ICWE 2009
San Sebastian, Spain
June 23, 2009

Model-Driven Development of Service-Oriented Systems

Nora Koch
LMU München and Cirquent GmbH

in co-operation with Martin Wirsing,
Philip Mayer, Matthias Hölzl, Rong Xie
and many other SENSORIA members

Aim of the tutorial

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

Plan of the tutorial

I. Setting the scene

- the context – SENSORIA
- what we mean by “service-oriented systems”
- what we mean by “model-driven development”

II. Engineering of service-oriented systems

- development process
- modelling
- metamodel and model transformations
- tool support
- model-driven development @ work
- pattern language

I. Setting the Scene

Context

Software Engineering for
Service-Oriented Overlay Computers

EU project

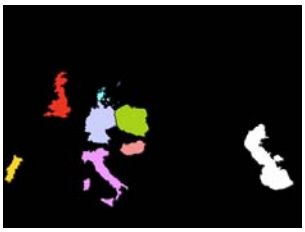
19 partners

2005 - 2010

more than 500 publications

2 spin offs

… more details

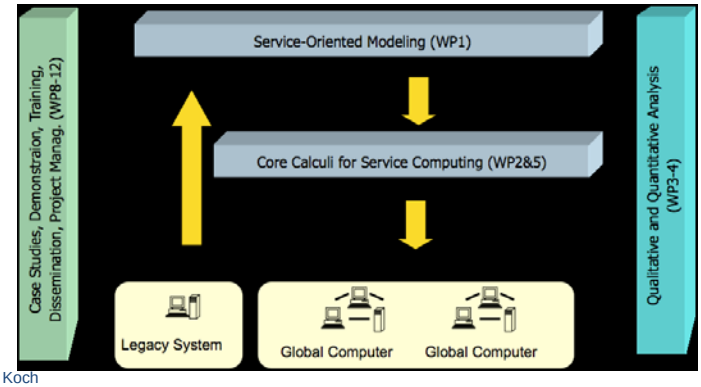


- 6th Framework (6FP)
- Information Society Technologies (IST)
- Global Computing (GC2)
- Future and Emerging Technologies (FET)

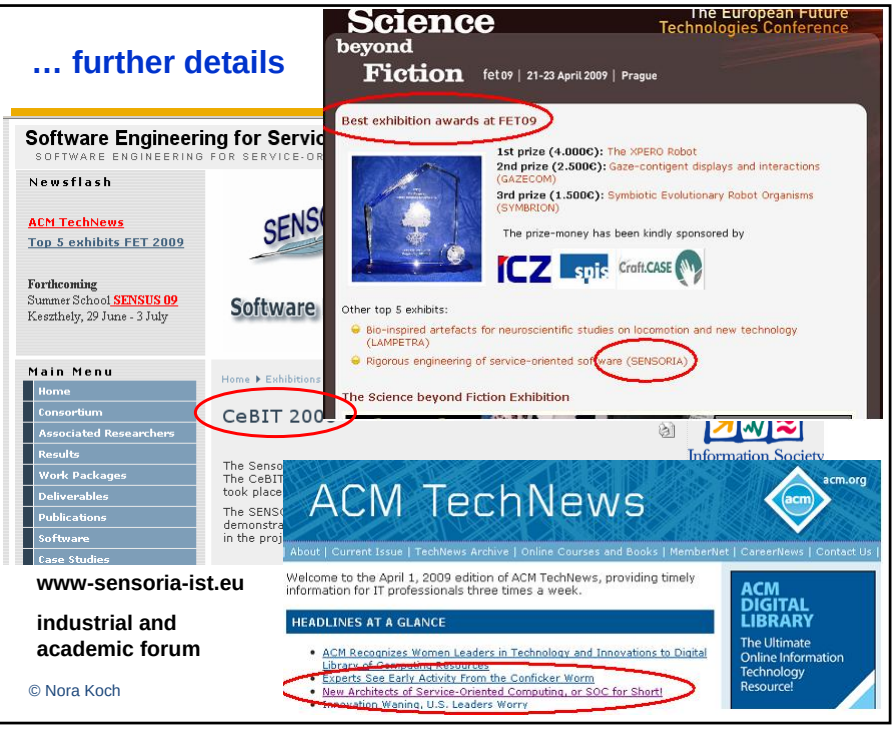
- LMU Munich (Coordination)
- Universitá di Trento
- University of Leicester
- Warsaw University
- Technical University of Denmark at Lingby
- Universitá di Pisa
- Universitá di Firenze
- Universitá di Bologna
- Istituto di Scienza e Tecnologie della Informazione
- University of Lisbon
- University of Edinburgh
- ATX Software SA
- Telecom Italia S.p.A.
- Imperial College London
- University College London
- Cirquent GmbH
- Budapest University of Technology and Economics
- S&N AG
- School of Management of Politecnico di Milano

Project overview

- Rigorous approach to engineering service-oriented systems integrating
  - foundational theories, techniques, and methods
  - pragmatic software engineering



… further details



www-sensoria-ist.eu

industrial and academic forum

What we mean by “service-oriented systems”

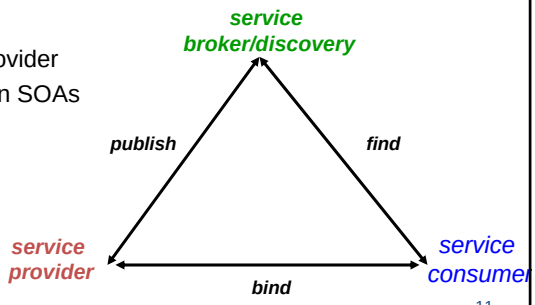
- Service
  - autonomous, platform-independent computational entity that can be described, published, categorised, discovered
  - services can be consumed without having to care about their maintenance, destruction, etc. (difference to components)
    - like gas, power, telephone, etc.
- Service-Oriented Systems (SOS)
  - use loosely coupled services
  - massively distributed, interoperable, evolvable applications
  - consist of providing, consuming and publishing services, i.e. establishing a community or marketplace
    - like applications spread over the web, e.g. online banking, hotel reservation, flight booking, etc.

… more terminology

- 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
    - “distributed computing” is such another paradigm
- Service-Oriented Architecture (SOA)
  - an architectural style to realize SOC
    - “client/server” is an architectural style for realizing distributed computing

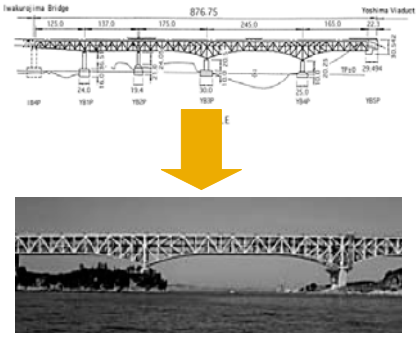
Stakeholders/Parties in service-oriented architectures

- 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
  - binds consumer and provider
  - offered as middleware in SOAs
- SOA triangle



What is a "model"

- A *description* of (part of) a system written in a *well-defined* language (Equivalent to *specification*) [Kleppe, 2003]
- 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]



Vallecillo, ICWE 2004

Examples

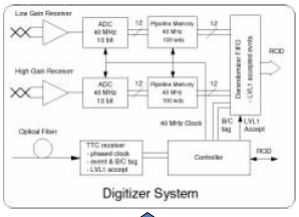
- City models
  - underground and bus maps, commuting models,...
- Building/house maps
  - floor plans, electric wiring, water and central heating pipes, etc.
- Scientific models
  - mathematical models
  - statistical models
  - simulation models
- Software system models
  - requirements (use cases)
  - 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
  - …

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



PCB and PCB model

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)

What is meant by "model-driven development"

- 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
  - instead expresses that models are mainly used for communication and documentation
- Model-Driven Architecture (MDA™)*
  - is the best known MDE initiative

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

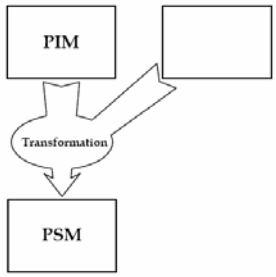
MDA terminology

- 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 realization of the software systems with respect to the chosen software technology platforms

OMG MDA Guide,2004

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



More about MDD,MDE, MDA, ... later

II. Engineering of Service-Oriented Systems

Motivation

- Service-oriented architectures (SOAs)
  - promise to organize and understand organizations, communities and systems maximizing agility, scalability and interoperability
  - built by IT industry in an ad-hoc and undisciplined way
- Challenges for service-oriented computing (SOC)
  - specification of correct behaviour of SOAs
  - automated composition of services (orchestration)
  - long running transactions
  - performance, security and safety
  - deployment and re-engineering

MDE in SENSORIA

- 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

… more details

- Modelling front-end

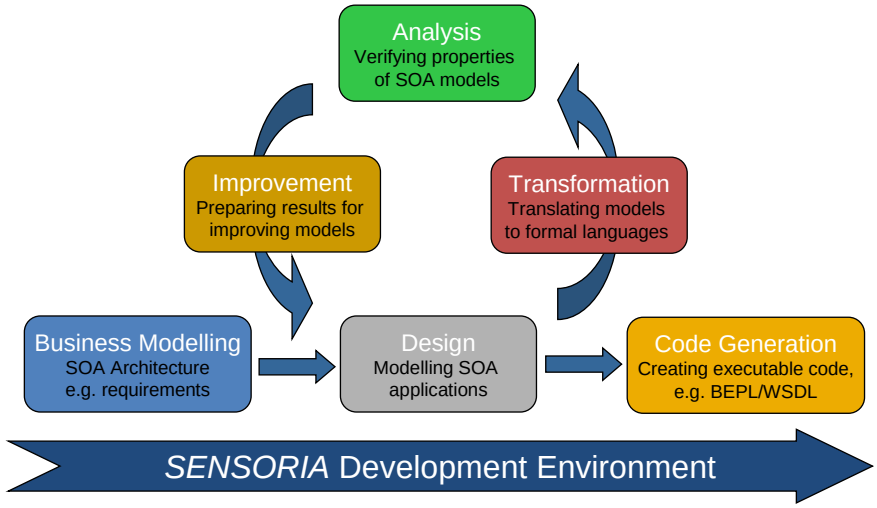
  Service-oriented applications are designed using high-level visual formalisms such as the industry standard UML or domain-specific modelling languages.
- Hidden formal analysis of services

  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.

