Semantic Integration in Conceptual Modeling of Service Architectures

Gustas, Remigijus

Abstract — ***The primary goal of service architecture is to align the business design with the information technology innovations in order to make both organisational and technical system parts more effective. Thus service architecture is not necessarily bound to the technical aspects of system development. It can be defined by using conceptual models that are independent of any implementation technology. Service-oriented architecture (SOA) provides principles of system decomposition into reusable, sharable and interoperable components, which require high degree of business data consistency. Unfortunately, the conventional information system analysis and design methods cover just a part of required modeling notations for engineering of service architectures. They do not provide effective support to maintain semantic integrity between business processes and data. Service-orientation is a paradigm that can be realized as a set of novel principles that can be used in conceptual modeling of enterprise architectures. Realizing a future vision on service-oriented analysis and design requires reassessment of existing conceptual modeling theories, concepts and practices. The most fascinating idea about service concept is that it applies equally well to organizational as well as to technical components. Principles of service-orientation could be successfully used for separation of concerns by breaking down enterprise system functionality into coherent non overlapping subsystems, which are represented by a set of service requesters and service providers. The concept of service is rather well understood in different domains and it can be expressed in different traditional modeling dimensions. Therefore, service-oriented descriptions can be used for semantic integration of the static and dynamic aspects of enterprise architectures.***

Index Terms — ***Conceptual modeling, Enterprise architecture, Intersubjective and objective views in system analysis and design, Semantic integrity, Service-Oriented modeling***

Manuscript received March 30, 2007.

Remigijus Gustas is a full professor at the Department of Information Systems, Karlstad University, Sweden (e-mail: Remigijus.Gustas@kau.se).

1. INTRODUCTION

Enterprise systems are evolving by adopting new configurations of service architectures, which prescribe and motivate various IT solutions. Service orientation promotes flexibility and interoperability by minimizing requirements for shared understanding. Enterprise architectures (EA) can be changed by replacing or recomposing more specific services. Traditionally, graphical representations of EA are built fragment by fragment and when all is done, then typically business and technical design does not fit each other. It is quite expensive and time consuming to maintain integrity and consistency of multiple specification fragments. Service architectures are intrinsically complex engineering products that can be defined on different levels of abstraction and represented by using several dimensions. One of the reasons why the traditional information system engineering methods do not provide effective support is that service architectures are difficult to visualize across disparate modeling dimensions such as the "why", "what", "who", "where", "when" and "how" [1]. Another problem is that the same implementation dependent artifacts are used in both system analysis and system design phases. It makes descriptions of service architectures less comprehensible for business experts.

The idea of computation independent modeling was introduced by the Object Management Group [2]. Two levels of computation independent models can be distinguished: semantic and pragmatic. The pragmatic requirements correspond to the "why" dimension. They typically refer to desirable or undesirable situations, which are expressed as intentions of stakeholders in terms of goals, problems and opportunities [3]. Pragmatics is supposed to motivate and drive the overall system analysis and design process. One of the main challenges in service-oriented analysis and design is mapping from the pragmatic to semantic modeling level. Semantic descriptions of services must follow the basic conceptualization principle

by representing only computation independent aspects. Such representations are less complex and more comprehensible for business process experts. They can be successfully used by non-technicians who play a key role in system integration. It is recognized that UML support for such task is quite vague, because semantic integration principles of different diagram types are still lacking [4].

Service-oriented analysis and design does not exclude the object-oriented (OO) point of view that is adopted by RUP, but rather suggest two additional semantic and pragmatic levels above the syntactic level of abstraction. Computation oriented modeling languages abstract from concrete implementation artifacts. This is a reason why specifications at the syntactic level are more comprehensible for software designers, but not readily accessible and understandable for business consultants and managers.

2. *Two System Development Traditions*

There are two significant qualities that characterize system development traditions: intersubjectivity and objectivity. Methods that put into foreground modeling of the external behavior have the intersubjective bias [5]. From the intersubjective point of view, service is a unit of functionality, which is exposed to environment. External behavior helps to understand a usage aspect of self-contained service components. Intersubjective bias is especially obvious in the enterprise modeling language Archimate [6]. Services can be also characterized by internal state changes [7]. Semantics of changes are typically represented by using state transition links. Transitions are triggered by operations, which specify the permissible ways for changes to occur in different classes of objects. Various types of OO diagrams that are intended for conceptual modeling of static and dynamic aspects have the objective bias to system development. Such tradition is very strong in the conventional system development approaches.

