Model-Driven Development of Service-Oriented Systems

Nora Koch
LMU München and Cirquent GbmH

in cooperation with the SENSORIA team

in particular Martin Wirsing (coordinator),
Philip Mayer and Matthias Hölzl
LMU München

Seville, 29.4.2010
Outline

- Part 1: Setting the Scene
  - Project overview
  - Service-oriented systems
- Part 2: Model-Driven Development of Service-Oriented Systems
Aim

- to provide you with an overview to a model-driven development approach for service-oriented systems that we developed in the SENSORIA project
  - methodological aspects of the engineering process
  - a modelling language
SENOSORIA Project

- EU project
  - 6th Framework Programme (FP6) 2005 – 2010
  - 19 partners
  - Coordination: LMU
  - www.sensoria-ist.eu

Software Engineering for Service-Oriented Overlay Computers

Seville – 2010 – Nora Koch ©
Service-orientation

- **Service**
  - autonomous, platform-independent computational entity that can be described, published, categorised, discovered

- **Service-Oriented Systems (SOS)**
  - use *loosely coupled* services
  - massively *distributed, interoperable, evolvable* applications

- **Service-Oriented Computing (SOC)**
  - the compute paradigm behind service-oriented systems, i.e. for organizing and utilizing distributed capabilities that may be under the control of different ownership domains

- **Service-Oriented Architecture (SOA)**
  - an architectural style to realize SOC
  - promise to organize and understand organizations, communities and systems maximizing agility, scalability and interoperability
Stakeholders/Parties in SOAs

- **Service providers**
  - offer services that correspond to ‘market’ demands

- **Service consumers/requesters**
  - are applications, not people
  - are decoupled from the providers
  - binding to services at run time, not design time

- **Service brokers**
  - manage registries
  - binds consumer and provider
  - offered as middleware in SOAs

- **SOA triangle**
Challenges for service engineering

Software engineering for SOS

- Specification and discovering services
- Correctness and consistency
- Automated composition of services (orchestration)
- Compensation of long running transactions
- Analysis of performance, security and safety, adaptive behaviour, …
- Deployment and re-engineering
Development process and analysis of service-oriented systems is supported by

- language primitives for global service-oriented computing
- qualitative and quantitative analysis methods
- sound engineering methods and deployment techniques
Language primitives

- SRML
  - formal founded modelling language
  - properties of required and provided services are specified in temporal logic
- UML family of profiles for SOC
- JOLIE
  - Java Orchestration Language + Interpreter
  - JOLIE is an open-source project developed in Java
- Process calculi for services
  - to describe, discover and compose systems
  - to prove that their behaviour is consistent with the expectation of the designer
Qualitative and quantitative analysis methods

- **Adaptive and Dynamic Service Compositions**
  - to ensure safety and correctness of adaptive and dynamic service composition specifications

- **Model checkers**
  - which verifies safety properties on the behaviour of services

- **BPEL Analysis and Back-Annotation.**
  - end-to-end method which facilitates analysis of several liveness and safety properties of BPEL orchestrations

- **SRMC - SENSORIA Reference Markovian Calculus**
  - allows for performance predictions which are valid whichever configuration of service providers is selected
(Re)engineering and deployment techniques

- Engineering
  - Eclipse-based SENSORIA development environment (SDE)
- Deployment techniques
  - end-to-end support for dynamic service composition from modelling to runtime
  - deployment and brokering with Dino
- Re-engineering
  - prototype for re-engineering legacy applications to SOA
Further results: Spin-off companies

- **AGILOGIK**
  - 2009
  - monoidal soft constraint solver for optimization problems
  - Steingaden, Germany

- **Italiana Software**
  - 2007
  - design and implementation of SOAs with Jolie
  - Imola, Italy

- **OptXware**
  - 2005
  - model transformations with VIATRA2
  - Budapest, Hungary
Part 2: Model-Driven Development of Service-Oriented Systems

- Models
- Metamodels
- Model transformations
- Tool support
- Model-driven development @ work
Use of models in SENSORIA

- **To specify SOSs**
  - structure, behaviour, ...
  - separate concepts at different conceptual levels
  - communicate with stakeholders

