If We've Got SaaS, Do We Need BPO?

It's been a long road to where we are today with software-as-a-service, but the business model is finally economically viable. Understanding how we got here sets the groundwork for answering the question: Do we need SaaS in BPO? By Naomi Bloom

In last month's column, I described the world of HRM software before we got SaaSy. During the late 1990s, the HRM software industry experimented with many variations on the previous business models (what if customers could adopt a pay-as-you-go model?), deployment approaches (what if providers hosted and upgraded the software for the customer?), but without widespread attention to the underlying single tenant architecture.

Vendors calling themselves ASPs (application service providers) thought they could manage very rapid upgrades, thereby allowing them to come to market more quickly and to build out functionality behind the scenes. They also thought that they could make the buy decision a lot easier for their customers, thereby shortening the sales cycle considerably.

There was a lot of good thinking here about deployment options and business models, but these vendors quickly discovered that their installation, operations, and support costs were totally unaffordable. Furthermore, many of them came to market without solid underlying object models or architectures, which broke down quickly under the weight of enhancements. And with no big up-front license payments to fund them, these ASP pioneers lost money during the early years of every customer relationship.

Table 1 lists some of the lessons from these experiments in off-premise deployment and pay-as-you-go business models, along with those of HRO pioneers such as ADP and Ceridian, which brought us to architectural (as opposed to marketing only) SaaS. Software-as-a-service is the industry's response to these lessons. It is an architectural, business model, and deployment approach taken by the vendor/owner of an application software package. SaaS applications are NEVER implemented in-house by their customers, but they must be designed to provide substantial integration capabilities to those still in-house or other SaaS applications with which they must interoperate. While the server environment might be outsourced by the software's owners, the SaaS vendor is accountable for delivering incredible up-time and response time.

SaaS applications are based on a single set of common code and data or object definitions that are consumed in a one-to-many model. While there may be reasons of operational performance to load balance or to configure the software differently for different target markets, there is only one logical instance of the application. The model is about sharing business rules, work flows, and other configurations.

Finally, SaaS applications are always on a pay-per-use or subscription basis. While there is usually a subscription period of at least one year with incentives provided for longer subscriptions, the more comprehensive and complex the software, the longer a subscription period needs to be.

Small or less complex organizations may be well-served and get much of the benefit of SaaS via pre-configured versions of on-premise/single-tenant software delivered on a hosted, subscription basis—where the vendor has done an excellent job of achieving the maximum possible operational efficiency without multi-tenancy. Much larger or more complex organizations may well prefer the sense of control that comes with having their own software and database instance delivered on a hosted, subscription basis.

But these other approaches simply can't achieve the lower costs of software delivery, the higher levels of client-specific configuration, or the highly beneficial cross-client analytics that can be achieved by a SaaS vendor's equally disciplined and automated operations. So, if we've got SaaS, do we need BPO? Next month's column reveals all. HRO

Naomi Lee Bloom, Managing Partner, Bloom & Wallace, can be reached at 239-454-7305 or naomibloom@mindspring.com. You can also follow her on Twitter @InFullBloomUS.

Table 1: Lessons From the Past

- Without being able to run their software on a one-to-many basis, a vendor's costs for deploying/upgrading/supporting the software are just too high, not to mention the opportunities for operational errors.
- Off-premise doesn't mean disconnected; all the same interfaces that keep that on-premise portfolio of applications connected must be recreated when an off-premise application needs those same connections.
- Configuration by client is just as important in off-premise as it is in on-premise software, perhaps even more so, and it takes much greater and more accurate domain models and architectural analysis to understand and abstract all those business rules/processes/UI etc. capabilities for which configuration is needed into the chosen configuration technology.
- Without full architectural multi-tenancy, no vendor of off-premise software can begin to cope with extensively configured customer instances at a profit-making cost unless no otherwise-capable competitor has full multi-tenancy.
- While it's certainly possible to use single-tenant software as the platform for some flavor of on-demand, subscribed software, and virtualization techniques can reduce the costs and risks of deploying this approach. However, full architectural multi-tenancy will always be less costly than single-tenancy when constructed correctly.
- When you add more architectural multi-tenancy capabilities needed for customer-satisfying on-demand software, not only are the total costs of ownership reduced tremendously, but improvement in total costs of service delivery are also substantial.