Modelling Service-Oriented Systems with UML4SOA

Nora Koch
LMU München and Cirquent GmbH
Germany
Aim of the talk

to provide you with an overview of the modelling approach used within the SENSORIA project for the development of service-oriented systems
Plan of the talk

I. Setting the scene
   - the context – SENSORIA development process
   - what we mean by “models” and “model-driven development”
   - what we mean by “service-oriented systems”

II. Modelling service-oriented systems
   - domain-specific modelling languages
   - UML profiles
   - modelling SOAs by example
   - metamodelling
   - SOA models in the MDA context
I. Setting the Scene
SENSORIA development approach

- SENSORIA approach to model-driven service engineering
  - from business models to implementations
  - via model transformations
- Formal analysis
  - functional service verification
  - type correctness
  - sensitivity analysis
  - scalability analysis
- Flexible service development support
  - service development patterns
  - development environment
Modelling front-end
Service-oriented applications are designed using high-level visual formalisms such as the industry standard UML or domain-specific modelling languages.

Hidden formal analysis of services
Back-end mathematical model analysis is used to reveal performance bottlenecks, or interactions leading to errors or violation of service contracts.

Automated model transformations
Formal representations are generated by automated model transformations from engineering models.

Service deployment
As a result, service models of proven quality serve as the basis for deployment transformations to generate configurations for standards-compliant platforms.
Graphical representation of the development process

**Analysis**
- Verifying properties of SOA models

**Improvement**
- Preparing results for improving models

**Transformation**
- Translating models to formal languages

**Business Modelling**
- SOA Architecture
  - e.g. requirements

**Design**
- Modelling SOA applications

**Code Generation**
- Creating executable code, e.g. BEPL/WSDL

**SENSORIA Development Environment**

SENSUS 2009, Nora Koch
What is a model?

A description or specification of the system and its environment for some certain purpose. A model is often presented as a combination of drawings and text. [MDA Guide, 2003]
Characteristics of models

- **Abstract**
  - emphasize important aspects, hide irrelevant ones

- **Understandable**
  - expressed in a form readily understood by users

- **Accurate**
  - faithfully represent the modelled system

- **Predictive**
  - can be used to derive correct conclusions about the system

- **Inexpensive**
  - cheaper to construct and study than the system

*Selic, IEEE, 2003*
Examples

- City models
  - underground and bus maps, commuting models, ...
- Building/house maps
  - floor plans, electric wiring, water and central heating pipes, etc.
- Scientific models
  - mathematical models
  - statistical models
  - simulation models
- Software system models
  - requirements (use cases)
  - structure (class diagrams)
  - behavioural models
  - choreography models
  - load balancing and deployment models, ...

- MDA proposed “everything is a model”
  - a process is a model
  - a platform is a model
  - a transformation is a model
  - a metamodel is a model
  - a system is a model
  - a program is a model
  - a measure is a model
  - a test is a model
  - a pattern is a model
  - ...

SENSUS 2009, Nora Koch
Usefulness of models

- Specify the system
  - structure, behaviour, ...
  - separate concepts at different conceptual levels
  - communicate with stakeholders
- Understand the system
  - if existing (legacy applications)
- Validate the system
  - detect errors and omissions in design ASAP
    - mistakes are cheaper at this stage
  - prototype the system *(execution of the model)*
  - formal analysis of system properties
- Drive implementation
  - code skeleton and templates
  - complete programs (if possible)

Vallecillo, ICWE 2004
Models in the “model-driven development” context

- Model-Driven Development/Engineering (MDD™/MDE)*
  - refers to a range of engineering approaches that are based on the use of software models as a primary form of expression
  - has a focus on architecture and corresponding automation
  - objective is to generate code from the models

- Model-Based Development
  - expresses instead that models are mainly used for communication and documentation

- Model-Driven Architecture (MDA™)*
  - best known MDE initiative

* Note that MDA, MDD are trademarks of the OMG; MDE is not
MDA terminology

- Computational Independent Model (CIM)
  - describes the business context and business requirements
  - focuses on the environment of the system

