Model-Driven Solutions: Open Source Tooling for Implementing OMG's new SoaML Standard

London, UK
30 April 2009

Ed Harrington
ed-h@modeldriven.com

Copyright © 2009 Data Access Technologies, Inc. and Model Driven Solutions
• Problem is… the business folks have no idea what the Eggheads are talking about - a.k.a.

“The SOA Chasm”
Agenda

• OMG and SoaML
  – Submitters
  – Goals
  – Service
  – SoaML Metamodel
  – SOA Diagrams Supported

• Open Source Deployment:
  – Relating the Parts for Model Driven SOA
  – Business Focused SOA Using Model Driven Architecture
  – Tiered Deployment
  – Custom Business Logic Components
  – Application Provisioning

• Questions
Service Oriented Architecture Modeling Language
UML Profile and Metamodel for Services (UPMS)

Principle Authors:
Jim Amsden, IBM
Dr. Arne J. Berre, SINTEF
Cory Casanave, Model Driven Solutions

Service vocabulary, Specification, Contract, Correlation to Business Process...
The Submission Team

• **Submitters**
  – 88Solutions
  – Adaptive
  – EDS
  – Model Driven Solutions
  – Capgemini
  – Fujitsu
  – Fundacion European Software Institute
  – Hewlett-Packard
  – International Business Machines
  – MEGA International
  – MID GmbH
  – Rhysome
  – Softeam
  – Telelogic AB

• **Supporters**
  – Everware-CBDI
  – General Services Administration
  – VisumPoint
  – Mega
  – BAE Systems
  – DERI – University of Innsbruck
  – DFKI
  – France Telecom R&D
  – NKUA – University of Athens
  – Oslo Software
  – SINTEF
  – THALES Group
  – University of Augsburg
  – Wilton Consulting Group
Goals

- **Intuitive and complete** support for modeling services in UML
- Support for **bi-directional asynchronous services** between multiple parties
- Support for **Services Architectures** where parties provide and use multiple services.
- Support for **services defined to contain other services**
- Easily mapped to and made **part of a business process specification**
- **Compatibility with UML, BPDM and BPMN** for business processes
- Direct mapping to web services
- **Top-down, bottom up or meet-in-the-middle modeling**
- **Design by contract** or **dynamic adaptation** of services
- To specify and relate the **service capability and its contract**
- **No changes to UML**
Service

• **Open Group definition: A service** is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit; provide weather data, consolidate drilling reports)
  – Is self-contained
  – *May be* composed of other services
  – Is a “black box” to consumers of the service

• **OMG working definition: A service** (noun) is the work or action performed by one for another, enabled by one or more capabilities.

• “Here, the access to the service is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service contract. A service is provided by a participant acting as the *provider of the service—for use by others. The eventual consumers of the service may not be known to the service provider and may demonstrate uses of the service beyond the scope originally conceived by the provider.” [OASIS RM]
SOA Diagrams Supported

- **Service Structure Diagram** – the specification of a service including service contract, service interfaces, events and service choreography. The service structure and choreography diagrams combine to fully define the contract between service providers and consumers, independent of implementation and technology concerns.

- **Service Choreography Diagram** – the specification for how the providers and consumers of a service interact to fulfill the service contract. Indicates what information is sent between provider and consumer as well as when the information is communicated.

- **Services Architecture Diagram** – a high-level diagram of the participants in a SOA as well as the services they provide/use to meet their business objectives.

- **Message Type Diagram** – the model of SOA message data as well as the tie between SOA messages and the UML information model. Message types are then used in the Service Structure Diagram.
SOA Diagrams Supported (cont.)

- **Composite Application Component Diagram** – the model of components and sub-components that provide and use services as a composite application component that can be provisioned for deployment.

- **Activity Diagram** – the business process and activities within a composite SOA component that provides and uses services.

- **Capabilities Diagram** – the capabilities diagram shows the capabilities behind the services and what other capabilities each service depends on.

