Model-Driven Software Engineering

Model-Driven Development of SOA Applications I

Dr. Jochen Küster  (jku@zurich.ibm.com)
Agenda

- Brief Introduction to SOA
- The SoaML Profile
- Summary and References
Brief Introduction to SOA
What is a Service-Oriented Architecture (SOA)?

No single definition – “SOA is different things to different people”

- A set of services that a business wants to expose to their customers and partners, or other portions of the organization.
- An architectural style which requires a service provider, a service requestor (consumer) and a service contract (a.k.a. client/server).
- A set of architectural patterns such as enterprise service bus, service composition, and service registry, promoting principles such as modularity, layering, and loose coupling to achieve design goals such as separation of concerns, reuse, and flexibility.
- A programming and deployment model realized by standards, tools and technologies such as Web services and Service Component Architecture (SCA).

Adapted from: [IBM SSS]
Motivation: Evolve and integrate former monoliths into a Service-Oriented Architecture (SOA) ecosystem

- **Component-Based Development**
- **Messaging Backbone**
  - Point-to-Point connection between applications
  - Simple, basic connectivity
- **Enterprise Application Integration (EAI)**
  - EAI connects applications via a centralized hub
  - Easier to manage larger number of connections

**Service-Oriented Architecture**
- Integration and choreography of services through an Enterprise Service Bus
- Flexible connections with well defined, standards-based interfaces

**Flexibility**
SOA building blocks on the three levels of abstraction

- Business-Aligned Service Descriptions (Interface Contracts)
- Separation of Concerns and Modularity
- Loose Coupling and Messaging
- Service Repository/Registry
- Service Composition (Process Choreography)
- Enterprise Service Bus (ESB)
- Development Tools
- Execution Runtimes (e.g. SCA, J2EE)
- XML & Web Services Standards
- Internet Protocols
Overview of layers of a Service-Oriented Architecture
Key Concepts of SOA - Services

- A **service** is value delivered to another through a well-defined **interface** and available to a community (which may be the general public).

- A service results in work provided to one by another.

- A **service provider** is an organization which provides services and their implementations.

- A **service consumer** is a software which uses a service offered by a service provider.
Realization of Service-Oriented Architecture Solutions

- Many different (middleware) technologies for realizing SOA principles and patterns
  - CORBA
  - Web Services (WSDL, SOAP, UDDI)
  - REST-based integration

- Multiple programming languages
  - Java (JEE), C# (.NET)

- For more details:
  - Service Oriented Systems Engineering by Prof. Abraham Bernstein
The SoaML Profile
Motivation: Modeling of SOA Applications

- Ok, we know that SOA puts forward integration and choreography of services through an Enterprise Service Bus, together with flexible connections with well defined, standards-based interfaces.

But:
- How to model service providers/service consumer in a SOA?
- How to develop SOA applications in a model-driven approach?
Recap: Key Concepts of MDA

- **Platform Independent Model (PIM)** captures domain-related specifications
- PIM does not contain platform details, independent of a platform
- **Platform Specific Model (PSM)** captures specifications with platform details
- For expressing PIM and PSM, domain-specific languages are used
- Model transformations transform PIMs into PSMs
Recap: Sample Architecture

- How to model service layer in a platform independent form?
Case Study: Purchase Order

Global business-to-business scenario:

- A community of independent dealers, manufacturers and shippers want to be able to work together cohesively and not re-design business processes or systems when working with other parties in the community.
- They want to be able to have their own business processes, rules and information.
- The community decides to define a service oriented architecture for the community to enable this open and agile business environment.

Local company case study:

- Establish a common means of processing purchase orders
- Ensure that orders are processed in a timely manner and deliver the required goods
- Help minimize stock on hand and inventory maintenance costs
- Minimize production and shipping costs

Adapted from: [SoaML]
Modeling of SOA Applications - Revisited

- How to model services of a SOA?
  - SoaML

- How to develop SOA applications in a model-driven approach?
  - Follow an MDA approach using SoaML as the language for the PIM
The Service oriented architecture Modeling Language

- SoaML = Service-oriented architecture Modeling Language
- SoaML is an OMG Specification (current version 1.0, December 2009)

- SoaML provides a UML profile for the specification and design of services in an SOA solution
- SoaML is a language for modeling SOA solutions at different levels
Scope of SoaML

- Identify services, the requirements they are intended to fulfill, and the anticipated dependencies between them

- Specify services including their functional capabilities, the protocols or rules for using them, and the service information exchanged between consumers and providers.