- Platform Independent Model (PIM)
  - specifies structure and functionality of the software system independent of software technology platforms
  - suitable for use with a number of different platforms

- Platform Specific Model (PSM)
  - describes the realisation of the software systems with respect to the chosen software technology platforms
MDA in a nutshell

- MDA supports the idea of
  - designing software systems using model(s) in the development
    - CIM, PIM, PSM
  - transforming CIMs to PIMs, PIMs to PIMs and PIMs to PSMs
    - based on model transformation technologies
  - models are first class entities

- MDA promotes to build different views (models) of a system following a separation of concerns

- MDA/MDE is changing the software development paradigm from code-centric to model-centric
MDE requirements

- 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…
Services, SOS, SOC and SOA

- **Service**
  - autonomous, platform-independent computational entity that can be described, published, categorised, discovered
  - services can be consumed without having to care about their maintenance, destruction, etc. (difference to components)

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

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

- **Service-Oriented Architecture (SOA)**
  - architectural style to realize SOC
Stakeholders/Parties in service-oriented architectures

- **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
  - bind consumer and provider
  - offered as middleware in SOAs

- **SOA triangle**
II. Modelling service-oriented systems
Modelling languages

- Need of a language to describe models
  - textual language and/or
  - graphical intuitive representation with symbols and icons
- Option 1: Use of standard modelling languages
  - Unified Modeling Language (UML™) or Business Process Modeling Notation (BPMN™)
  - no specific concepts for SOC
- Option 2: Domain-specific modelling languages (DSML)
  - make models more intuitive
  - reduce modelling efforts due to replacement of a set of elements by one language construct
Option 2: Domain-specific modelling languages for SOAs

- Objective is to have a domain-specific graphical representation and clear semantics for service-oriented concepts
  - Option 2.1: Definition of a proprietary language, like SENSORIA Reference Modelling Language (SRML)
    - free definition of syntax and semantics
    - high cost: requires the definition of all required concepts and proprietary tools
  - Option 2.2: Define a UML2 profile
    - using the extension mechanism that allows to customise 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
    - limited to UML guideline
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...)
UML extensions for SOA modelling

- **SoaML profile** (OMG standardization process beta1 version)
  - 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
UML extensions: 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
  - not a prescriptive approach
Service Oriented Architecture Modeling Language

- Answer to Request of Proposal of the OMG
  - for a *UML Profile and Metamodel for Services* (UPMS), Sept. 2006
- Submission and supporters
  - SINTEF, Norway (co-ordination), European Software Institute (ESI)
  - Capgemini, Fujitsu, Hewlett-Packard, IBM, Telelogic AB, Thales Group, France Telecom R&D, etc
  - University of Insbruck, University of Augsburg, University of Athen
  - SHAPE project (FP7) is the main contributor
- Results
  - Merge of approaches, June 2008
  - 1\textsuperscript{st} revised submission, August 2008
  - 2\textsuperscript{nd} revised submission, November 2008
UML4SOA

- Set of UML profiles developed within the scope of SENSORIA
  Contributors
  - Ludwig-Maximilians Universität München (LMU), Germany
  - Budapesti Műszaki és Gazdaságtudományi Egyetem (BME), Hungary
  - Imperial College London, UK
  - Università di Pisa, Italy
  - University of Leicester, UK

- Results
  - deliverables
    - D1.4a - UML for Service-Oriented Systems, August 2007
    - D8.2a - D8.2.a: Requirements Modelling and Analysis of Selected Scenarios (Automotive Case Study), August 2007
  - publications
    - MDD4SOA: Model-Driven Service Orchestration
      P. Mayer, A. Schroeder, N. Koch. (EDOC 2008)

...
Profiles for modelling different aspects of SOSs: structural aspects (SoaML), behavioural aspects (Orchestration), non-functional properties (NFP), reconfiguration (Modes), policies (StPowla), requirements (SRML), and performance (MARTE).
SOA models in the MDA context

