Adobe Article for Loosely Coupled – part 3 of 4

Title: Humans as part of the SOA equation

Abstract:

While many architectures are aimed at the application to application space, eventually interactions with humans must transpire. There are several popular methodologies for round tripping between an enterprise architecture embracing SOA and a human actor. In this article - part three of a four part series - we'll explore many of these methodologies, expounding on some of the virtues of each.

Author: Duane Nickull, Senior Standards Strategist, Adobe Systems

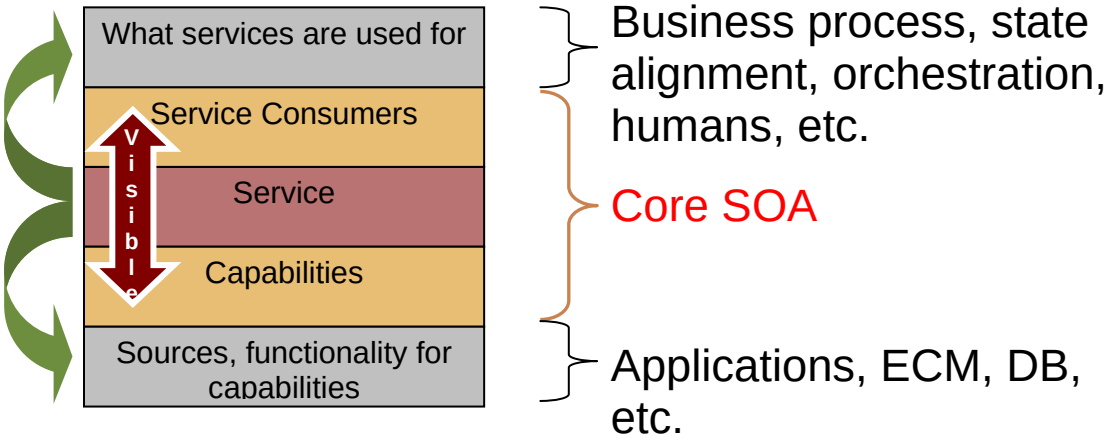
Article:

While many architectures are aimed at the application to application space, eventually interactions with humans must transpire. In the first article in this series, we introduced the OASIS Reference Model (RM) for SOA as an abstract framework for understanding significant relationships among the entities in a service oriented environment. The second article explored using the model during the architectural process. This time, we cover the role of the 'human element' inherent in an SOA.

Service Oriented Architecture (SOA) is an architectural paradigm and solution to the problem of connecting capabilities and requirements via the action boundary called the "service." One of the core tenets of this interaction is "managed transparency," an act by which the consumer of a service should not care how the service is performed, but only that it is performed, and from the service provider's side, only that a request has been issued and a result has been communicated back to the consumer. The service itself does not (and should not) really care about anything that exists above the consumer in the service stack since it cannot see why the service is being used.

The common convention for interpreting architectural 'stack' diagrams is that each layer of a stack can see only itself (N) and the layers immediately above (N+1) and below (N-1). If you draw a stack as below in which the service consumer is the layer above (N+1), you can see that a service (N) should not care what it is consumed for (N+2). A service itself is blind to the fact that it is being used as part of a process or is being called on for something else. Of course, there may be exceptions to this where an architect has a good reason for creating such a dependency, but in general the axiom is accepted. In the exception (i.e. – making a service mindful of some state that exists above the service consumer), dependencies of this nature make it more difficult to change infrastructure later since these dependencies must be carefully analyzed before any proposed change is made.

