simpleSOAD® 2.0
Architecture & Governance

Paris, December 8, 2010
Libero Maesano & Fabio De Rosa
libero.maesano@simple-eng.com
fabio.de-rosa@simple-eng.com

22° International Conference on Software & Systems Engineering and their Applications - Paris – Dec 7-9 2010

www.simple-eng.com
# Table of contents

- Business automation
- Service orientation
- Contract-based
- Model-driven
- Service contract
- Services architectures
- Contract modeling
- Motivation model
- Business model

- Functional specification
- Platform Independent Model
- PI Functional Model
- PI Interaction Model
- PI Sample Model
- Interoperability PSM
- Implementation PSM
- Conclusion
Business Automation

- Tens, hundreds, thousands,... applications, systems, devices will be connected and will collaborate without human intermediation...
- ...allowing the automation of business processes that support daily activities
- **Dependability** and **security** requirements will reach levels until now reserved to critical sectors
- Current design methods and tools are challenged
- **Service orientation** and **Service Oriented Architecture** are the breakthroughs
Service orientation

- **System** = entity that interacts with other entities, i.e. other systems, including hardware, software, humans, organizations and the physical world (Avizienis et al. 2004)
- A system has **function**, **structure** and **behavior**
- Systems have **boundaries**:  
  - internal vs. external (interface) structure  
  - internal vs. external behavior
- **Service** = activity performed by a system (provider) that engenders effects carrying value for another system (consumer)
- **Service** = **function** + **interface** + **external behavior**
- **Service orientation** is a design and implementation approach that separates  
  - the **service specification** – function, interface and external behavior - from  
  - the **system(s) specification(s)** – internal structure(s) and behavior(s) of the system(s) that provide (and consume) the service
Contract-based service orientation

- The service is fully described in a service contract.
- Avizienis et al. (2004) state: "correct service is delivered when the service implements the system function"
- Service orientation reverses this point of view: a system acts correctly if it provides – and consumes – services in compliance with the corresponding service contracts.
- Service contract = set of clauses detailing:
  - Functionality (function)
  - Interaction between service parts (provider and consumer)
  - Security constraints
  - Quality of service constraints
- The service contract doesn’t include any specification of:
  - Function, interface and external behavior outside its scope
  - Structure and internal behavior of the service parties
- The service contract is a first-class object.
- Contract-first approach: the service contract definition precedes the system design and implementation (even for legacy systems !)
## Model-driven Engineering

**A service contract is a layered set of formal models**

<table>
<thead>
<tr>
<th>Service contract</th>
<th>MDA Model</th>
<th>Service Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Business Motivation Model (BMM)</td>
<td>Service Motivational Model - Service BMM (BMM)</td>
</tr>
<tr>
<td></td>
<td>Business Model (BM) or Computationally Independent Model (CIM)</td>
<td>Service Conceptual Model – Service BM / Service CIM (SBVR)</td>
</tr>
<tr>
<td></td>
<td>Platform Independent Model (PIM)</td>
<td>Service Logical Model - Service PIM (UML 2, SoaML, QFTP, UML Testing Profile)</td>
</tr>
<tr>
<td>Interoperability Platform Specific Model (Interoperability PSM) Platforms: Web services, REST, CORBA,...</td>
<td>Service Interoperability PSM (Web services platform: XSD, WSDL, UDDI, WS-SecurityPolicy, WSRM Policy,...)</td>
<td></td>
</tr>
<tr>
<td>Implementation Platform Specific Model (Implementation PSM) Platforms: JEE, .NET, ...</td>
<td>System Physical Model(s) – System Implementation PSM</td>
<td></td>
</tr>
</tbody>
</table>

Contract design = Modeling, model mapping and transformation
## References

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SoaML</td>
<td></td>
</tr>
</tbody>
</table>
Service models