- **Computation Independent Model (CIM)**
  - Business Model
    - Enterprise Services
      - Roles, Collaborations, Dependencies, Workflows
  - Design Model
    - Services
      - Components, Interfaces, Messages, Data
  - Technical Specification
    - Technical Services
      - WSDL, BPEL, XML Schema, Java, Jolie

Source: Data Access Technologies, Inc
SOA modelling by example

- 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.
Process as orchestration of services

- UML activity diagram selected for the representation of orchestration of services

A **scope** is used to group service specific actions and scopes. It may have associated event, exception and compensation handlers. A scope can be represented as a UML StructuredActivityNode or an Activity.
Specifying service capabilities

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

**SoaML**

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 service contract specifies the service without regards for realisation 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 services 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.
It is important not to over-specify any of the parties, usually it is not required to specify the internal structure of a participant allowing each party maximum freedom in how they achieve their goals.

However, it is possible to provide a high-level services architecture of a participant.

Define how a set of internal and external participants use services to implement the responsibilities of the participant.

A participant architecture represents the architecture of a specific participant. It is represented as a UML class or component.
SOA models in the MDA context

**Computation Independent Model (CIM)**

**Platform Independent Model (PIM)**

**Platform Specific Model (PSM)**

**Business Model**

*Enterprise Services*
*Roles, Collaborations, Dependencies, Workflows*

**Design Model**

*Services*
*Components, Interfaces, Messages, Data*

**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 interface
  - “implements” (“provides”) provider interfaces (represented as realisation)
  - “requires” consumer interfaces (represented as a «use» dependency)
A **service point** provides the provider interface and requires the consumer interface.

A **request point** requires the provider interface and provides the consumer interface (reverse of service point).

**Need of a change in UML**
Change in UML Metamodel

Required by SoaML

- Port is modified to indicate the direction of a Port, whether
  - the Port is providing the operations available through a Port or
  - the Port is consuming them
- Current situation in the UML
  - conjugate types must be created explicitly

![UML Diagram]

SoaML
A **service interface** implements (“provides”) provider interfaces (represented as realisation).

A **service interface** “requires” consumer interfaces (represented as a «use» dependency).

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

- **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 among requester and provider
Synchronous and asynchronous service invocation

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.

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

Service interaction **receive** blocks until message is received.

A **raise** action causes normal execution flow to stop and invokes associated exception handlers.
Detailing service invocation
Partner services and data handling

Pins containing interaction information

**link:** partner

**snd, rcv:** data to be send or received

**Implicit declaration of variable in a rcv pin.**

**Use of variable after declaration**

Variables belong to the scope they are declared in

Use of “::” for referring of variables of parent scopes

Accept activity is part of the scope Decision
Data handling

- 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 scope modelling the compensation handler will be triggered by a compensate or compensateAll.
A **compensateAll** triggers all active compensation handlers.
### 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</td>
<td>service point</td>
</tr>
<tr>
<td></td>
<td>participants</td>
<td>request point</td>
</tr>
<tr>
<td></td>
<td>service contract</td>
<td>service interface</td>
</tr>
<tr>
<td></td>
<td>service architecture</td>
<td>message type</td>
</tr>
<tr>
<td></td>
<td>participant architecture</td>
<td></td>
</tr>
<tr>
<td><strong>Behavioural aspects</strong></td>
<td>scope</td>
<td>send, receive, send&amp;receive reply, raise, pick, wait</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ink, snd, rcv</td>
</tr>
<tr>
<td></td>
<td></td>
<td>compensate, compensateAll</td>
</tr>
<tr>
<td></td>
<td></td>
<td>compensation, exception, event data</td>
</tr>
<tr>
<td><strong>Diagram type</strong></td>
<td>class diagram</td>
<td>class diagram</td>
</tr>
<tr>
<td></td>
<td>composite structure diagram</td>
<td>composite structure diagram</td>
</tr>
<tr>
<td></td>
<td>activity diagram</td>
<td>activity diagram</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sequence diagram</td>
</tr>
<tr>
<td></td>
<td></td>
<td>state machine</td>
</tr>
</tbody>
</table>
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
Defining a modelling language