- **To understand the SOS**
  - legacy applications

- **To validate SOSs**
  - detect errors and omissions in design ASAP
  - prototype the system (*execution* of the model)
  - formal analysis of system properties

- **To drive implementation**
  - code skeleton and templates
  - complete programs (if possible)
SENSORIA
Model-driven development process

Feedback
Formal Methods
Requirements
Modelling
Transformation
Code Generation
Runtime
Reengineering
Legacy Systems
Modelling languages

- Objective is to have a domain specific graphical representation and clear semantics for service-oriented concepts
  - **Option 1:** Definition of a proprietary language, like SENSORIA Reference Modelling Language (SRML)
    - high cost: requires the definition of all required domain specific concepts and proprietary tools
  - **Option 2:** Use of a standard, like Unified Modeling Language (UML™), Business Process Modeling Notation (BPMN™)
    - diagrams are more difficult to read
  - **Option 3:** Define a UML2 profile
    - using the extension mechanism that allows to customize the UML for specific domains and platforms
    - defining stereotypes, tagged values and constraints to restrict and extend the scope of UML
    - UML CASE tools can be used
UML2 profile(s)

- **Main Aim**: to have a powerful yet readable graphical modelling language for SOAs – based on UML
  - “minimalist” extension
    - use UML constructs wherever possible
    - use other extensions if available
    - only add new model elements where needed
  - reducing efforts of building SOA models
    - covering domain specific aspects, such as
      - service contracts
      - long running transactions and compensation
      - loose coupling of services

- **Secondary Aim**: to employ transformers from such models to common implementation languages (BPEL, Java...)

- SoaML / UML4SOA
- MDD4SOA
UML extensions for SOA modelling

- **SoaML profile** (OMG standardization process beta1 version)
  - Service-oriented architecture Modeling Language
  - for structural aspects of services

- **UML4SOA profile** (developed within the scope of the project)
  - for behavioural aspects, e.g. orchestration
  - for non-functional aspects
  - for reconfiguration
  - for policies
  - for requirements

- **MARTe profile** (OMG standardization process beta2 version)
  - for performance analysis
UML4SOA, SoaML, MARTE

- Defined as UML profiles
  - provide a set of elements for modelling SOAs
  - use UML extension mechanisms (stereotypes)
  - no changes to UML (exception SoaML propose one change)

- Use of the profiles
  - to build models at different levels of abstraction
  - in combination with UML model elements
  - is not a prescriptive approach
SOA models in the MDA context

- **Computation Independent Model (CIM)**
  - **Business Model**
  - **Enterprise Services**
    - Roles, Collaborations, Dependencies, Workflows

- **Platform Independent Model (PIM)**
  - **Design Model**
  - **Services**
    - Components, Interfaces, Messages, Data

- **Platform Specific Model (PSM)**
  - **Technical Specification**
  - **Technical Services**
    - WSDL, BPEL, XML Schema, Java, Jolie

Source: Data Access Technologies, Inc
Finance Case Study: Credit Portal Scenario

- Stakeholders (parties) of the service-based scenario are customers, clerks and supervisors.
- Login is required, if a customer wants to request a credit by using the credit portal.
- The credit request process requires from the customer credit data, security data and balance data.
- Based on the uploaded information the system calculates a rating that is used for an automatic decision, a clerk or supervisor decision.
- In case of a positive decision the process informs the customer and waits for his decision.
- Once the credit offer is accepted, the process stores the credit offer in an agreement system and the process is finalised.
- In case of a negative decision the customer is informed about this decision and the process ends, too.
Specifying service capabilities

- Capabilities are used
  - to identify needed services
  - to organize them into catalogues or network of capabilities
  - prior to allocating those services to particular service providers and requesters

A capability is the specific ability to provide a service. It is modelled as UML class.

SoaML Specification for the UML Profile and Metamodel for Services (UPMS), OMG 2008
Identifying parties involved in SOAs

- Provider and consumers of services are represented as participants
  - in the business domain: person, organization or system
  - in the systems domain: system, application or component
- Participant can play the role of
  - providers in some interactions
  - consumers in others

A participant represents some party that provides and/or consumes services. It is modelled as UML class.
A service contract is the specification of the agreement between providers and consumers of a service. It is modelled as a UML collaboration.