“Model” of the model-driven development process



Project results

- Languages
- Techniques
- Methods
- Tools

to support this development process of service-oriented systems

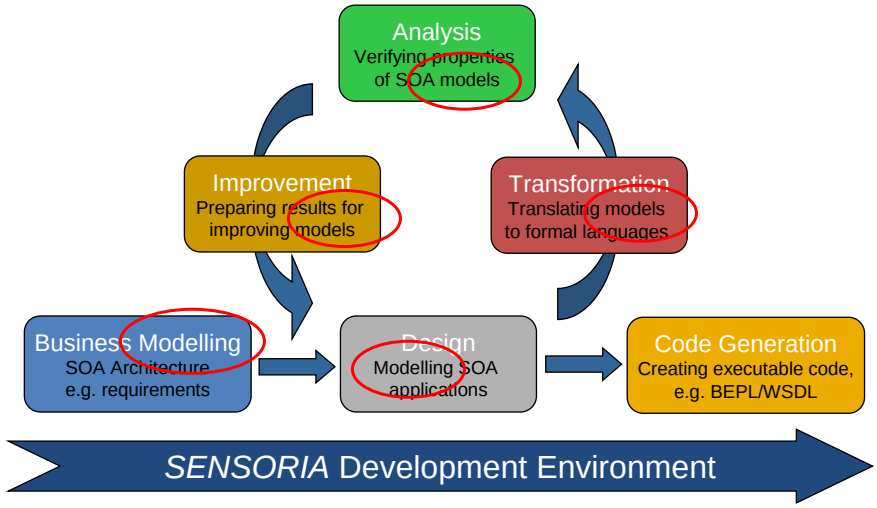
... concrete results

- Service ontology
- Modelling languages
  - UML4SOA, SRML, StPowla
- Process calculi
  - SCC, SOCK, Stock, COWS, …
- Languages for programming service-oriented systems
  - Jolie
- Transformation tools supporting MDE process
  - SRML Use Case Wizard
    - UseCases2SRML
  - MDD4SOA
    - UML2BPEL/WSDL, UML2Jolie, UML2Java
    - BPEL/WSDL transformers (ActiveBPEL, Tomcat)
  - VIATRA
    - SOA2WSDL, UML2Axis

... concrete results (continued)

- Languages, tools and techniques for qualitative and quantitative analysis
  - StockKlaim, MoSL, PEPA, WS-Engineer, CMC/UMC, Lysa
- Service broker
  - Dino
- Re-engineering tool
  - CoreStudio
- CASE tool
  - SRML modelling environment
- Tool suite
  - SENSORIA Development Environment (SDE)

Model-driven development process

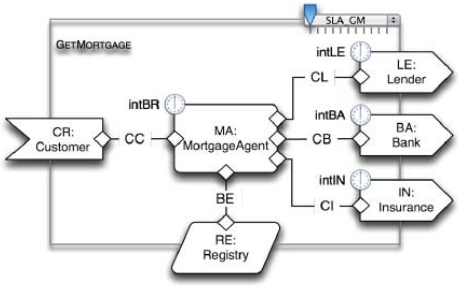


Modelling languages

- 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

Option 1: SENSORIA Reference Modelling Language (SRML)

- Modelling language with a formal semantics
- Offers descriptions of business logic based on conversational interactions
- Inspired by SCA (standards proposed by IBM, BEA, Oracle, SAP, Siebel,…)
- Proprietary language needs proprietary CASE tool




```
INTERACTIONS
r&s getProposal
   idData:usrdata,
   income: moneyvalue,
   preferences:prefdata
   proposal:mortgageproposal
   cost:moneyvalue
```


Option 3: UML2 profile

- **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

➲ UML4SOA

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

➲ MDD4SOA

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

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

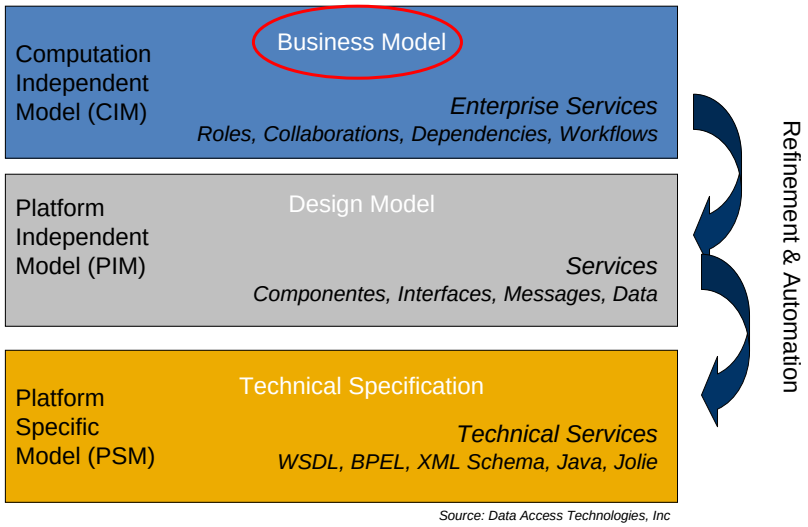
Service Oriented Architecture Modeling Language

- 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
  - Univeristy of Insbruck, University of Augsburg, University of Athen
  - SHAPE project (FP7) is the main contributor
- Results
  - Merge of approaches, June 2008
  - 1st revised submission, August 2008
  - 2nd revised submission, November 2008
- Meetings SoaML and UML4SOA groups
  - EDOC 2008, Munich, Sept. 2008
  - next, Sept. 2009

MARTE profile

- Defined for modelling of real-time and embedded systems
- Concerns also model-based analysis, i.e. provides facilities to annotate models with information required to perform specific model analysis
- Focuses on performance and schedulability analysis

SOA models in the MDA context



Source: Data Access Technologies, Inc

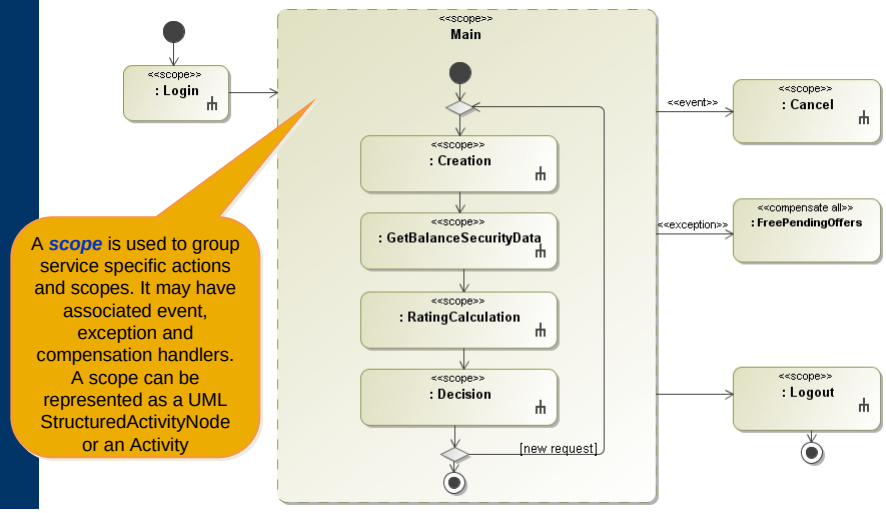
SOA modelling by example

- 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

UML4SOA

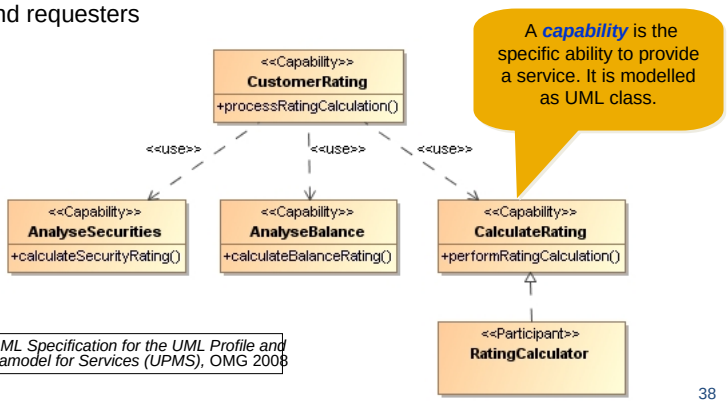
- UML activity diagram selected for the representation of orchestration of services



Specifying service capabilities

SoaML

- 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

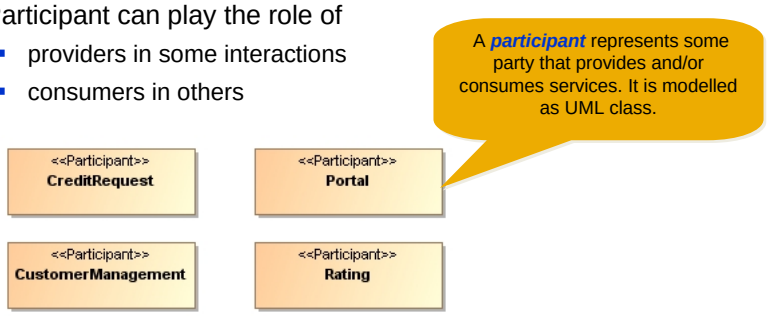


SoaML Specification for the UML Profile and Metamodel for Services (UPMS), OMG 2008

