Enterprise architecture in the age of cloud services

Using a hybrid SaaS, software as a service

Fabio Castiglioni

September 25, 2012

This article shows how to use an enterprise architecture to specify requirements for public cloud service, using a hybrid software as a service (SaaS) example. Enterprise architecture (EA) is a key tool to help cloud customers understand how to take advantage of the new business model that is enabled by the technology and how to fit external services into their current applications and technical environments. In reading this article, an IT architect will learn how to use EA notations and IBM Rational System Architect to communicate effectively with business users and other stakeholders, including the service provider.

Introduction

The idea that a good enterprise architecture (EA) is a key enabler for an effective adoption of a service-oriented architecture (SOA) has been raised by many years (see the Ibrahim and Long citation in Related topics), and many customers have paid for the absence of an EA "due diligence" at the price of project failure or half-failure. The architecture big picture, the end-to-end link between business processes and IT services, and the day-to-day governance mechanism that come with an established enterprise architecture are all essential elements for an SOA that keeps its promise of a technology capable of transforming the business and the enterprise.

Now, I can hear your mind buzzing, thinking something like, "I must have opened the wrong article. This was supposed to be about cloud, not SOA."

The fact is that, whether we are discussing cloud or SOA, we are dealing with services.

Saying "services" means that we willingly accept structuring our IT with a granularity that is probably sub-optimal for the management of the specific problem at hand, and we are also willing to accept to over-engineer some of its components to support an increased level of flexibility rearrange business operations. (Think about standardizing interfaces, for example. It doesn't happen for free, and it pays back only when you are actually reusing the function.) This level of flexibility is, in itself, only a means to achieve flexible business processes. Those, in turn, have meaningful economic returns only if they support flexible, informed business strategies.
Implementers of a cloud (or a SOA) architecture must have this cause-effect chain well laid out before them. Otherwise, their decisions will always miss some key aspect of the problem.

This means that having an enterprise architecture model of the area to be transitioned is performing the *due diligence* of identifying business-to-IT relations, dependencies, requirements, and constraints.

In this article we will explore how to represent this model, and how to understand from it what needs to be done and why, from the perspective of a Consumer of cloud services. This might be an organization that wants to leverage the cloud technology to inject a new level of flexibility in its operations.

**Cloud service Consumer scenarios**

A Cloud Reference Architecture, like the ones from IBM or the National Institute of Standards and Technology (NIST) of the United States Department of Commerce, structures the cloud business, starting from the set of involved actors. Each actor has a defined role.

![IBM Cloud Reference Architecture](image)

The IBM reference architecture identifies the following roles:

- The Cloud Service Creator who develops new services to be consumed through the cloud infrastructure
- The Cloud Service Provider who administers and operates the cloud infrastructure
- The Cloud Service Consumer who uses the services hosted in the cloud infrastructure

NIST lists the following roles instead:

- Cloud Provider (similar to the IBM one)
• Cloud Consumer (same)
• Cloud Auditor who can conduct independent assessment of the cloud services
• Cloud Broker who has the capabilities to intermediate, compose, add value to the services of the Provider
• Cloud Carrier who has the capabilities to provide transport services to the Cloud Service Provider

As you can see, Provider and Consumer are the central roles. Although the Provider’s business and IT models are very similar to the ones of a traditional outsourcer, the Consumer is the one that leverages the innovation capabilities of the cloud the most.

Going back to the IBM Cloud Reference Architecture, the Consumer can choose four classes of services:

• Infrastructure services (known as infrastructure as a service, or IaaS) where the Consumer uses services equivalent to a hardware system
• Platform services (PaaS, for platform as a service) where the services are equivalent to a complete hardware and software infrastructure
• Software services (SaaS, for software as a service) where the Consumer uses a business application
• Business process as a service (sometime referred to as BPaaS) where the Consumer has outsourced part of the business processes to an external provider

Provider and Consumer might be two departments in the same company (IT operations and IT development, for example) that use a private cloud, or two separate business entities, one of which is in the business of providing services through the cloud. The latter is the most interesting case, because it involves changes in the enterprise business model, not merely in the model of one of its organizational entities.

Of the four classes of service, this article concentrates on the specific case of the acquisition of new business capabilities by using a business application delivered through a cloud service (public SaaS).

One last point involves the ability of the cloud model to be a "platform." Providing services is the objective, of course, but the quality of the provided services depends on the technical and business support functions provided by the platform (the two pink boxes in the IBM Reference Architecture).

