COLLABORATIVE CONSULTING

Collaborative Point of View

Document Centric SOA

Jeremy Deane, Technical Architect, jdeane@collaborative.ws, November 2007

TABLE OF CONTENTS

LIST OF FIGURES

877-376-9900
www.collaborativeconsulting.com

COLLABORATIVE CONSULTING

Collaborative Point of View

1. Document Centric SOA

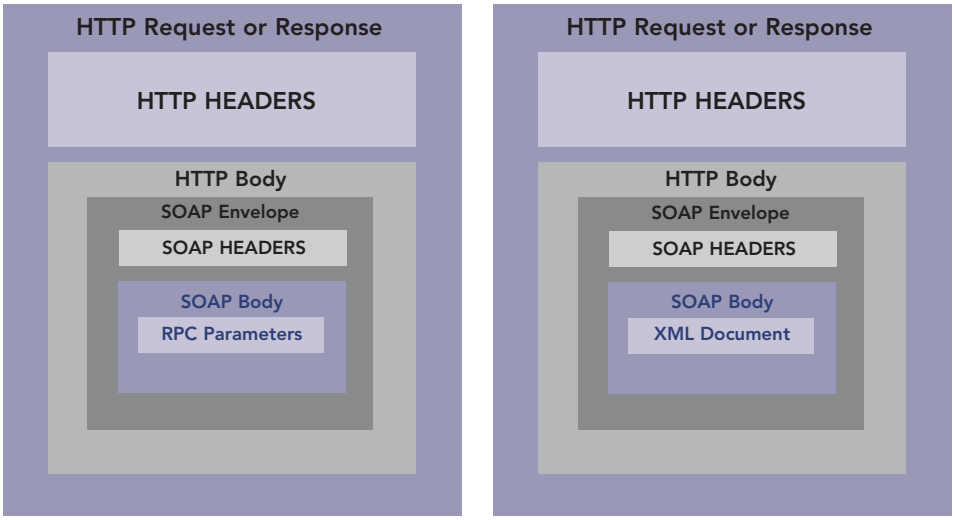
1.1 Background

Service Oriented Architecture (SOA) is an approach for building software services, regardless of location or ownership, that map directly to business processes. This approach promotes a flexible enterprise architecture that adapts quickly to ever-changing business requirements. For instance, a new service can be composed of existing services, creating a new business process, or can simply delegate to another service, extending a business process.

Many of the early SOA adopters implemented services based on Simple Object Access Protocol (SOAP) and remote procedure calls (SOAP-RPC). However, SOAP-RPC services tended to have fine-grained interfaces resulting in redundant exchanges between the consumer and service provider. Synchronous calls blocked both the consumer and service provider during each exchange. Thus, this type of interaction tightly coupled the consumer and service provider.

The current industry trend is to implement document-style services. This type of service still exchanges a message consisting of a header and a body but instead of RPC parameters, the body now consists of an XML document. For instance, document-style SOAP services can be implemented by defining the body of the message as XML, also known as a document-literal approach (SOAP-XML).