- **Provisioning Diagram** – the provisioning diagram defines the connection between an architecture and an implementation of that architecture. The provisioning diagram is used by the Open Source ModelPro engine to produce a deployable project. The provisioning diagram specifies the service components to be provisioned, what technologies will be used to implement each service component and where developers should augment the generated project with custom code.
A services architecture describes how participants work together for a purpose by providing and using services expressed as service contracts. It is modeled as a UML collaboration.

A participant represents some party that provides and/or consumes services. Participants may represent people, organizations or systems.

A service contract is the specification of the agreement between providers and consumers of a service as to what information, products, assets, value and obligations will flow between them. It specifies the service without regard for realization, capabilities or implementation.
A Participant can realize a Capability

A ServiceInterface can Expose a Capability
A business process represents the desired behavior among the various participants in a services architecture. This is modeled here as a UML activity.

Each participant is given a swimlane which contains the actions carried out by that participant within the business process.

The overall behavior emerges as an orchestration of the actions carried out by each of the participants. Interactions with participants must be consistent with the service contracts by which they interact.
Relating to Other Standards

- SoaML integration with BPMN 2.0 and BPDM will be related to the ongoing BPMN 2.0 standardization
- Extensions for Agents and semantic services will also relate to semantics, ontologies and other OMG metamodels like ODM and SBVR
- Limited BMM integration is included to tie services to the business

  ![Diagram]

  - Business requirements can be captured using the OMG Business Motivation Model (BMM).
  - Any UML BehavioredClassifier including (for example a ServicesContract) may realize the BMM Motivation concept of \textit{motivation realization}. This allows services models to be connected to the business motivation and strategy linking the services to the things that make them business relevant.
Open Source Deployment:

• Automatic Generation of Services
• Implementation Technologies (currently) Supported:
  – Web Services: XSD and WSDL
  – Eclipse IDE: project and build files
  – Java and JEE Implementations: Java source (user override capability) and required libraries for:
    • Services
    • Messages
    • Components
    • Data and Session Beans
  – Application Servers: Configuration and JAR files (tested on JBoss and Glassfish)
  – Constantly expanding
Relating the Parts for Model Driven SOA

ModelPro (ModelDriven.org) Open Source MDA Tools

- ModelPro Provisioning Engine
- SoaML Cartridge for JEE
- Provisioning Profile

OMG SoaML UML Profile

Users SOA Model

Provisioning Model UML Tool

Uses

Automates

Uses

Automates

Uses

Uses

Application

Deploy

Manual Platform Application Artifacts

Automated Platform Application & IDE Artifacts

Platform & Tools (E.G. Eclipse/Netbeans/.NET)
Business Focused SOA Using Model Driven Architecture

MDA Terms

Business Concerns

Business Model
Enterprise Services (e-SOA)
Roles, Collaborations & Interactions
Process & Information

Logical System Model
Technology Services (t-SOA),
Components
Interfaces, Messages & Data

Technology Specification
JMS, JEE, Web Services
WSDL, BPEL, XML Schema

Copyright © 2009 Data Access Technologies, Inc. and Model Driven Solutions

30 April 2009
Page 17
“To-Be Claims Processing” Tiered Deployment

Provisioning is the generation of some or all system implementation artifacts directly from the models.

A component can be deployed on a specific physical tier. A tier is modeled as a UML node.

The technical architecture for a tier defines the implementation platform to use when provisioning components on that tier.

Within the context of the technical architecture, component deployments can be stereotyped to affect how they are provisioned.

Shared services may be provided to multiple applications.

Note: The provisioning specification is part of Open Source ModelPro Project – not the SoaML Standard
Custom Business Logic Components

Application components provide service implementations with user supplied logic. These “plug into” the users architecture as composite application components.

Framework components add infrastructural capabilities by extending the platform (E.G. JBI) and are called by the provisioned code or platform configuration.

As MDA progresses, there will be less and less need for custom components, but the capability will remain.
Application Provisioning

- Platform technologies are provisioned from the model based on the technology specified
  - XSD
  - WSDL
  - Application Server Configuration
  - Java Interfaces & Implementation
  - XSLT
  - IDE Project
  - SQL
  - Documentation
  - Tests
  - ...