Identifying parties involved in SOAs

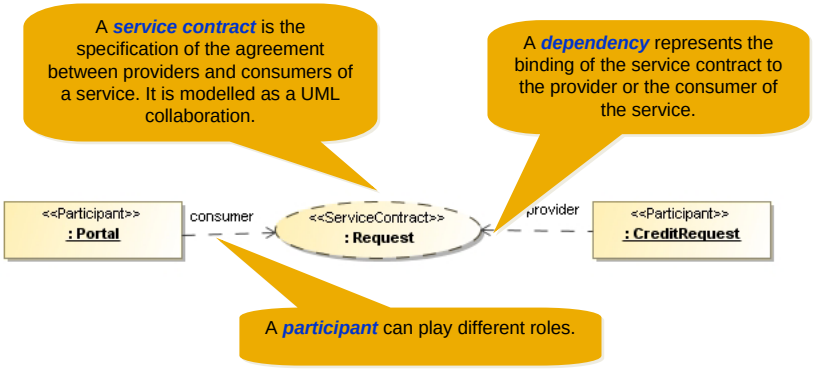
SoaML

- 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



Modelling service contracts

SoaML

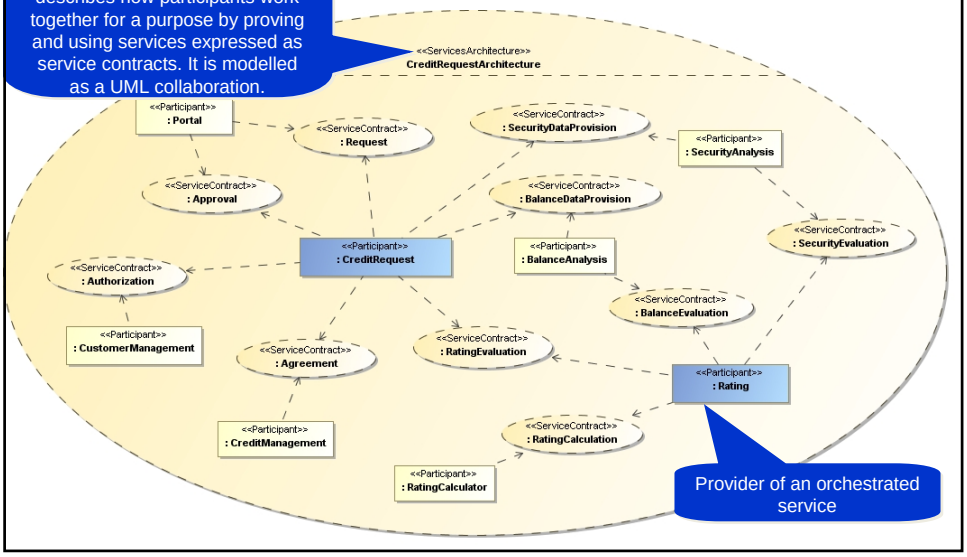


- A service contract specifies the service without regards for realization or implementation.
- A UML2 collaboration defines a set of cooperating entities to be played by instances (its roles), as well as a set of connectors that define communication paths between the participating instances.

Representing service architecture

SoaML

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.

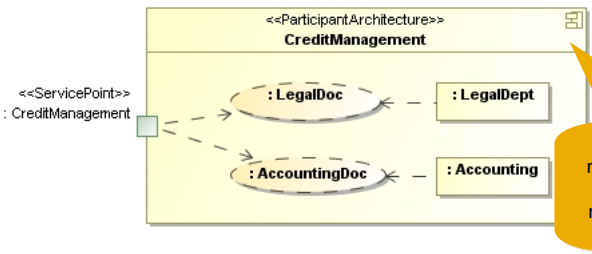


Provider of an orchestrated service

Participant architecture

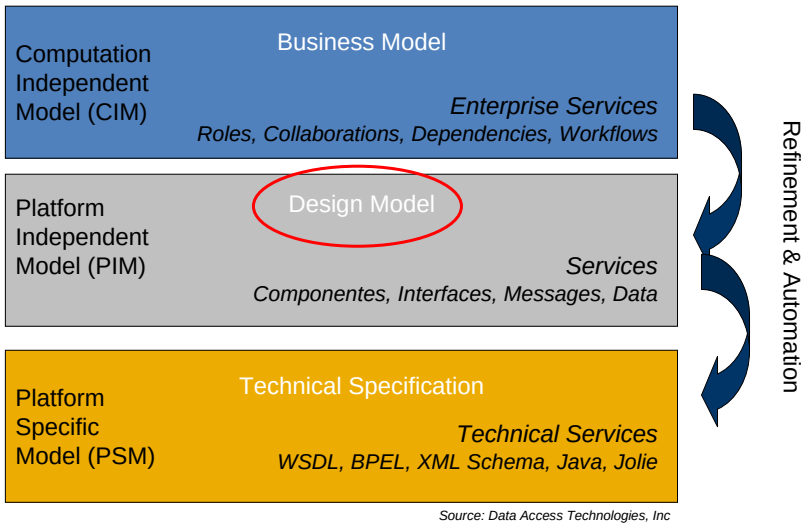
SoaML

- It is important not to over-specify any of the parties, i.e. 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
- Defines 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

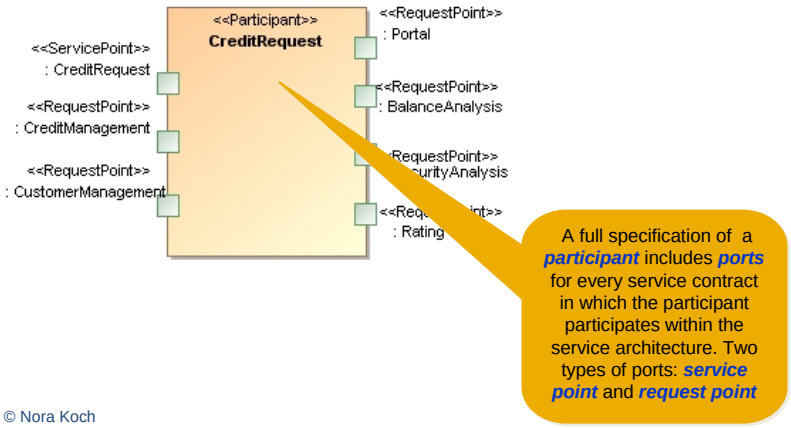


Source: Data Access Technologies, Inc

Refining specification of participants with ports

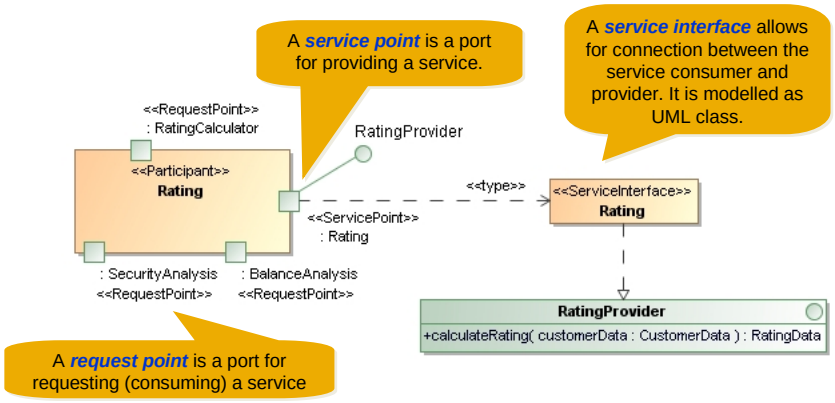
SoaML

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



Modelling service interfaces

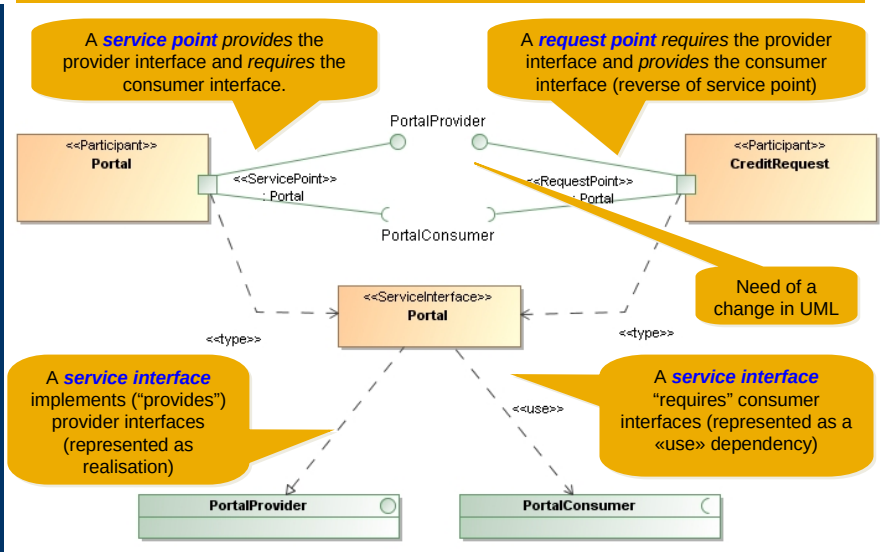
SoaML



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

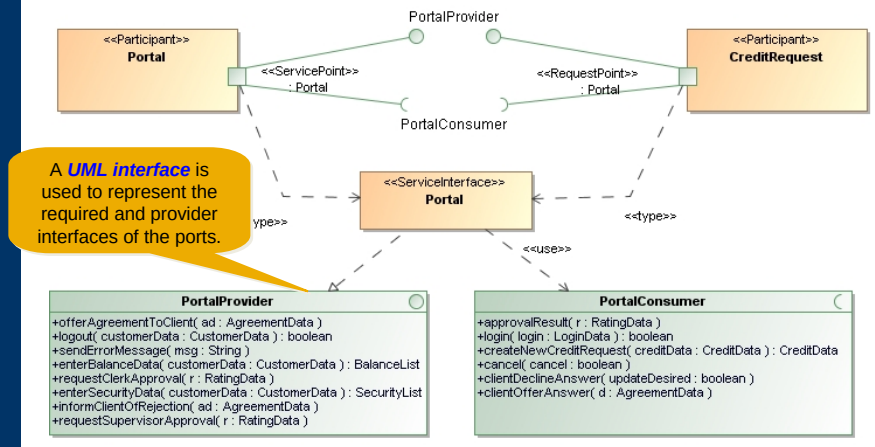
Service point and request point
Reverse interfaces

SoaML



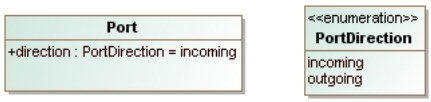
Service point and request point
Reverse Interfaces

SoaML



Change in UML Metamodel
Required by SoaML

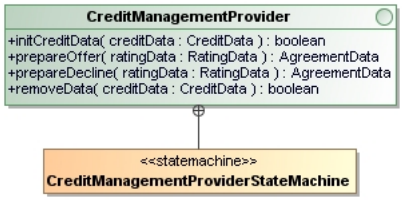
- Port is modified to indicate the direction of a Port, whether
  - the Port is providing the operations available through a Port or
  - the Port is consuming them
- Current situation in the UML
  - conjugate types must be created explicitly

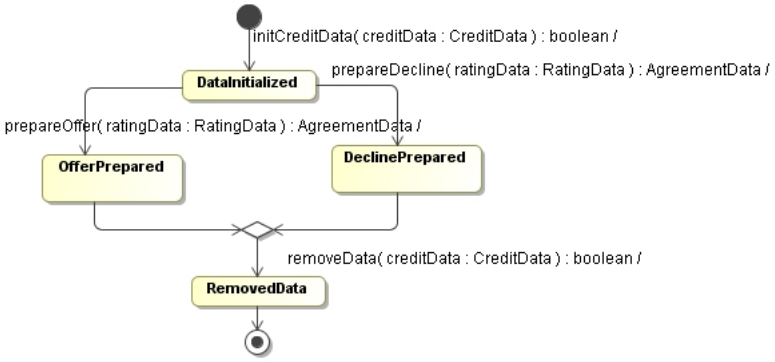


Interface behaviour

UML

- UML4SOA
  - propose protocol state machine
- SoaML
  - propose activity diagrams or sequence diagrams