Figure 1 Remote Procedure Call vs. Document-Literal



SOAP- RPC

SOAP- XML

1.2 Exchanging Business Documents

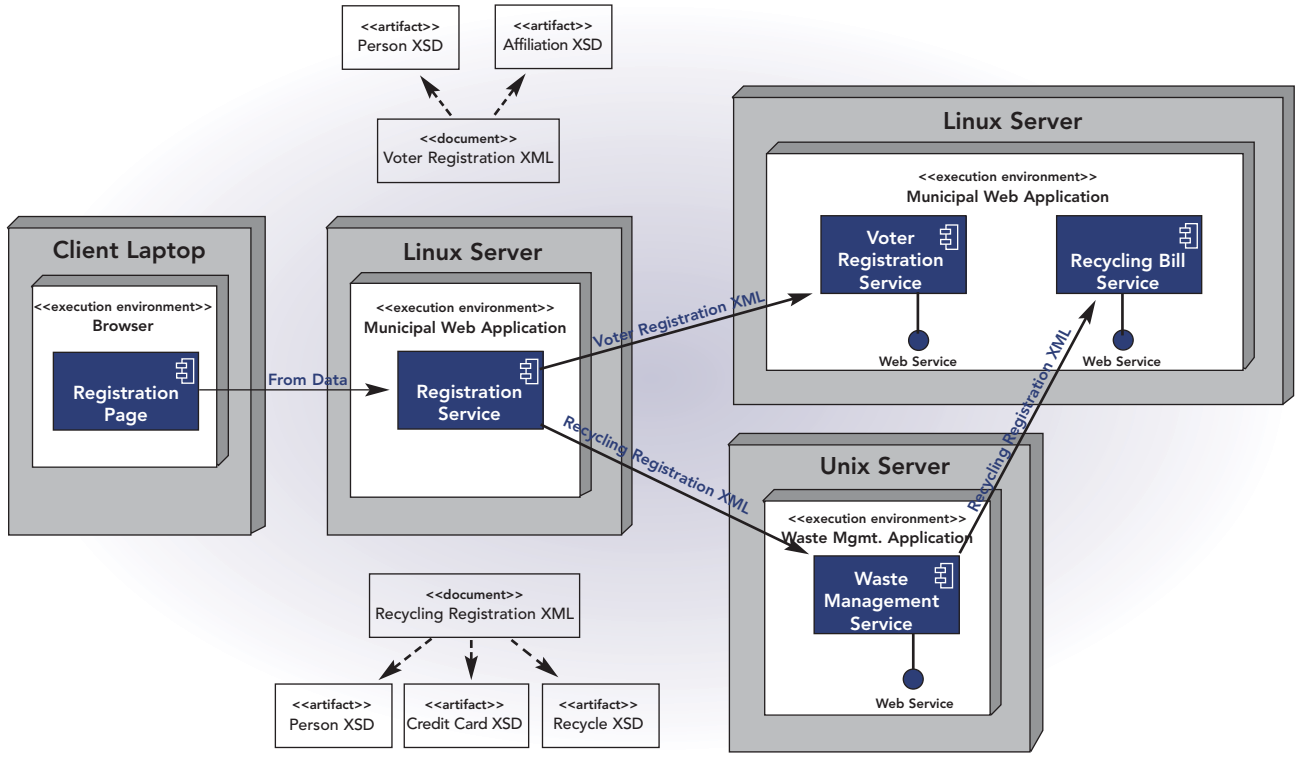
A service exchanging a document, promotes loose coupling in that a consumer easily could use a different service provider that accepts the same document or the service provider could implement the underlying document processing logic in a variety of ways without changing its interface. A document centric approach can also limit temporal coupling, the degree to which a consumer and provider are locked during their exchange, because it supports both synchronous and asynchronous interactions . In addition, document centric services generally have course-grained interfaces and consequently, scale and perform better then finer-grained remote procedure calls.

Documents consist of Extensible Markup Language (XML), ideally in canonical form[1], with their structure defined by an XML schema (XSD). It also is possible for XML documents to be defined by more than one XSD. In aggregate, the schemas define a vocabulary and grammar for creating business documents. Consequently, an enterprise domain model, decoupled from the various application details, is created. And any XML-enabled application can process or exchange these business documents based on the enterprise domain model.

[1] *W3C Canonical XML Recommendation*

Collaborative Point of View

Figure 2 Document Centric Services



The Document Centric Services Model, Figure 2, comprises several services exchanging documents. Data is submitted from a client browser to a web application for processing. The web application transforms the HTML Form data into XML documents defined by several XML schemas. These XML documents are then passed onto services for processing where the entire document or parts of the document are read or updated using a Pipes and Filters approach.

