19. Service-Oriented Computing

INFO 210 - 31 October 2007

Bob Glushko
Plan for Today's Class

Abstracting the idea of "Service"

Key Concepts for Services
- Coupling - *loose vs tight*
- Transparency
- Discovery
- Integration

Deconstructing Four "Services Perspectives"
- Service Oriented Architecture
- Service Oriented Computing
- Web Services
- Web-based Services
Abstracting the idea of "Service"

Many of the traditional concepts, techniques, and curricula for service design and operations originate in and emphasize person-to-person services

These do not fit so well when person-to-person services are replaced or complemented by self-service or for automated information-intensive services provided by one computational or automated process to another

The concept of SERVICE SYSTEM was proposed to raise the level of abstraction about the service concept to make these different kinds of services more comparable

For the next two lectures we will see how far we can push this with "SERVICE-ORIENTED" Computing
What Is a Service?

Provider

Client / consumer / requestor

Interface -- the (sometimes implicit) description of what the service does and how to request it

- ... or the information about the service provided to any potential service client / consumer / requestor

- A service can offer multiple interfaces ... you might think of different "quality of service" that way

- The interface can change without affecting the service that is delivered
Loose Coupling

Two businesses anywhere in the world can do business with each other using telephones, fax machines, or email.

Their interactions are:

- *Ad hoc*
- *Asynchronous*
- *Location and Implementation Transparent*
Service Discovery

Two businesses can be in a long-term relationship with recurring interactions, but initially the service client / consumer / requestor needed to discover the existence of the service provider.

And for infrequently used or specialized services, the service consumer usually doesn't have any knowledge of potential service providers.

So the service consumer needs to DISCOVER the existence and contact information (and the interface) for the service provider in a directory.

The service provider learns of the existence of the service consumer when the latter requests its service.
The Integration Requirement

But firms don't rely on phones, fax machines, or email to do all of their business because they have too many interactions, and these kinds of service requests must often be manually transformed and re-entered as input to the destination application or requested service.

This requires INTEGRATION -- the controlled and automated sharing of data and business processes among any services, applications, or data sources, intra- or inter-company.
The Integration Challenge

Business processes span multiple departments (or companies), and the business applications run by separate departments (or companies) may not have been designed to share information with each other.

"Stovepipes" most often describe the non-integrated systems within a given company, but can also be applied to non-integrated systems within a well-defined supply chain.

Stovepipe applications naturally occur when business activities are organized hierarchically -- where engineering, manufacturing, sales, marketing, etc. are separated into different departments -- because hierarchy often limits the interactions between them to the "hand-offs" of "finished work" to each other.

- Recall "Staple Yourself to an Order" reading from the Service System lectures.
Automated Information Sharing and Service Requests

Businesses use a range of techniques to transfer information from one service or application to another:

- Batch file transfer
- "Real-time" messages, database inserts
- Cross-platform distributed applications that use remote procedure calls or other remote object protocols
- Using a browser and Web portals for "integration by eye" between a service provider and consumer is a ubiquitous and default form of "integration" -- but this is the Web equivalent of the telephone and fax machine with no automation
Enterprise Portals
Enterprise Portal Applications

Portal applications replace the different interfaces to multiple systems with a single, user-friendly screen that accesses only the parts of a back-end system that the employee needs.

Purpose is to create a unified experience with a "single sign-on".

You can think of this trying to recreate something like Yahoo for the enterprise (Intranet).

Nearly every major software vendor has created an enterprise portal solution that is an attempt to "up-sell" from the application server platform.
"Tight coupling" between two businesses, applications or services means that their interactions and information exchanges are completely automated and optimized in performance...

... by taking advantage of knowledge of their internal processes, information structures, technologies or other private characteristics that are not revealed in their public interfaces

... and usually implemented with a custom program that fit only between the two of them

Tight coupling is most often used, and usually limited to, situations in which the same party controls both ends of the information exchange

But tight coupling can be a "Faustian bargain" that trades efficiency for flexibility (John Hagel)
The Integration Challenge

Can we have integration and loose coupling at the same time?

The idea of service-oriented integration says we can

But we can get there from here?
Co-Evolution of Business and Technology Architecture

- Vertical 1980s and Earlier
- Horizontal 1980s and 1990s
- Ecosystem The New World

- Monoliths
- Structured
- Client/Server
- 3-Tier
- N-Tier
- Distributed Objects
- Components
- Services
Deconstructing the Four Services Perspectives

The concepts of discovery, coupling, and integration are present at all levels or perspectives on the "services stack," but they are manifested in business terms at the "top" and in technological terms at the "bottom"

- Service Oriented Architecture / Component Business Models
- Service Oriented Computing
- Web Services
- Web-based Services
Service Oriented Architecture - A Conceptual Perspective and Design Philosophy

Business processes are increasingly global and involve widely dispersed parts of an enterprise or multiple enterprises.