Details of what is provisioned for a particular technology are beyond the scope of this presentation
Thank You

Additional Information:

- OMG
- SOA ML

http://www.omg.org/cgi-bin/doc?ad/08-11-01

- ModelPro

http://www.modelpro.org

- MagicDraw

http://www.magicdraw.com/

(Fully supported modeling tool from MagicDraw with an integrated and supported version of ModelPro, included)

Ed Harrington, EVP & COO
Model Driven Solutions
ed-h@modeldriven.com

Copyright © 2009 Data Access Technologies, Inc. and Model Driven Solutions
30 April 2009
Page 21
Supplemental Materials

• Social Security Administration / ORSIS SOA Modeling Example
Value derived from the architecture

Business Concerns

One GSA Business Model
Business Services (b-SOA)
Roles, Collaborations & Interactions
Process & Information

Logical System Model
Technology Services (t-SOA),
Components
Interfaces, Messages & Data

Technology Specification
Web Services
WSDL, BPEL, XML Schema

Component Acquisition Specification

OMB 300

FEA/FTF
BRM
SRM
DRM*

Deployment

Data

Test & Simulation

Components

Adapters

Web Services

Business Driven Technology

Facilitating Business Processes
Focus on the Business Model

Business Concerns

Business Model
Business Services (e-SOA)
Roles, Collaborations & Interactions
Process & Information

Logical System Model
Technology Services (t-SOA),
Components
Interfaces, Messages & Data

Technology Specification
JEE, JMS, Web Services
WSDL, BPEL, XML Schema
Social Security Administration / ORSIS
Service Oriented Architecture (SOA)
Modeling Example

Ed Seidewitz
Computation Independent Model (CIM)

- RIB Claims Processing Services Architecture
  - RIB Claims Processing Business Process
- Apply for RIB Service Contract
  - RIB Application Service Interface
- Query for SSN Service Contract
  - SSN Query Service Interface
- Establish RIB Claim Service Contract
  - RIB Establishment Service Interface
- RIB Claims Processing Participants
A services architecture describes how participants work together for a purpose by providing and using services expressed as service contracts. It is modeled as a UML collaboration.

A participant represents some party that provides and/or consumes services. Participants may represent people, organizations or systems.

A service contract is the specification of the agreement between providers and consumers of a service as to what information, products, assets, value and obligations will flow between them. It specifies the service without regard for realization, capabilities or implementation.
A business process represents the desired behavior among the various participants in a services architecture. This is modeled here as a UML activity.

Each participant is given a swimlane which contains the actions carried out by that participant within the business process.

The overall behavior emerges as an orchestration of the actions carried out by each of the participants. Interactions with participants must be consistent with the service contracts by which they interact.
A service contract is the specification of the agreement between providers and consumers of a service as to what information, products, assets, value and obligations will flow between them. It specifies the service without regard for realization, capabilities or implementation. It is modeled as a UML collaboration.

The service contract defines the roles to be played by consumers and providers of the service. Many service contracts have only two roles, one a consumer and one a provider. But any number are allowed.

The service contract also defines the connections across which roles may interact.
A service contract may have a behavior that choreographs the allowed interactions between parties in the contract. This is modeled here as a UML activity.

Each role in the contract is given a swimlane which contains the actions that are expected to be carried out by that role.

Roles may exchange information via message objects.
The behavior of a service contract may also be modeled using other kinds of UML interaction models. It is modeled here as an interaction using a sequence diagram.

Each role in the contract is given a lifeline which acts as the source and target for the sending of messages.

Messages are modeled as being passed via calls to operations on the interfaces to the roles.

Condition flows can be modeled using interaction fragment constructs within the sequence diagram.
The messages passed between roles in a service contract are specified using message types. Message types are modeled as UML classes.

A message type may have data attributes but no operations or other behavior.

Note: Message information model has not been fully elaborated yet.
A service interface defines the interface and responsibilities required for a participant to play a role in a service contract. It is the means for specifying how a participant is to interact to provide or consume a service according to the contract. It is modeled as a UML class.