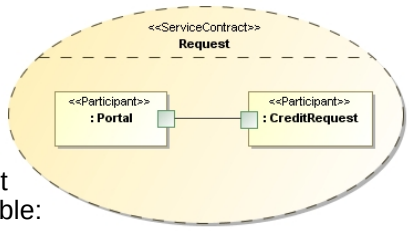




Service channel

SoaML

- Communication path between service points and request points within an architecture that
  - connects consumers and providers
  - defines the coupling in the system
  - extends UML connector

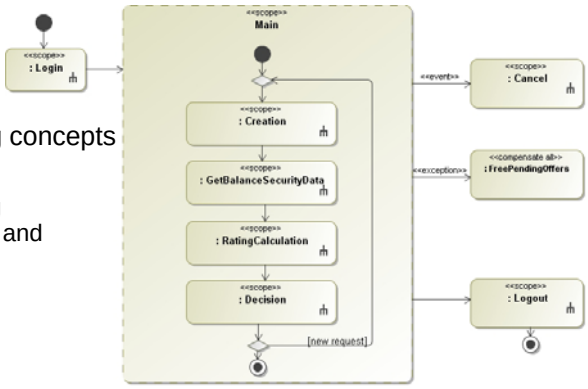


- Connection possible if request and service point are compatible:
  - both have the same port type (interface or service interface)
  - type of the service point is a specialisation or realisation of the type of the requested point
  - both have compatible needs and capabilities, i.e. the service must provide an operation for every operation used through the request and reverse

Orchestration of services

UML4SOA

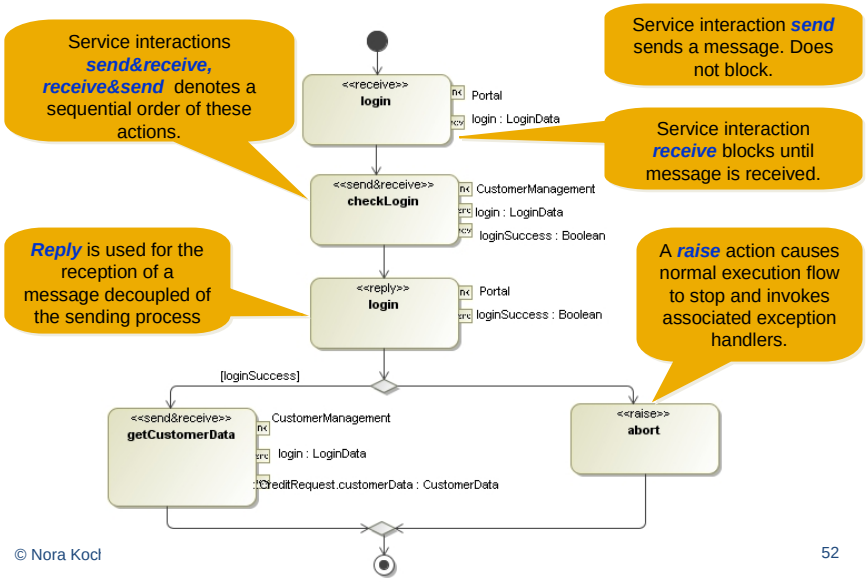
- Service orchestration is the process of combining existing services to form a new service to be used like any other service.



- Key distinguishing concepts
  - partner services
  - message passing among requester and provider
  - long-running transactions
  - compensation

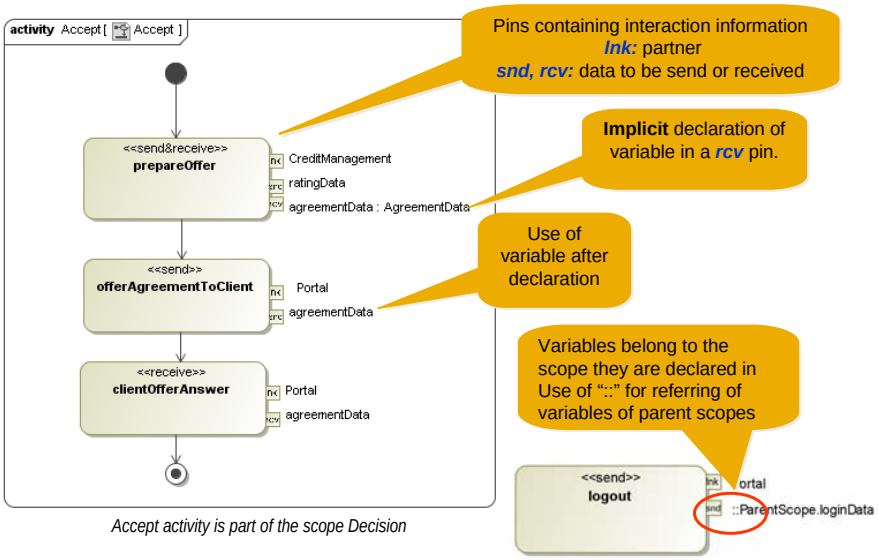
Message passing among requester and provider
Synchronous and asynchronous service invocation

UML4SOA



Detailing service invocation
Partner services and data handling

UML4SOA

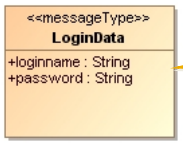


Accept activity is part of the scope Decision

Data handling

SoaML/UML4SOA

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



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

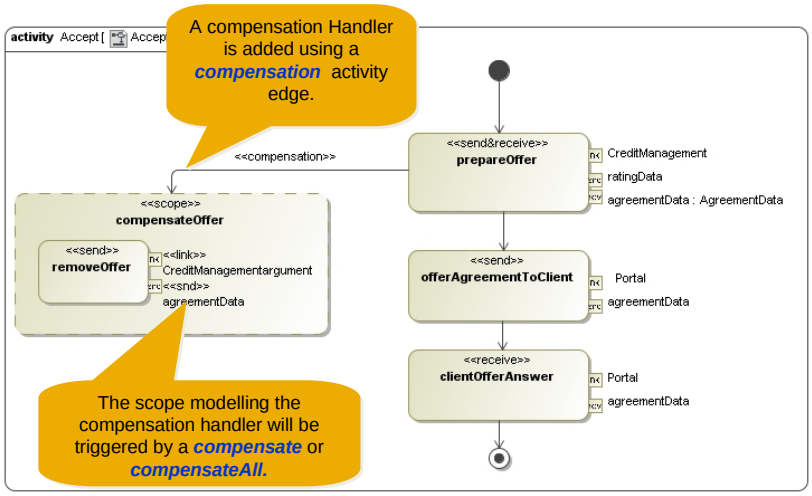
- Use in behavioural diagrams
  - support for typed, scoped variables in the orchestration
  - data handling support

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

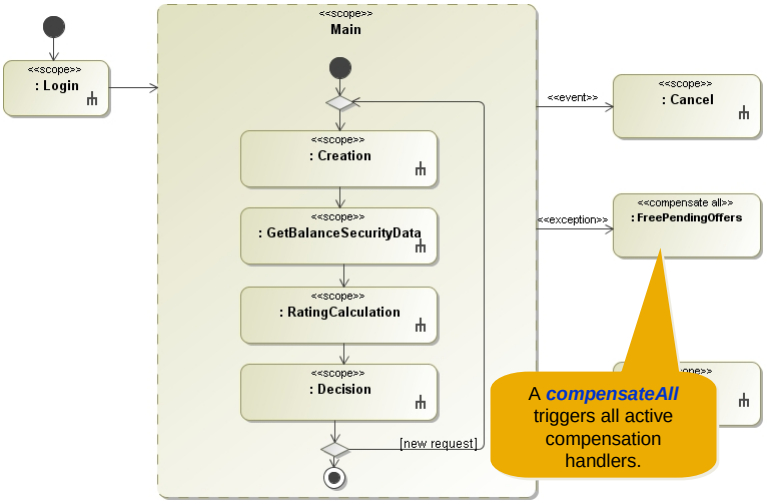
UML4SOA

- Require compensation mechanisms, e.g. compensation handlers



Compensation

UML4SOA



SOA model elements and diagram types

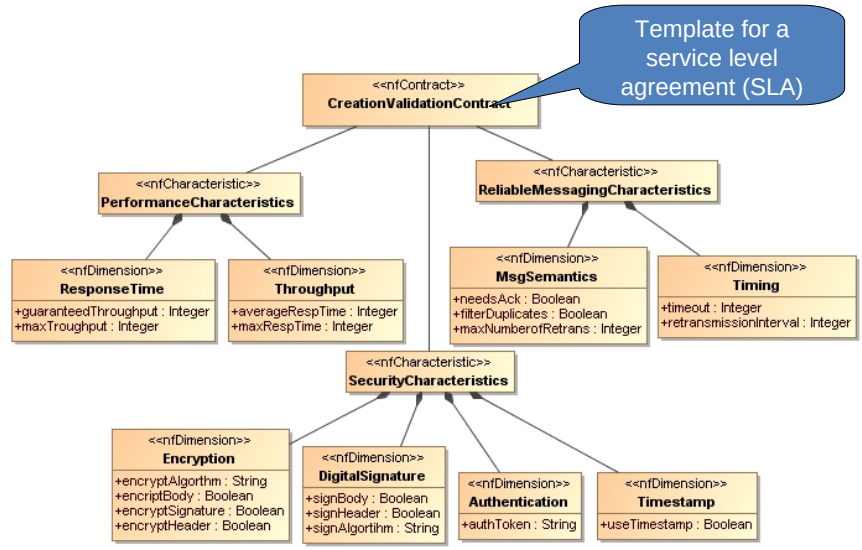
	Business model	*Design model*
Structural aspects	capabilities participants service contract service architecture participant architecture	service point request point service interface message type
Behavioural aspects	scope	send, receive, send&receive reply, raise, pick, wait lnk, snd, rcv compensate, compensateAll compensation, exception, event data
Diagram type	class diagram composite structure diagram activity diagram	class diagram composite structure diagram activity diagram sequence diagram state machine

