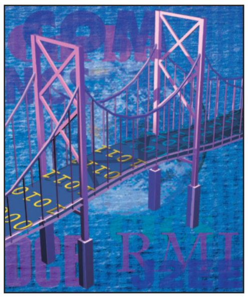

The Social Side of Services

Steve Vinoski • *IONA Technologies*

I recently read a weblog at ZDNet.com (http://blogs.zdnet.com/service-oriented/?p=479?=rss&tag=feed&subj=zdblog) that announced a poll had found that one out of every seven efforts to develop a service-oriented architecture (SOA) fails. My response, like that of many others, was to wonder about the poll's details and validity:

- What were the poll's criteria for success or failure?
- Were the statistics behind the poll sufficient for drawing such conclusions?
- What were the respondents' qualifications with respect to SOA development?
- Isn't one in seven actually a pretty good rate, considering that the failure rate for general IT projects is typically accepted to be at least as high as 50 percent?

Although this particular poll's methods and conclusions were dubious, the core question — whether IT developers are succeeding with SOA — is important. After all, simply employing service-oriented approaches won't guarantee that your project will succeed. On the other hand, there's no guarantee that you'll succeed even if you already have a firm grasp of the technical issues that you'll face with SOA. Technologists like to believe that projects live or die based solely on whether the technology behind them works well or not, but that's rarely the case. Nontechnical factors have a far greater influence on project success than we technologists would care to admit.

A Gathering of Services

It's hard to argue with the principles behind service orientation. Few would propose that properties such as encapsulation, information hiding, separation of interface and implementation, low coupling, and high cohesion are undesirable. Why, then, can it sometimes be so difficult to convince your organization to adopt service-oriented approaches?

If services could have human qualities, you'd want them to be gregarious and extroverted. After all, they can't stand alone. They fare best when they're part of a larger collection or network of services. When such a network exists within enterprises or organizations, it implies that they've accepted service orientation as a desirable approach to solving their IT problems. The alternative — in which services are few and far between — means that the organization has neither fundamentally adopted service orientation nor embraced its underlying principles. The few services that crop up in such settings are likely to wither and die.

Another way of stating this is that SOA success is based, in part, on achieving a critical mass of useful services. The technical line of reasoning for reaching and managing this critical mass typically goes like this:

1. For services to operate as a collective, they have to know about each other.
2. For services to know about each other, they must either be hardwired together or be able to dynamically find one another.
3. Hardwiring would be bad, as it implies high coupling and potential difficulties in replacing one service implementation with another somewhere down the line.
4. To facilitate dynamic discovery, then, services need a place that they can advertise themselves and meet other services.
5. Of course — a registry!

This line of reasoning usually fails because it ignores the hard problem of how services are created. It assumes that development teams will bring new services into being and that, because those services will want to interact with each other, the real problem is just providing a state-of-the-art registry to support those interactions. Over the years, I've seen plans and proposals for some pretty sophisticated registries, usually involving registration and query capabilities for all kinds of extensible service properties. However, few registries ever provide all the functionality their designs call for, and the few that actually do are usually so complicated that nobody uses them. Most of the service-oriented systems I've seen in production use pretty simple name-based service registries, in which applications find services only by name.

The problem is that service networks are, and will be for the foreseeable future, created by people, rather than by the services themselves. In other words, succeeding with SOA isn't just a matter of getting your services up and running and letting them interact via a registry. If you want to succeed with SOA, you have to work the human side of the equation. Otherwise, all the technology in the world won't help you.

Getting Buy-In

IT used to be a specialized and experimental enough area that it could get away with being the "tail that wags the dog." IT projects didn't always have to line up with business objectives, or even deliver useful results. But in today's post-dot-com era, companies expect IT to contribute to the bottom line, like traditional areas such as manufacturing, marketing, and sales. Ultimately, organizations that look to adopt service-oriented approaches do so because of the potential for reduced costs and added value. However, such prospects aren't tangible enough to address the people side of the SOA equation. They provide only a big picture assessment that not everyone in an organization can fully relate to or appreciate.

Pushing your organization to adopt service-oriented approaches and create a production SOA network requires a multipronged effort that depends somewhat on where you fit into the organization. Based on my own experiences, I can say that different barriers to SOA exist at different levels of the company. If the descriptions that follow come across negatively, it's only because I'm focusing exclusively on real-world hurdles you might run into.

Trench-Level Barriers

Developers typically focus only on the technical merits of proposed changes. Unless their current approaches and processes aren't working at all, developers, like most people, tend to resist change. They typically analyze any newfangled SOA proposal for holes or other evidence indicating why the plan couldn't possibly work. Of course, this generalization isn't true of all developers, given that some always want to be the first to try everything. You're probably one of these bleeding-edge developers yourself, given that you're the one pushing to adopt SOA. If so, don't fall into the trap of thinking that everyone is an early adopter like you. Be aware that many developers are actually technology laggards or conservatives who generally won't adopt new technologies or methods unless they absolutely have to. Alternatively, some developers suffer from the "I have a hammer so everything looks like a nail" syndrome, and so will likely object to SOA adoption if they deem it to be outside their area of expertise. Still others might feel they're already struggling to keep up with the rapid rate of technological change, and they'll view your proposal as yet another interference with their productivity.

Nontechnical factors have a far greater influence on project success than we technologists would care to admit.

