simpleSOAD® 2.0 - Architecture & Governance

Libero Maesano and Fabio De Rosa
SIMPLE ENGINEERING
31-35, rue St. Ambroise 75011 Paris (F)
libero.maesano@simple-eng.com, fabio.de-rosa@simple-eng.com

Abstract: This paper introduces the simpleSOAD® methodological framework for design and implementation of services and services architectures, founded upon a contract-based, model-driven (MDA) approach and the OMG standards. Contract-based service orientation is presented first. The service contract is a first-class object, is independent from the systems that endorse it, and acts as a functional and behavioral specification for these systems. Furthermore, the principles of application of the model-driven engineering approach to service design are given. A service contract is a layered set of models: BMM (motivational), BM (conceptual), PIM (logical) and Interoperability PSM (physical). They are all based upon largely accepted OMG standards. The service contract is designed top-down with the help of methods and tools of model mapping and transformation. The Implementation PSM of a system that intends to provide the service can be partially generated by the contract model. Future work will concern phases and activities of the SOA life cycle till now poorly covered, such as Validation & Verification, Test and Governance.

Keywords: SOA, MDA, MDE, contract-based, model-driven, BMM, SBVR, SoaML, QFTP.

1. INTRODUCTION

With the spreading of business automation a large number of applications, systems and devices will be connected and will collaborate without human intermediation. Such architectures, involving companies, government and for-no-profit organizations, will allow the automation of business processes that support daily activities. The requirements of dependability and security [1] for such architectures will reach levels until now reserved to critical sectors such as defense and infrastructure utilities. Service orientation and Service Oriented Architecture (SOA) are the breakthroughs that allow organizations to implement distributed architectures of loosely coupled systems to achieve flexible, dependable and secure business automation.

simpleSOAD® is a methodological framework, developed by SIMPLE ENGINEERING since 2004, for design and implementation of Service Oriented Architectures. Since the time of its inception, simpleSOAD® has been employed in successful projects in manufacturing, rail transport and government. simpleSOAD® 2.0 - Architecture & Governance deepens and solidifies the simpleSOAD® conceptual apparatus and core methods and provides: (i) standardization, with the full adoption of emerging OMG standards [2] [6] [7] [9] [12]; (ii) extension of the framework towards hitherto partially covered SOA life cycle stages - motivational analysis, business modeling, legacy systems reverse modeling, validation & verification, test - and towards SOA governance.

Section 2 aims at clarifying the peculiar characteristics of the contract-based service orientation. The service is described in a service contract independent from service implementation and acts as a functional and behavioral specification of the systems that intend to endorse it. Section 3 presents the application of the Model-Driven Engineering (MDE) approach to the service contract design and to service-aware system design and implementation. Section 4 reports on related work and introduces the main sources of inspiration of simpleSOAD®. Section 5 draws some conclusions and sketches the main perspective of SIMPLE ENGINEERING research, which is about SOA testing environments and on intelligent tools for supporting SOA testing strategy on the basis of probabilistic reasoning.

2. CONTRACT-BASED SERVICE ORIENTATION

Service Orientation is a design and implementation approach that separates the specification of a service from the specification of the system that provides it. A system is an entity that interacts with other entities, i.e., other systems, including hardware, software, humans, organizations and the physical world. A service is an activity performed by a system that engenders effects carrying value for another system, the former and the latter respectively fulfilling the provider part and the consumer part in the service relationship. The service is the result of the provider's internal behavior that causes an external behavior perceived by the consumer as the service performance.
**simpleSOAD®** adopts a contract-based service orientation: a service must be fully described in a service contract [9]. An authoritative approach states: "correct service is delivered when the service implements the system function" [1]. The contract-based service orientation reverses this point of view: correct service is fully described in the contract, and the system acts correctly when it delivers service in compliance with the contract.

The service contract includes a set of clauses detailing:
(i) **Functionality** - the provider activity's effects that carry value for the consumer.
(ii) **Interaction** - the exchange of communicative actions between the provider and the consumer in order to coordinate and deliver the service performance.
(iii) **Security** - the needs and constraints about confidentiality, accountability, authentication, authorization.
(iv) **Quality of service** - the needs and constraints about availability, reliability, safety, integrity, maintainability.

