Service contract clauses as business rules

Fabio De Rosa¹, Ariele P. Maesano², Libero Maesano³

¹ fabio.de-rosa@simple-eng.com - SIMPLE ENGINEERING, Rome, Italy
² ariele.maesano@lip6.fr - Laboratoire d’Informatique de Paris 6 UPMC, Paris, France
³ libero.maesano@simple-eng.com - SIMPLE ENGINEERING, Paris, France

24th International Conference on Software & Systems Engineering and their Applications
ICSSEA ’12 – Paris Oct. 23-25
Table of contents

- Problem statement
- Elements of the solution
- An example: Withdrawal
- Conclusion
Problem statement

Contract-first Service Oriented Architecture

- **Services** are not **systems** – they are activities carried out by systems (providers) that bring value to other systems (consumers).
- Each service shall be precisely defined by an artifact - the **service contract** - that must be considered as a bundle of rights and duties by the service parties.
- The service contract is the formal model of the service that the parties must agree.
- MDA approach: the service model is layered
  - **Computation Independent Model** (CIM) – *declarative model* of the service
  - **Platform Independent Model** (PIM) – *abstract computational model* of the service – independent from a specific platform (UML/OCL Model)
  - **Platform Specific Models** (PSMs) – *concrete computational models* of the service – dependent on the specific platform – halved in two parties
    - **Interoperability Platform(s) SM** – allowing *interoperability* of non interoperable implementations (SOAP, REST, ...) - included in the service contract
    - **Implementation Platform(s) SM** – supporting the parties’ implementations (JEE,.NET,...) – outside the service contract

The service model is not the system model!
Problem statement (II)

Service model modularity - separation of concerns

- **operation model** - *what* the provider *does* for the consumer (not *how* it does it!)
- **interaction model** - *how* the provider and the consumer *interact* in order to coordinate the service *invocation, deliberation, delivery* (or refusal), reporting (*conversations – choreographies*)

- **security model** - *authentication, authorization, confidentiality, integrity, accountability, non repudiation*
- **performance model** – *volume, throughput, speed, scalability, availability, reliability, integrity, maintainability*

The **service operation model** is the **core contract component** – the *same* service operation may be carried out through several different interaction, security and performance agreements

The **focus here** is on the **service operation expression at the CIM level**
The problem statement (III)

**Formality vs. understandability**

- Computation Independent Model: previously called *Business Model*
- How to express the service operation in a notation that can be easily *understood* and *validated* by the *business stakeholders*

*natural language*?

- How to express the service operation in a *formal notation* that allows:
  - *mechanical checking* (*model checking* for properties such as consistency)
  - *automated transformation* towards computational models (*model-driven engineering*)
  - *automated creation* of test cases (*model-based testing*)

*UML/OCL*?

- Understandability vs. mechanical checking, transformation and automated generation are conflicting exigencies:
  - *Natural language* is understandable but it is hard to be mechanically checked and automatically transformed
  - *UML/OCL* is formal but not understandable by the business stakeholders – it is suitable for the computational model (PMI)
Elements of the solution

Contract-based design formulation of the service operation

How to express a service operation in a *declarative* manner?

Design by Contract™ by Bertrand Meyer, that we call ‘contract-based design’\(^1\), is an approach for designing software that prescribes that software designers should define not only syntactic interface specifications for software components, but also semantic ones that extend the ordinary definition of abstract data types with pre-conditions, post-conditions and invariants.

An *operation*, equipped with a *signature*, can be precisely defined by:

- Its *pre-conditions* – to be assumed on entry
- Its *post-conditions* – to be guaranteed on exit
- Its *invariants* – to be assumed on entry and guaranteed on exit

\(^1\) Design by Contract is a registered trademark of Eiffel Software in the United States - In order to avoid the confusion between the general design approach and the specific Eiffel provisions and mechanisms, we will employ the generic term ‘contract-based design’
Elements of the solution (II)

Business rules

- Business rules are all around in the “real” economy, but yet for whatever reason they are seldom featured in requirements for the digital economy.

