Service-Oriented Information Systems Engineering: A Situation-Driven Approach for Service Integration*

Nicolas Arni-Bloch and Jolita Ralyté

University of Geneva, CUI, 24 rue General Dufour
CH-1205 Geneva, Switzerland
{Nicolas.Arni-Bloch,Jolita.Ralyte}@cui.unige.ch

Abstract. In this work we propose a Metamodel of Information System Service (MISS) and introduce a situation-driven approach for ISS integration. This approach is based on situational method engineering principals and is defined as a collection of inter-related method chunks.

1 Applying SOA to Information System Engineering

The community of IS developers increasingly adopts service-oriented approach to IS engineering. In this work we consider a particular type of services – IS Services (ISS) – that have to be integrated into enterprise legacy IS in order to avoid IS fragmentation. Following the traditional SOA [1, 5], integration of an ISS into an IS would be limited to the exchange of messages between services. That means that only needs for services and their capabilities to response would be considered but not the information overlap that different services could have. This allows to resolve "point-to-point" integration and represents a step towards integrated IS but this not allows to resolve the problem of information overlap management. The overlap supported by data redundancy maintained between services will continue to generate extra cost and bad quality of data and processes.

Before publishing a service in the registry of the enterprise, it is necessary to guaranty the integrity of the data but also the alignment of the rules and processes on the IS policies. Besides, the impact of ISS integration has to be evaluated taking into account technical, informational and business aspects. At the technical level it means to consider the cost of the interoperability between language, framework or hardware components. The evaluation at the business level consists in analysing the capability of the enterprise to support new processes and to provide the necessary data. Finally, at the informational level links between data and the compatibility between processes and rules has to be considered. Therefore, we argue that the ISS integration cannot be limited to the exchange of messages between services but has to deal with information overlap. To enable the integration of ISS, it is necessary to extend service description with its informational knowledge: (1) the definition of service data structure and semantics, (2) the definition of service behaviour in terms of actions that can be

* This work was partially supported by the Swiss National Science Foundation. N° 200021-103826.

Z. Bellahsène and M. Léonard (Eds.): CAiSE 2008, LNCS 5074, pp. 140–143, 2008.

executed by the service and effects that these actions provoke, and (3) the definition of rules (data integrity and process rules) to be respected when realising the service. Taking into account the complexity of the integration process is critical and requires an advanced methodological support. In particular, in order to deal with the diversity of integration situations, the method for service integration has to be flexible, modular and configurable.

2 Metamodel of Information System Service (MISS)

We define an Information System Service (ISS) as an autonomous coherent and interoperable component of an information system that offers capabilities and owns resources to realize these capabilities. These resources can be technical (hardware or software), informational (information, behaviour and rules) and organizational (organizational role, actors). At the informational level, an ISS is considered through three spaces: static, dynamic and rules. Fig. 1 represents the Metamodel of Information System Service (MISS).

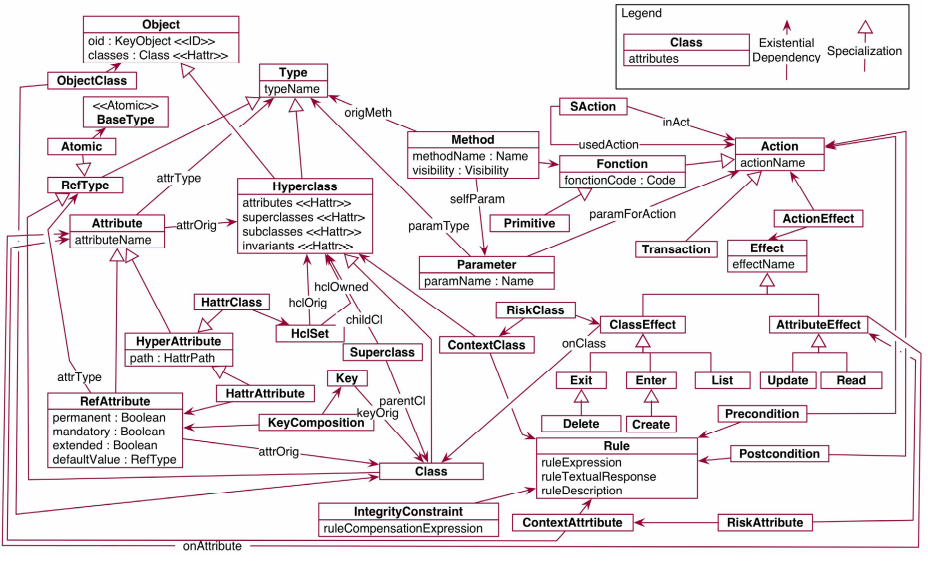

Fig. 1. Metamodel of Information System Service: the informational part of an ISS

Static Space. The static space of the MISS represents the data structure of an ISS by using the following concepts: *class*, *key*, *attribute*, *method* and *object*. Besides, we use the concept of *hyperclass* [4] in order to define the set of classes required by the service to realize its capabilities and to guaranty the completeness and coherence of its data structure.