The service contract must be precise, non ambiguous and detailed, but it shall not include any specification, other than the service specification. It must not include neither specifications of function and external behavior of the parties outside the scope of the service, nor specifications of the internal structure and behavior and of the implementation technology of the systems providing and consuming the service. While service contracts are obviously shared between the parties, implementations of systems that embody the parties are private. Thus the service contract acts as a functional and behavioral specification, delimited to the scope of the service, for the parties' design and implementation.

A service contract is usually established between two parties, but it can also be multi-party - a classical example being the explicit definition in the contract of a trusted third party monitoring the service performance. Service contracts can be specialized and aggregated. Compound contracts [9] are the result of the recursive aggregation of "atomic" contracts. Service contract aggregation can range from stateless assembled contracts, where component contracts are simply juxtaposed in the compound contract and can be solicited independently of each other, to stateful choreographed contracts, in which compound contract choreography drives the chaining and interleaving of the component services performances. Contract aggregation is not what is commonly - and perhaps inappropriately - called service composition. The latter refers to a situation in which the provider, in order to be able to supply the service, needs to consume other services provided by other systems. So, the so-called service composition actually refers to dependence relationships between participants of a services architecture, not to aggregation relationships between service contracts.

Service Oriented Architecture is a distributed architecture design and implementation approach in which the collaboration among systems (organizations, IT systems, users) is carried out through the exchange of services governed by contracts. A services architecture [9] is a network of roles, connected by service contracts, played by participants, i.e. classes of systems able to put in place and orchestrate the capabilities needed to enact the roles that collaborate in order to achieve business goals. Each role is the result of the composition of the parts of the service contracts it endorses as a provider or as a consumer.

### 3. MODEL-DRIVEN SERVICE ORIENTATION

The requirements of flexibility, dependability and security of services and services architectures cannot be met without good design processes, tools and methods. The Model Driven Architecture (MDA) program of the Object Management Group (OMG) is aimed to establish these processes, tools and methods on the basis of the modeling and model transformation approaches. The simpleSOAD® framework adopts the MDA approach to service and services architecture design and implementation: a service contract, as a functional and behavioral specification of the systems endorsing it, is a layered set of models (as shown in the next table).

<table>
<thead>
<tr>
<th>MDA Model</th>
<th>Service Contract Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Motivation Model (BMM) [2]</td>
<td>Service Motivation Model - Service BMM</td>
</tr>
<tr>
<td>Business Model (BM) [7]</td>
<td>Service Conceptual Model - Service BM</td>
</tr>
<tr>
<td>Platform Independent Model (PIM) [9] [6] [12]</td>
<td>Service Logical Model - Service PIM</td>
</tr>
<tr>
<td>Interoperability Platform Specific Model (Interoperability PSM) Platforms: Web services, REST, CORBA,...</td>
<td>Service Physical Model(s) - Service Interoperability PSM Web services platform: XSD, WSDL, UDDI, WS-SecurityPolicy, WSRM Policy, ...</td>
</tr>
</tbody>
</table>

The service orientation modifies the design strategy. The traditional approach that considers IT systems an a posteriori designed support of an a priori established business model is no more valid, as the mere adoption of
the service orientation changes the business practices. Thus the BMM and the BM levels are really important: motivation analysis (modeling) allows to elicit and formalize the why of a service for the parties, and conceptual modeling, thanks to SBVR [7], allows to establish, on the basis of a body of shared meanings between service parties, a representation of the contract clauses that is formal but understandable by the business experts and users.

From the functional point of view, simpleSOAD® classifies atomic services in two categories: (i) state/transition services and (ii) purely informative services. The effect of a state/transition service performance is a transition, valuable to the consumer, between allowed states of resources managed by the provider. The effect of a purely informative service is the delivery of valuable information by the provider to the consumer, without any state change. The functional description of a state/transition service is formally established in the conceptual model (BM) as a service operation, together with a precondition on the state of the resources that allows the service performance and a postcondition on the state of the resources after the service delivery [4]. The operation is objectified [7], and the precondition and postcondition are expressed as business rules on the basis of the object and fact types of the service vocabulary - the following is an example of precondition in SBVR suggested notation, whose meaning is intuitive.

```
It is obligatory that a withdraw that is on an account is accepted at a date/time if and only if the balance of the account at the date/time is greater than the amount of the withdraw.
```