- The very first goal of the business rule research field and practice has been to give the rules the status of “first class” requirements and to distinguish them from other requirement artifacts such as use cases and process models.

- An essential feature of business rules is understandability.

Controlled natural language - formulation style that:

  - uses only words from a specified, relatively limited set of words in some defined natural language.

  - uses only specified forms of phrase, from a relatively limited set.

- Very important step: release by the OMG of the Semantics of Business Vocabulary and Business Rules (SBVR – 2008) - controlled natural language becomes formal (equipped with a formal meta-model)!
Elements of the solution (III)

Semantics of Business Vocabulary and Rules (SBVR)

- SBVR-compliant Business vocabulary
  - contains all the specialized terms, names, and fact type forms of concepts that a given organization or community uses in their talking and writing in the course of doing business
  - supporting basis for expressing business rules
- SBVR-compliant Business rules
  - Criteria for making decisions
  - Consistent with formal logics:
    - structural (definitional) rules are alethic claims (necessity)
    - operative (behavioral) rules are deontic claims (obligation)
  - Business rules are logical expressions, not CA or ECA rules!
- SBVR does not mandate any particular notation
- Specification includes a recommended proposal for expressing business rules with controlled natural English language
An example: Withdrawal

Writing convention
- terms - that design **noun concepts** - are underlined (term)
- verbs - that design **fact types** - are in italics (verb)
- names - that design **individual concepts** - are doubly underlined and with the first letter capitalized (Name)
- keywords - that design **quantifiers, logical operators, modal operators** ... - are in boldface (keyword)

Bank vocabulary
- the expression of operation clauses on a **banking service** needs the definition of **object-type** and **role** entries such as ‘account’, ‘balance’, ‘amount’, ‘customer’ and of **fact-type** entries such as ‘customer **withdraws** amount from account’, ‘account number **identifies** account’, ‘account has state’, ‘account has balance’ ...
An example: Withdrawal (II)

Objectification (crucial formulation device)

Propositions based on the ternary fact type ‘customer withdraws amount from account’ can be objectified by adding to the service vocabulary the noun concept ‘withdrawal’ and additional binary fact types such as ‘customer effects withdrawal’, ‘withdrawal involves account’ and ‘withdrawal takes out amount’.

The relationship between the objectified fact type and the objectifying object types and fact types is established through a structural rule such as:

It is necessary that each withdrawal that is effected by a customer and that involves an account and that takes out an amount is an actuality that the customer withdraws the amount from the account.

An actuality is a state of affairs that happens in the actual world

Withdrawal operation signature

Add vocabulary entries such as: ‘withdrawal operation’, ‘argument’, ‘result’, ‘withdrawal operation has argument’, ‘withdrawal operation has result’, ‘argument specifies account number’, ‘argument specifies amount’, ‘result specifies account number’, ‘result specifies account balance’.

And a structural rule such as:

It is necessary that each withdrawal operation has exactly one argument that specifies exactly one account number and exactly one amount and has exactly one result that specifies exactly one account number and exactly one account balance.
An example: Withdrawal (II)

Withdrawal operation pre-condition

bearing on resources states and operation arguments expressed as an operative rule

It is obligatory that
if
a withdrawal operation that has an argument that specifies an account number and that specifies an amount handles a withdrawal
then
the account that is involved in the withdrawal is identified by the account number immediately before the withdrawal occurs
and the account is in the extension of the concept ‘account’ immediately before the withdrawal occurs
and the account state of the account equals Active immediately before the withdrawal occurs
and the balance of the account is greater than the amount immediately before the withdrawal occurs