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

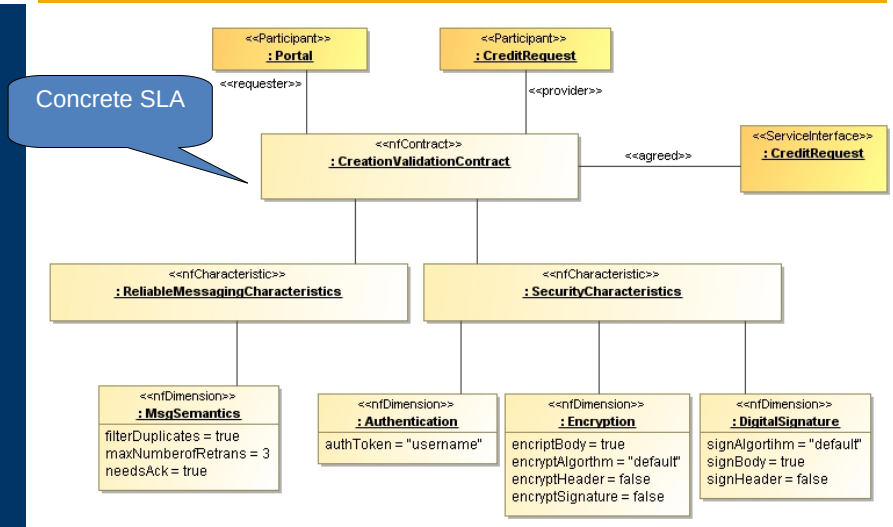
Quality of services

- 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

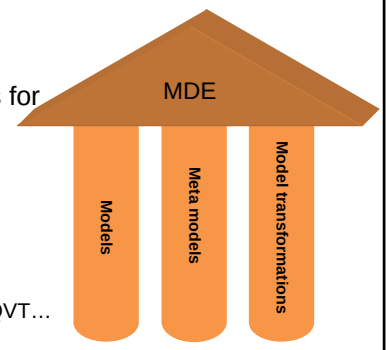


Modelling a concrete configuration



Coming back to MDE

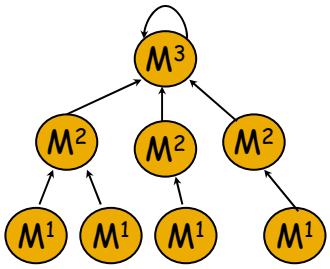
- MDE approaches
  - are based on the constructions of models
  - propose transformation of models
  - implement model transformations based on the metamodel of the modelling language
- MDE approaches require languages for
  - specification of models
    - UML, BPMN, …
  - description of metamodels
    - UML, MOF, OCL, …
  - definition of model transformations
    - Java, graph transformations, ATL, QVT…



What is meant by “metamodel”

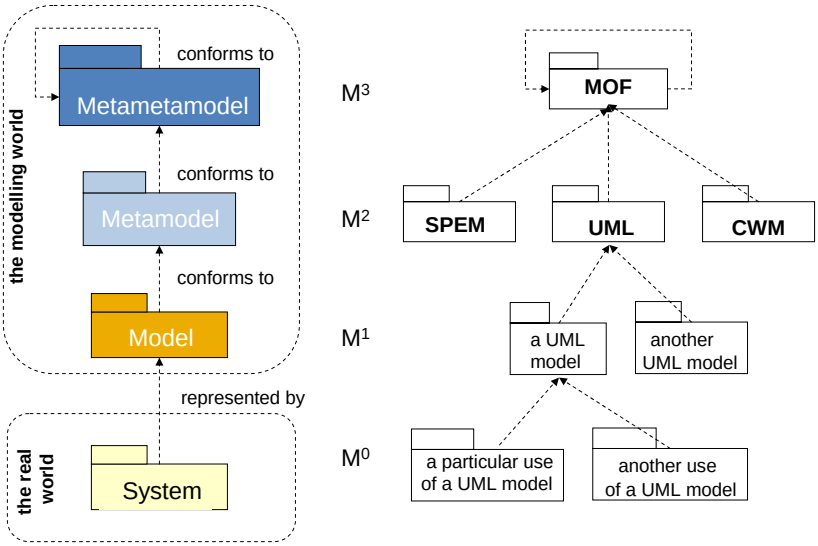
- A model of a modelling language [Seidewitz, 2003]
  - That is, a metamodel makes statements about what can be expressed in the valid models of a certain modelling language.
- A model that defines the language for expressing a model [MOF, 2000]
- A *meta-metamodel* is a model that defines the language for expressing a metamodel,e.g. Meta Object Facility (OMG). The relationship between a meta-metamodel and a metamodel is analogous to the relationship between a metamodel and a model.

MDA principles

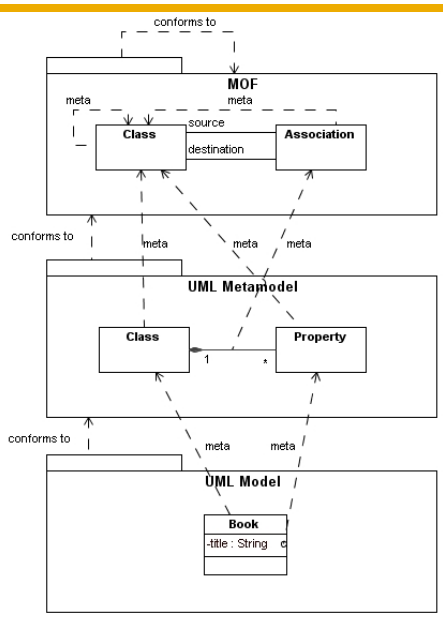


- Models are specified using a modelling language (M^1)
- A modelling language is described by a metamodel (M^2)
- Metamodels belong to a library of domain specific languages (DSLs)
- Metametamodel: there is a unique language for describing these metamodels (M^3), i.e. the Meta Object Facility (MOF)

Four-layers metamodel hierarchy

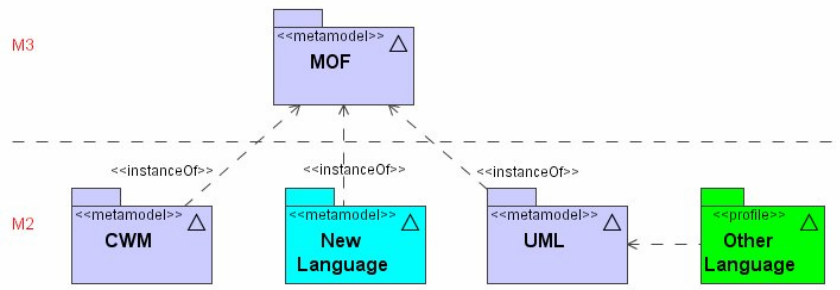


Four-layers metamodel hierarchy (example)



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

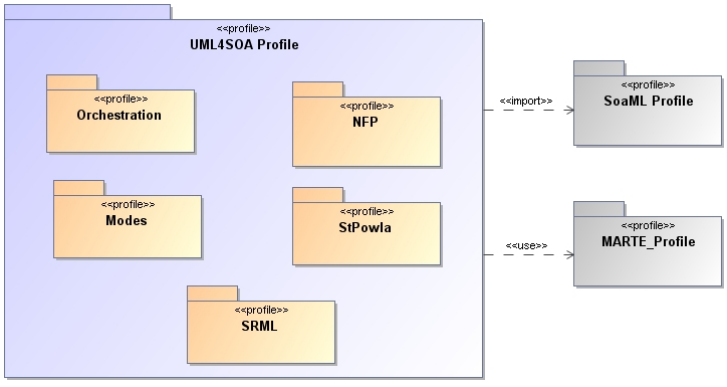
- Decorative
  - vary only the concrete syntax (visual presentation)
  - few value
- Descriptive
  - extend the syntax of a language such that additional information can be expressed
  - limited power as purely syntactical
- Restrictive
  - descriptive and impose semantic restrictions
  - has the capability 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

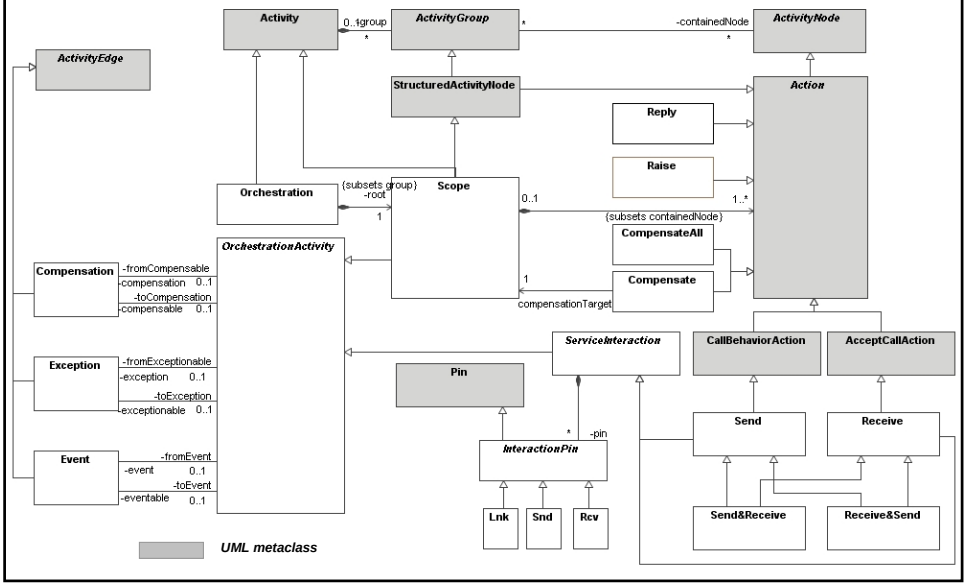
Several profiles for SOAs in SENSORIA



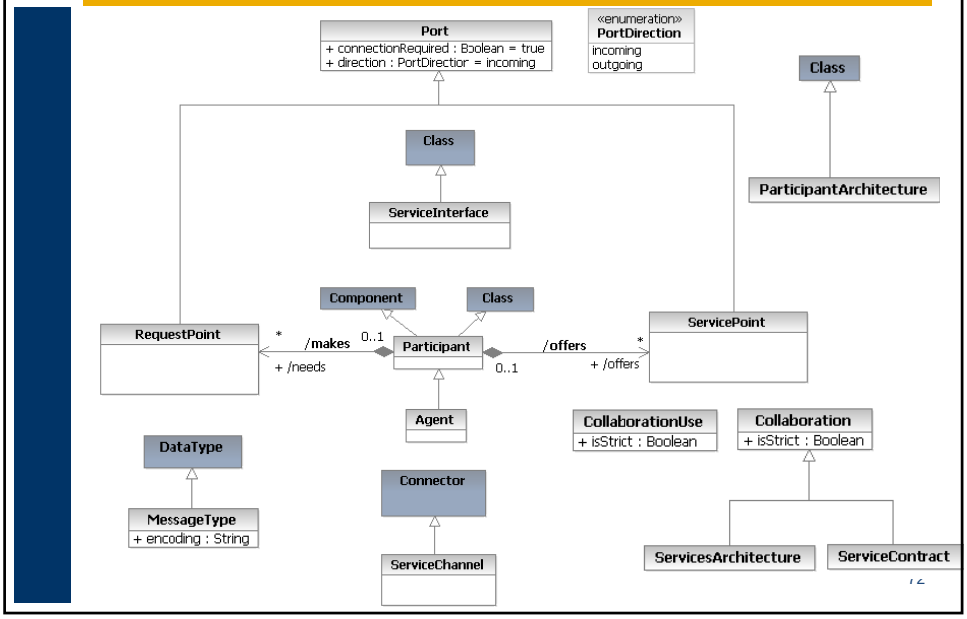
Profiles for modelling different aspects of SOSs: structural aspects (SoaML), behavioural aspects (Orchestration), non-functional properties (NFP), reconfiguration (Modes), policies (StPowla), requirements (SRML), and performance (MARTE).