Adobe Article for Loosely Coupled – part 3 of 4



SOA Stack Diagram

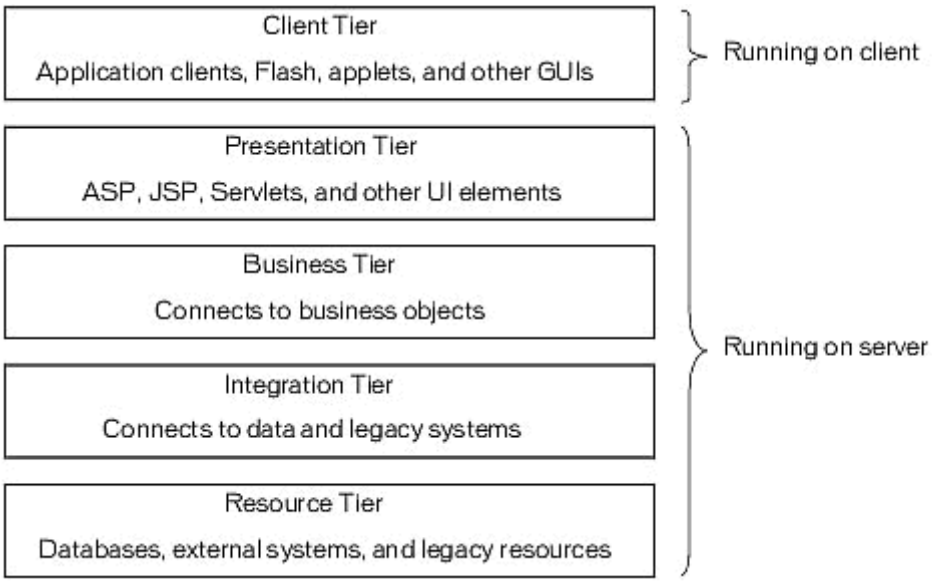
Since human beings live in the world above the service consumer, the service itself will not necessarily care or be aware that a human is there. It is highly likely however, that humans are part of a service call and exist as either the trigger of some event that results in a service call and/or are the ultimate consumers of the service result themselves. While contradictory to the title of this article, one could state that humans are not directly part of SOA any more than business process is, however both business processes and humans utilize the SOA infrastructure.

To enable such interactions, architects should take great care to abstract any details of the service itself away from the human. The human should view the "V" (view) component of MVC and not care about the C (controller) or M (model) components.[i] Accordingly, architects need a mechanism to present consistent views to the human that are abstract of any dependency upon the service.

One of the most popular ways to do this is to use a container on the client side for all interactions with the consumer. This is not new thinking, as client-server architecture has used this for a long time. Commodities such as web browsers have become commonplace for interactions with multiple services. The model is easy to understand when the ratio is one human consumer to one service, however as IT infrastructure changes over time, a single service may be divided into multiple endpoints that all must be interacted with to provide the human being the ultimate end result. Popular patterns include client-server and Rich Internet Applications (RIAs).

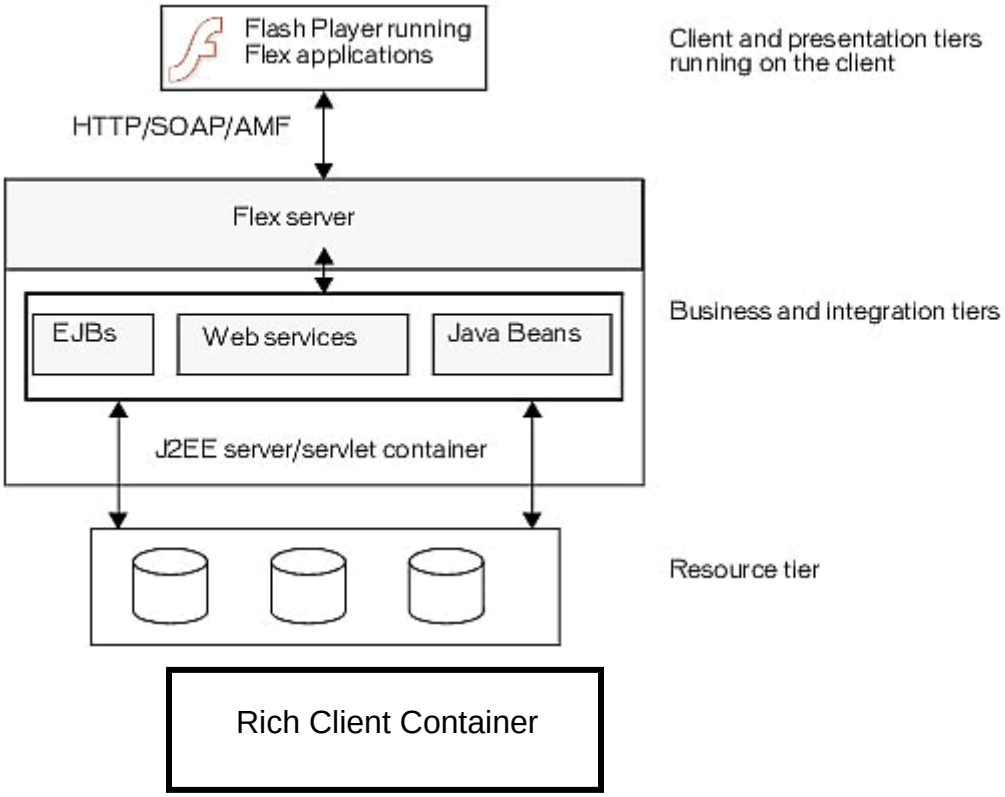
Client server is a simple two tier architecture that incorporates the final view into the client component. There are variations of this model. The first is the "thin" client whereby the server contains the processing (control) components while the client tier merely presents the end result to the human with minimal processing. The human, while not depicted below, interacts only with the Client tier and has no connection directly with the server tiers.