A dependency represents the binding of the service contract to the provider or the consumer of the service.

A participant can play different roles.

- A service contract specifies the service without regards for realization or implementation.
- A UML2 collaboration defines a set of cooperating entities to be played by instances (its roles), as well as a set of connectors that define communication paths between the participating instances.
A **service architecture** describes how participants work together for a purpose by proving and using services expressed as service contracts. It is modelled as a UML collaboration.
SOA models in the MDA context

Computation Independent Model (CIM)

Business Model
Enterprise Services
Roles, Collaborations, Dependencies, Workflows

Platform Independent Model (PIM)

Design Model
Services
Componentes, Interfaces, Messages, Data

Platform Specific Model (PSM)

Technical Specification
Technical Services
WSDL, BPEL, XML Schema, Java, Jolie

Source: Data Access Technologies, Inc
Refining specification of participants with ports

- Add ports for provided and consumed services
- A port has as type a service interface or an interface

A full specification of a participant includes **ports** for every service contract in which the participant participates within the service architecture. Two types of ports: **service point** and **request point**.
Modelling service interfaces

A service point is a port for providing a service.

A request point is a port for requesting (consuming) a service.

A service interface allows for connection between the service consumer and provider. It is modelled as UML class.

A UML interface is used to represent the required and provider interfaces of the ports.

- A service interface
  - implements ("provides") provider interfaces (represented as realisation)
  - "requires" consumer interfaces (represented as a «use» dependency)
Interface behaviour

**UML**

- **UML4SOA**
  - propose protocol state machine
- **SoaML**
  - propose activity diagrams or sequence diagrams
Orchestration of services

- Service orchestration is the process of combining existing services to form a new service to be used like any other service.

- Key distinguishing concepts
  - partner services
  - message passing among requester and provider
  - long-running transactions
  - compensation
Message passing
Synchronous and asynchronous service invocation

Service interaction `send` sends a message. Does not block.

Service interaction `receive` blocks until message is received.

Service interactions `send&receive`, `receive&send` denotes a sequential order of these actions.

`Reply` is used for the reception of a message decoupled of the sending process.
Detailing service invocation
Partner services and data handling

Initialization

<<receive>>
createNewCreditRequest
(creditData : CreditData )

<<send&receive>>
initCreditData
(CreditManagementProvider::)

<<reply>>
createNewCreditRequest
(creditData : CreditData )

Pins containing interaction information

lnk: partner
snd, rcv: data to be send or received

Use of variable after declaration

Implicit declaration of variable in a snd pin.
Data handling

SoaML/UML4SOA

- Declaration of structured types
  - extends metaclass data type and class

- Use in behavioural diagrams
  - support for typed, scoped variables in the orchestration
  - data handling support

A **message type** is used to specify information exchanged between service consumers and providers (message passing).

A **data action** can be used to explicitly declare the type of a variable or for manipulation of data (copy, calculation, etc.).
Long running transactions

- Require compensation mechanisms, e.g. compensation handlers

A compensation Handler is added using a compensation activity edge.

The service activity modelling the compensation handler will be triggered by a `compensate` or `compensateAll`.
A `compensateAll` triggers all active compensation handlers in the reverse order.
### SOA model elements and diagram types

<table>
<thead>
<tr>
<th></th>
<th>Business model</th>
<th>Design model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structural aspects</strong></td>
<td>capabilities, participants, service contract, service architecture, participant architecture</td>
<td>service point, request point, service interface, message type</td>
</tr>
<tr>
<td><strong>Behavioural aspects</strong></td>
<td>service activity</td>
<td>send, receive, send&amp;receive reply, raise, lnk, snd, rcv, compensate, compensateAll, compensation, exception, event data</td>
</tr>
<tr>
<td><strong>Diagram type</strong></td>
<td>class diagram, composite structure diagram, activity diagram</td>
<td>class diagram, composite structure diagram, activity diagram, state machine</td>
</tr>
</tbody>
</table>

