A Trip Without a Roadmap

Commentary on Queue's RIA development case study, "Making the Move to Ajax"

Peter Christy, Internet Research Group

The associated account of the efforts made at SalesCentrix to develop a new RIA capability for an existing product is part of an effort by ACM to create something analogous to those case studies that have become so central to the modern business school experience. It's a formidable challenge. For one thing, most business school case studies at least include references to supporting documents. Often, even the underlying financials can offer some valuable clues. In this instance, however, the company in question has vanished, leaving us with nothing more than this case study to evaluate. Thus, it's impossible for me to distinguish between observations on the technical choices that were actually made by the company and the manner in which those choices have been represented in the case study. So if any of the original participants should have occasion to read this, I apologize in advance for any misunderstanding.—PC

Viewed broadly, programming projects fail far more often than they succeed. In some cases, failure is a useful step toward success, but all too often it is simply failure—effort wasted (apart from leaving the participants somewhat smarter than before, hopefully).

In this case study, the seeds of failure are all too obvious, at least with the benefit of hindsight. The two warning signs that really jump out are the absence of any coherent statement of requirements or goals (note, for example, that the notion of performance appears to have been an afterthought) and the fact that the team chose a technical approach they had few qualifications for—namely, writing PC software while avoiding any use of Microsoft technology. So if the job here was just to explain failure after the fact (much like explaining why the stock market went up or down on any given day), we could just stop right here. But perhaps there's something we can learn from the experience just the same. Let's focus on the issue of requirements.

In that respect, the case study starts ominously: "SalesCentrix, a small start-up in Vancouver, realized in 2006 that something would have to be done with its SaaS (software-as-a-service) solution for synchronizing data between QuickBooks and Salesforce to let users take better advantage of the Web." The user might need to do a task faster or better, but in this case the requirement can't be established literally as "to take better advantage of the Web" (although that does seem accurately to describe the path the company took). The case study then quickly jumps to the conclusion that the right way to go about this would be to build an RIA (rich Internet application). Certainly, RIA is one way of using the Web.

Salesforce is the successful pioneer of commercial SaaS, a form of application delivery that can only grow in importance over time. Salesforce got a lot of the important architectural issues right (e.g., a highly reliable, multitenant database system with metadata-based personalization), and it managed to choose a market (customer relationship management/sales management) where the Web and SaaS are particularly attractive because of the mobile nature of the target users.

Small businesses were Salesforce's initial focus since they couldn't realistically run their own

CRM systems internally. Given that many small businesses use Intuit's QuickBooks for accounting, the SalesCentrix idea of facilitating the coordination of QuickBooks and Salesforce is also one that seems to make good sense. For a new Salesforce user, QuickBooks is often the best source for a lot of customer information. Moreover, the ability to pre-populate Salesforce with QuickBooks data is clearly a good idea—as is the ability to continue that synchronization such that once a prospect converts to a customer in Salesforce, a new entry is automatically created in QuickBooks.

What isn't clear is which part of this problem SalesCentrix was hoping to address through RIA to let users take “better advantage of the Web,” because the case study never really touches on these issues. This leads the skeptical technological anthropologist to the plausible hypothesis that the SalesCentrix effort quickly turned into a matter of technology looking for a problem to solve (an all-too-common occurrence). To be sympathetic to the engineers, RIA and Google were quite hot at the time that this effort got under way, providing more than enough motivation for the SalesCentrix team to be interested in using them. But to what end?

Here the case study leaves the technological anthropologist who happens along later precious little to work with. Getting synchronization set up is known to be a thorny problem when the goal is synchronizing Outlook to some random mobile phone running some version of the firmware. In this particular case, the necessary generality of the SalesCentrix solution (with many versions of Outlook running on essentially infinite mobile platform variants) would clearly make it very challenging to help users get the details right.

But that isn't the problem here. Certainly there is some QuickBooks version sprawl to account for, but it's probably less than what might be expected simply because changes in tax laws and reporting requirements serve as relatively strong motivators for QuickBooks users to keep up to date. More important, however, at any moment in time there is precisely one version of Salesforce simply because it is delivered as SaaS, meaning all users are on the same system and are updated at the same moment. It's just not clear to me why getting the setup right is a problem that in turn requires the user to “take better advantage of the Web.” The technological anthropologist is once again left thinking that this smacks of (interesting) technology in search of a problem.

What's to be learned from this? The lesson probably has to do with the value of product marketing and product management. Sadly, programmers can't be relied upon to spend adequate time thinking about the needs of software users. Some developers believe they know more about what the users should do than the users themselves do. Some focus on what they have built rather than how it will be used. An effective product manager can bridge that gap by reflecting the user requirements back to the development team. Optimally, this dynamic has considerable give and take. Innovation and design are usually best driven by the engineers because they can best see how technical progress enables new solutions (RIA is a good example). The product manager can then relate that innovation to the challenges and opportunity faced by the users. The answer lies not in relying on either perspective in isolation, but rather in effectively blending the two perspectives.

How are we to avoid replaying this sad history? A good start is to not imagine what users want or need, but instead to talk with them about it. Try to walk in their shoes and understand how your software might help make their lives better (or not, as in this case). If you don't have time to do that yourself, then you should at least work with good product marketing people and listen to what they have to say. ☐

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

PETER CHRISTY is one of the principals at the Internet Research Group, a consulting firm focused on Internet ventures.

© 2009 ACM 1542-7730/09/0400 \$10.00