherever the data is used. This was all but impossible in version 1.0 and required a lot of tedious programming in 2.0. Complex data binding, event routing and validation are built into 3.0 and **accessible without programming**.

Out-of-Browser

Prior to 3.0, Silverlight applications were served up from a web server and ran inside a web browser window. If you closed the window the application went away. In version 3.0 you have the option of allowing the application to be stored locally to the user's workstation and launched from the Start menu or a shortcut. When launched this way the web browser is not present and the application has the feel of a local window. The application will run **even if there is no Internet or network connection**. The application receives status notifications of network availability so it can be aware and alter behaviors. For example,

offering to sync local storage (discussed later in this article) with a central sever only when the network connection is present.

Versioning and Updates

When installed locally a Silverlight 3.0 application can check automatically and asynchronously for updates on every launch and updates can be automatically installed. Running instances of the application are alerted when updates become available. This is a critical feature when you consider the support nightmare of having hundreds or thousands of users with slightly different versions installed on their workstations and no ability to detect much less force an update for bug fixes or improvements.

Deep Linking

Unlike a typical web application, Silverlight is not based on a large collection of individual web pages but rather a series of windows and forms that appear from a single .ASPX web page. This makes it difficult for a user to close the application and later return to the same form state containing the same data. Silverlight 3.0 offers a new development template called a Navigation Application where the web browser's URL and query string can direct the user to a specific application state and data. The query string is a logical construction and has nothing to do with the physical organization of the application or data.

Local Data Storage

Silverlight applications have always had access to Windows isolated storage, a protected, sandboxed allocation of local file system storage. Silverlight 3.0 extends protected services out to the entire local file system to which the user has access, making it possible to load/save data to the desktop, My Documents folder and so on. The Silverlight application is restricted to accessing files under the user's control through a File-Open dialog, and is not allowed to see the physical file path or otherwise rummage around the file system.

Local Connection

If you divide a large Silverlight solution into multiple independent applications they can now communicate between one another directly, rather than having to make a roundtrip to a common server. This encourages a modular approach to design and simplifies versioning and updates of large applications.

Development Tools

Developer tools and productivity lagged in the first release of Silverlight, requiring a significant commitment on the part of the developer to learn arcane XML markup and Javascript coding without any assistance. Silverlight 2.0 was greatly improved from a visual tools standpoint but still required a lot of unassisted programming to perform common actions like data access, control binding and validation.

The Silverlight 3.0 developer experience is now comparable to what ASP.NET developers have come to expect from Visual Studio 2008. Common application implementation tasks are now fully declarative and assisted by IntelliSense along with intelligent dialog windows and property settings. Coupled with the much more streamlined and rational application platform, you'll find that developers are substantially more productive and spend more time adding line-of-business value rather than implementing basic plumbing.

A good example is **Exception Handling**. Silverlight applications use Web Services to send and receive data over the Internet. Prior to Silverlight 3.0 when a web service ran into trouble and faulted, the developer was presented with a terse and entirely useless error message. This made debugging and testing a slow and ponderous process. In Silverlight 3.0, web services and applications can be configured to pass detailed exception information so developers know exactly what happened, where, to cause the exception.

Another example is the way Silverlight streamlines and simplifies the process of hooking up **web services as data sources** to which you can bind. Prior to version

3.0 this was a tedious process involving a lot of coding.

Your Existing Investment in ASP.NET

Your organization probably has a huge investment in ASP.NET + AJAX web applications and is in no position to abandon all of that, or even to attempt a wholesale conversion.

A more reasonable approach is to design and implement all *new* projects in Silverlight 3.0 to gain the benefits of a much faster and more efficient development process and greatly improved user interface experience.

Rather than seeing this as a bright line of separation between your existing ASP.NET investment and your new Silverlight applications, you can take advantage of the Silverlight 3.0 **HTML Bridge** features and smoothly integrate Silverlight functionality into your existing ASP.NET applications. This includes client-side Javascript making calls to Silverlight-based methods, HTML form and control events attached to Silverlight-based event handlers and much more.

Wireframes and Prototyping

This overview ends with a new product, called **Expression SketchFlow**, that operates separately but in parallel with Visual Studio to make the application design process more streamlined and efficient.

A major obstacle to delivering complex yet attractive and highly functional web application is the near total disconnect between designers and developers. This disconnect occurs because designers and developers use totally incompatible tools (for example, Photoshop versus Visual Studio) and their sole point of contact is a one-time hand-off of a static, graphical representation of the user interface, often called a *wireframe*. As users and stakeholders watch the application come to life and begin to steer the user interface the application diverges from the wireframe and steadily becomes more difficult to influence in a meaningful way without a re-write.

Instead, SketchFlow is used by designers to rapidly create an interactive, dynamic model of the user interface, including realistic sample data. The tool uses the same XAML user interface markup as used by Silverlight so you're **guaranteed** that anything a designer dreams up in SketchFlow will be implemented directly in Silverlight.

It doesn't stop there. A SketchFlow project can be packaged up as a single self-installing *SketchFlow Player* that is **distributed to the target users** and other stakeholders. They can interact with the prototype and add notes and overlay graphical indicators where things should be moved or problems that need to be resolved. This input is saved and returned to the designers who aggregate the commentary from all users, make revisions, and repeat the cycle. Developers then receive a very detailed and tested specification that can be directly mapped to a starting point for the application's user interface.

Summary

Silverlight 3.0 is ready for doing real business. Compared to the alternatives, the development process and supporting infrastructure is rational and highly efficient and effective while the user interface experience is vastly superior. Couple this with the out-of-browser/offline capability and easy update management process and you have a very compelling platform for your next development project. You don't need to justify using Silverlight; you have to justify *not* using it!

Contact Alto if you'd like a demonstration of any of the products or techniques discussed in this article. Our next solution briefing is devoted to these concepts.