From: www.cio.com

How to Evaluate Cloud Code, Part 1

– David Taber, CIO

October 13, 2011

Cloud-based applications tend to focus the users' creative impulses on what the vendors are calling "declarative programming:" point-and-click specification of behavior, plus configurations, settings, rules, and formulas. These are good choices for ease of use, containing multi-tenant workloads, and testability. But if your cloud app has large user counts or sophisticated use cases, declarative development will take you only so far: soon enough, you'll be merrily coding in both the presentation layer (view) and the business logic (controller). In some cloud environments, you'll even be working directly with metadata and the transaction layer (model).

Of course, if you're using a cloud-based development or deployment environment, you'll be working with all three of the MVC layers from day one. For IT pros, that's nothing new.

But the world of cloud development does present some new challenges, because it's still very much in the early stages. For example:

• **There's no center:** Not only are the users decentralized, the developers can be anywhere and the services are in several locations. There's no central repository for the application model, code, test databases, or application artifacts unless you create one with your favorite version control system and wiki. So...go do that.

• **You'll typically be working in several languages at once:** Even when working within just one application, you'll be using plenty of cross-language references. For example, your code might call java libraries, internal application functions, and scripts. Your UI may call jquery, JavaScript, JavaScript frameworks, native file-system utilities, AJAX, Flex, and other languages. This is very powerful and efficient, but it can lead to code that is very tough to troubleshoot and debug. Try to keep the total number of languages in use to less than 5. Go ahead...try.

• **Different developer tools will affect the code structure:** This is true from analysis and design through to test and deployment: some tools allow (even encourage) much more elaborate and extensible coding structures than others. Some of the fancier tools are available only on windows, so you're Mac and Linux-based developers will have a different experience of your system, both at development and deployment time. This goes double for debugging. And double again for UI.

Cloud Code Evaluation Criteria

Assuming that the cloud code functions and meets the user requirements, how else should code be evaluated? Let's look at some basics:

• **Performance:** In a cloud application, this is typically measured by "screen refresh time," as that's what the users will perceive as "time to complete." Do this evaluation on a typical user machine (say, 3-year old laptop), not a 3 GHz quad-core behemoth with 16 GB of RAM. Two elements typically dominate perceived performance: the length of time before the cloud's servers respond to requests, and the screen-paint time. The three killer things to look for when evaluating cloud code: silly queries, excessive network chattiness (particularly when several Web services have to be called), and overblown page length or view state.

• **Deployability:** While a bug-fix for an individual module could be pushed in seconds, the right thing to do is a fairly comprehensive test cycle before any deployment (for production systems, Salesforce.com even requires this). With large applications, though, the deployment may include modifications to metadata, configuration tables, artifacts, and test data in conjunction with the code. Of course the deployment can be automated with a CVS system and scriptable deployment engine. And the test sequences can be automated as well. The real questions are: is that automation actually in place, and does the test cycle complete in a reasonable time? One system we worked on recently took hours to run the compulsory test cycle. This made the application operationally problematic, and we highly recommend screening vendors on this axis.

• **Depth and Breadth of Documentation:** Call me Don Quixote, but cloud apps need to have more documentation than traditional apps do. And they have to be documented in a different way. We're all still on the learning curve, so make sure to evaluate both purchased apps and integrators' code on this basis.

And Then There's the Big Kahuna: Maintainability

Not to sound like a consultant, but "it all depends on your situation and what you're trying to achieve." For example, if you're simply using a commercial cloud application, you won't be trying to maintain it yourself. So "maintainability" translates into "SLA." (If you're using an open-source cloud application, it translates to "how available are consultants who know this code base?")

But if you're developing your own cloud application or extending an existing one with pages, triggers, and classes, you need to evaluate the code for ease of analysis, extension, and troubleshooting. A common misconception is that it's always best to have the most generic, extensible code possible. While extensibility and avoidance of hard-coding are good things, taken to extremes these practices can be indistinguishable from code obfuscation. Make sure to evaluate the code's "understandability index" by having your current staff attempt to do a code walk-through on their own. If the coding techniques are so abstract as to be opaque, you'll be forever dependent on the integrator who originally created the code.

Next week, we'll follow up with 15 tips to avoiding code breakage -- practices that can save you trouble and time, as your application requirements evolve.

David Taber is the author of the new Prentice Hall book, "Salesforce.com Secrets of Success" and is the CEO of SalesLogistix, a certified Salesforce.com consultancy focused on business process improvement through use of CRM systems. SalesLogistix clients are in North America, Europe, Israel, and India, and David has over 25 years experience in high tech, including 10 years at the VP level or above.

Follow everything from CIO.com on Twitter @CIOonline.