1.3 Service Implementation Options

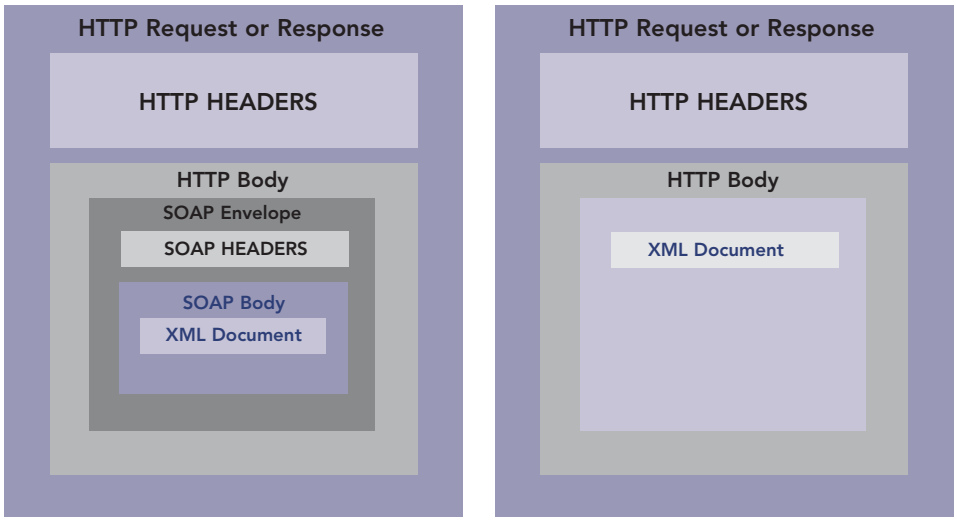
Document centric services can be implemented using SOAP-XML or REST-XML[2]. Both support a business process, however, they differ in their approach. SOAP-XML, invoked using WSDL defined interfaces, focuses on an activity. REST-XML, invoked using an application protocol, focuses on a resource, identified by a unique ID. The difference is in the semantics, registering to vote is an activity while submitting a voter registration document is updating a resource.

A key benefit of SOAP-XML is transport independence. Unfortunately implementing SOAP-XML services requires adhering to a large set of fluctuating WS-* specifications. Consequently, SOAP-XML services tend to be more complex than REST-XML services. For example, if adhering to the all specifications, it would take the following steps/tasks to implement a SOAP-XML service:

[2]*Representational State Transfer (REST) originated from Roy Thomas Fielding's PhD Dissertation*

Collaborative Point of View

Figure 3 SOAP vs. REST



SOAP- XML

REST - XML

1. Model the process using Business Process Modeling Notation (BPMN)
2. Define a service interface using WSDL and register the service with a Universal Description, Discovery and Integration (UDDI) repository.
3. Generate Business Process Execution Language (BPEL) scripts using the BPMN that access the services from the service registry.
4. Define policies governing access to the service using WS-Policy.

On the other hand, REST is based on a small set of widely-accepted standards, such as HTTP and XML, requiring far fewer development steps, toolkits and execution engines. Thus, the three key benefits of a RESTful approach include a lower cost of entry, quicker time to market, and flexible architecture. The clear choice for implementing document centric services is to use a RESTful approach.

1.4 RESTful Document Centric Services

A RESTful service provides access to a resource, identified by a Universal Resource Indicator (URI), over HTTP[3]. The HTTP verbs (e.g. GET, PUT, POST, and DELETE) define the interactions between consumer and service. A service returns a resource representation containing information about the resource's state. In turn, when a service receives a resource representation from a consumer, the service can update the existing resource or create a new resource. Finally, a consumer can request the service to delete a resource.

The most common resource representation format is an XML document, optionally defined by an XSD. A resource representation often contains not just data but links to other resources as well. The resource representation and links represent a snapshot of the application state within a shared context including the consumer and service provider. While each service request results in an updated application state, a specific resource's state only changes in the case of a PUT, POST or DELETE.

The Request and Response messages implement a standard format that includes a header and a body. Since the RESTful services are stateless, the security information must be passed in each request. To secure a request, authentication tokens and digital signatures are placed in the header while the body of the message is encrypted.