Dynamic Space. The dynamic space of an ISS represents the behaviour of the service capabilities. The main concepts of the dynamic space are: *action* and *effect*. An action is an object that defines a behavior having effects on other objects. An action accepts

parameters that denote objects to be given to the action for its execution. An action is described by a *process* to be executed and produces one or more *effects* that specify the type of its result. Finally, the execution/enactment of an action is constrained by a set of *preconditions* and a set of *postconditions*. The effects of an action are defined by using a set of primitives: create, enter, exit, update, read, list and call.

Rule Space. The objective of this space is to preserve the coherence, correctness and consistency of the ISS during its exploitation. A *rule* is an expression/algorithm that returns a boolean value when evaluated. The classes and attributes that participate in the validation of the rule define its validation context. Rules are used as a basis for the specification of the *integrity constraints* and *conditions*. An integrity constraint is a rule that has to be verified in each state of the services or at each modification of it. A *condition* is a rule that has to be valid at some point of process execution.

3 A Situational Approach for ISS Integration

The process of ISS integration cannot be limited to a simple prescribed set of activities because of the multitude of integration situations that has to be considered. To be able to deal with this diversity of integration situations we construct our method following the principles of Situational Method Engineering (SME) and particularly the chunk-driven SME approach [2]. We define our approach as a collection of interrelated method chunks each of them addressing some specific activity in the ISS integration process and organised into a map-based [3] process model.

The ISS integration mainly consists in resolving the information overlap situations between the selected ISS and the legacy IS. We identify four main intentions that have to be considered in the ISS integration process:

1. To identify and characterise the information overlap between the ISS and the IS,
2. To adapt the overlap part in the ISS specifications in order to allow their integration with the IS specification,
3. To integrate the IS specifications and the ISS specifications, and
4. To consolidate the integrated specifications.

Therefore, our process model (Fig. 2) is based on these four intentions and identifies several strategies that have to be considered in order to achieve these intentions. The first step consists in identifying overlap situations in the specifications of the ISS and the IS, analysing these situations, identifying elements to be integrated (e.g. classes to be merged) and detecting elements that need to be unified before their integration. The *semantic analysis* strategy helps to identify naming inconsistencies in the static space of the IS and the ISS, while the *functional analysis* strategy focus on the identification of similar capabilities provided by both IS and ISS and specified in their dynamic space. The *structural*, *action-driven* and *rule-driven* strategies continue the identification and characterisation of the overlap in the static, dynamic and rule spaces respectively.

Next, the overlap situations requiring some unification have to be settled by applying the appropriate *semantic unification* and/or *transformation* operators. The third step consists in selecting the appropriate integration operator for each overlap situation and applying it. The strategy *"with integration operator"* in fact represents a bundle of exclusive strategies each of them dealing with specific static, dynamic or rules space integration operator.

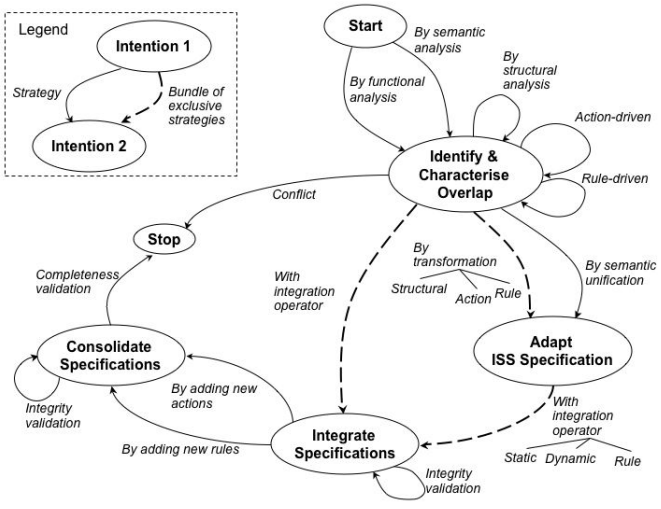

Fig. 2. Process model for ISS integration (Map formalism [3])

Finally, the integrated specification has to be consolidated. The integration of an ISS into an IS can create new situations which didn't exist neither in the initial IS nor in the ISS. Therefore, it can be necessary to *add new integrity rules* or new *actions* in order to guarantee the completeness and the coherence of the integrated specification.

Our approach supporting ISS integration aims to provide one or more method chunks for each section (a triplet <*source intention, target intention, strategy*>) of this map. Currently, we focus our effort on identifying and evaluating different situations that can occur in the ISS integration process and defining method chunks satisfying these situations. A tool support is also under development to store and enact the method chunk knowledge.

References

1. OASIS Reference model for service oriented architecture 1.0. Technical report (2006), http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf
2. Ralyté, J., Rolland, C.: An Approach for Method Reengineering. In: Kunii, H.S., Jajodia, S., Sølvberg, A. (eds.) ER 2001. LNCS, vol. 2224, pp. 471–484. Springer, Heidelberg (2001)
3. Rolland, C., Prakash, N., Benjamen, A.: A Multi-Model View of Process Modelling. Requirements Engineering 4(4), 169–187 (1999)
4. Turki, S., Léonard, M.: Hyperclasses: towards a new kind of independence of the methods from the schema. In: Proc. of ICEIS 2002, pp. 788–794 (2002) ISBN: 972-98050-6-7
5. Erl, T.: Service-Oriented Architecture (SOA): Concepts, Technology, and Design. Prentice Hall PTR, Englewood Cliffs (2005)