+ use of plain UML, e.g. SOA's protocols
Quality of services

- Defined by non-functional properties (NFP)

- Example: Credit Portal Scenario
  - The Portal and the CreditRequest should communicate via a secure and reliable connection
  - All requests sent to the CreditRequest should be acknowledged
  - As the credit request handles confidential data, all requests should be encrypted in order to protect the privacy of the customers
  - Messages sent by the CreditRequest must be clearly accountable, i.e. non-repudiation of messages must be guaranteed
Modelling approach for NFP of services

Template for a service level agreement (SLA)
Modelling a concrete configuration

Concrete SLA

- **CreationValidationContract**
  - **ReliableMessagingCharacteristics**
    - **MsgSemantics**
      - filterDuplicates = true
      - maxNumberOfRetrans = 3
      - needsAck = true
  - **SecurityCharacteristics**
    - **Authentication**
      - authToken = "username"
    - **Encryption**
      - encryptBody = true
      - encryptAlgorithm = "default"
      - encryptHeader = false
      - encryptSignature = false
    - **DigitalSignature**
      - signAlgorithm = "default"
      - signBody = true
      - signHeader = false
MDE approaches
- are based on the constructions of models
- propose transformation of models
- implement model transformations based on the metamodel of the modelling language

MDE approaches require languages for
- specification of models
  - UML, BPMN, …
- description of metamodels
  - UML, MOF, OCL, …
- definition of model transformations
  - Java, graph transformations, ATL, QVT…
Language definition mechanisms

- Options for defining a new modelling languages
  - New MOF-based modelling language
  - UML extension (profile)
UML Profile

- Extension of the UML for domain specific model element
  - providing a different notation
  - enriching model elements with additional semantics (e.g. request point)
  - representation of domain specific patterns (e.g. compensation)
  - annotations (marks) facilitating model transformations in a model-driven approach (e.g. Ink)

- Use of extension mechanisms of the UML
  - stereotypes
  - tagged values
  - constraints

- Risks
  - too many stereotypes
  - selection of inadequate UML metaclass
  - decorative and redefined stereotypes (→)
Creating a UML profile

- Specification of a metamodel for the specific domain
  1. identification of the domain specific concepts and their relationships
  2. construction of a model capturing concepts and relationships (metamodel)
  3. UML elements for this concepts? (minimalist extension)

- Specification of the profile
  1. creation of stereotypes for identified elements (#3 is false)
  2. identification of appropriate UML metaclasses
  3. stereotypes and metamodel elements related by an "extension" (multiple metaclasses)
  4. define semantics of new elements
UML4SOA metamodel: Orchestration
Conservative extension of the UML
UML4SOA metamodel: Orchestration (cont.)
Conservative extension of the UML
Profile metamodel mapping (excerpt)
Extension model (excerpt)
SOA models in the MDA context

Computation Independent Model (CIM)

Business Model

Enterprise Services
Roles, Collaborations, Dependencies, Workflows

Platform Independent Model (PIM)

Design Model

Services
Componentes, Interfaces, Messages, Data

Platform Specific Model (PSM)

Technical Specification

Technical Services
WSDL, BPEL, XML Schema, Java, Jolie

Source: Data Access Technologies, Inc
MDD4SOA

- MDD4SOA
  - Transformation mechanisms from models to executable orchestration of services
    - source: UML4SOA models
    - target platforms: BPEL/WSDL, Java, Jolie
    - fully automatic generation of code
    - implemented in Java

Mayer et al, EDOC 2008
Demonstration’s aim
- to show how model-driven development of SOSs can work

Consists of
1. building an orchestration model with UML4SOA
2. defining a tool chain of transformations in SDE
   - model2model, model2code, deployment
3. execution of the tool chain
   - input: UML4SOA model
   - output: application
4. running the deployed application
5. changing the model
6. go to 3
SENSORIA Development Environment (SDE)

- **Eclipse-based** integration platform for developing SOA-based software
  - SDE Core
  - integrated tools