The IBM Cloud Reference Architecture identifies these support services:

• Service Offering Catalog and Management
• Service Request Management
• Order and Subscription Management
• Contracts and Agreement Management
• Pricing, Metering and Billing
• Customer Account Management
• Rating
• Clearing and Settlement, Accounts Payable, Accounts Receivable
• Service Delivery Catalog
• Service Automation Management
• Change and Configuration Management
• Image Lifecycle Management
• Provisioning
• Incident and Problem Management
• IT Service Level Management
• Monitoring and Event Management
• IT Assets and License Management
• Capacity and Performance Management
• Platform and Virtualization Management

Some of these services clearly support the Providers' business and technical processes, some require the participation of the Consumers and might actually be new for them, such as billing for the payment of the service, correlation of external monitoring information to control the quality of service purchased, and so forth.

Note:
This article will not focus on them for reasons of space, even if we suggest including their (EA) analysis as part of the technical due diligence for the adoption of any cloud-based service.

The TOGAF framework and the ArchiMate model

The Open Group Architecture Framework (TOGAF) is an enterprise architecture framework. The development of TOGAF Version 1 in 1995 was based on the Technical Architecture Framework for Information Management (TAFIM) developed by the US Department of Defense. Central to TOGAF is the Architecture Development Method (ADM), which describes a process in 8+1 phases to manage the development of an enterprise architecture.

ADM is iterative at three levels: over the whole process, between phases, and within phases. It is a generic method, and it can be tailored to suit the specific needs of the organization. For example, some US federal agencies that use ADM have developed architecture deliverables specific to their particular departmental needs.

ArchiMate, an Open Group Standard, is an open and independent modeling language for enterprise architecture that provides instruments to describe, analyze, and visualize the relationships among business, application and technology domains in an unambiguous way.

Just as an architectural drawing in classical building architecture describes the various aspects of the construction and use of a building, ArchiMate offers a common language for describing the construction and operation of business processes, organizational structures, information flows, IT systems, and technical infrastructure. This insight helps stakeholders to design, assess, and communicate the consequences of decisions and changes within and between these business domains. (Source: The Open Group)

ArchiMate is organized into three core layers and two extensions.
Business layer
This layer models the organization structure and the services it produces, business roles and processes, and business objects such as products and contracts.

Application layer
It describes application components and their interactions, logical data entities and their relationships, and the resulting services offered to the upper (business) layer.

Technology layer
This one models hardware and software systems and the connecting networks, showing how they translate into services provided to the upper (application) layer.

The AchiMate 2.0 specifications added two extensions to the core layers.

Motivation
Motivational concepts are used to model the motivations, or reasons, that influence, guide, or constrain the design or change of some part of the enterprise architecture.

Implementation and migration
This extension includes concepts for modeling change programs and migration planning, as well as to support program, portfolio, and project management.
As any good architectural model, in addition to architectural layers, ArchiMate specifications support Viewpoints. Viewpoints are for communicating certain aspects of an architecture. These aspects are determined by the concerns of a stakeholder. Therefore, what should and should not be included in a specific viewpoint is entirely dependent on the stakeholder’s concerns.

This article complements the motivation business, application, and technology models with a nonfunctional viewpoint of the architecture, as described by Castiglioni and Pedulla (see Related topics), using Corso ArchiMate 2.0 plug-in for IBM® Rational® System Architect.
Using EA to integrate cloud service in the enterprise

Even from this cursory introduction, it is clear that the ArchiMate model is about services and service realization. This fits very well with the cloud model.

We can easily define a SaaS as application services exposed to support a business process, implemented through application components.

Equally easily, we can model a PaaS as a technology service (or a collection of services) to support components and data of a specific application.

Thus, functional requirements for a SaaS can be described by:

- The motivation of involved stakeholders
- The business model that it must support
- Interfaces exposed to or from other applications
- Services requested to the Consumer’s technology level if connectivity between the cloud and the in-house systems is required
- The nonfunctional aspects of the SaaS solution

Sample scenario: Externalizing a process to the cloud

To better understand the use of EA models to specify how to use a SaaS solution, we will extend the ArchiMate sample from The Open Group. This example uses a fictitious insurance company called Archinsurance, and it assumes that Archinsurance has identified the existing CRM application as an inhibitor to the achievement of its business objectives.

Sharing motivations with all stakeholders

The Motivation model captures the motivation and the requirements of the stakeholders. As the flow diagram in Figure 3 illustrates, the CEO motivation is twofold:

- Increase the number of repeated sales to a given customer
- Earn money before year-end

The CFO is concerned by the ROI of capital expenditure in a period of tight cash, while the CIO is mainly worried about the floor space available in the existing data center. People in the Security department oppose every off-site solution, because they fear theft or loss of company vital data, such as customer records, and require the implementation of company security practices.
From the CEO, CFO, CIO, and Security department requirements, it seems that the choice is between a cloud solution and more traditional outsourcing, with additional points in favor of the first. However, the security requirements are difficult to accommodate and will restrict the number of eligible providers to the ones that are capable of offering data encryption both in-flight and at rest.