UML4SOA metamodel: Orchestration Package

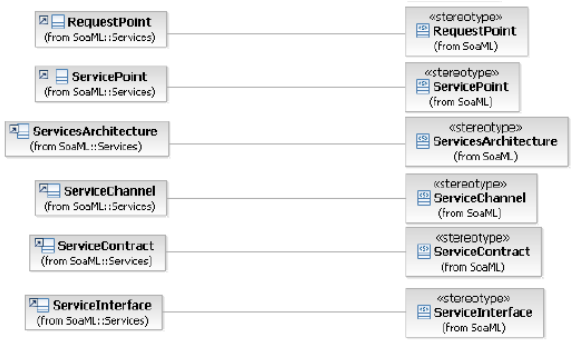
Conservative extension of the UML



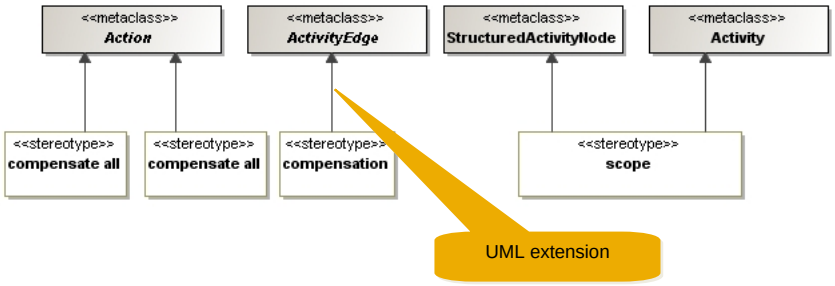
SoaML metamodel



Profile metamodel mapping (excerpt)



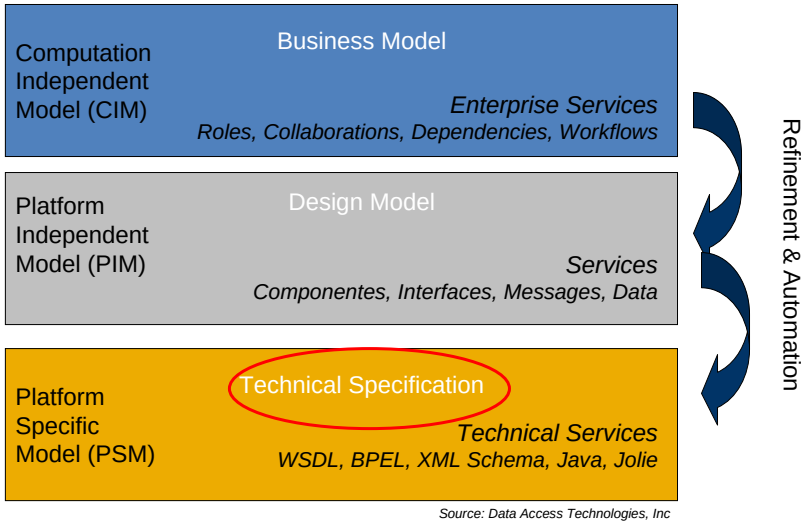
Extension model (excerpt)



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

SOA Models in the MDA Context



Source: Data Access Technologies, Inc

Programming language Jolie

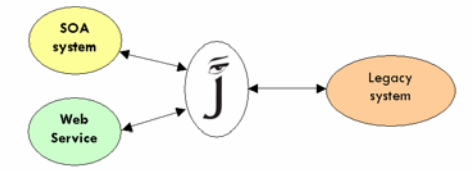
- Service-oriented paradigm
  - in Jolie everything is a service
  - used to create new services and compose existing ones
  - mechanisms for managing data, communication and service composition services
- Suitable for programming distributed applications
  - no distinction between local and remote services
  - endpoint locations and communication protocols can be changed dynamically thus allowing to build a dynamic system, fully reconfigurable at runtime

```
main {
    getInfo(request)(response)  {
            getTemperature@Forecast(request.city)(response.temperature)
            |
            getData@Traffic(request.city)(response.traffic)
    };
    println@Console("Request served!")()
     }
```

service concurrently retrieves information from a forecast service and a traffic service:

Jolie

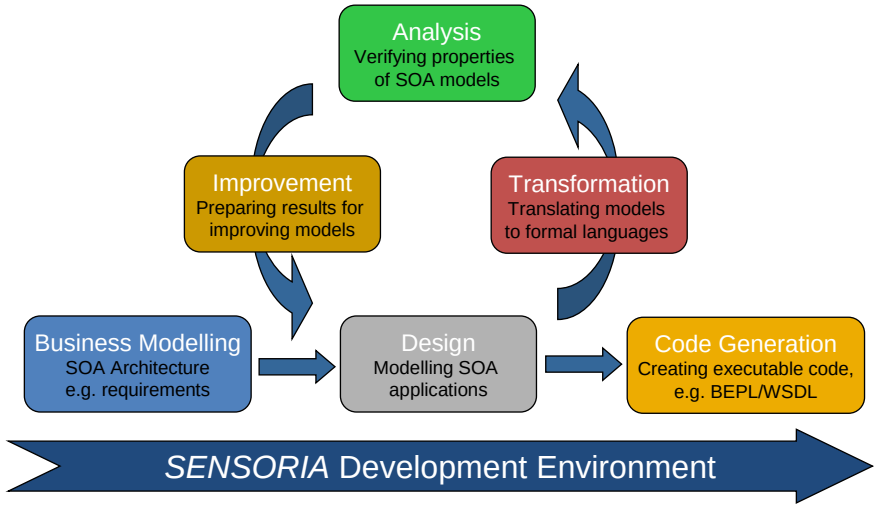
- Integration with other technologies
  - Jolie can be used for interacting with existing web services, own services and integrating legacy systems and current technologies within a SOA



- Open-source project
- Java Orchestration Language Interpreter Engine
- Spin-Off of the University of Bologna

italianaSoftware

Verification in model-driven service engineering

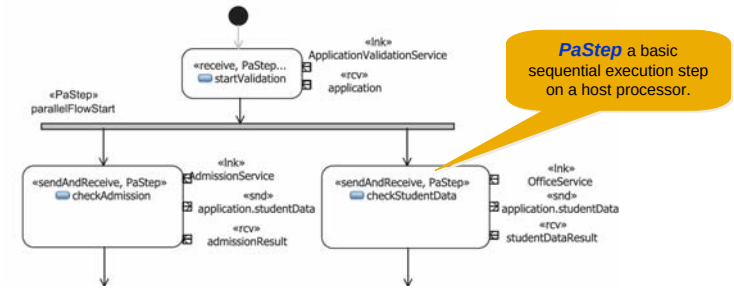


Quantitative and qualitative analysis methods

- Analysis using formal techniques
  - performance analysis
  - service level agreement analysis
  - security and behavioural analysis
- Methods and tools based on
  - stochastic simulation
  - model checking
  - logic
- Model-based analysis
  - in early phase of the development process

Performance analysis at model level

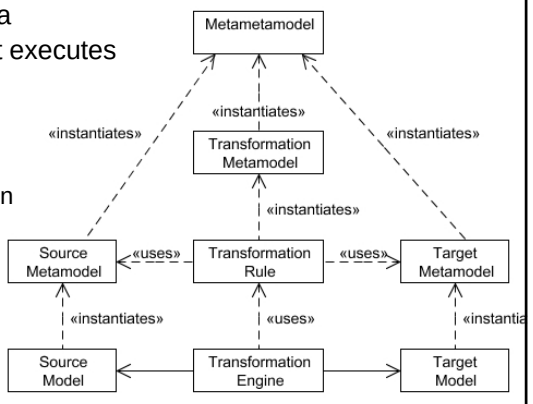
- Using formal techniques for SOA
  - prediction of service level agreement and performance
    - annotation of UML diagrams with rates of time consuming actions of the workflow (stereotypes of MARTE profile)
    - translation of the activity diagrams into stochastic process calculus PEPA
    - prediction with the tool SRMC (SENSORIA Reference Markovian Calculus)



Gilmore et al, LNCS 4346,2006

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)

MDD4SOA and VIATRA

- 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

mdd4SOA

- VIATRA2 ((VIsual Automated model TRAnsformations)
  - general tool based on graph transformations and abstract state machines
  - used within the project for deployment transformations
  - Eclipse project

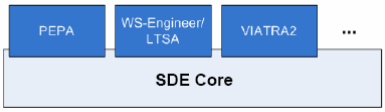
Mayer et al, EDOC 2008
Varró et al,SAC, 2006

Tool support

Tool	Area	Case Study	Integration in SDE
SDE	*Integration*	*All*	*X*
UML4SOA profile	Modelling	Finance, eUniversity, Automotive	X
MDD4SOA Transformer	Transformation	Finance, eUniversity, Automotive	X
VIATRA2	Transformation	Finance, eUniversity	X
LTSA/WS-Engineer	Analysis	eUniversity	X
PEPA	Analysis	eUniversity	X
LySa	Analysis	Finance	X
UMC/CMC	Analysis	Automotive	X
DINO	Runtime	Finance, Automotive	X
SRML Editor	Modelling	-	-
ADR2GRAPHS	Visualisation	Automotive	-
Jolie	Modelling	eUniversity	X
Reengineering Tool	Reengineering	Finance	-