- Distinctive features of the SDE Core
  - uses a SOA approach itself
  - tools are orchestrated by the specification of a tool chain
  - **tool-as-service concept**: Orchestrations of tools are now usable as tools themselves
  - enables SOA developers to use tools without the need to understand the underlying formal languages

- Tool chain in SDE
  - defined as a SDE script
  - drawn with the graphical orchestration tool
  - executable in the Eclipse environment
SDE: Eclipse-based environment

- SENSORIA Browser
  - tools as services
- SENSORIA Blackboard
  - e.g. results
- SENSORIA Shell
  - e.g. tool orchestration scripts
## Tool support (excerpt)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Area</th>
<th>Case Study</th>
<th>Integration in SDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDE</td>
<td>Integration</td>
<td>All</td>
<td>X</td>
</tr>
<tr>
<td>UML4SOA profile</td>
<td>Modelling</td>
<td>Finance, eUniversity, Automotive</td>
<td>X</td>
</tr>
<tr>
<td>MDD4SOA Transformer</td>
<td>Transformation</td>
<td>Finance, eUniversity, Automotive</td>
<td>X</td>
</tr>
<tr>
<td>VIATRA Transformations</td>
<td>Transformation</td>
<td>Finance, eUniversity, Automotive</td>
<td>X</td>
</tr>
<tr>
<td>WS-Engineer</td>
<td>Analysis</td>
<td>Finance, eUniversity, Automotive</td>
<td>X</td>
</tr>
<tr>
<td>SRMC</td>
<td>Analysis</td>
<td>Finance, eUniversity, Automotive</td>
<td>X</td>
</tr>
<tr>
<td>LySa</td>
<td>Analysis</td>
<td>Finance</td>
<td>X</td>
</tr>
<tr>
<td>UMC/CMC</td>
<td>Analysis</td>
<td>Finance, Automotive</td>
<td>X</td>
</tr>
<tr>
<td>Dino</td>
<td>Runtime</td>
<td>Finance, Automotive</td>
<td>X</td>
</tr>
<tr>
<td>SRML Editor</td>
<td>Modelling</td>
<td>Automotive, Finance</td>
<td>-</td>
</tr>
<tr>
<td>ADR</td>
<td>Visualisation</td>
<td>Automotive</td>
<td>-</td>
</tr>
<tr>
<td>sCOWS_LTC/sCOWS_AMC</td>
<td>Analysis</td>
<td>Finance</td>
<td>-</td>
</tr>
<tr>
<td>Reengineering Tool</td>
<td>Reengineering</td>
<td>Finance</td>
<td>-</td>
</tr>
</tbody>
</table>
Selection of tools, techniques, methods, languages, ...

- **SENSORIA approach**, in particular the integrated tools in SDE encompasses
  - the whole development process of service-oriented software
  - from systems in high-level languages to deployment and re-engineering

- **Difficulty to identify the “best” techniques and tools (SDE plug-ins)**
  - for solving a particular problem arising in the development process

- To ameliorate this problem we defined a **catalogue of patterns**
  - serves as an index to our results
  - illustrates, in a concise manner, the advantages and disadvantages of the individual techniques
Automotive case study

- Scenario On Road Assistance
  - Driver is on the road with his car
  - Diagnostic system reports a low oil level; the car is being no longer driveable
  - Driver contacts the on road assistance system
  - Car position is located
  - System finds appropriate services in the area (garage and rental car)
  - Based on the drivers preferences the best services are selected
  - Driver is required to deposit a security payment by credit card

- On Road Assistance as orchestration of services
  - services: car position, finding garage and car rental station, selection of best service, charge credit card

- Application: visualisation of invoked services
  - Each service has associated a user interface (web page)
Orchestration model
On Road Assistance scenario

activity orchestration2

<<send&receive>>
start

<<send&receive>>
chargeCredit
chargeCreditPL
cardInfo
transactionMsg

<<send&receive>>
getPosition
getPositionPL
userId
location

findGaragePL
gPosition
garageList

<<send&receive>>
findGarage

findRentalStationPL
carPosition
rentalCarList

selectBestGaragePL
garageListInfo
garageId

<<send&receive>>
selectBestGarage

selectBestRentalCarPL
rentalCarListInfo
rentalCarId

<<send&receive>>
selectBestRentalCar
2. Defining tool chain in SDE

- Converter UML4SOA to BPEL/WSDL
  - transformation from UML2 models to an Intermediate Orchestration Model (IOM)
  - transformation from IOM to BPEL/WSDL*

