Feature Articles

TopCoder @ Work: Incorporating new technologies

Discuss this article
Normal view


By timmac & rhudson
TopCoder Members

The situation: The boss has just returned from a technology conference, no doubt full of torturous new ideas. You perspire in your seat, waiting for the inevitable conversation. What twisted brand of innovation will he force upon your project today? Last year's mandatory integration of XML over HTTP launched the development team into long weekends of unpaid overtime, as "unanticipated" browser compatibility issues plagued your flagship product. You fear similar distress as the boss rounds the corner to your cubicle. His mouth forms a single word -- "AJAX" -- and you brace yourself for another twelve months of nightmares thanks to a "time-saving" methodology.

Consider this situation instead: you've just returned from a technology conference, full of exciting new ideas about "AJAX" that will certainly be ignored back in the office. A faulty integration of "XML over HTTP" last year seemed to completely rattle the boss. His concern about overtime -- spent in vain to make the application "cross-platform" -- could have been mitigated by mandating a single browser for this company-internal application. As you round the corner to his office, you swear you can see his lips form a single word -- "No" -- and your resign yourself to another twelve months of dodging innovation in favor of hacking together archaic methodologies.

These two very common scenarios demonstrate how the application of new technology -- in this case, AJAX -- can go terribly wrong. Both developers and managers are susceptible to the charms of innovation without considering whether implementation is feasible. This article will describe the right way that you, as a software developer, can research, propose, and "sell" a new technology to your organization and your client.

History

For our article, we'll pretend that your company just migrated a client from a Microsoft Access "application" to a fairly plain ASP.NET website that duplicates the core functions of the application. The new website, however, lacks the interactivity of its desktop counterpart -- the site uses frequent postbacks and is not as responsive, or informative, as the old interface.

The client has requested that the new application be able to return a list of matching products based on a full or partial inventory identifier. If we do this in our minimal web application, we will need to do a postback, which will hit the server and reload the page.

Or, we could implement AJAX to do a quick, interactive product lookup without leaving the page. The client would perceive the application as seamless, responsive, and flicker-free.

Research

Implementing a new technology is rarely a matter of making a few code changes, no matter how simple the technology may appear. You should assess the technical feasibility, the integration path, the cost, and the time required to implement AJAX.

Technical Feasibility: *Is AJAX compatible with our application?*
In our environment, we'd have to adjust our ASP.NET application to use a new framework. Most likely, we'd move to ATLAS (Atlas.asp.net). The client, in turn, would have to use an AJAX-compliant web browser -- most likely Mozilla Firefox or Internet Explorer 6.0.

The Integration Path: *How will we introduce the changes into our product?*
We determined that we would have to install Atlas; how will this impact our current application? Can we introduce AJAX-related functionality incrementally? We could respond to the open request and "test" AJAX for our product lookup example without changing the rest of the application. We will develop an estimate for the "bare bones" implementation of the technology that

accounts for our integration tasks.

Cost: *What is the true price of implementing a new technology?*
As developers, we tend to consider only making code changes and performing basic unit tests. Yet, at the project level, we incur more expenses. AJAX involves installing and integrating a new framework and development methodology (Asynchronous JavaScript over XML). Moving from a synchronous to asynchronous solution introduces new timing issues for testers. Adding XML and JavaScript to the website may require training for developers. When the application is delivered to the user, changes must be documented and integrated into plans for the trainers. These and other issues are summarized below:

1. Staffing -- do we need more people to integrate this, or do we require a skill we don't have on the team? Can we acquire knowledge through training?
2. Training -- will the client require more training to use the new system features? Will our developers require training in this new technology? Will our support, testing, and marketing staff require additional training?
3. Documentation - for users and developers; what new standards must you integrate and delineate in order to ensure smooth integration?
4. Testing - adding features, especially asynchronous, interface-centric ones, adds testing plans and testing time to the application.
5. Hardware - will we need new or adapted hardware to integrate this new technology?
6. Software - will we require any new tools or applications to integrate this new technology?
7. Support - will we require outside vendor support for this integration?

Time: *How long will it take?*
Programming hours -- the lingua franca of the developer -- account for only a small portion of the time to integrate a new technology. Invest the time to estimate the seven costs listed above before you propose the project.

The Proposal: *Sell AJAX to the stakeholders*
We have completed our estimates and have determined that AJAX is indeed feasible, but so far we have considered only technical issues. In order to persuade our company-- and our client -- to accept our proposal we need to sell our ideas to the project stakeholders. Stakeholders represent the complete audience for our application, including:

1. Direct managers - people to whom we report
2. High-level and department managers - people who determine and support the company's technology choices
3. Customer-facing employees-- in large companies, these are the sales and marketing folks
4. The customer - the people who represent the client
5. The user - the people who actually work with our application

Understand the beliefs and attitudes of your direct manager. He may be eager to "leap" into a new technology, or be recalcitrant to try "another XML solution" after past failures. However, if you have prepared a valid estimate and integration plan, you can convince the eager manager that the technology will require time to implement, and impress the hesitant manager with your extensive planning. Show that you can mitigate the risks and failure points of the past and you can sell the upgrade.