RESTful services can be written in any language that provides an interface to HTTP. In fact, several frameworks exist for both scripting and compiled languages including Django (Python), Restlet (Java), and Ruby on Rails. Another approach is to build RESTful services using a resource-oriented computing platform, such as 1060 NetKernel. These frameworks and platforms abstract the complexities of working with HTTP and XML, resulting in rapid deployment of RESTful services.

[3] *A RESTful service can also be invoked asynchronously using the Request-Reply Integration Pattern*

Collaborative Point of View

Figure 4 RESTFul Document Centric Service

While REST and XML provide a foundation for implementing Document Centric Services, service provisioning, monitoring and management require additional enterprise capabilities. An Enterprise Service Bus (ESB) solution provisions services from a central location eliminating the point-to-point communication between consumer and service provider. A Service Governance solution monitors and manages service usage.

1.5 Service Provisioning

An Enterprise Service Bus (ESB) provides a platform for service provisioning. The core capabilities that enable provisioning across an enterprise include addressing, routing and transformations. The ability to specify the location of a service regardless of transport is addressing.

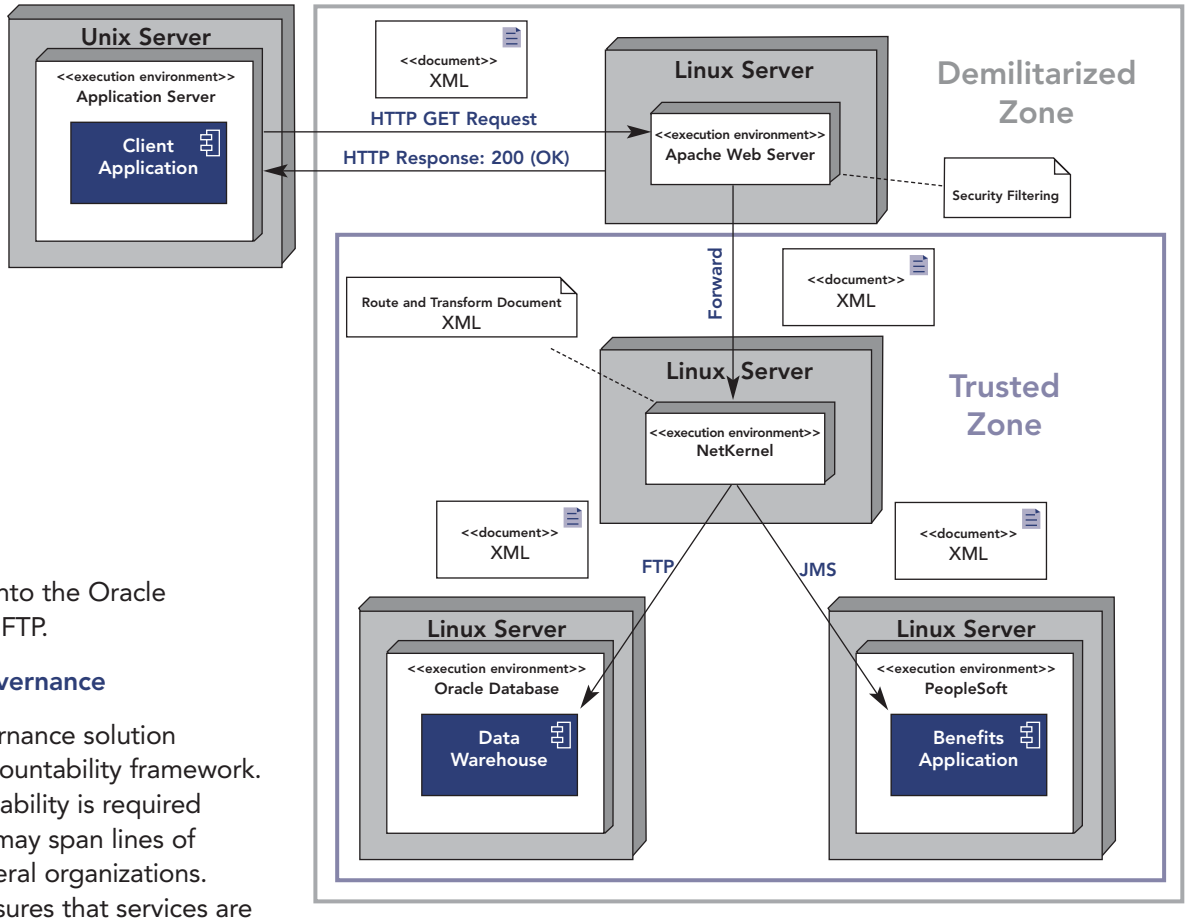
Service routing defines a message path across a number of servers or nodes and transformations are implemented using XML technologies such as XSLT and proprietary adapters. Advanced ESB capabilities include registration and orchestration of services.