SDE: SENSORIA Development Environment

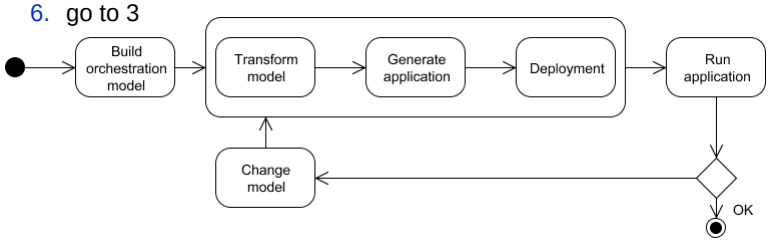
- Eclipse-based integration platform for developing SOA-based software
  - SDE Core
  - integrated tools



- 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

MDD4SOA@work

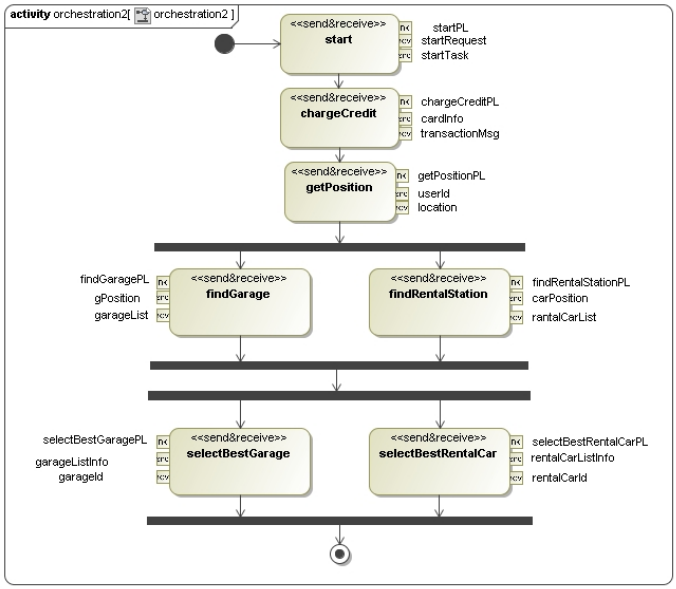
- Demonstration's aim
  - to show how model-driven development of SOSs can work
- Consists of
  1. building an orchestration model with UML4SOA
  2. defining a tool chain of transformations in SDE
     - model2model, model2code, deployment
  3. execution of the tool chain
     - input: UML4SOA model
     - output: application
  4. running the deployed application
  5. changing the model
  6. go to 3



1. Building an orchestration model with UML4SOA

- 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)

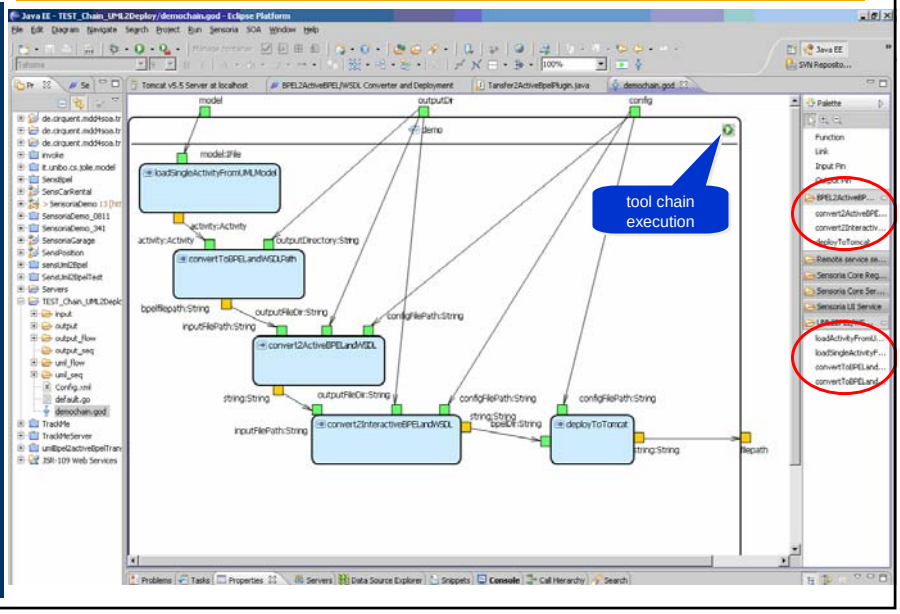
Orchestration model for “On road assistance”



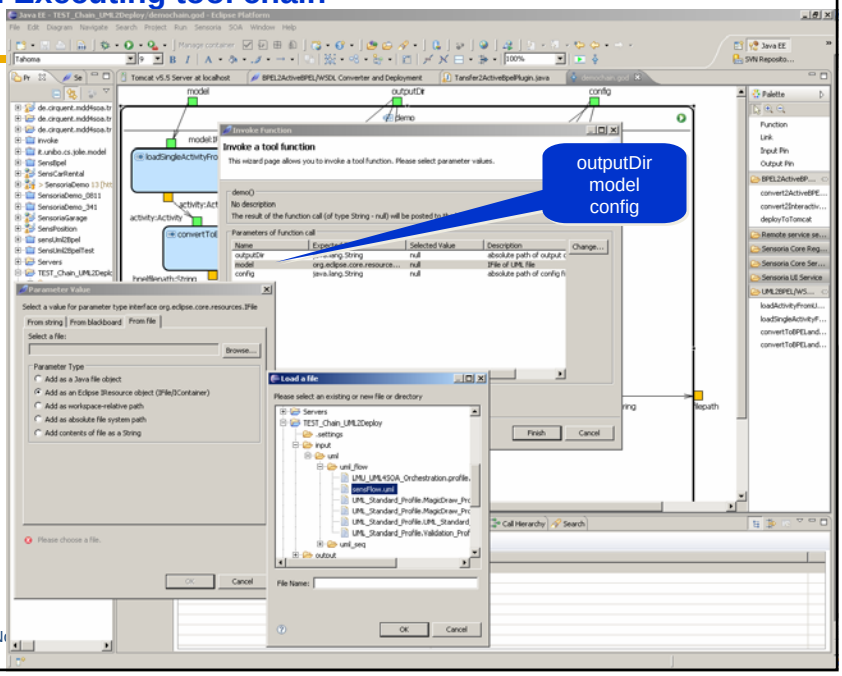
2. Defining tool chain in SDE

- 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 BPE /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)

