21. Service-Oriented Computing

INFO 210 - 5 November 2007

Bob Glushko
Plan for Today's Class

Simple vs Composite Services

Composite Services vs Service Systems

SOA and Web Services with "Hub" or "Marketplace" Architectures

Software as a Service
A Simple Service

A simple service is a request for a service that the provider can directly provide to the service consumer

- ...because the provider already has every required resource and doesn't need to search or wait or compute
- ... or because the context is static (or slow-changing)
- ... or because the service doesn't depend on the identity of the service requestor
A Simple (Information-Based) Service

A simple information service is a request for information that the service provider can directly provide to the service consumer

- ...because the provider already has the information and doesn't need to search or wait or compute for it
- ... or because the information is static (or slow-changing)
- ... or because the service doesn't depend on the identity of the service requestor

Example: maps, ISBN, phone directory, stock quote
A Less Simple Service

A less simple service is a request for service that the provider has to apply some business logic to answer

- ... because some of the required inputs have to be searched for, prepared, or generated

- ... or because it depends on the identity of the service requestor
A Less Simple (Information-Based) Service

A less simple service is a request for information that the service provider has to apply some business logic to answer

- ... because it has to be dynamically generated
- ... or because it depends on the identity of the service requestor

Example: account balance, catalog with contract pricing
Composite Services

A composite service provides a single user interface to a set of services linked by overlapping information requirements, business rules, and processes.

The application logic is reused from the component services, which can still function as services in their original contexts.

A common type combines services from internal legacy applications, where the lack of integration had made it difficult to provide good customer service.
Motivations for Internal Composite Services

-business strategy
perspective - can provide improved and new services to customers by combining customer information, order management, product information, etc.

-business operation
perspective - Reduce errors and time for information to flow from one "stovepipe" system or process to another by replacing manual methods of information sharing like re-typing or phone with automated interchange

-technology
perspective - reuse of legacy applications extends their useful lives and can be technically easier than reimplementing their functionality in a new application
Example: Why Enterprises Want Composite Services

An existing customer calls a service representation to increase an order

The service representative must:

- locate information about the customer (Customer Database)
- locate the existing order (Order Management Application)
- determine if the order can be changed or whether a new order must be created
- determine whether to accept the order based on the customer's payment history and credit (Accounting Application; Credit Check Application)
Before Service Composition
After Service Composition
Inter-Enterprise Composite Services

Rather than being provided by internal legacy systems, the separate services being composed might be provided by other firms

This makes the composite service the initiator, and possibly the controller of a long-running multistep collaboration

The interactions between the composed services is called a choreography

where there is distributed coordination with equivalent responsibility, where the sequence is determined by the service outputs, or where we don't need to take a position on how the overall process is controlled

The coordination is said to be orchestrated when one of the services controls the invocations of the others, serves as the conductor, or when all the services are controlled by an intermediary service (the "traffic cop" analogy)
Motivations for External or "Inter-Enterprise" Composite Services

*Business strategy*
perspective - can focus on core competencies by using services provided by other firms in component business models

*Business operation*
perspective - can improve collaboration with "real-time" information exchange with suppliers and other partners

*Technology*
perspective - composite application platform can exploit "on demand" services and reuse the models and techniques needed to integrate (and interoperate) applications and services
Drop Shipment
A Simple Service?

Welcome to My Financial Picture™

My Financial Picture is a free and convenient service available to Merrill Lynch clients. It provides a current view of all your financial information in accounts you enroll both from Merrill Lynch and other financial institutions. You may also include reward programs to consolidate your points for travel, hotel, and other rewards.

- Enroll Now!
  It’s easy to set-up My Financial Picture:
  1. Select individual online sites to add
  2. Enter your ID and password for each site
  3. Enter financial information not available online
  4. View all your accounts on one secure site

- Use My Financial Picture
  - To see an up-to-date summary of account balances, including banking, investments, credit card accounts, etc.
  - To list information not available online, including personal property, real estate holdings, etc.
  - To see the big picture so that you can make better financial decisions.
  - To enjoy one ID and password convenience to view your current information online.
  - To get better informed advice and guidance if you have a relationship with a Merrill Lynch Financial Consultant.

Merrill Lynch Research
Simple vs Composite Services -- Not That Simple

If a service interface is the published or public description of what the service does and how to request it then all services are simple from the service consumer's perspective.

That's because any of the services that have been composed "behind the scenes" are invisible to the service consumer... THAT'S THE POINT OF SERVICE-ORIENTED COMPUTING.

This *component anonymity* gives service providers flexibility in implementing and providing services because of *transparent substitutability*.

But the *granularity or abstraction* of the service presented to the service consumer matters a lot.
"Order Service" or Set of Services?