- Service models are contracts
- The Motivation Model is formulated by business strategists and can be understood and validated by high level management (thanks to BMM)
- The BM/CIM is created by business analysts and can be understood and validated by business people (thanks to SBVR)
- PIM is built by architects, from the BM, is cross-validated by business analysts and can be understood by implementers
- PSM’s (Interoperability and Implementation) are crafted by technical architects and engineers from the PIM
- Service contract models are shared between parties
- System constructive (implementation) models are private to parties
- What about model mapping and transformation?
Service Oriented Architecture

- Distributed architecture design and implementation approach ...
- ... in which the collaboration among systems is carried out through the exchange of services governed by contracts
- A services architecture is a network of roles, connected by service contracts, enacted by participants, i.e. classes of systems that collaborate in order to achieve business goals
  - Each role is the result of the composition of the parts of the service contracts it plays (as a provider or as a consumer)
- The specification of the functions, interfaces and external behaviors of a participant is the union of the service contracts it endorses (as a provider and as a consumer)
- A participant is the class of system implementations that are compliant with the specified functions, interfaces and external behaviors
- Business Automation enabler
simpleSOAD® 2.0 Contract modeling

- Strong contract-based, model-driven service orientation
- Methodological framework developed by simple engineering since 2004
- Applied in manufacturing, railways, government ... projects
- simpleSOAD® 2.0: full adoption of OMG emerging standards - BMM, SBVR, UML 2, SoaML, QFTP, UMLTP
  - simpleSOAD® 2.0 Profile and Metamodel extends and specializes the standards
Motivational model

- IT systems are no more a posteriori designed supports of an a priori established business model
- The mere adoption of the service orientation modifies the business practices
- Motivational analysis allows to elicit and formalize (thanks to BMM and SBVR) the WHY of a service for the parties
- Means-ends analysis – means-ends model (of the parties and of the service)
  - ends that motivate means
- Impact analysis of business events (new regulations, changes in the environment, strategic change...) - upon the established means-ends model
  - events that motivate changes in the means-ends model
- Bargaining and cooperative games equilibrium
- The updated means model is the starting point of the design cycle
- Motivation modeling is the fulcrum of the governance cycle
Business Model

- The Business Model (Computationally Independent Model) is a declarative formal service model expressed in structured natural language (SBVR).
- On the basis of a body of shared meanings, represented by a business vocabulary, it allows to enounce service functional clauses as business rules.

Vocabulary
- concepts and terms - object types, roles, fact types
- Capturing the service “linguistic game” (Wittgenstein)
- Fixing concepts: “Don’t ask for the meaning, ask for the use” (Wittgenstein)
- Making the conceptual body as simple as possible -“Entia non sunt multiplicanda praeter necessitatem” (Occam’s razor)

Business rules
- structural vs. operational rules
- invariants, preconditions, postconditions, questions
- Business rule = a rule that is under business jurisdiction
- Business rules are not business routines
BM - Service functionality

All services, as activities performed by a provider that engenders effects carrying value for a consumer, can be classified in two categories:

- **State/transition services**
  - The service performance is a transition - valuable to the consumer - between states of resources managed by the provider.

- **Pure informative services**
  - Delivery of valuable information by the provider to the consumer, without any state change.

**Functional description**

- State/transition service: operation, precondition, postcondition (thanks to B. Meyer “Design by Contract”)

- Pure informative service: question
BM Functional model

State/transition service: contract clauses (precondition, postcondition) as SBVR business rules
Supporting concepts (object types, roles, fact types) are defined in the service business vocabulary

<table>
<thead>
<tr>
<th>Guidance Type:</th>
<th>operative business rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>withdrawal precondition</td>
</tr>
<tr>
<td>Supporting fact types:</td>
<td></td>
</tr>
<tr>
<td>withdrawal is on account</td>
<td></td>
</tr>
<tr>
<td>withdrawal has amount</td>
<td></td>
</tr>
<tr>
<td>withdrawal is accepted at date/time</td>
<td></td>
</tr>
<tr>
<td>account has state at date/time</td>
<td></td>
</tr>
<tr>
<td>account has balance at date/time</td>
<td></td>
</tr>
<tr>
<td>amount[1] is greater than amount[2]</td>
<td></td>
</tr>
<tr>
<td>Supporting noun concepts:</td>
<td></td>
</tr>
<tr>
<td>account withdrawal balance amount date/time Active</td>
<td></td>
</tr>
</tbody>
</table>