A business needs to be able to quickly and cost-effectively change how it does business and who it does business with (suppliers, business partners, customers).

A business also needs more flexible relationships with its partners and "assets" to handle variable demands.
Service Oriented Architecture - Business Components

So we need to think of "what a business does" in more granular terms:

- Business functions or services are "components"
- A business model is a composition or assembly of components

These business components can be a mix of core, internal ones that a business does itself and outsourced ones provided by other businesses
Definition of Component

A component offers a business service to other components

A component has:

- A business purpose
- One of more activities
- Resources
- Governance
- Services
"The notion of componentization allows an enterprise to deconstruct, analyze, and then reconstruct into value nets, in which partnerships with customers and suppliers operate in a network supported by real-time information flows and integrated IT systems"

"The process of deconstruction/reconstruction is realized through business components, which correspond to distinct business functions"

"In the on demand environment, the component-based firm links its components efficiently and seamlessly both internally and across the firm’s boundaries with best-of-breed components provided by external partners"
Proponents of "Service Oriented Architecture," "Service Oriented Computing," and "Web Services" have a vision:

- ... the component-based firm links its components efficiently and **seamlessly** both internally and across the firm's boundaries...
- ... the business components can be **plugged in** or unplugged with relative ease...
- ... permitting applications to be **constructed on the fly**...
Service Oriented Architecture and the "Plug and Play" Economy

"Plug and Play" was a marketing term used by Microsoft Windows in 1995.

"Plug and Play" as a metaphor for the integration of business components was first used around 1996 from Marty Tenenbaum.

For small businesses to participate in the web economy, the solution they used must be technically simple and relatively inexpensive to start with.

These requirements for "easy and cheap" integration are especially important in the lower tiers of supply chains.
Service Oriented Architecture -- Discovery and Integration

A service-oriented business needs to avoid the "lock-in" by technology or data format created by tightly-coupled integration.

The particular integration technology is less important than the philosophy or business model that requires it – treating different organizations, applications, and devices as loosely-coupled cooperating entities regardless of where they fit within or across enterprise boundaries.

Instead of thinking "how do these two applications exchange information," think "how do these two businesses exchange information."

This highlights the principles of discovery and transparent substitutability of service providers.
"Plug and Play" (Cherbakov et al.)
Modeling a Business as a Set of Components

Business processes are typically "factored" into components according to the "best practice" patterns in each industry.

An emphasis on business model / business process / information exchange patterns facilities component reuse / reassembly into new combinations - virtual enterprise, composite services.

"What components do" is defined in abstract, technology-independent terms so we don't have to care about the computer, operating system, or software application that performs each business process.

This level of abstraction reduces integration and communication costs between components and is the essence of service orientation.
Component Business Map -- Generic

<table>
<thead>
<tr>
<th>Directing</th>
<th>New Business Development</th>
<th>Relationship Management</th>
<th>Servicing and Sales</th>
<th>Product Fulfillment</th>
<th>Financial Control and Accounting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Planning</td>
<td>Sector Planning</td>
<td>Account Planning</td>
<td>Sales Planning</td>
<td>Fulfillment Planning</td>
<td>Portfolio Planning</td>
</tr>
<tr>
<td>Business Unit Tracking</td>
<td>Sector Management</td>
<td>Relationship Management</td>
<td>Sales Management</td>
<td>Fulfillment Planning</td>
<td>Compliance</td>
</tr>
<tr>
<td>Staff Appraisals</td>
<td>Product Management</td>
<td>Credit Assessment</td>
<td></td>
<td></td>
<td>Reconciliation</td>
</tr>
<tr>
<td>Staff Administration</td>
<td>Product Directory</td>
<td>Credit Administration</td>
<td></td>
<td></td>
<td>Customer Accounts</td>
</tr>
<tr>
<td>Production Administration</td>
<td>Marketing Campaigns</td>
<td>Sales</td>
<td></td>
<td></td>
<td>Document Management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contact Routing</td>
<td></td>
<td></td>
<td>General Ledger</td>
</tr>
</tbody>
</table>
# Component Business Map -- Rental Car Operations

