Service Oriented Architecture

Service Oriented Analysis and Design (SOAD) in Practice
Part 4

Adomas Svirskas
Vilnius University
October 2005
Agenda

• Service identification and definition
• Business process specification
• Service interactions
The Layers of Design

Service Oriented Architecture Stack

Coupling and Scope [8]
SOA Modeling [2]

- Top-down, business-level modeling techniques such as CBM can provide the starting point for the SOA modeling activities.

- However, creating a SOA solution will almost always involve integrating existing legacy systems by decomposing them into services, operations, business processes, and business rules.
SOA Modeling [2]

- Existing applications and vendor packages are factored into sets of discrete services that represent groups of related operations (bottom-up approach)
- Business processes and rules are abstracted from the applications into a separate BPM, managed by a business choreography model.
Service Identification [1]

- A combination of top-down, bottom-up, and middle-out techniques of domain decomposition, existing asset analysis, and goal-service modeling.

- *Domain decomposition*, which consists of the decomposition of the business domain into its functional areas and subsystems, including its flow or process decomposition into processes, sub-processes, and high-level business use cases.
**Service Identification [1]**

- In the *bottom-up portion* of the process or *existing system analysis*, existing systems are analyzed and selected as viable candidates for providing lower cost solutions to the implementation of underlying service functionality that supports the business process.
- In this process, API’s, transactions, and modules from legacy and packaged applications are analyzed and leveraged.
- In some cases, componentization of the legacy systems is needed to re-modularize the existing assets for supporting service functionality.
Service Identification [1]

- The middle-out view consists of goal-service modeling to validate and unearth other services not captured by either top-down or bottom-up service identification approaches. It ties services to goals and sub-goals, key performance indicators, and metrics.
Service classification/categorization

• This activity is started when services have been identified. It is important to start service classification into a service hierarchy, reflecting the composite or fractal nature of services: services can and should be composed of finer-grained components and services [1]

• Classification helps determine composition and layering, as well as coordinates building of interdependent services based on the hierarchy. Also, it helps alleviate the service proliferation syndrome in which an increasing number of small-grained services get defined, designed, and deployed with very little governance [1]
Service Decomposition [2]
Direct and Indirect Analysis [2]

- *Direct* requirements analysis through stakeholder interviews and CBM are an obvious and well-suited way of identifying candidate services.
- Some *indirect* techniques are necessary
  - When mining for candidate services, product managers and other business leaders should be interviewed. For example, what are the planned payment and billing models?
  - The organizational chart of the enterprise that is supposed to use the system under construction should also be consulted.
  - Any existing use case models from non-SOA projects should also be consulted for advice.
  - The terminology used on marketing presentations for the system under construction is another great source of input.
Service Granularity [2]

- To select the right level of abstraction is a key service modeling issue.
- Model as coarse-grained as possible, without losing or compromising relevance, consistency, and completeness.
- There is room for fine-grained service abstractions in any SOA, assuming that there is a business need.
- SOA does not equal Web services and SOAP, different protocol bindings can be used to access services residing on different levels of abstraction.
- Bundling of several related services into coarser-grained service definitions, which is a variation of the facade pattern.
Service Allocation [1]

- Service allocation consists of assigning services to the subsystems that have been identified so far. These subsystems have enterprise components that realize their published functionality.
- Service allocation also consists of assigning the services and the components that realize them to the layers in your SOA. Allocation of components and services to layers in the SOA is a key task that require the documentation and resolution of key architectural decisions that relate not only to the application architecture but to the technical operational architecture designed and used to support the SOA realization at runtime.
Naming Conventions [2]

- An enterprise-wide naming scheme (XML namespaces, Java package names, Internet domains) should be defined.
- A simple example would be to recommend always assigning a service with a noun, and its operations with verbs (as in OOAD)
Service harvesting and knowledge brokering [2]

• This is a knowledge management and lifecycle issue: how can services successfully be prepared and made available for reuse once they have been conceptualized?

• Services should be identified and defined with reuse (and harvesting) as one of the main driving criteria of the SOA in mind. If a component (or service) has no potential for reuse, then it should probably not be deployed as a service. It can be connected to another service associated with the enterprise architecture, but will not be a service in its own right.

