

Building the Architecture for SaaS

By Michael Oliver

Enterprise Architecture isn't something you can just buy for your SaaS application, any more than Architecture is something you can buy for your new headquarters building. But if you want your new headquarters to operate efficiently and send a message about your relationship to technology and the future by having a 'green' building, then you are probably going to have to incorporate "Sustainable Architecture" or "Environmental Architecture" or some flavor of architecture which is suitable for constructing an environmentally friendly building. So to with SaaS. It requires a certain kind of architecture to truly meet the design goals of modern enterprise software meant to support multiple tenants in Cloud environments.

The Gartner Group states Enterprise Architecture, "it isn't a thing", and Zapthink says about Service Oriented Architecture, "it isn't a product". Software as a Service requires a Service Oriented Architecture for the Enterprises that subscribe to the service, and that SOA isn't a thing or a product that can be bought and bolted on any more than 'Green architecture' can be bolted onto a skyscraper. It has to be built-in, designed in, architected in right from the start.

It's great that many SaaS Vendors have provided API's as well as Web Services to create some openness and accessibility to their products. We think that the time is right to take the next step, a bigger step into the realm of architecture. We are proposing to take it to the next level of professionalism and craft an architectural discipline; a set of considerations and principles that foster commonality, agility and reliability in the SaaS community, hence SaaSOA

There are a number of unique attributes of any Service Oriented Architecture needed to support Software as a Service, and our newly coined SaaSOA will need to address them all.

Every SaaS Application has unique requirements just as every Enterprise has unique requirements, so there is no single SaaSOA "Specification", just as there is no "Specification" for building a green headquarters building. There are "Criteria" that must be met to be considered as a good example of a particular style of architecture though. For the SaaSOA, that criteria includes the concept that it MUST allow for the "Future State" where subscribers use the SaaS solution in their Enterprise Architecture. This idea that architecture must consider the almost inevitable fact that at some point, the SaaS applications used by an organization will need to be used in concert with internal applications built upon the organizations Enterprise Architecture, is both predictable, and disruptive. Usually the domain of Enterprise Architecture is somewhat of an ivory tower exercise, and subscribes to the idea that all applications must be built according to the overarching architectural plan. But of necessity, SaaS applications must be more flexible, because the idea that all subscribers are going to change their Enterprise Architecture to fit that of one particular SaaS application is both naïve and self-defeating. Which of the many SaaS applications an organization uses will be the one whose architecture would be adopted?

SaaSOA Architectural Elements

- **Loosely Coupled Components and Services** – Without this as a cornerstone of the Architecture, the dependencies of tight coupling will eventually start to climb the steep slope of maintenance and integration costs.
- **Configuration and Convention over Programming** – Like Lego Blocks, a SaaSOA needs to depend on small components based on the “separation of concerns” principle, and shared services that can be composed into business solutions at or near run time. Add a Component to a solution and by convention the component starts working with minimal code.
- **Database Agnostic** – The choice of RDBMS should be left to the needs of the SaaS Solution Provider.
- **Topology Agnostic** – The Cloud Computing Topologies and increasingly the Enterprise Architectures that are evolving requires the ability to deploy SaaS Solution Loosely Coupled Components and Services in any Topology.
- **GUI Agnostic** – The choice of GUI technology needs to be left to the needs of the Enterprise subscribing to the service. Firewall Issues, Performance Issues, Look and Feel as well as the ability to customize and re-brand the SaaS solution requires that the GUI technology that is best for one may not work at all for another. If the Architecture is Loosely Coupled, then AJAX/DHTML, or Flash or Flex or other user interfaces technologies may easily be substituted, even on a Tenant by Tenant basis.
- **Data Source Agnostic** – If the SaaS Solution is tightly bound to data sources on the SaaS site, then integration with legacy applications by tenant will become more difficult. A SaaSOA that doesn’t care where the Data Sources are located can more easily adapt to the “Future State” when an Enterprise Tenant needs to change a Legacy Application, and the source of the data changes.
- **Data Schema Agnostic** – If the SaaSOA uses a Meta Model Driven Data Architecture, then changes to schemas for database tables can change on the fly without loss of data or breaking the application. This Meta Model can be on a Tenant by Tenant basis with each tenant having a slightly different schema with additional columns or constraints.

If the Enterprise Architecture gurus all agree that for any given Enterprise the goal is to make the “Future State”, whatever that state needs to be, more achievable and cost effective by adopting technologies, practices and strategies that are more agile and reliable, (which we think they do, despite differences in those strategies, practices and technologies), then there are implications to that consensus. For a Software as a Service application to be most attractive to an organization, the SaaS application architecture needs to be more compatible with the Subscriber’s Enterprise Architectures. SaaS Solutions that are architected with SaaSOA are better suited to the Enterprise Architecture needs of their subscribers:

- SaaS Solution needs to do more than just satisfy that Solution’s requirements, it also needs to interact with the rest of the world, in this case the subscriber’s world, by addressing the Enterprise Application Integration (EAI) needs of their subscribers.
- Gartner advises in their Enterprise Architecture practice that one pitfall to avoid is the Enterprise Architecture IT Group failing to understand that THEY don’t dictate what

needs to be done and the Enterprise isn't at their command. The same can be said for SaaS Vendors, if they take the position that their subscribers must conform to THEIR Architectural demand, then they are no better than the Enterprise IT Architecture group dictating to the Enterprise how they do things. In both cases the rationale is wrong and will face resistance and possibly even resentment, neither of which are conducive to sales. The future needs of the Enterprise should always come first. It's the paradox of Enterprise Architecture; if the need to plan for the constant evolution and change of systems wasn't so important to efficient investment, there would be no need or advantage for it. No matter how good a technical approach to Enterprise Architecture or SaaS may be, if it doesn't serve the goal of making the desired Future State of the Enterprise easier and less expensive to meet, it fails to meet the need.

- We are seeing a migration to Cloud computing because it has so many benefits. Enterprises that don't Architect for this "Future State" are not serving the needs of their Enterprise. SaaS Vendors therefore that do not Architect their solutions for this "Future State" of Cloud Computing are not serving the needs of their Subscribers.

Summary

As organizations of all sizes look to the Cloud and SaaS to improve agility and reduce costs, they need to consider the vendors solution on the basis of its Architecture to support the "Future State" that may exist within their organization. Will it truly be able to integrate into their Enterprise Architecture in a way that allows them to use the data and functions of the SaaS application as they would an application they designed for their own internal use? The lock-in that many SaaS vendors impose could easily outweigh the benefits and cost savings they bring if they impede the ability to integrate with or evolve the overall application portfolio of an organization. A set of Web Services to access a SaaS Vendor's API and data does NOT make it a Service Oriented Architecture and NOT make it a flexible and adaptable SaaS SOA that is able to fit in with the Enterprise Architecture of the organization. Enterprise Architects may want to pay more attention to the SaaS applications their organization is using. It's their role to look out for a guide the big picture of EA for the organization, and increasingly that is going to be extending outside the organizations firewalls and into the Clouds and murky proprietary worlds of SaaS applications.