Guidance Type: operative business rule
Note: withdraw precondition

The functional description of a purely informative service is formally established as a question, i.e. a query expressed in SBVR on the basis of the service vocabulary. The creation of a service vocabulary is the first step towards the functional definition of the service at the conceptual level. The service vocabulary should be minimal: concepts that are not useful for the service functional description should be dropped out (Occam's razor). If necessary, the service conceptual model must be augmented, without semantic shift, with complementary formulations that allows the mechanical transformation into the service logical model (PIM). Examples of such complementary formulations are the objectifications of n-ary (n > 2) fact types [7]. As a result, the part of the augmented vocabulary candidate to PIM transformation, that is called sustainable vocabulary, is only made of object types and binary fact types. Moreover, the sustainable vocabulary must include a reference scheme for each object type, the specifications of multiplicities for each binary fact type and closed vs. open world assumptions [3] [5]. Of course, business rules (preconditions, postconditions, invariants, questions) must be reformulated on the sustainable vocabulary.

The service logical model is a PIM, built with SoaML [9]. Atomic and compound services are formally specified by ServiceContract(s), Provider(s), Consumer(s), ServiceInterface(s), (UML) Interface(s), MessageType(s), Capability(s). The functional definition of the service is represented, both for state/transition and purely informative services, as the signature of a private operation implemented by the Provider, together with, for state/transition services, precondition and postcondition expressions in OCL - as shown in the following example.

```
context WithdrawProvider::withdraw(p: WithdrawInput) : WithdrawResult
pre: p.account.balance > p.amount
post: result.balance = p.account.balance and
       p.account.balance = p.account.balance@pre - p.amount
```

The functional specification of a purely informative service is a query operation signature together with an OCL body expression. Like the business rules of the conceptual model that are built upon the BM vocabulary, the OCL expressions of the logical model are built upon the PIM domain model (the service ontology), that can be obtained by mechanical transformation of the sustainable vocabulary into classes, attributes and associations that are stereotyped with the simpleSOAD® Profile stereotypes [11]. object type(s) are mapped to ObjectType(s)«ObjectType» classes, binary fact type(s) between an object type and a lexical object, i.e. a text sub-concept mapped to a Text - a «Text» DataType - are mapped to attributes of the corresponding ObjectType typed by the Text. binary fact type(s) between two object types are mapped to FactType(s) «FactType» associations. Reference schemes for object types, multiplicities of roles and open vs. closed world assumptions are mapped to UML constructs too. A SIMPLE ENGINEERING ongoing project aims to build a standard QVT transformation between the MOF-based SBVR model [7] and the simpleSOAD® ontology model [11].
In the specification of the service interaction, the service performance is embedded in a service transaction that is made up of three phases: (i) a deliberation phase, in which the service preconditions are evaluated and the parties exchange information and commitments about the future service performance; (ii) a delivery phase, in which the service operation is executed and the service is delivered; (iii) a reporting phase, in which the parties exchange information and assessments about the past service performance [13]. The service transaction is coordinated by the exchange of communicative actions between the consumer and the provider. The communicative act approach [8] allows to consider separately the intention of a communicative act from its content, i.e. the state of affairs object of the intention. simpleSOAD® supplies a library of performative verb patterns in order to build performative (intention-explicit) communicative actions. These performative verbs are employed to coordinate the service performance (request, undertake, decline, report, query, inform, subscribe, notify...), to drive the service delivery (stop, go, next, halt, suspend, restart...) and to prematurely exit the service transaction (cancel, renounce, refuse, quit...). simpleSOAD® also brings in a library of conversation patterns (as UML Protocol State Machines) that meaningfully combine communicative acts together with timeout events specifying the semantics of silence. The design of the service interaction is achieved with the specification of the communicative acts contents as MessageType(s) and the definition of communicative acts as public operations in UML Interface(s), realized by the Provider and used by the Consumer and, conversely, realized by the Consumer and used by the Provider. The service functionality and interaction PIM is completed by the definition of the Capability(s) that realize the service parties, and the definition of the service choreography (an UML Activity owned by the ServiceContract) representing a control flow involving all the communicative and internal actions of the parties. Last but not least, the security and quality of service clauses are represented as QFTP [6] annotations on the ServiceContract.