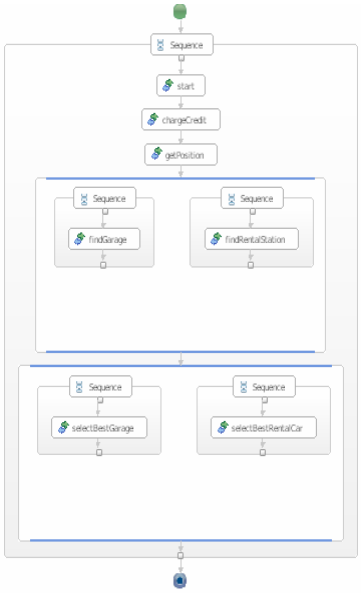


3. Executing tool chain



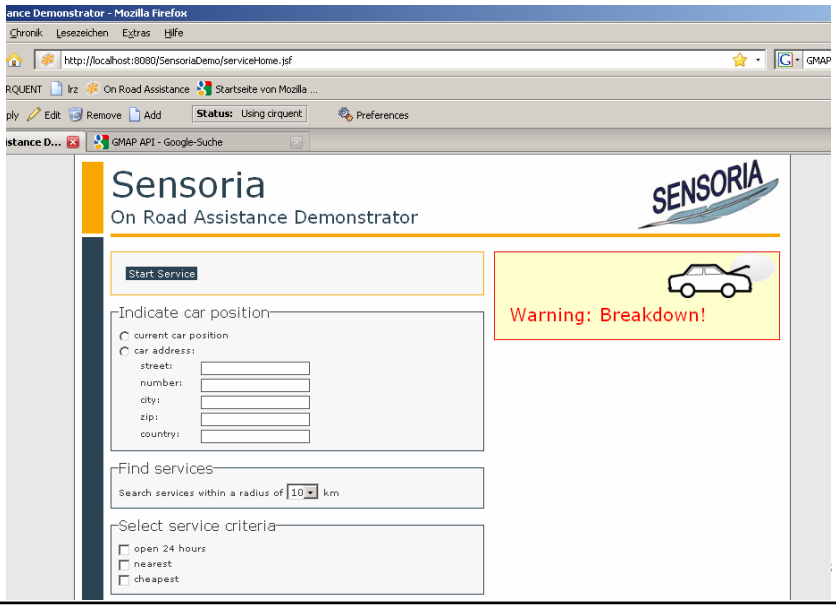
Looking at transformation results

BPEL model



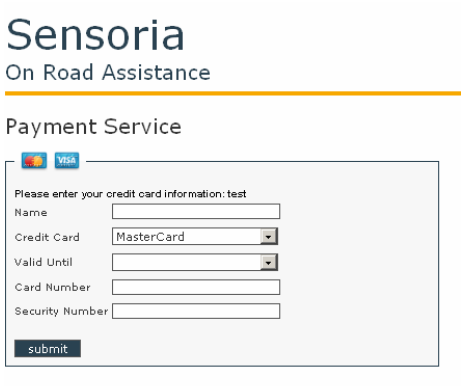
4. Running the deployed application

Home Page - Setting of Preferences



4. Running the deployed application

Credit card charge



4. Running the deployed application

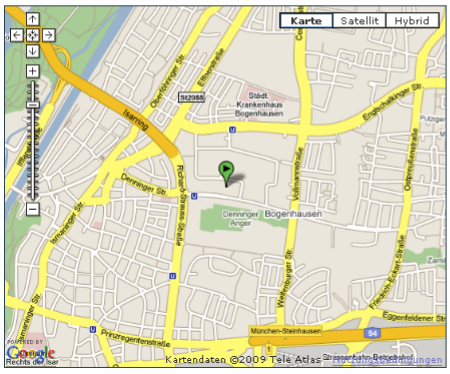
Car position

Sensoria

On Road Assistance

SENSORIA

Car Location



Next step

search rental car station nearby
search garage nearby

continue

Current Location

4. Running the deployed application

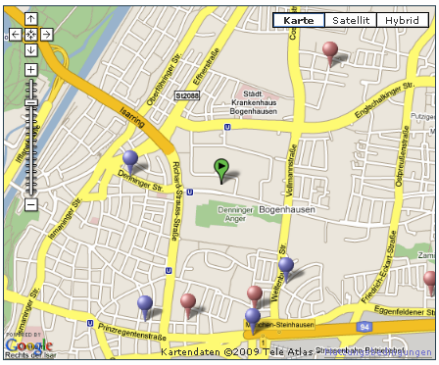
Garage and rental car services

Sensoria

On Road Assistance

SENSORIA

Garage nearby your car
rental car station nearby your car



Next step

search best garage
search best rental car station

continue

Current Location

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

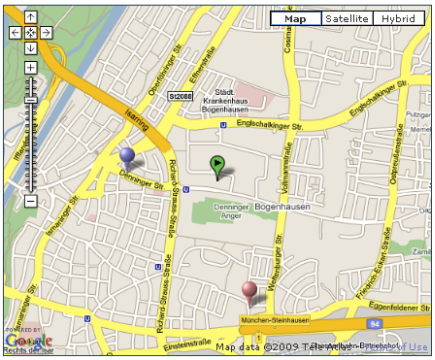
4. Running the deployed application

Selection best services

Sensoria

On Road Assistance

The best Garage
The best rental car station



Next step

start new service

continue

Current Location

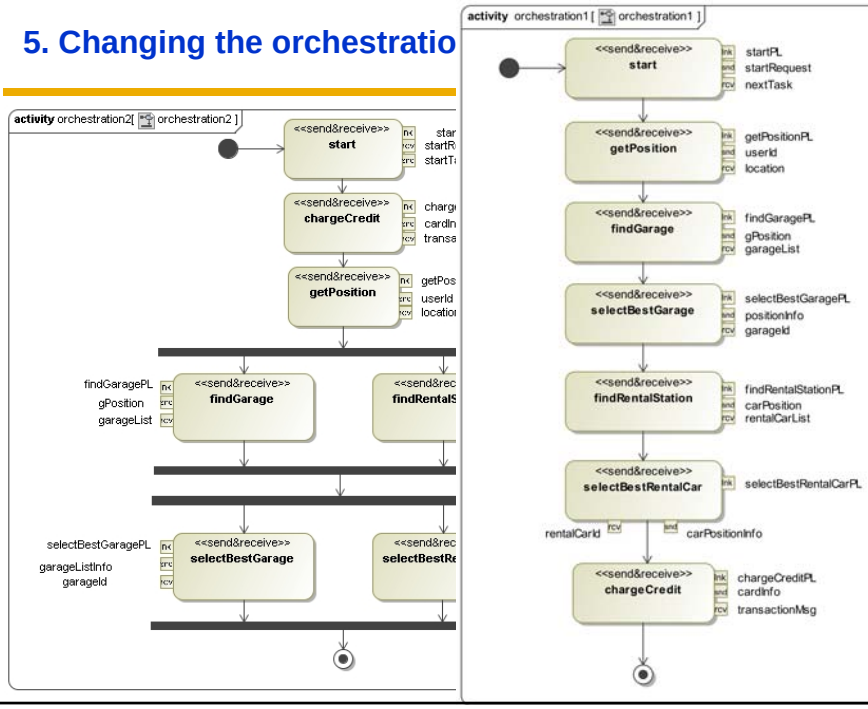
The best garage

Garage Denninger get route

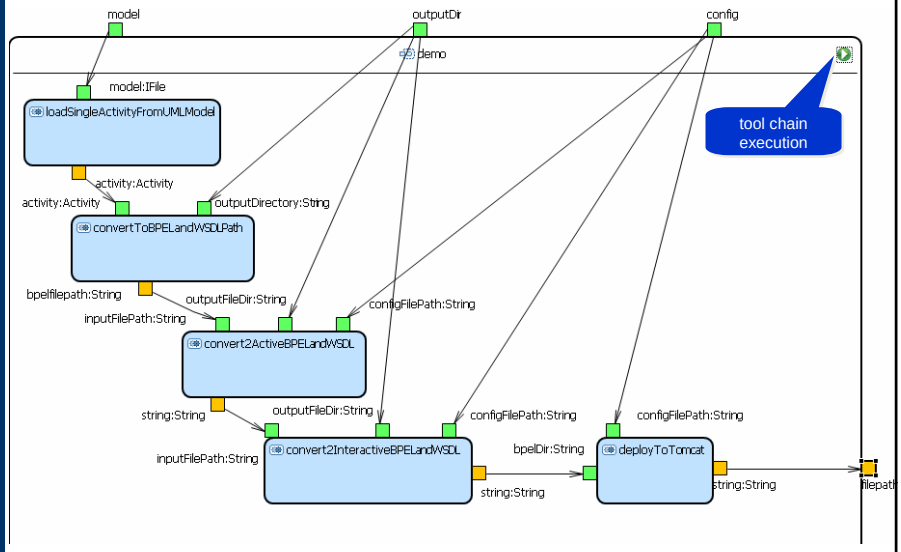
The best rental car station

Car Rental Gotthelf get route

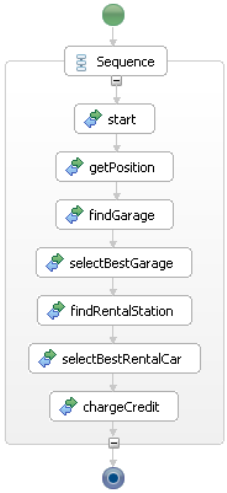
5. Changing the orchestratio



6. Back to the tool chain

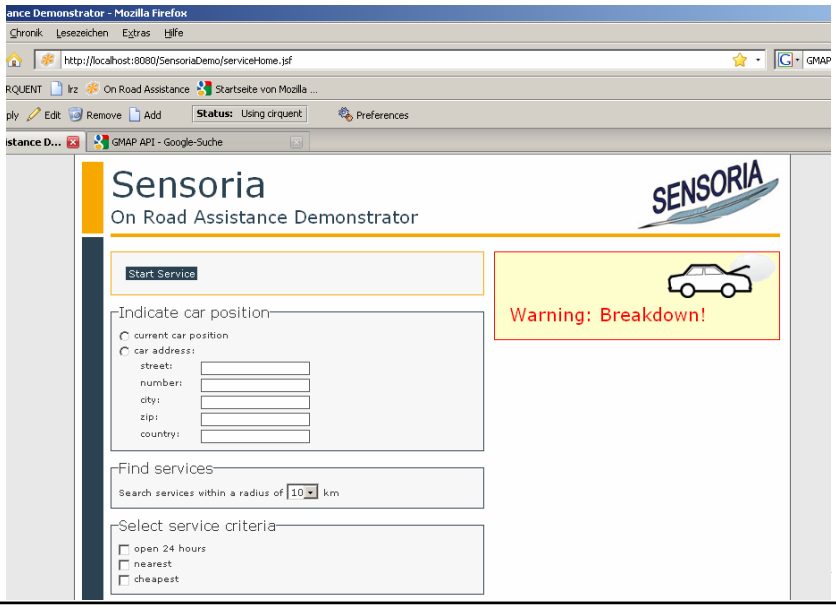


Looking at transformation results: BPEL model



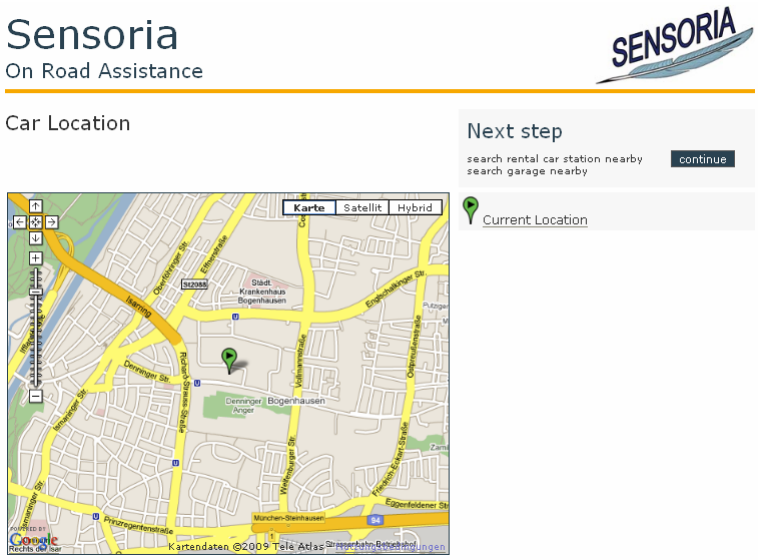
4. Running the deployed application again

Home Page - Setting of Preferences



4. Running the deployed application again

Car position



4. Running the deployed application again

- Different order of web pages
  - Credit card charge at the end
  - Only list of garages
  - etc.

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-in)
  - for solving a particular problem arising in the development process
- To ameliorate this problem we are developing a catalogue of patterns that
  - serves as an index to our results
  - illustrates, in a concise manner, the advantages and disadvantages of the individual techniques

Scalability analysis pattern

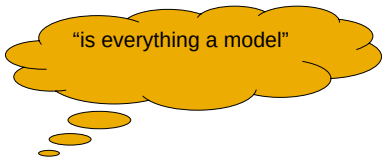
- Context
  - a large-scale service provider using replication to scale his service provision to support large population.
- Problem
  - understanding the impact of changes in number of servers or number of users subscribed to his service
- Forces
  - being able to support large-scale use is an indicator of quality in planning
  - heavy demand due to large user populations require service replication, but replication represents costs
- Solution
  - develop a high-level model of the system and apply continuous–space analysis to the model to make predictions about the large-scale system
- Related patterns
  - sensitivity analysis
- Tools
  - PEPA Eclipse plug-in project

Patterns catalogue

- Patterns defined so far …
  - Service modelling
  - Service specification and analysis
  - Functional service verification
  - Sensitivity analysis
  - *Scalabilitiy analysis*
  - Declarative orchestration
  - Model-driven development

A pattern-based approach to augmenting service engineering with formal analysis, transformation and dynamicity, Martin Wirsing et al., ISOLA 2008

Conclusions



- Service Engineering Approach
  - modelling of SOSs
  - metamodels and UML profiles for SOC
  - transformations to analysis models
  - automatic generation of SOAs
  - pattern language
  - MDD4SOA@work

Bottom line: Ideas to take home

- Relevance of domain specific modelling language
  - UML profile
  - must be simple, few constructs
- Automated development approach
  - model-based
  - model-driven (transformations)
  - pattern-based
- Importance of flexible tool support
  - easy (graphically) integration of diverse tools

Publications

- OMG, www.omg.org
- SENSORIA project, www.sensoria-ist.eu
- SHAPE project (SoaML), www.shape-project.eu

Thank you for your time and attention !

Nora Koch
kochn@pst.ifi.lmu.de

Further information
www.sensoria-ist.eu