Although several ESB implementations such as Codehaus Mule support REST, only 1060 NetKernel is built upon a RESTful Micro-Kernel. NetKernel is a middleware server providing the core ESB capabilities such as addressing, routing and transformations. In addition, NetKernel can act as the service registry and service orchestration engine. The server supports a number of different transport protocols (e.g. HTTP, JMS, SMTP) enabling integration across a heterogeneous enterprise. Finally, NetKernel has a SOAP engine enabling it to consume SOAP-XML services also.

In figure 5, NetKernel Enterprise Service Bus, a client application, sends an XML document to a Web Server using an HTTP PUT Request. The Web Server applies security filters to the request, such as a SQL injection check, forwards on the request to NetKernel, and informs the client that the request was accepted but might be processed asynchronously. NetKernel sends the original XML document to the PeopleSoft application using JMS. Finally, NetKernel uses another resource to transform the XML document into an ASCII text file and

Collaborative Point of View

Figure 5 NetKernel Enterprise Service Bus



sends the file onto the Oracle database using FTP.

1.6 Service Governance

A Service Governance solution provides an accountability framework. Service accountability is required since a service may span lines of business or several organizations. Governance ensures that services are not overused, or worse misused, by enforcing policies and maintaining service level agreements.

A number of exceptional products exist for implementing SOA-XML service governance. However, not all the steps in a business process flow are based on a SOAP transaction. In fact, part of a flow could involve sending an ASCII file using FTP or part of the flow could be implemented using REST-XML.

Conventional governance products cannot provide the type of visibility necessary to implement enterprise-wide governance. One unique product that provides a complete solution is Actional SOA Management[4].

Actional auto-discovers the process flows within an enterprise, regardless of transport (e.g. HTTP, JDBC, and JMS), and determines the dependencies between service consumers and producers. It is then possible to analyze the process flows to determine if operational issues exist. In addition, policies can be defined and applied to parts of or the entire process flow. In other words, the product provides centralized policy management and distributed

[4] *Actional SOA Management Product Suite*

Collaborative Point of View

run-time enforcement.