**BUYER**

**PURCHASING SERVICE**
- Input: Employee requisitions
- Output: Purchase Order to Supplier

**ORDER MANAGEMENT SERVICE**
- Input: Supplier Purchase Order Confirmation

**SUPPLIER**

**ORDER SERVICE**
- Input: Buyer Purchase Order
- Output: Purchase Order Confirmation
Order Service -- More Granular View

**BUYER**

PURCHASING SERVICE
Operations:
1. Select supplier
2. Select items
3. Check out
4. Managed confirmed order

**SUPPLIER**

ORDER SERVICE
Operations:
1. Authentication
2. Generate catalog
3. Confirm availability
4. Confirm order to Buyer
5. Notify fulfillment
Services -- Granular View

- Login
- Contract lookup
- Catalog browsing, with shopping cart and check-out
- Purchase Order Request creation
- Notify fulfillment process
- Confirm order to buyer
## RosettaNet Process Hierarchy

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Segments</th>
<th>Partner Interface Processes (PIPs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Partner Product &amp; Service Review</td>
<td></td>
<td>3A1 Request Quote</td>
</tr>
<tr>
<td>2 Product Information</td>
<td></td>
<td>3A2 Request Price &amp; Availability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3A3 Request Shopping Cart Transfer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3A4 Request Purchase Order</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3A5 Query Order Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3A6 Distribute Order Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3A7 Notify of Purchase Order Update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3A8 Request Purchase Order Change</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3A9 Request Purchase Order Cancellation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3A10 Notify of Quote Acknowledgment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3A11 Notify of Authorization to Build</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3A12 Notify of Authorization to Ship</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3A13 Notify of Purchase Order Information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3A14 Distribute Planned Order</td>
</tr>
<tr>
<td>3 Order Management</td>
<td>3A Quote &amp; Order Entry</td>
<td>3B Transportation &amp; Distribution</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3C Returns &amp; Finance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3D Product Configuration</td>
</tr>
<tr>
<td>4 Inventory Management</td>
<td>4A Collaborative Forecasting</td>
<td>4B Inventory Allocation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4C Inventory Reporting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4D Inventory Replenishment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4E Sales Reporting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4F Price Protection</td>
</tr>
</tbody>
</table>
Level of Service Abstraction

Constraint(s)

INPUT

Service

OUTPUT

Resource
Or Actor
Performing
The Service
Appropriate Abstraction for Service Composition

INPUT → Service 1 → Service 2 → OUTPUT
Granularity Implications

Reusability / Composability

Flexibility

... for other "ilities"
Let's go back to my introduction to "Service Systems" from lecture 5, about 8 weeks ago...
From "Service" to "Service Systems"

The SERVICE SYSTEM is the correct framework or perspective for understanding how services work.

A service system takes a more abstract view that de-emphasizes the obvious differences between person-to-person services and computational or automated ones.

A service provider's interaction with a service consumer transforms it or something else to create or "co-create" value.

THE SYSTEM OF RELATIONSHIPS AMONG THE PARTICIPANTS IN A SERVICE SYSTEM FOLLOWS PATTERNS OF VALUE CREATION AND BUSINESS ARCHITECTURE.

SERVICE SYSTEMS CAN BE COMPOSED FROM OTHER SERVICE SYSTEMS.
"Service System" Definition (Glushko)

A set of interconnected provider-consumer relationships and the flow of information through them

Every service has at least one service producer and one service consumer

A set of related services can define a SERVICE CHAIN or SERVICE NETWORK or VALUE CHAIN

Designating the last consumer in a service chain as the POINT OF VIEW establishes a perspective or context in the service system
Front and Back Stages
A Simple Service
A Simple Service in a Service System?

A simple service isn't really simple because it is offered in a cultural / social / legal context that determines many aspects of service delivery and quality.

And even if the provider has "every required resource" to deliver the service, these might have been originally obtained from some other provider.

So what makes a service "simple" is that these contextual constraints are ubiquitous or implicit and don't need to be be explicitly considered by the provider or consumer in the service encounter and that the resources are obtained asynchronously with service delivery.

Put another way, the "zone of visibility" or "front stage" for simple services includes only the immediate service provider.
A Composite Service
Composite Services in a Service System?

A composite service is a specialized type of point of view in a service system

This point of view combines the information provided by each of set of services to provide a new service

The services that are brought together typically have some common semantics and can satisfy some related goals for the service consumer, but don't necessarily have to be related in value chains or processes

Example: a composite travel service that lets the consumer select a flight, hotel, and rental car satisfies a set of related goals for the traveler, but the most important relationships among the choices are temporal and geographical, not causal or process ones
Service-Oriented Architecture for E-Government in Ireland

Design case study about the use of SOA and Web Services to implement e-Government services in Ireland

Uses a "hub" architecture to make services more readily reusable / composable

Abstractions of "service," "message," and "hub" make it easy to understand technology and business benefits and their interrelationships
The E-Government "Hub" or "Public Service Broker"
The PSB Architecture

The "Service & Data Exchange Catalog" is a registry that contains the models that define the structure and semantics of the messages exchanged to request and satisfy a service.

The PSB hub is "hollow" in functionality, doing little more than message routing and security functions -- but it provides the platform into which other value-added services are "plugged in."

Among the most important of these plugged-in services are the adaptors that transform message content and protocols to ensure interoperability and composability of services.
A Network of Hubs
A "customer facing service" is split into three distinct components:

- The user interface through which people interact with the service
- The integration part (at the hub) that orchestrates the routing of messages to the (one or more) services that provide the information to the user interface
- The fulfillment part -- "where the work gets done" to create / compute / process information that gets put into a message
Some Example Messages and Integrated Services

Messages like:

- This child has been born...
- Please look up the following car registration...
- Is the person with the following identifier entitled to a free telephone allowance...