- Defining service consumers and providers, what requisition and services they consume and provide, how they are connected and how the service functional capabilities are used by consumers and implemented by providers

- The policies for using and providing services
### SoaML Stereotypes and Descriptions

<table>
<thead>
<tr>
<th>SoaML Stereotype</th>
<th>UML metaclass</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>Port</td>
<td>A Service represents a feature of a Participant that is the offer of a service by one participant to others using well defined terms, conditions and interfaces. A Service designates a Port that defines the connection point through which a Participant offers its capabilities and provides a service to clients.</td>
</tr>
<tr>
<td>ServiceInterface</td>
<td>Class</td>
<td>Provides the definition of a service and defines the specification of a service interaction as the type of a Service or Request port</td>
</tr>
<tr>
<td>Participant</td>
<td>Class</td>
<td>A participant is the type of a provider and/or consumer of services. In the business domain a participant may be a person, organization or system. In the systems domain a participant may be a system, application or component.</td>
</tr>
<tr>
<td>Port</td>
<td>Port</td>
<td>Participants provide or consume services via ports. A port is the part or feature of a participant that is the interaction point for a service – where it is provided or consumed. A port where a service is offered may be designated as a «Service» port and the port where a service is consumed may be designated as a «Request» port.</td>
</tr>
</tbody>
</table>

Adapted from: [SoaML]
Service, Participant and Port Example

SoaML stereotype

- The **ServiceInterface** defines how a consumer of a service must interact.

- **Participant** Productions offers a Scheduling **Service** via a scheduling **Port**.

- **Port** specifies that **Participant** Productions offers that service.

Adapted from: [SoaML]
SoaML Stereotypes and Descriptions

<table>
<thead>
<tr>
<th>SoaML Stereotype</th>
<th>UML metaclass</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider</td>
<td>Interface, Class</td>
<td>Provider models the type of a service provider in a consumer/provider relationship. A provider is then used as the type of a role in a service contract and the type of a port on a participant.</td>
</tr>
<tr>
<td>Consumer</td>
<td>Interface, Class</td>
<td>Consumer models the type of a service consumer. A consumer is then used as the type of a role in a service contract and the type of a port on a participant.</td>
</tr>
<tr>
<td>MessageType</td>
<td>DataType, Class, Signal</td>
<td>The specification of information exchanged between service consumers and providers.</td>
</tr>
</tbody>
</table>
Provider, Consumer and ServiceInterface Example

- «ServiceInterface» Place Order Service is the root of the service interface and represents the service – the terms and conditions under which the service can be enacted and the results of the service.

- provider : Order Taker – this defines the role of the provider

- consumer: Order Placer – this is the role of the consumer

- Order Taker – this is the interface for a place order service provider. It lists all operations and signals a providing participant may receive when enacting this service.

- Order Placer – this is the interface for a place order service consumer. It lists all operations and signals a consuming participant will receive when enacting the service.

Adapted from: [SoaML]
Choreography Example

- The service choreography is a **behavior** owned by the service interface.
- It defines the required and optional **interactions** between the provider and consumer.
- There are two primary interaction sets – the quote request resulting in a quote and the order resulting in an order confirmation.

Adapted from: [SoaML]
The manufacturer is the provider of the place order service and has a «Service» port.

The dealer is a consumer of the place order service and uses a «Request» port.

Note that the manufacturer’s port provides the “Order Taker” interface and requires the “Order Placer” interface.

Adapted from: [SoaML]
SoaML Stereotypes and Descriptions

<table>
<thead>
<tr>
<th>SoaML Stereotype</th>
<th>UML metaclass</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServicesArchitecture</td>
<td>Collaboration</td>
<td>The high-level view of a Service Oriented Architecture that defines how a set of participants works together, forming a community, for some purpose by providing and using services.</td>
</tr>
<tr>
<td>ServiceContract</td>
<td>Collaboration</td>
<td>A ServiceContract is the formalization of a binding exchange of information, goods, or obligations between parties defining a service.</td>
</tr>
<tr>
<td>Capability</td>
<td>Class</td>
<td>A Capability models the ability to act and produce an outcome that achieves a result that may provide a service specified by a ServiceContract or ServiceInterface irrespective of the Participant that might provide that service.</td>
</tr>
</tbody>
</table>
Capabilities identify or specify a cohesive set of functions or resources that a service provided by one or more participants might offer.

Capabilities can be used by themselves or in conjunction with Participants to represent general functionality or abilities that a Participant must have.

Capabilities represent an abstraction of functionality

Adapted from: [SoaML]
Sample Architecture Revisited

- SoaML can be used to specify Service Layer in a platform independent and interoperable way
- SoaML allows to express patterns and principles of an SOA using a commonly accepted modeling language
Benefits of SoaML

- Enables **interoperability** and **integration** at the model level
- Provides a higher level of abstraction
- Separate from **platform variability**, allows for flexible platform choices
- Addresses **business integration** and **service interaction concerns** at the architectural level by using architecture as the bridge between business requirements and automated IT solutions
- Enables SOA both on and between existing platforms through model-driven architecture (MDA)
- Leverages and integrates with existing OMG standards for end-to-end life cycle development and management

- Compare these advantages to MDA advantages!

Adapted from: [SoaML]
Summary of Lecture

- Introduction to SOA and SOA key concepts
- SoaML profile for modeling SOA applications in a platform-independent way

References:

Further References

- M. Huhns et al. Service-oriented computing: key concepts and principles, IEEE Internet Computing, Volume 9, Issue 1, Jan-Feb 2005, Page(s):75 - 81