It is obligatory that a withdrawal that is on an account is accepted at a date/time if and only if the state of the account at the date/time equals Active and the balance of the account at the date/time is greater than the amount of the withdrawal

<table>
<thead>
<tr>
<th>Guidance Type:</th>
<th>operative business rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>withdrawal postcondition</td>
</tr>
<tr>
<td>Supporting fact types:</td>
<td></td>
</tr>
<tr>
<td>withdrawal is on account</td>
<td></td>
</tr>
<tr>
<td>withdrawal has amount</td>
<td></td>
</tr>
<tr>
<td>withdrawal is executed at date/time</td>
<td></td>
</tr>
<tr>
<td>account has balance at date/time</td>
<td></td>
</tr>
<tr>
<td>date/time[1] is immediately after date/time[2]</td>
<td></td>
</tr>
<tr>
<td>Supporting noun concepts:</td>
<td></td>
</tr>
<tr>
<td>account withdrawal balance amount date/time</td>
<td></td>
</tr>
</tbody>
</table>
BM Functional model (II)

Pure informative service: contract clauses (question) as SBVR concepts
Supporting concepts (object types, roles, fact types) are defined in the service business vocabulary

<table>
<thead>
<tr>
<th>Get Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>General concept:</td>
</tr>
<tr>
<td>Closed projection:</td>
</tr>
<tr>
<td>Supporting fact type:</td>
</tr>
<tr>
<td>Supporting noun concepts:</td>
</tr>
</tbody>
</table>

- First step: creation of the service vocabulary (body of shared meanings)
- Business rules are stated on the basis of the vocabulary
- The service vocabulary is augmented, without semantic shift, with complementary formulations (sustainable conceptual model), to allow mapping to the service logical model (PIM)
- Examples of such complementary formulations are the objectifications of n-ary (n > 2) fact types (ex.: withdrawal)
Platform Independent Model

Service Logical Model

- **PI functional model**
  - sustainable vocabulary to the object model (UML 2 + simpleSOAD® profile)
  - business rules (service clauses) to operation signatures with OCL preconditions, postconditions (state/transition) and bodies (pure informative)

- **PI interaction (interface + external behavior) model**
  - service conversation, interfaces, service interfaces, service choreography (SoaML + simpleSOAD® profile)

- **PI security model**
  - security annotations on the functional and interaction models (QFTP + simpleSOAD® profile)

- **PI quality of service model**
  - quality of service annotations on the functional and interaction models (QFTP + simpleSOAD® profile)

- **PI sample model**
  - samples – cross-verification of the model – oracles (UMLTP + simpleSOAD® profile)
**PI Functional model**

Object model from the sustainable vocabulary

OCL specification of service operation from business rules

---

**State/transition “atomic” service: withdraw**

**context** Bank::withdraw(p: TupleType(account:AccountCode, amount:Decimal)) :

TupleType(number:PositiveInteger, dateTime:DateTime, balance:Decimal)

**pre: let** accountInSet = Account.allInstances() -> select(a | a.code = p.account) **in**

if accountInSet -> notEmpty()
then let account = accountInSet -> asOrderedSet() -> first() **in**

account.state = 'active' **and** account.balance > p.amount

else false endif

**post: let** account = Account.allInstances() -> select(a | a.code = p.account) -> asOrderedSet() -> first()

result.balance = account.balance **and**

account.balance = account.balance@pre - p.amount

---

**Pure informative “atomic” service: getBalance**

**context** Bank::getBalance(p:AccountCode) : TupleType(balance:Decimal, now:DateTime)

**pre: Account.allInstances() -> collect(code) -> includes(p)**

**body: let** account = Account.allInstances() -> select(a | a.code = p) -> asOrderedSet() -> first() **in**

Tuple(balance = account.balance, now = Clock::now())
Service conversation

Ideally, service conversation passes through three phases:
1. Deliberation
2. Execution
3. Reporting

Conversation pattern: **SynchronousRequest**

- simpleSOAD® interaction patterns - thanks to Winograd & Flores (1986)
- Communicative act patterns (performative verbs)
  - request, decline, undertake, report, query, inform, ...
- Conversation patterns (StateMachines)
  - SynchronousRequest, SynchronousDeferredRequest, ConversationForAction, SynchronousQuery, ...