Demonstrate ultimate financial benefits to high-level managers and customer-facing employees. If the customer values interactivity, then providing that feature creates "add on" sales. You might suggest adding other interactive features with AJAX to make the application more "desktop-like."

Sell department managers on reduced support costs. Customers may call the help desk less frequently if the application appears more responsive. Instead of hanging browsers and the dreaded time-outs, users face what appears to be a functional application screen while waiting for updates.

Finally, we list the customer and user as separate parties. In many cases, company managers request features for the users in that company. Though your customer may request a feature, if the users are not satisfied they may reject the changes. Consider the attitudes of the people who will actually be working with your system as you develop your sales plan.

Prove It: *Show, don't tell*
Documentation and tailored rhetoric are no substitute for a visual representation, especially an interactive one. Create a prototype of your new feature; in a complex system you may only be able to show screen shots or mock animations. Ideally, you'll present something that is actually functional; this makes your application seem tangible and proves that you can master the technology, at least at a rudimentary level. The danger to working prototypes, however, is that someone will misinterpret a project as being done simply because something is in front of them that seems to be working. In other words, while a good prototyper will include real-looking data and functionality, a realistic developer will reiterate the conceptual nature of the prototype.

Sell It

Now that we have thoroughly considered the project, and the new technology we wish to use, we are able to complete our sales package, our complete persuasive sell to a manager or client. This includes cost benefit, time savings, and appeals to specific audience concerns, needs, or characteristics. Different people, and even entire departments, have their own agenda, culture, needs and requirements. In the case of an external client, you are left to contend with both your customer as well as the interested parties within your organization. It may also be necessary to work with the other departments in your organization, such as sales, support, and IT, to cover all bases of infrastructure.

Finally, you are presenting your package to management, and hopefully, after all goes well, a modified form of your presentation is presented directly to the client. Whether or not the package is accepted, you already have some new skills, even from the research you have gone through in determining feasibility for this particular project. Not only have you improved your skills with developing proposals, but you will also have something to talk about during a future job interview. Inevitably, a prospective employer will eventually ask you to "tell me about a challenge or problem you faced", or "tell me about a recent project where you showed leadership".

If the project is accepted, then there is a good chance you will find yourself in a leadership position, with an excellent opportunity to learn a new technology. Remember that climbing the learning curve takes time, and if your company doesn't necessarily provide pet project time, then this may fall into your own spare time. But, there is a time investment either way, and for anyone with a long-term interest in this career field, you are still helping yourself.

Don't Sell It

One more scenario we have not considered in detail, is when not to sell new technology. While it might sound very different than the typical mindset of an ambitious developer, an experienced developer is also looking to find the best tool for the job. Consider the following dialog between a manager and a developer:

Developer: We need to modify the registration system.
Manager: Ok, what do you plan to do?
Developer: Right now, it asks the customer to enter their serial number, and select the product they are registering…
Manager: Yes, and they frequently select the wrong product…
Developer: It would make more sense to just have them enter their serial number, and let us simply tell them what product it is.
Manager: GREAT! We can rewrite the system using AJAX and it will be faster and less error prone.
Developer: But wait, is there any advantage to using AJAX, or would it just make things more complicated?
Manager: Haven't you read all the latest info on how great AJAX is? It's the coolest new technology.
Developer: Well, yes, but it doesn't provide us any real advantage here.

Sometimes sticking to a simpler, if older, technology really is a better solution. Consider if AJAX was the chosen course of action above, then the results might be something like:

- The AJAX request transfers less data and, in theory, is faster. However, the user cannot confirm their entry until the data is returned. Instead of clicking submit and then waiting, the user has to wait before clicking the submit button. Thus, even though it's faster, the user perceives the application as being slower.
- Once the data is finally submitted, the same server side validations still happen, to prevent possible hack attempts (we are building this securely… right?)
- This system might be used by any customer, anywhere, so we have a wide variety of browser configurations to consider. We

don't want to anger our sales staff with calls from customers who cannot register their product.

- How many members of the sales and support staff in the organization will need some training on the new interface? Probably not too many… oh, wait, the operations staff might want to know how this works, too, and maybe the engineers will need to know about this. Wait, I forgot to consider our reseller partners as well…

If you've prepared the documentation we discussed above, then you'll be prepared to justify your determination.

Denouement

Congratulations! You have achieved perfection -- you combined your persuasive discourse with a killer presentation, and cemented your position in the company for years to come. As your manager vigorously pumps your hand with congratulations, your ears still ring with lavish praise from the customer.

- Or -

Your risk-averse manager rejected your sales package…

- Or -

Your tech-hungry manager gladly accepted your sales package, after removing a few weeks from the schedule…

In any case, you now have enough experience developing proposals and justifications to add the following to your resume: "Developed a proposal to adapt a web application with AJAX technology to create interactive, real-time client experience."

Have you had an experience with a new technology that mirrors, or differs from, this article?
Tell us about it.