The keywords ‘if’ and ‘then’ respectively introduce the antecedent and consequent of a material implication. With the fact type ‘state of affairs$_1$ occurs immediately before state of affairs$_2$ occurs’, the SBVR Structured English supports objectification by proposing a convenient mechanism that is based on the word ‘occurs’ being in the designation of a fact type after a placeholder. An implicit form of a fact type can be used that objectifies a propositional expression in the position of the placeholder and leaves out the word ‘occurs’. The fact type ‘state of affairs$_1$ occurs immediately before state of affairs$_2$ occurs’ conveys the meaning of the well-known Allen’s “X meets Y” relation between the temporal interval X in which the state of affairs$_1$ occurs and the temporal interval Y in which the state of affairs$_2$ occurs.
An example: Withdrawal (III)

Withdrawal operation post-condition

bearing on resources' states “immediately after” and operation results, and their relationships with resource states “immediately before” and operation arguments expressed as an operative rule.

It is obligatory that

if

a withdrawal operation that has an argument that specifies an account number \(_1\) and that specifies an amount

and that has a result that specifies an account number \(_2\) and that specifies a balance \(_1\) handles a withdrawal

then

the account involved in the withdrawal has a balance \(_2\) immediately after the withdrawal occurs

and the account has a balance \(_3\) immediately before the withdrawal occurs

and the balance \(_2\) equals the balance \(_3\) minus the amount

and the account number \(_2\) identifies the account immediately after the withdrawal occurs

and the account is in the extension of the concept ‘account’ immediately after the withdrawal occurs

and the account number \(_2\) equals the account number \(_1\)

and the balance \(_1\) equals the balance \(_2\)
An example: Withdrawal (IV)

Withdrawal operation invariant

- bearing on the subsistence of states of the service world that shall not change as an effect of the service operation execution
- we consider that they are implicitly asserted (Common Sense Law of Inertia)
- there is no need of explicitly stating invariants as “dumb” operative business rules such as:

It is obligatory that
if
a withdrawal operation handles a withdrawal
then
the account that is involved in the withdrawal and that is in the extension of the concept ‘account’ immediately before the withdrawal occurs is in the extension of the concept ‘account’ immediately after the withdrawal occurs
and the account number that identifies the account immediately before the withdrawal occurs identifies the account immediately after the withdrawal occurs
and the account state that is of the account immediately before the withdrawal occurs is of the account immediately after the withdrawal occurs
and ...
An example: Withdrawal (V)

Explicit/implicit meta-requirements of the approach

The service operation must be realized by an activity that exhibits the characteristics listed below:

- **Robustness** - the activity can be performed only if the service operation pre-condition rule is satisfied
- **Atomicity** - the activity shall be “all or nothing” and does not show intermediate states - only the initial and the final ones are accessible
- **Isolation** - the activity shall be conducted as the only one that accesses the resources that are referred to in the service operation pre/post-condition rules
- **Accuracy** - the activity effects – state changes & service operation results – shall satisfy the service operation post-condition rule
- **Durability** - the activity effects shall be permanent “all things being equal”
- **Safety** - the activity must not produce any effect that is not regulated by the service operation post-condition rule
Conclusion, present and future

Service operation contract-based formulation with SBVR makes real the dream of formality and understandability – these conceptual tools can be used now - hand-made transformation to UML/OCL (PIM)

we are working on:

- controlled natural French language
- controlled natural Italian language – within HL7 Italia

Ongoing and future research programme in the MDA community:

- editing tools and formal checkers that help formulating the vocabulary and the rules
- tools supporting the automatic transformation of the vocabulary and rulebook into artifacts based on PIM level notations, such as UML/OCL and eventually OWL
- automatic transformation of elements of the service PIM (UML/OCL, OWL) into the interoperability PSM (SOAP, REST platforms) and the implementation PSM (JEE, .NET platforms) - this issue has already been broadly investigated in the research field and by the authors, and henceforth can be considered an engineering rather than a research task
- model-based testing – automated, systematic (checking pre/post conditions, unintended side effects ...) and purpose-driven generation of test cases from the service model (from SBVR – UML/OCL to TTCN-3 ....)

MIDAS project - www.midas-project.eu
Questions?

Thank you for your attention

More information:
http://blog.simple-eng.com