<table>
<thead>
<tr>
<th>Plan</th>
<th>Products</th>
<th>Rentals Management</th>
<th>Rental Fleet Logistics</th>
<th>Business Administration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marketing and Customer Management</td>
<td>Rental Product Strategy</td>
<td>Location and Channel Strategy</td>
<td>Fleet Strategy</td>
<td>Corporate / LOB Strategy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Channel Design and Layout</td>
<td></td>
<td>Real Estate Planning</td>
</tr>
<tr>
<td>Management</td>
<td>Promotions Management</td>
<td>Channel and Location Profitability</td>
<td>OEM Performance Management</td>
<td>Alliance Management</td>
</tr>
<tr>
<td></td>
<td>Pricing Management</td>
<td>Location Operations Management</td>
<td></td>
<td>Business Performance Reporting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reservations Management</td>
<td>Inbound Logistics</td>
<td>Legal and Regulatory Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Workforce Management</td>
<td></td>
<td>Real Estate and Construction Management</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Risk Management:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Stock Ledger:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HR Management (Career Development, Training, Recruiting)</td>
</tr>
<tr>
<td>Execute</td>
<td>Customer Service</td>
<td>Purchasing/ Sourcing</td>
<td>Location Operations</td>
<td>HR Administration / Payroll</td>
</tr>
<tr>
<td></td>
<td>Preferred Member Management</td>
<td>Demand Forecasting</td>
<td></td>
<td>Corporate Audit</td>
</tr>
<tr>
<td></td>
<td>Customer Communications</td>
<td>Time and Attendance</td>
<td>Fleet Servicing</td>
<td>Corporate Accounting (GL, AP, AR, Treasury, etc.)</td>
</tr>
<tr>
<td></td>
<td>Mass Marketing and Advertising</td>
<td></td>
<td>Fleet Management</td>
<td>Indirect Procurement</td>
</tr>
<tr>
<td></td>
<td>Target Marketing</td>
<td></td>
<td></td>
<td>PR and Investor Relations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IT Systems and Operations</td>
</tr>
</tbody>
</table>
Rental Car Process Model -- India
The "Componentized" Bank
"Componentized" Bank Value Chains

Disaggregation of the value chain

Customer

Internet

Call Center

Branch

Internal and external distribution units

Rheinhyp/Eurohyp AG

Other product suppliers

Rheinhyp/Eurohyp AG

Settlement unit (prompter AG)

Settlement
Service Oriented Computing

Service Oriented Computing extends the abstract principles of Service Oriented Architecture to consider concerns that inherently arise because SOA must be realized in some computing technology.

These include:

- Transaction management and coordination
- Security
- Orchestration
- Resource management to ensure performance and quality of service
Alternatives for Service Realization

Build services internally -- pay entire cost

Outsource the building of services -- pay entire cost

Reuse an external service
  - Pay on subscription / lease basis
  - Pay by use
Web Services

Web Services -- with a capital "S" -- generally means a particular set of specifications for doing service-oriented integration with XML documents as the "payload" that conveys the information required by the service interface

(Or put another way -- the interface is specified using an XML schema that defines in a formal way the information the service expects and how it should be structured)

The most important Web Service specifications are those for a service's public interfaces (Web Service Description Language) and for the messaging protocol used to send and receive XML documents through those interfaces (SOAP)
The Web Services Model

Service Registry

Find

Bind and Invoke

Service Provider

Publish

Service Consumer

Service Description

Service
Web Services Standards & Mailing a Letter

To whom should we address the letter? ==> need directory so we can "discover" service providers (UDDI)

To what address do we send it, and what should the letter say? ==> need service description (WSDL)

How to address the envelope and put the letter in it? ==> need message packaging protocol (SOAP)
The Service Discovery Myth

Many discussions about services highlight the concept of service discovery and a specification called UDDI (Universal Description, Discovery and Integration)

UDDI was proposed as a kind of services "white" and "yellow" pages directory that would enable services to be registered by their providers and discovered by potential users, all by automated means.

But UDDI is mostly used for "internal" service directories and rarely for "public" ones.