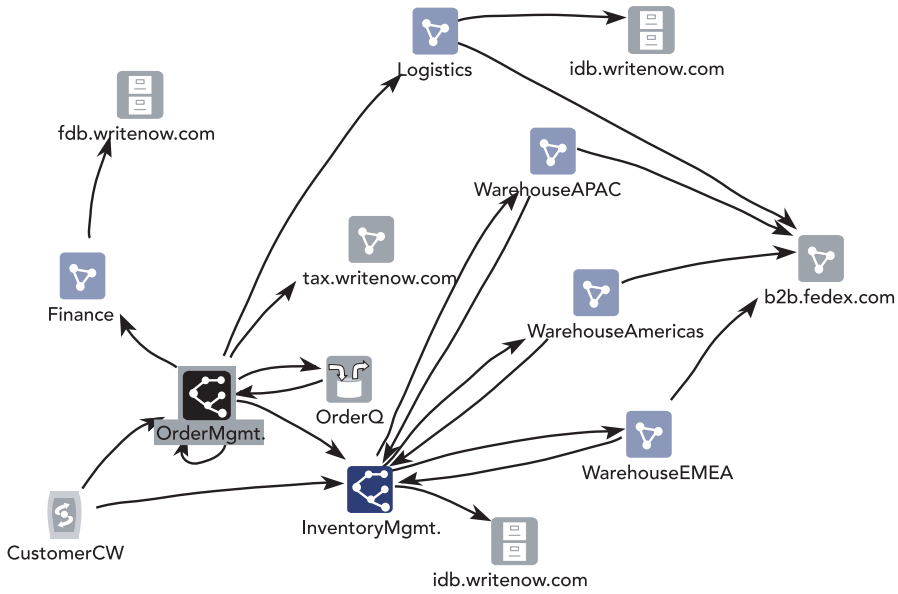
1.7 Summary

A Document Centric SOA consists of reusable services exchanging business documents. Using schemas to define the contents of the documents creates an enterprise domain model independent of a platform or application. An enterprise domain model provides an organization with architectural agility since integrated systems do not rely on each other's message structure or data types. Thus the replacement of a system of record, such as legacy application, may require an update to the ESB but will have little or no impact on the consuming applications.

Using a RESTful approach to implement document-centric services lowers the initial cost of entry and increases the time to market. REST is preferred over SOAP because REST is based on the stable principles of the web rather than a set of ever changing vendor-driven specifications. In addition, REST is far less complex than SOAP, requiring less development and fewer processing steps. And because RESTful services are stateless and cacheable, they often provide superior performance and scalability.

As an organization's SOA matures, it can deploy enterprise systems for provisioning, monitoring and managing document-centric services. An Enterprise Service Bus centralizes access to services eliminating the point-to-point communication

Figure 6 Actional Business Process Visibility



Document Centric SOA

between consumer and service provider. Decoupling the service consumer from the service provider results in additional enterprise architectural agility. A Service Governance solution adds an additional layer within the enterprise architecture ensuring document-centric services comply with all applicable regulatory, competitive and operational requirements.

In his book Service Oriented Architecture, Thomas Erl proposed a set of principles that hold true for all software services. Services are loosely coupled, reusable, and stateless. In addition, services are discoverable and can comprise other services. Finally, service interfaces are defined by a formal contract that conceals the underlying logic. A document-centric SOA not only adheres to these principles but offers an approach that, if implemented incrementally, reduces the initial cost of entry to SOA. In the long run, this approach maximizes an organization's ROI, increases its capabilities and enables true enterprise governance.

Collaborative Point of View

2. References

Service Oriented Architecture

- OASIS SOA Reference Architecture
  http://www.oasisopen.org/committees/download.php/19679/soa-rm-cs.pdf
- Service-Oriented Architecture: Concepts, Technology, and Design by Thomas Erl
  http://www.soabooks.com/chapters2.asp
- Resource Oriented Computing
  http://1060.org/upload/IntroductionToResourceOrientedComputing-1.pdf
- Understanding Enterprise Service Bus Part I-III by Rick Robinson
  http://www-128.ibm.com/developerworks/webservices/library/ws-esbscen/
- Understanding SOA Governance by Lori MacVittie
  http://www.networkcomputing.com/showArticle.jhtml?articleID=191203018&queryText=centrasite

Web Services

- W3C Web Services Architecture
  http://www.w3.org/TR/ws-arch/
- Web Services: Concepts, Architectures and Applications by Alonso, Casati, Kuno & Machiraju
  http://www.inf.ethz.ch/personal/alonso/WebServicesBook
- Web Services Interoperability Organization
  http://www.ws-i.org/
- Resource-oriented vs. activity-oriented Web services by James Snell
  http://www-128.ibm.com/developerworks/xml/library/ws-restvsoap/
- RESTful Web Services by Leonard Richardson, Sam Ruby
  http://www.oreilly.com/catalog/9780596529260/

Extensible Markup Language (XML)

- Extensible Markup Language (XML)
  http://www.w3.org/XML/
- XML Schema
  http://www.w3.org/XML/Schema
- XSL Transformations (XSL)
  http://www.w3.org/TR/xslt