Mid-Level Barriers

Middle managers and technical leaders tend to fall into one of two distinct categories. If they already have a good reputation within the organization for delivering on time and within budget, they might view SOA adoption as too risky. Secretly, they might also view it as a challenge to their own tried-and-true methods or as an undermining of their existing span of control. If, on the other hand, they're looking to make a name for themselves, they might jump eagerly at the chance to help champion SOA and prove that it can save money or increase productivity. Succeeding with SOA could pave the way for them to achieve promotions up the corporate ladder, added responsibility, and higher earnings.

Barriers at the Top

Upper management looks to the bottom line. They'll want to know how much SOA adoption will cost and what they'll get in return for investing in it. They'll be unlikely to care about the precise technologies being proposed, preferring to leave such details to those in the trenches who are paid to worry about them, but they won't adopt SOA or any other new approach unless they understand how it can help them deliver on their business objectives.

Hierarchical Differences

As you see, a lot depends on where you fit into the organization. If you're the chief information officer and you've convinced the CEO and other key executives that adopting SOA will benefit the company, then you can roll it out in your own department as a mandate. If you do this and SOA is a big change for your teams, you should expect some level of push-back — some of your key people might resign, and you might even have to let some people go if they're too resistant to change. In a large organization, you might mitigate this somewhat by mandating that only new projects use service-oriented approaches because you can't hope to push all your projects to immediately adopt SOA.

At the other end of the spectrum, you might be a developer who believes that SOA can improve the way your company does things. Lacking the power and position required to mandate SOA, you'll need to find like-minded people within your organization. It's obviously helpful if such people are known technical leaders or respected managers. You might also need to carry out your own skunk-works project to prototype a service-oriented system. If your manager supports your ideas, he or she might give you time to build such a prototype; otherwise, you'll need to do it in your own spare time. The goal of the prototype must be to show improvements over the current approaches your company employs. Don't focus the prototype solely on technical issues. Instead, strive to have it display business value as clearly as possible. If it shows only technical superiority, you might get a "hey, that's cool" from your fellow developers, but it's unlikely to get the attention of any budget holders. You'll need to win converts from the business side of the house if you want to have any hope of succeeding.

Socializing

Regardless of your position in the organization, the key to promoting SOA's adoption is communication. Essentially, you have to employ marketing and sales tactics to socialize your ideas and win over the key people who can help make your dreams a reality.

For example, consider the "elevator pitch" scenario: what if, by chance, you wind up in an elevator with a key person, such as the CIO, and you have 20 seconds to convince him or her that your ideas are worth exploring further? Even if you think such a scenario is unlikely, plan for it because, regardless of whether it happens, you'll still end up giving that pitch to a variety of people on your own team and the management team you report to. Make sure the message is consistent and compelling, and enhance it every time you present it, based on feedback from previous listeners. Whenever possible, tune your message's highlights to the person you're talking to.

Two relatively modern ways to socialize service-oriented approaches within your organization are to employ wikis and weblogs. According to Wikipedia (itself, a wiki at www.wikipedia.org), a wiki is "a type of Web site that allows users to easily add and edit content and is especially suited for collaborative authoring," whereas a weblog or blog is "an online publication with regular posts, presented in reverse chronological order." Both are powerful Web-based socialization tools that can help you build the shared vision needed for SOA to take root.

Because wikis allow for collaborative authoring and editing, they essentially give people equal voices. You might create an initial wiki page explaining your ideas for SOA adoption and then seek feedback and contributions from others. They could then easily edit your page or link their own new wiki pages to yours. Because everyone can contribute, it's not unrealistic that the result could be a set of hyperlinked pages that present a comprehensive and compelling case for using service-oriented approaches within your organization.

Be sure your wiki pages include references to SOA use cases within other companies, including your competitors where possible. Such a wiki could help start a grassroots movement within your developer teams and might create an internal SOA community that could collectively effect a bottom-up adoption scheme. Such a wiki could even serve as a human-readable services registry, to help spread the word about who's building what and to provide documentation and access details for available services.

Blogs are not collaborative like wikis, but they provide an easy way to get messages across. Starting an SOA blog and regularly updating it with descriptions of the benefits of service orientation, how it might apply to your systems, links to articles about SOA and industry use cases, and news about your own SOA prototyping efforts can go a long way toward getting the word out and making sure that your message is heard consistently and repeatedly. Blogs typically allow readers to comment on your postings as well, meaning you can obtain valuable feedback and incorporate it into your promotions for SOA adoption.

Despite their utility, blogs and wikis are no substitute for face-to-face conversations with key stakeholders. Whether you're the CIO or just another developer, convincing key people to buy into service orientation is ultimately the only way to get it adopted. Keep in mind, however, that building service-oriented systems is hard. Even if you get buy-in from the right people, it doesn't mean that actually building and deploying services will then be trivial. After all, you still have to deal with changes in development processes, training, tools, and perhaps new avenues of technical collaboration with other teams. Nevertheless, actively addressing the social side of the equation greatly increases your chances for success with SOA.

Steve Vinoski is chief engineer for IONA Technologies. He's been involved in middleware for more than 17 years. Vinoski has helped develop middleware standards for the Object Management Group (OMG) and the World Wide Web Consortium (W3C). Contact him at vinoski@ieee.org.