From: www.cio.com

Avoid 3 Cloud App Development Sand-Traps

– David Taber, CIO

March 23, 2011

Somebody once said that cloud apps are just like enterprise software, only more so. OK, nobody ever said that. Actually, most everyone says they're much easier and faster than traditional software projects. And done right, cloud projects can be. Because done right, they are smaller, simpler, and more separable than tightly-coupled software projects were.

But done wrong, they have all the pitfalls of large software projects...plus some new ones you never had in the client/server world. Because in cloud projects, you are typically trying to leverage existing services (in mashups, RESTful calls, or SOAP interactions) that you don't control. The more of those existing, external services you use, the higher the likelihood of surprises that are rarely pleasant.

In recent weeks, I've come across cloud projects that have been burdened with executive mandates that practically guarantee failure. Watch out for decisions or statements that follow these patterns, as they increase the risk of cost, schedule, or quality problems:

1. "We can't give you guidance on the priority of deliverable items: they're all hard requirements!" As an executive, it's easy to fall into this kind of thinking...particularly if you've already sold your customers and constituencies on the benefits of the project. But the reality is that tradeoffs will be made during the development, test, and deployment of any project — the only question is whether they will be hidden/political/random or open/rational/economic. There are no two requirements that are of exactly equal cost or equal value/benefit to the business. So to pretend that they are of equal importance (or worse, of infinite/non-negotiable importance) leads to distorted business decisions, at best. At worst, this kind of dictate leads to teams that cannot make decisions and will not communicate bad — but real — news up the chain of command. The end result is an "everything's fine until the last minute" syndrome, which classically leads to nasty surprises at the end of the project.

In cloud projects, the real opportunity is to have quick, low-cost work-arounds that provide a bare-bones feature in early iterations, and to gradually expand the feature coverage in later project cycles. The all-or-nothing mandate makes you miss this opportunity, costing the project time, money, and risk. If Agile projects teach us anything, it's that requirements should be treated as malleable and adaptable to user feedback. This goes double for CRM projects, where the usability of a feature can be more important than the underlying function (because without user adoption, the feature will never take root).

2. "We need to have all parts of this project on a common tool, library, or infrastructure element." Of course you want to keep the number of moving parts to a minimum, and it's important to have as few learning curves as possible. But cloud projects need to be flexible — because things move fast out there. You may discover that one version of a PHP library works really well with one cloud service, but is not supported by another cloud service. One payment gateway's API may have a detail that is not supported by the gateway your firm has standardized on. You may need to use SOAP,

REST, and naked XML interchanges in tandem.

A key advantage of the cloud approach is it provides a lot of insulation from the underlying implementation details. So don't overdo it with the "one size has to fit all" directives.

3. "We don't have time to do user testing along the way. Leave that till the very end." This is a classic for people with a waterfall, command-and-control project management style, because it satisfies their thirst for certainty. But that comfort is illusory. You don't have to be a Demming devotee to know that product cost, schedule, and risk are reduced by driving quality into every stage of your development (manufacturing) process, not tacking it on at the end. To postpone user testing is to fall into traps described in the literature over decades (from the *The Mythical Man Month* to *The Agile Manifesto*).

In cloud projects, this issue is amplified by the fact you are leveraging services that you neither built nor control. The cloud services may do exactly what your users want...or they may not. Until you actually do a trial deployment of your cloud functionality and all the elements you're integrating from outside, you won't know that the project meets user needs. For example, a simple map mash-up may work fine with a single icon, but be hopelessly slow or visually messy when trying to paint "my closeby customers."

When leveraging public cloud services, you may not discover reliability or performance issues until you've run things under a representative load for a couple of weeks. During unit testing by developers, everything may look fine — but that's because they're doing their tests at 2 in the morning. Try the public clouds during busy hours, and you may discover that you need to switch to a different service. This can be a costly discovery if it's late in the project.

This goes double for CRM projects, as the value of the system comes as much from the data in the system as from the software functionality. Any user test that's done with faked data is almost pointless — and will give a false sense of security. I strongly advise that the user tests be done with a big chunk of real data from the very beginning, and that final acceptance testing not start until the entire data set is in the system. Too often, you won't see the bugs and shortfalls until users are working with the real thing.

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.