© simple engineering 2004 - 2010 – Some rights reserved.
Maesano & De Rosa – ICSSEA 2010
Withdraw service

PI Functional & Interaction Model

One functional model
Several interaction models
• Conversation patterns
• Exchange modes
• MessageTypes
• Choreography

context Bank::withdraw(p: TupleType(account:AccountCode, amount:Decimal)) : TupleType(number:PositiveInteger, dateTime:DateTime, balance:Decimal)
pre: let accountInSet = Account.allInstances()->select(a | a.code = p.account) in if accountInSet->notEmpty() then let account = accountInSet->asOrderedSet()->first() in account.state = 'active' and account.balance > p.amount else false endif
post: let account = Account.allInstances()->select(a | a.code = p.account)->asOrderedSet()->first() in result.balance = account.balance and account.balance = account.balance@pre - p.amount
PI Sample model

Fact model

Thanks to Nijssen & Halpin (1989), the object model is mechanically transformed (one shot) into a 5th normal form relational model.

Fact (relational) bases are populated with facts of “possible worlds”

Samples: instances of service performance

A sample is:

- A fact base representing the state before the service performance
- An interaction diagram representing an instance of the conversation
- The corresponding set of MessageType instances

Samples are:

- Functional checking samples (the fact base verifies the precondition)
- Robustness checking samples (the fact base does not verify the precondition)
- Security checking samples
- ...

Samples are PIM first-class objects

Samples are templates for test cases
Interoperability PSM

- Interoperability platforms: Web services, REST, CORBA, RMI, ...
- Mechanical generation of XSD from MessageTypes
- Mapping of service interface to WSDL
- Mapping of security QFTP annotations to WS-SecurityPolicy
- Mapping of message reliability QFTP annotations to WSRM Policy
- Mapping of services architectures to UDDI V3 data structures
Implementation PSM

- Mechanical transformation of the object model to Java & C# class models
- Mapping of the object model to relational model
- Class model automatic persistence (with Hibernate, Nhibernate and other frameworks)
- Library of service wrapper/adapter patterns, covering a very large spectrum of legacy system architectures
- Mapping to standard frameworks (Axis 2, ...)
Conclusion

Today - simpleSOAD® 2.0
- Contract-based, model-driven, service oriented methodological framework
- Simple, straight, quick and complete design approach based upon standards

Tomorrow - Research project: Stochastic decision aid for SOA testing
- In the contract-based, model-driven, service oriented approach the validation question ("Are you building the right system?") becomes: "Are you modeling the right service contract?"
- ... and the verification question ("Are you building the system right?") becomes: "Are you building a system compliant with the contract?".
- We are focusing our attention on system verification – i.e. black-box testing of participants and grey-box testing of services architectures
- The starting point is the availability of service samples from the design phase that act as templates for test oracles
- We are looking for automated test environments (test definition and execution) for SOA testing – TTCN-3 is a strong candidate
- We are designing and developing a decision aid tool helping to optimize the testing strategy (“What is the next test?”) based upon stochastic reasoning and inference, in partnership with the Laboratoire d'Informatique de Paris 6 (LIP6) and the Centre National de la Recherche Scientifique (CNRS).
Bibliography


Jan L. G. Dietz: Enterprise ontology: theory and methodology; Springer, 2006

B. Meyer: Object-oriented software construction; Prentice Hall, 1997


SIMPLE ENGINEERING - Stochastic Decision Aid for SOA Testing - Interim Report - R2010073101

SIMPLE ENGINEERING - simpleSOAD® 2.0 Reference Manual - R2010070101

Thanks

Questions ?