- Converter BPEL/WSDL to active BPEL/WSDL
  - transformation of BPEL/WSDL* to code executable by ActiveBPEL Engine 4.0 (open source)
    - Replacement of namespace and service location within BPEL/WSDL
    - Create process deployment description files (catalog.xml, *.pdd)

- Transformation active BPEL to interactive BPEL
  - transformation for adding user interaction mechanisms
    - additional receive & reply for each invoke for communication between user and BPEL process
    - extension of reply with a list of next actions

- Deployment on a web server (Tomcat)
Tool chain in SDE
Graphical orchestration of tools (Eclipse plug-ins)
3. Executing tool chain
Looking at transformation results

BPEL models
Start Service

Indicate car position
- current car position
- car address:
  - street: 
  - number: 
  - city: 
  - zip: 
  - country: 

Select service criteria
- open 24 hours
- nearest

set request parameter for Dino
price level: Inw
Payment Service

Please enter your credit card information:

- Name
- Credit Card: MasterCard
- Valid Until
- Card Number
- Security Number

Submit
Car Location

Next step
- search rental car station nearby
- search garage nearby

Current Location
Garage nearby your car
rental car station nearby your car

Next step
search best garage
search best rental car station

Garage nearby your car
Garage Denninger  get route
Garage Neckar  get route
Garage Riedenburger  get route
Garage Zaubzer  get route

rental car station nearby your car
Car Rental Gotthelf  get route
Car Rental Steinhauser  get route
Car Rental Eva  get route
Car Rental Ina  get route
5. Changing the orchestration model

```
activity orchestration1

<<send&receive>> start

<<send&receive>> getPosition

<<send&receive>> findGarage

<<send&receive>> selectBestGarage

<<send&receive>> findRentalStation

<<send&receive>> selectBestRentalCar

<<send&receive>> chargeCredit
```

```
activity orchestration2

<<send&receive>> start

<<send&receive>> getPosition

<<send&receive>> findGarage

<<send&receive>> selectBestGarage

<<send&receive>> findRentalStation

<<send&receive>> selectBestRentalCar

<<send&receive>> chargeCredit
```
Back to the tool chain (step 3)
Looking at transformation results
BPEL models

Seville – 2010 – Nora Koch ©
Sensoria
On Road Assistance

Car Location

Next step
- search rental car station nearby
- search garage nearby

Current Location
Rental car station nearby your car

Next step
search best rental car station

Current Location

Rental car station nearby your car
AutoHertz August get route
AutoHertz Theresien get route
AutoHertz Briener get route
AutoHertz Karl get route
Sensoria
On Road Assistance

The best rental car station

Next step
search garage nearby

Current Location

The best rental car station
AutoHertz August
get route
Garage nearby your car

Next step

search best garage

Continue

Garage Deutsch Herzenheim München
Garage Denninger
Garage Neckar
Garage Zaubzer
Garage Riedenburger

The best rental car station

AutoHertz August

Get route
Sensoria
On Road Assistance

The best Garage

Next step
start new service

Current Location

The best garage
Garage Denninger

The best rental car station
AutoHertz August
Sensoria
On Road Assistance

Payment Service

Please enter your credit card information:

Name
Credit Card
Valid Until
Card Number
Security Number

submit

MasterCard
Bottom line: Ideas to take home

- Relevance of domain specific modelling language: SoaML+
  - UML profile
  - must be simple, few constructs

- Automated development approach: MDD4SOA
  - model-based
  - model-driven (transformations)
  - pattern-based

- Importance of flexible tool support: SDE
  - easy (graphically) integration of diverse tools
Thank you very much for your attention!

SENSORIA web site
www.sensoria-ist.eu

Nora Koch
kochn@pst.ifi.lmu.de