The operations used to pass messages to a role are collected into an interface for that role.

The service interface uses the interface of the consumer role.

The service interface realizes the interface of the provider role.

A service interface defines the interface and responsibilities required for a participant to play a role in a service contract. It is the means for specifying how a participant is to interact to provide or consume a service according to the contract. It is modeled as a UML class.
The use of a service contract is modeled as a UML collaboration use. Participants are bound the specific roles they play in the contract. Participation in a service contract requires that the participant type have a port with the corresponding service interface. A port is a connection point for providing or consuming services.

A request point is a port for requesting (consuming) a service. Note that the sense of provided and required interfaces is reversed at a request point: The port requires the provider interface and provides the consumer interface. The relative interface dependencies of the request point and service point “fit together” to allow a legal connection between the service consumer and provider.

A service point is a port for providing a service. The port provides the provider interfaces and requires the consumer interface.
Query for SSN Service Contract

<<ServiceContract>>

Query for SSN

querier : SSN Query Consumer

responder : SSN Query Provider
SSN Query Messages

- SSN Query Request
  - Name
  - DOB

- SSN Query Response
  - SSN

- SSN No Match Notice
SSN Query Service Interface

SSN Query Provider
+query for SSN( query request : SSN Query Request )

SSN Query Consumer
+provide SSN( query response : SSN Query Response )
+notify no match( no match notice : SSN No Match Notice )

<<Service Interface>>
SSN Query
Establish RIB Claim Service Contract
Establish RIB Claim Behavior

consumer: RIB Claim Establishment Consumer

provider: RIB Claim Establishment Provider

Request to Establish RIB Claims for an Applicant → RIB Claim Establishment Request → Search for SSN

RIB Claim Exists Notice: Notify User That Claim Exists → Establish Claim

RIB Claim Established Notice: Notify User That Claim is Established

Receive Response to Request
RIB Claim Establishment Messages

- RIB Claim Establishment Request
  - SSN
- RIB Claim Established Notice
- RIB Claim Exists Notice
RIB Claim Establishment Service Interface
RIB Claims Processing Participants

The full specification of a participant includes ports for every service contract in which the participant participates within the services architecture.

- **Applicant**
  - **RequestPoint**: RIB Application
  - **ServicePoint**: RIB Application

- **Claims Handler**
  - **RequestPoint**: SSN Query
  - **RequestPoint**: RIB Claim Establishment

- **SSN Matcher**
  - **ServicePoint**: SSN Query

- **RIB Claims Processor**
  - **ServicePoint**: RIB Claim Establishment
Producing the logical systems model

- **Business Concerns**
  - Business Model
  - Business Services (b-SOA)
  - Roles, Collaborations & Interactions
  - Process & Information

- **Logical System Model**
  - Technology Services (t-SOA), Components
  - Interfaces, Messages & Data

- **Technology Specification**
  - Web Services
  - WSDL, BPEL, XML Schema
Platform Independent Model (PIM)

- As-Is Claims Processing Services Architecture
  - Human Participants
  - System Participant Architectures
- MCS: Potential Tiered Replacement Architecture
- Claims Processing System: Potential Replacement Architecture
  - Citizen Self Service
  - Claims Rep Assisted Service
As-Is Claims Processing Services Architecture

The as-is claims processing architecture is modeled here as a services architecture showing how the roles CIM-level business architecture are currently being played.

The business process being carried out is defined by the CIM-level services architecture, which defines the process roles and desired behavior.

A customer is a participant who is external to, and being served by, the enterprise carrying out the business process.

A worker is a participant who is internal to the enterprise carrying out the business process.

A system is a participant that whose responsibilities are being automated using an IT system.
As-Is Claims Processing Human Participants

At the PIM-level, some participants may be known not to be automated. Such participant types generally represent positions filled by people in the enterprise.

Participants at the PIM level can realize (one or more) participants at the CIM level. This indicates the intended way the PIM-level participants are to participate in various business processes. The PIM-level participant model must have ports that conform to all the ports of the CIM-level participant.
A **user interface** is the provision of services in a form directly accessible by external human participants.