- When you use an ATM, would the machine search for the bank's service interface in a directory?
- Most service relationships are established "offline" and then the information about how to access the service is built into the service requestor's implementation
- Would you trust the information published in a UDDI directory?
Web Services Abstract View

Service Registry

Functions

Business Process

Service

Service Description

Service Communication Protocol

Transport

Quality of Service

Policy

Security

Transaction

Management
# Web Services Standards Stack (2005)

<table>
<thead>
<tr>
<th>Business Domain Specific extensions</th>
<th>Business Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Management</td>
</tr>
<tr>
<td>Distributed Management</td>
<td>WSDM, WS-Manageability</td>
</tr>
<tr>
<td>Provisioning</td>
<td>WS-Provisioning</td>
</tr>
<tr>
<td></td>
<td>Security</td>
</tr>
<tr>
<td>Security</td>
<td>WS-Security</td>
</tr>
<tr>
<td>Security Policy</td>
<td>WS-SecurityPolicy</td>
</tr>
<tr>
<td>Secure Conversation</td>
<td>WS-SecureConversation</td>
</tr>
<tr>
<td>Trusted Message</td>
<td>WS-Trust</td>
</tr>
<tr>
<td>Federated Identity</td>
<td>WS-Federation</td>
</tr>
<tr>
<td></td>
<td>Portal and Presentation</td>
</tr>
<tr>
<td>Portal and Presentation</td>
<td>WSRP</td>
</tr>
<tr>
<td></td>
<td>Transactions and Business Process</td>
</tr>
<tr>
<td>Asynchronous Services</td>
<td>ASAP</td>
</tr>
<tr>
<td>Transaction</td>
<td>WS-Transactions, WS-Coordination, WS-CAF</td>
</tr>
<tr>
<td>Orchestration</td>
<td>BPEL4WS, WS-CDL</td>
</tr>
<tr>
<td></td>
<td>Messaging</td>
</tr>
<tr>
<td>Events and Notification</td>
<td>WS-Eventing, WS-Notification</td>
</tr>
<tr>
<td>Multiple Message Sessions</td>
<td>WS-Enumeration, WS-Transfer</td>
</tr>
<tr>
<td>Routing/Addressing</td>
<td>WS-Addressing, WS-MessageDelivery</td>
</tr>
<tr>
<td>Reliable Messaging</td>
<td>WS-ReliableMessaging, WS-Reliability</td>
</tr>
<tr>
<td>Message Packaging</td>
<td>SOAP, MTOM</td>
</tr>
<tr>
<td></td>
<td>Metadata</td>
</tr>
<tr>
<td>Publication and Discovery</td>
<td>UDDI, WSDL</td>
</tr>
<tr>
<td>Policy</td>
<td>WS-Policy, WS-PolicyAssertions</td>
</tr>
<tr>
<td>Base Service and Message Description</td>
<td>WSDL</td>
</tr>
<tr>
<td>Metadata Retrieval</td>
<td>WS-MetadataExchange</td>
</tr>
</tbody>
</table>
WS-* ("star" or "splat")

The major platform and enterprise software vendors (Microsoft, IBM, Sun, Oracle, BEA, HP, SAP...) have developed and "standardized" a few dozen specifications for extending the basic Web Services specifications to handle issues that emerge in complex distributed applications and service systems.

These specifications cover things like security, multi-hop addressing, process choreography, policy assertion, performance management, ...

Their proponents argue that these additional specifications are essential for service oriented computing to be viable for enterprise-level applications and services.

But they've made Web Services (with a capital "S") seem needlessly complex for a great many applications where they might have been useful.
Web-based Services

This is a category coined by Erik Wilde for his courses at the I-school to mean "Web Services and any services that use any Internet protocol"

This includes services implemented using the basic HTTP protocol and its mechanisms for providing "better service" using content negotiation (provide different information to the client based on the type of browser, etc.)

This broader category makes it easier to understand and make tradeoffs in the design and implementation of services
The Missing Web(-based) service Specifications

Web Services and Web-based Services share the principle that services are requested and delivered using documents (almost always encoded using XML)

But neither specifies any particular standard for what those documents mean - that is, there are no standard ways of encoding semantics in the document payloads

Many standards have been proposed -- for example, the Universal Business Language proposes a library of standard semantic "building blocks"

And unless services use the same payload standards or transform to them, integration challenges will remain and services will never be "seamless," "plug and play," or composed "on the fly"
Integration Is NOT Interoperation – the Transformation Requirement

Integration means that one application can extract or obtain information from another one.

It doesn't mean that the information will work "as is" for the receiving application or service.

Transformation of the incoming information is often required before it can make sense to the receiving application or service.

- Easy transformations are like field length, data type or unit conversion.
- More sophisticated transformations involve semantic processing (differences in vocabulary or even language translation).

This transformation could be performed by a gateway or connector service.
Readings for Next Lecture

Dean Jacobs "Enterprise Software as Service" ACM Queue
July/August 2005

Abhijit Dubey and Dilip Wagle, “Delivering Software as a Service,” The

Sean McGrath and Fergal Murray "E-Government Architecture in
Ireland" XML 2004 Conference