In addition, the SaaS solution selected cannot be a "pure" cloud service, because it must include a mechanism for protecting the data from unauthorized access and data alignment between the cloud and the data center. Therefore, it will be a hybrid solution.

**Identifying functional requirements from the business model**

As mentioned before, this example is based on the assumption that the new application will support the same business model and processes of the previous one. Our EA business model will provide the guidance to identify what needs to be supported.

As an example, the current CRM supports the Handle Claim process by providing Customer Administration Services and Printing Services. The services identified in this way will be the source of functional requirements for the new SaaS solution (for example: "The printout of the claim must contain the following data."), as well as nonfunctional ones (for example: "The formatting and printing of a claim cannot take more than 3 minutes").
Designing the new application model

When we are satisfied that our would-be SaaS solution can meet all functional requirements, the next step is to design how it can verify its interface requirements with other applications (in-house ones and maybe even those on the cloud).

Figure 5. The new application portfolio

In this example, the Archinsurance application portfolio will change from the one on the left in Figure 5 to the one on the right, where CRM has become a component external to the enterprise. The role of the CRM component is unchanged, but it is important to carefully consider its interfaces with other components, such as Customer Data Access and Policy Data Management, because those could change, both in signature (the data exchanged) and in protocol (the technology used for the exchange). (Notice that the direction of relationships in ArchiMate is inverted compared to relationships in UML models.)

To focus this aspect, we redraw our application model, making the application interfaces and, if necessary, the technical components implementing them explicit. For the two interfaces shown in
Figure 5, given the technical capabilities of the SaaS implementation, this means adding, at the edge of our network, an ESB technical component that is able to offer web services interfaces to the company data. We will use a secure ESB and a secure FTP service toward the enterprise data warehouse to satisfy the requirements of the company’s manager of security.

For the interface with the web portal, we have decided that the real requirement is for a simple HTTP redirect supported by a single user ID and password schema. Therefore, the contact agent will use a single ID and password pair to log on to internal systems and to the new CRM application.

The interface is between the LDAP data and the data in the authentication mechanism used by the CRM in the cloud (in the example, we push LDAP updates to CRM as events). The LDAP authentication and CRM cooperate in providing the (new) user authentication function, just as the CRM data and the internal data mart cooperate in providing the new data warehouse function.

The new application model will guide us in the design of the changes needed to accommodate the new SaaS service with our existing in-house infrastructure.

**Figure 6. Diagram of the revised application model**

![Diagram of the revised application model](image)

**Describing the non-functional Viewpoint**

The last item that we need to consider is the nonfunctional Viewpoint.

Our focus here will be twofold:

- What nonfunctional requirements (NFRs) the SaaS solution must be able to support
- What we need to do to connect the two systems and how this will affect the NFRs of the in-house infrastructure
In fact, the new service will have to satisfy to a complex set of NFRs, some of them originated by the business needs (for example, the printing time), some by other stakeholders, such as the manager of security.

In addition, some requirements might add to form complex combinations that will stress the new service (for example, response time with a large number of concurrent users), and we will have to verify those before declaring the SaaS solution acceptable.

The inner architecture and infrastructure of the Service Provider is not of our concern (except for due diligence). Therefore, whatever acceptance test we perform will be a black box one. However, the infrastructure must be taken into consideration. In our schema, the XML firewall that secures the Archinsurance ESB and the HTTP/FTP firewall must be configured to take into account the new flows between the Company and the Service Provider.

**Figure 7. CRM SaaS nonfunctional Viewpoint**

![Diagram showing CRM SaaS nonfunctional Viewpoint]

**Summary**

This article has shown how an EA model can provide excellent guidance in the adoption of a SaaS solution, both in terms of requirements on the Provider and as specification and design for what is needed in the in-house infrastructure to adapt to the new service. The example of this approach uses the TOGAF ArchiMate notation provided in Rational System Architect through a Corso plug-in.

**Acknowledgements**

Thanks to Pietro della Peruta and Francesco Pedulla, Executive IT Architects at IBM, for their review and suggestions.
Related topics

• Learn more about how nonfunctional requirements (NFRs) can be integrated into the architectural description of complex, software-intensive systems.
  • F. Castiglioni and F. Pedulla. Representing nonfunctional aspects using TOGAF ArchiMate. IBM developerWorks (April 2012).
  • V. Temnenco. Using a combined SOA and TOGAF environment for increased productivity: Part 2. Service-oriented architecture from top to bottom. IBM developerWorks (2010).

• Browse the Unicom System Architect, product page
• Subscribe to a developerWorks newsletter.
• Evaluate IBM software.