- Models are specified using a modelling language ($M^1$)
- A modelling language is described by a metamodel ($M^2$)
- Metamodels belong to a library of domain-specific languages (DSLs)
- Metametamodel: there is a unique language for describing these metamodels ($M^3$), i.e. the Meta Object Facility (MOF)
What is meant by “metamodel”

- A model that defines the language for expressing a model [MOF, 2000]
- A *meta-metamodel* is a model that defines the language for expressing a metamodel, e.g. Meta Object Facility (OMG).
Four-layers metamodel hierarchy

- **Metametamodel**
- **Metamodel**
- **Model**
- **System**

The real world is represented by a model, which conforms to a metamodel, which in turn conforms to a metametamodel. The metametamodel conforms to a metamodel, which conforms to a metamodel, which in turn conforms to a model.

The diagram shows layers as follows:

- **M⁰**: a particular use of a UML model
- **M¹**: another use of a UML model
- **M²**: SPEM
- **M³**: MOF

These layers correspond to different modeling languages and standards:

- **UML**: a UML model
- **CWM**: another UML model
- **SPEM**

The modeling world is represented by a system, which is a particular use of a UML model.
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 (→)
Types of UML extensions

- Decorative
  - vary only the concrete syntax (visual presentation)
  - few value
- Descriptive
  - extend the syntax of a language such that additional information can be expressed
  - limited power as purely syntactical
- Restrictive
  - descriptive and impose semantic restrictions
  - has the capability to define a meta language on top of the base language
- Redefined
  - modify the core semantics of the language elements
  - no need of a base language

Glinz et al., UML 1999
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 Package
Conservative extension of the UML
SoaML metamodel
Profile metamodel mapping (excerpt)
Extension model (excerpt)
Specification of „new“ elements

- **Service Interface (excerpt)**
  - **Description**
    - defines the interface to a Service Point or Request Point and is the type of a role in a service contract….
  - **Extended Metaclass**
    - Class
  - **Attributes**
    - no new attributes
  - **Associations**
    - no new associations
  - **Constraints**
    1. A *Service Interface* must not define the methods for any its provided operations or signals….
  - **Semantics**
    - A *Service Interface* defines a semantic interface to a Service or Request. That is, it defines both the structural and behavioural semantics of the service necessary for consumers to determine if a service typed by a *Service Interface* meets their needs, and for consumers and providers to determine what to do to carry out the service…

- **Notation**
- **Examples**
- **Additions to UML2**
Model transformations

- Goal is automatic translation between source and target models
- Translation performed by a transformation engine that executes transformation rules
- Set of rules
  - seen as a model
  - based on a transformation metamodel
- MDA model transformations
  - CIM2PIM
  - PIM2PIM
  - PIM2PSM

Model transformation pattern (J. Bézivin, 2004)
CASE tool support for UML4SOA

- UML CASE tool (any)
  - profile import
  - definition of stereotypes and tagged values
Conclusions

- **Service Engineering Approach**
  - models in the development process of SOAs
  - domain-specific modelling language for specifying SOSs
  - metamodels for the UML extension
  - UML profiles for SOC

Do you think that “everything is a model”?
Bottom line: Ideas to take home

- Relevance of domain-specific modelling language
  - power of a UML profile
  - but must be simple and contain few constructs
- Systematic definition of a DSML
  - defining a metamodel
- Model transformations are based on
  - metamodels
  - marking of models, e.g. UML profile
Publications

- Standards
  - OMG, [www.omg.org](http://www.omg.org)

- Projects
  - SENSORIA project, [www.sensoria-ist.eu](http://www.sensoria-ist.eu)
  - SHAPE project (SoaML), [www.shape-project.eu](http://www.shape-project.eu)
Thank you for your attention!

Questions?

Nora Koch
kochn@pst.ifi.lmu.de