• However, even if reuse is planned for right from the beginning, the process still has to formalize the service harvesting process. Service usage by more than one consumer is an explicit SOA design goal. A build-time service registry, for example, an enterprise-wide UDDI directory can be part of the answer.
Example: Automotive Service Work Order [2]
Business Scenario [2]

• The work order is created when the customer calls to make an appointment.
• For each planned maintenance activity or operation, a separate work order item is created, containing details of the expected usage of parts, supplies, and labor.
• The inventory is checked to ensure that all necessary parts are in stock before the appointment is scheduled.
• A suitably-equipped service bay plus a suitably-qualified mechanic needs to be scheduled for each work order item.
Business Scenario [2]

• The estimated total cost is calculated, and the customer approves the appointment, or the scenario terminates and the work order is cancelled.

• Immediately before the appointment, the necessary parts, supplies, tools, and equipment are assembled in the selected bay.

• When the customer arrives, the planned activities are performed, plus any other activities that become apparent when the vehicle is inspected.
Business Scenario [2]

- Actual values for parts and supplies used and labor are recorded.
- On completion of all maintenance, the total cost is calculated.
- An invoice is created and presented to the customer.
Modeling from Scratch (OOAD)

Class diagram example [2]
OOAD Modeling

• If Work Order is constructed as an OO application, these software objects would contain all the necessary business rules and would understand the business processes that should be followed.

• There are some disadvantages of applying pure OOAD/CBD however
Towards SOA

- Many of the steps involve interfacing with existing legacy systems and databases such as billing, scheduling, and inventory systems, which were probably not designed adhering to the OO paradigm (applying an adapter or mediator pattern helps in such situations).
Towards SOA

• In order to make the system as flexible as possible, it would be helpful to have some of the rules externalized so that the processes or workflow can be modified without changes to the code. For example, rules like:
  – A standard service includes X liters of oil. Additional charges for oil should only be made if more than four liters are used, or if the customer requests a premium grade oil (such as a synthetic oil)
  – The customer should be contacted for confirmation if the estimate is exceeded by more than Y%.
SOAD Solution [2]

• Services are grouped on the basis of related behavior, rather than encapsulated (behavior plus data)
• For example, Work Order and Work Order Item are grouped together into Work Order Services
• Scheduling, Catalog, and Inventory services are separated
• As there are no services instances, there are no equivalents to relationships between services.
Services Model [2]

<table>
<thead>
<tr>
<th>Work Order Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations:</td>
</tr>
<tr>
<td>- Creative Work Order</td>
</tr>
<tr>
<td>- Operation: Create Work Order Item</td>
</tr>
<tr>
<td>- Calculate Estimated or Actual Costs</td>
</tr>
<tr>
<td>- Update Work Order Item</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Catalog Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations:</td>
</tr>
<tr>
<td>- Lookup/List available maintenance offerings by vehicle.</td>
</tr>
<tr>
<td>- List parts, supplies and labor for a specific maintenance item</td>
</tr>
<tr>
<td>- Lookup Parts and Supplies</td>
</tr>
<tr>
<td>- List valid alternative parts/supplies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scheduling Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations:</td>
</tr>
<tr>
<td>- Schedule skilled mechanic and service bay</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customer Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations:</td>
</tr>
<tr>
<td>- Lookup customer by tel #</td>
</tr>
<tr>
<td>- Create new Customer</td>
</tr>
<tr>
<td>- Get Customer Vehicles</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inventory Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations:</td>
</tr>
<tr>
<td>- Determine Quantity on Hand of Item</td>
</tr>
<tr>
<td>- Special Order Item</td>
</tr>
<tr>
<td>- Backorder Item</td>
</tr>
<tr>
<td>- Expected Arrival Date for Part</td>
</tr>
</tbody>
</table>
Services Model [2]

• Unlike the OO paradigm, this model does not represent a functional system
• There is no sense of flow, nor description of business events or rules
• In the SOA paradigm, business process choreography, maintained externally to the services, determines the sequence and timing of execution of the service invocations
• Conceptually, the entire business from first customer contact, through completion of the work and payment of the bill, represents a single, macro-level unit of work, with a lifetime of days to weeks. That unit of work generates revenue from a business perspective.
Process as State Transitions [2]

• However, in practice it is a series of intensive activities:
  – Scheduling an appointment
  – Selecting parts and supplies
  – Performing maintenance activities
• They are separated by relatively long periods of inactivity.
• In the IT system, there is no real process that lasts more than a few minutes; the state of the business process persists as data in a database between events.
State Transition Models

- This type of process can be well represented in state transition models.
- Business process choreography focuses on these transitions between states. Individual operations record the associated state changes permanently [2] [4].
Work Order Transition Model [2]
Business Interaction Model [2]
Conclusions

• For service identification, it is important to combine the three approaches of top-down, bottom-up, and cross-sectional, goal-model analysis

• SOAD notation and process are not complete, however it identifies key activities already
References