Instead of requiring a new mother to apply for benefits when her child is born, the "new child message" can be directly sent to the agency responsible for child benefits

- This made 30,000 annual applications for benefits unnecessary
Let's go back to my fourth lecture on "Service Systems" from September 24...

The topic was "Information Flow & Exchange Models"

We can now see how SOA is an alternate conceptual view of those models and Web Services a physical view.
"In the Middle" -- Hub/Bus Architecture for Information Exchange
Information Aggregation and Exchange Patterns "in the Middle"

Marketplaces and Auctions

- Bring together sellers (or their catalogs)
- Bring together buyers (or their RFIs or RFQs)
- Match buyers and sellers
- Provide critical mass and infrastructure for other service providers

A "supply chain hub" can be thought of as a "private marketplace" run by the manufacturer/assembler in which all the participants are there because of a business relationship with the entity running the hub.
Services Provided "in the Middle" [1]

- Setting or selecting standards for protocols and messages
- Protocol and format conversion so that messages can be delivered and made mutually intelligible
- Routing documents between senders and receivers
- Registry and directory services so that participants and service providers be located or "discovered"
Services Provided "in the Middle" [2]

Core services (order management, content management)

Interfaces for services provided by others (logistics, shipping, tax calculation, credit, payment, reputation, etc.)

Support for "business process choreography" to create virtual combinations of the shared services

Aggregation (actual or virtual) seller catalogs with semantic integration (normalized and standard classification and description)
"Information Hub" in Supply Chain
Software as a Service

Significant amount of enterprise or "packaged" software is being transformed from an installed product to a hosted service (CRM, HR, BI are significant sectors, Supply Chain and ERP are coming)

Instead of installing the software on a local machine, the customer pays on a subscription or per use basis to access the functionality using a Web browser

And Google offers S-as-a-S applications for email, spreadsheets, and office documents for "free" (in exchange for you know what)

Many "deployed" software applications have become hybrids with a significant hosted or "vendor-managed" S-as-a-S component
S-as-a-S Value Proposition(s)

For the customer:

- Incremental implementation is less risky than "big bang" of enterprise applications
- "Freedom from the ball and chain of expensive hardware and software" (Reduced acquisition and maintenance cost so greater ROI)
- Scaleability and Quality of Service is responsibility of vendor, not IT department
- Benefits of belonging to larger user community: "community intelligence" and economies of scale result in services that wouldn't be economically viable

For the service provider:

- Easier support and maintenance
- Easier and more continuous innovation
Lower Total Cost of Ownership for S-as-a-S (McKinsey)

<table>
<thead>
<tr>
<th>Sample deployment of customer-relationship-management (CRM) software (200-seat license)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total cost of ownership, $ thousand</strong></td>
</tr>
<tr>
<td>Software on premises</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>Implementation, deployment</td>
</tr>
<tr>
<td>Customization, integration</td>
</tr>
<tr>
<td>Basic infrastructure testing, deployment</td>
</tr>
<tr>
<td>Application infrastructure testing, deployment</td>
</tr>
<tr>
<td>Ongoing operations</td>
</tr>
<tr>
<td>Training</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Management, customization of business process change</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Data center facilities rental, operations; security, compliance; monitoring of incident resolution</td>
</tr>
<tr>
<td>Software</td>
</tr>
<tr>
<td>User licenses, subscriptions; maintenance</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Other</td>
</tr>
<tr>
<td>Unscheduled downtime</td>
</tr>
<tr>
<td>Unused licenses</td>
</tr>
</tbody>
</table>
Salesforce.com

Salesforce.com is a S-as-a-S success story

Competing against conventional "internal" CRM applications that are integrated with enterprise ERP systems, in 6 years went from start-up to 1/2 $B revenue with 1/2 M subscribers, wiping out Siebel Systems

Provides web access to "Customer Resource Management" services needed by "traveling salespeople"

- Schedules
- Customers
- Products and pricing

AppExchange is a "service marketplace" that runs inside of the Salesforce.com user interface that offers over 200 applications that it doesn't offer itself
Software-as-a-Service -- Downside

For the customer:

- Business processes might be dictated by S-as-a-S vendor
- Customizability can be limited to UI and "branding"
- Security -- is critical information potentially visible to others? Can I even get at it for other purposes?

For the service provider:

- Performance, availability, and reliability are harder to guarantee in a S-as-a-S environment than in an installed enterprise one
- Can the existing functionality of enterprise apps be decomposed into S-as-a-S components of appropriate granularity?
- Can an enterprise app vendor survive the transition in business model from "big ticket" software to sales to "on demand" revenue streams?
Readings for November 7

R. Glushko & L. Tabas, “Bridging the ‘Front Stage’ and ‘Back Stage’ in Service System Design


Robert J. Glushko & Tim McGrath “Introduction to Document Engineering” Ch 1, Document Engineering MIT Press, 2005