Adobe Article for Loosely Coupled – part 3 of 4



Classic Client Server Architecture model

Alternatively, clients can incorporate some moderate processing themselves by using JavaScript or other scripting languages to manipulate objects on the client. Two new variations of rich client interaction models with services are AJAX (Asynchronous JavaScript and XML) and Flex[ii]. Both use a container for presenting the human the final view of their interaction with the service but use complex processing and communication techniques to maintain state with multiple services in the back end. In both cases, Flex and AJAX embody MVC and use a controller on both the server and the client.

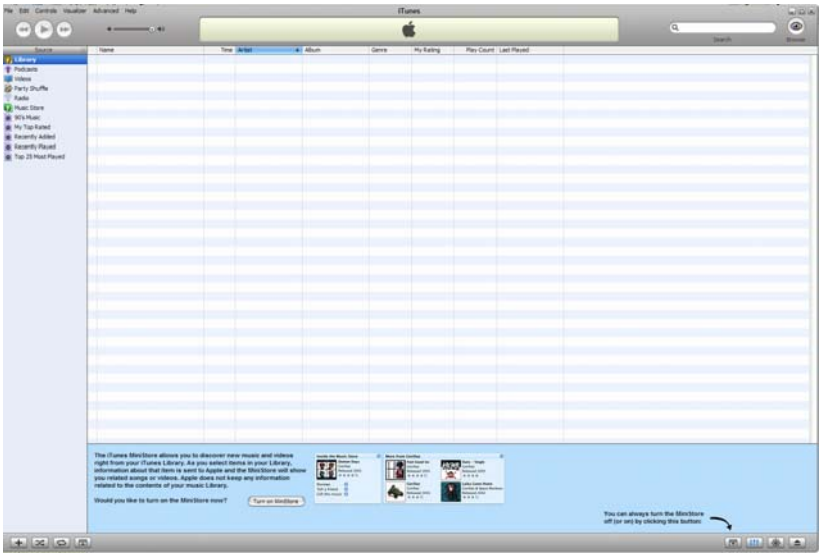


AJAX, Flex model for RIA

Adobe Article for Loosely Coupled – part 3 of 4

The use of this paradigm can enable rich internet applications (RIA's) to be developed that interact with multiple back end services. In the Flex example above, web services are used to communicate interactively with the services at the bottom. Note that the cardinality is not limited in any way to 1:1. A Flex or AJAX application may communicate directly with multiple services then abstract all the complexity of SOA and present the view and some control components to the human being above the client side.

Several examples of RIA's interacting between humans and services can be seen all over the Internet. Some applications like Apple's iTunes have viewer panes which are generated locally on the client machine (such as the user's music library). When the user switches panes to view the iTunes music store, the application connects directly to the Apple iTunes music store service, retrieves a result set then presents it to the user. To the human being, the end result is a seamless experience joining both service calls and local processing into one application.



The local Pane – music library

The music store pane – generated by making a service call to the iTunes music store service.

Adobe Article for Loosely Coupled – part 3 of 4

Final Analysis:

The end result of careful architecture and planning is a model for software using the Internet as a reliable medium for content delivery which begins to blur the lines between conventional applications and software as a service. This software can incorporate interactions with human beings as the ultimate consumers of service calls. SOA is the foundational paradigm for architecting such applications while standards like XML, web services and others make the interoperability between humans and services possible over the Internet.

---

[i] MVC: Model-View-Controller – an architectural pattern for separating the model, view and controller aspects of an application into independent components. See more at : http://ootips.org/mvc-pattern.html

[ii] Flex - http://labs.macromedia.com/flexproductline/

Author Bio: Duane Nickull, Senior Standards Strategist, Adobe Systems

The main focus of Duane's professional career has been working for Adobe and both the United Nations CEFACT committee and OASIS for the purposes of writing and building new architectures for global integration of multiple systems.

In addition to chairing the OASIS Service Oriented Architecture Reference Model Technical Committee (SOA-RM TC), Duane is Vice Chair of the United Nations Centre for Facilitation of Commerce and Trade (UN/CEFACT). Here, he oversees the UN's Electronic Business strategy and architecture and has served as the project team lead of the United Nations (UN/CEFACT) eBusiness Architecture Group, a technical advisor to the UN/CEFACT – OASIS Joint Coordination Committee (JCC) and a specially appointed liaison between the W3C, UN and OASIS standards consortiums.