CIO

Print Article Close Window

From: www.cio.com

How to Document Cloud Design Decisions

– David Taber, CIO

January 26, 2012

When developing and integrating cloud systems, the public interfaces and external "contracts" among services mean that design and architecture can evolve rapidly and in parallel. But when they do and the teams are not in the same room, this speed is an invitation to chaos. As two teams work on opposite sides of an interface (the service provider and the service consumer), it's easy for the teams' definition of variables and methods to fall out of sync. Of course the service provider team could update its document and notify the other team about a new semantic of a field value or behavior of a service. But the reality is too often that they don't, and the classic problem of distributed version control rears its ugly head.

Let's start with some assumptions about cloud design and architecture:

1. The teams value speed and smarts over rigor and process.
2. The teams use an Agile process that stresses iterative development, with continuous integration and testing cycles.
3. The teams don't have a strong methodological bias, and they are highly pragmatic. In particular, they don't insist upon UML or other specific modeling and interface definition tools.
4. The teams are likely to be spread across several organizations are definitely and not centrally controlled.
5. The team members are not within shouting distance of each other.

So what are the implications here? Generally speaking, the approach for design coordination will tend to depend on lowest-common-denominator tools. (Yes, there are agile toos, but their use is spotty at best.) So Microsoft Word, Excel, Visio and even Powerpoint will be likely starting points. Unfortunately, most of these tools have weak change-management features: Even Word's change-bar feature can be tough to unravel if there have been many people changing things.

One common approach is to store these files in Dropbox, BaseCamp, or similar file-sharing and project repositories. This certainly helps, as files and changes will be in full view of the team. If you're lucky, the file-sharing system's file locking actually works, and there might even be a solid versioning function. But, in most cases, it only really works when every change is stored in a file with a different version that follows a strict naming convention (along the lines of 10Jan11@DTaber#8NewFieldsV2_32).

And even that doesn't work if two teams have been working on a piece of technology at the same time. So there needs to be some sort of check-out protocol to prevent "update collisions." If your file-sharing system doesn't fully enforce file locking, a work-around is to have the latest version of checked-out files relevant file be empty, except for the identity of the person/team working on it.

Stop Partying Like It's 1999

If all of this smacks of old thinking and silly manual procedures it's because, well, that's exactly what it is.

One more modern approach is Google Docs, which provides a collaborative spreadsheet, drawing, and text-document system that supports distributed real-time evolution of documents. Particularly in the course of conference calls, Google Docs provide an easy-to-use, platform-independent, and free way to collaborate. The revision history allows inspection of at least a year's worth of versions, making it easy to see how the documents have evolved. The document can be reverted at any time to its previous state, in case changes need to be "backed out."

However, there isn't a way to rapidly document spot changes: There isn't a "visual diff" across versions, and I haven't found a way to export the revision history for use by analytical tools (although in theory it's possible using Google's v3 API feeds).

More importantly, Google Docs have their magical qualities only when viewed from within a Google account. For companies that need to control everything behind their own firewall or have other security concerns, the free Google Docs are off limits.

In these situations, we recommend you use a Wiki: It provides the shared documents, infinite revision history, and collaboration tools (such as "Talk" pages) that help distributed teams stay in sync even if they're in different time zones. With this collaborative power, however, come limitations: weak tabling features (feels like HTML hacking) and nothing in collaborative drawing.

Another relatively new tool is Chatter, which can be an effective way of collaborating within Salesforce.com. While it requires a lot of discipline to be effectively for engineering decisions (through the careful construction of Groups and hash-tag topics), it has the benefit of being free and supports versioning of attached documents.

What Doesn't Work

Three ideas for design collaboration it's time to give up on: email, recordings of design meetings, and paper. Let's just make that a Chinese New Year's resolution.

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.