The responsibilities for providing (or consuming) a service can be *delegated* to an internal participant.

A **participant architecture** is a services architecture that defines the implementation of the responsibilities of a participant in some higher-level architecture.

A PIM-level participant may have additional ports/interfaces to those required by the CIM-level participant being realized.
Alphadent System Architecture
As-Is Claim Processing Composite Structure

A service channel connector shows how a consumer is connected to providers of services. One end is always a request point, the other a service point.

The PIM-level architecture may include supporting participants that do not directly play business roles in the CIM-level business architecture model.
This is a simplified example of a logically tiered replacement architecture for MCS, in which internal file interfaces have been replaced by a service interface.

The replacement architecture plays the same business role as the original MCS.

A presentation component makes one or more services available to human participants by providing them through user interfaces and then forwarding requests to a service component for processing.

A service component implements one or more provided services via interfaces available to other system components (as opposed to humans).

These services are presumed to be available from some shared service provider.
This is a simplified example of a more sweeping replacement for the entire claims processing architecture.

An applicant presentation component allows for customer self-service, mediating the use of the same services internally that are directly available to a claims rep.

Service components are designed to provide services specifically related to different business entities.

For convenience, all shared service interfaces have been grouped into a single composite service interface.
To-Be Claims Processing Architecture: Citizen Self Service

The applicant user interface mediates the same basic business interaction that an applicant would have had with a Claims Rep.

In a self-service scenario, the Application Presentation component automates the claims handler role, replacing the human Claims Rep.

The service components handle requests made via the presentation component, playing the ident and processor roles in the business process.
To-Be Claims Processing Architecture:
Claims Rep Assisted Service

In an assisted-service scenario, the Claims Rep still plays the handler role for the Applicant.

The Claims Rep Presentation component acts as a façade allowing the Claims Rep access to the underlying services provided by the service components. Relative to the Claims Rep it effectively plays both the ident and processor roles.
Custom Business Logic Components

Application components provide service implementations with user supplied logic. These “plug into” the users architecture as composite application components.

Framework components add infrastructural capabilities by extending the platform (E.G. JBI) and are called by the provisioned code or platform configuration.

As MDA progresses, there will be less and less need for custom components, but the capability will remain.

Custom part is separate from the generated part.
Platform Specific Model (PSM)

- MCS Tiered Deployment
- Claims Processing System Tiered Deployment
Technology Architecture

**Business Concerns**

- **Business Model**
  - Business Services (b-SOA)
  - Roles, Collaborations & Interactions
  - Process & Information

- **Logical System Model**
  - Technology Services (t-SOA), Components
  - Interfaces, Messages & Data

- **Technology Specification**
  - JEE, JMS, Web Services
  - WSDL, BPEL, XML Schema
MCS Tiered Deployment

A component can be deployed on a specific physical tier. A tier is modeled as a UML node.

Service connections across tiers are modeled as UML assembly dependencies.

Shared services may be provided to multiple applications. (The provider shown is illustrative—there may be more than one provider for different sets of shared services.)

The provisioning specification is part of ModelPro—not the SoaML Standard.
Application components provide service implementations with user supplied logic. These “plug into” the users architecture as composite application components.

Framework components add infrastructural capabilities by extending the platform (E.G. JBI) and are called by the provisioned code or platform configuration.

As MDA progresses, there will be less and less need for custom components, but the capability will remain.
To-Be Claims Processing Tiered Deployment

Provisioning is the generation of some or all system implementation artifacts directly from the models.

The technical architecture for a tier defines the implementation platform to use when provisioning components on that tier.

Within the context of the technical architecture, component deployments can be stereotyped to affect how they are provisioned.
Application Provisioning

- Platform technologies are provisioned from the model based on the technology specified
  - XSD
  - WSDL
  - Application Server Configuration
  - Java Interfaces & Implementation
  - XSLT
  - IDE Project
  - SQL
  - Documentation
  - Tests
  - ...

Details of what is provisioned for a particular technology are beyond the scope of this presentation