"A ServicesArchitecture (or SOA) is a network of participant roles providing and consuming services to fulfill a purpose" [9]. Services architectures are specified at PIM level with SoaML Participant(s), Capability(s) and ServicesArchitecture(s). Each ServicesArchitecture is the result of the coupling of (i) a contracts architecture, i.e. the set of the concerned ServiceContract(s) and of their aggregation and generalization/specialization relationships, and (ii) a participants architecture, i.e. the Participant(s) roles in the architecture, the ServiceContract connections between these roles and, for each Participant, the composition of internal Capability(s) needed to perform its roles (in each ServicesArchitecture in which it is involved). Moreover, each participant role owns an orchestration (UML Activity) that depicts the control flow of its own communicative and internal actions. The role orchestration must be consistent with the choreographies of the ServiceContract(s) that it endorses and the Capability(s) of the Participant carrying out the role.

A peculiarity of contract-based service orientation is that the Platform Specific Model layer for a services architecture must be split into two sub-layers corresponding to (i) the Interoperability PSM, that is part of the contract and obviously shared between participants, (ii) the Implementation PSM, outside the contract and private to each participant that has full jurisdiction on it. The transformation of the service contract interaction model (PIM) into the service Interoperability PSM on the Web services platform is straightforward. It can be automatically performed by the most popular UML Modelers, at least for the automatic generation of XSD elements from MessageType(s) and WSDL elements from UML Interface(s). Also the transformation of QFTP annotations into WS-SecurityPolicy and WSRM Policy assertions is straightforward, and is the goal of an ongoing project. simpleSOAD® proposes for the Implementation PSM (JEE and .NET platforms) two general patterns of provider and consumer wrappers whose interface components are generated from the Interoperability PSM (Web services) and whose core components are generated from the PIM service ontology.
4. RELATED WORK

The simpleSOAD® main sources of inspiration are: (i) the fact oriented approach, (ii) Design by Contract and (iii) Language/Action Perspective. The fact oriented approach [5] fathered a family of methods of conceptual schema analysis and design, such as NIAM and ORM. It should be noticed that these methods and approaches had strongly influenced the SBVR model and design [7]. The sustainable conceptual model and the service ontology model are NIAM/ORM-like schemas expressed in SBVR (BM) and UML class/association (PIM) respectively. Design by contract [4] is a well known theory by B. Meyer that views software construction as based on contracts between clients (callers) and suppliers (routines), relying on mutual obligations and benefits made explicit by assertions. Design by Contract is applied to service functionality description, through Provider operation signatures with attached pre/post conditions. Language/Action Perspective [13] [3], based on the pioneering work of Winograd and Flores, applies the theory of communicative acts [8] to human collaboration through computer systems. The theory of communicative acts is employed by the simpleSOAD® framework to model the exchange of information, commitments and assessments that coordinates the service performance.

5. CONCLUSION AND FUTURE WORK

In our experience, the contract-based and model-driven simpleSOAD® approach of services and services architectures analysis and design has met its main goal, that is the realization of a sound, dependable and secure services architecture in situations in which the participants of the architecture belong to different independent organizations. In our opinion, an important success factor of the approach is that the contract model is standard, can be understood by people that do not participate to its design process, and even do not master the methodological framework: the model is independent from the method that engenders it. Our research focus is now on SOA Validation & Verification, Test and Governance.

In the contract-based, model-driven approach the validation question ("Are you building the right system?") is transformed in: "Are you modeling the right contract?", and the verification question ("Are you building the system right?") becomes: "Are you building a system compliant with the contract?". In other words, the availability of a validated formal contract transforms the system validation task into a system verification task - verification of the system compliance with the contract. This verification could be carried out by formal methods inside the system jurisdiction, but the SOA Integrator needs strategies, processes, methods, tools applicable to black box testing of participants and grey box testing of services architecture. The starting point is the availability of a collection of service samples, i.e. examples of service performance that are compliant with the contract, that is a by-product of the design phase, and is used for continuous model cross-checking. A sample is made of a service oracle - a correct instance of a service transaction - on a fact model (an instance of the service ontology that verifies the oracle precondition). We are building a general model, compliant with UML Testing Profile [12], of an environment for contract-based SOA functional testing together with an intelligent tool that helps the testers to optimize their testing strategy, based upon stochastic reasoning and inference [10]. This work
is being conducted in partnership with the Laboratoire d'Informatique de Paris 6 (LIP6) and the Centre National de la Recherche Scientifique (CNRS).

6. REFERENCES