The static aspect of intersubjectivity can be defined by using inheritance, composition and classification relations among enterprise actors. The dynamic aspect of intersubjectivity is expressed by interaction dependencies [8], which represent physical, information or a decision flows between two kinds of actors involved. Service providers are actors that typically receive service requests, over which they have no direct control, and transform them into responses that are sent to service requesters. Each *Service Response* is a function of a *Service Request*. This idea illustrated graphically in figure 1.

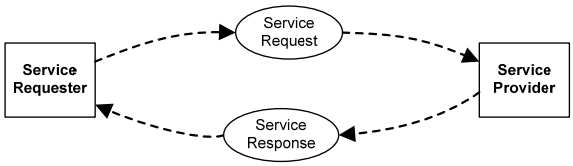

Figure 1: Intersubjective view of service

From the objective stand point, an action is changing business data from one consistent state to another. Quite often service outputs depend not only on inputs, but also on availability of stored data that result from other services. Such data are supposed to constrain service responses to the present or future inputs. For instance, if a reservation of trip is created, then it can be paid by using a trip payment service. Moving flows together with request and response actions, which create or remove objects of various classes, are crucial to understand the semantic aspects of services. This idea is illustrated by figure 2.

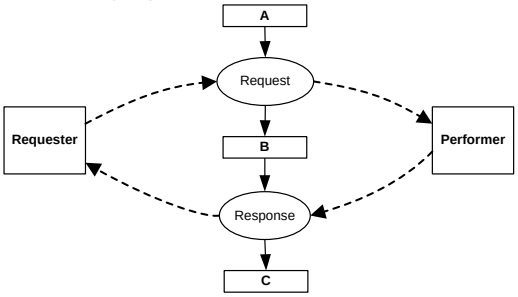

Figure 2: Intersubjective and objective views in a service interaction loop

The objective view of action can be defined by using transition links. The transition link from a class illustrates termination of an object and – to a class represents creation of object. For instance, a request action is supposed to remove an object from class A and to create an object in class B. Creation of object B is a precondition for initiation of response action by performer, which is supposed to remove business data about object B and create an object in class C. It should be noted that either precondition or poscondition class of action may be missing, but not both of them. Otherwise, an action makes no sense. Such action has no effect on the internal behavior of objects.

3. *Semantic Integration of Static and Dynamic Aspects*

The most conventional system development methodologies are biased on the objective tradition. They use totally different diagram types for defining separately an external and internal behavior. For instance, a used case diagram is capable to express just an external view of

system functionality. Most of the conventional methods typically use various diagram types for representation of many internal modeling dimensions in isolation. Nevertheless, there is an overlapping among them to some degree. For instance, the concept of operation in UML is represented in a class diagram ("what" dimension), activity diagram ("how" dimension), sequence diagram ("where" dimension) and state-transition diagram ("when" dimension). Furthermore, the atomic operations are typically aggregated into higher granularity functions that are represented as the elements of a use case diagram ("who" dimension). It should be noted that some use cases even can be interpreted as goals at the pragmatic level of abstraction. Interplay between the external and internal views of enterprise systems and services create big challenges even for the recently developed system analysis and design methods [8], which deal with an integrated modeling of static and dynamic aspects.

Enterprise models traditionally define how business, data, software application and technology architecture is perceived by different stakeholders. Since different modeling views and dimensions are highly intertwined, it is crucial to maintain integrity and consistency across multiple diagrams on various levels of abstraction. Traceability of changes from one diagram type to another is a bottleneck in traditional enterprise modeling approaches. Service-orientation can be applied for verification and validation of diagrams that are represented on the pragmatic, semantic and syntactic levels. Intersubjective and objective aspects of service loops are defined equally well for organizational as well as technical system parts. Organizational system parts can be represented by individuals, companies, divisions or roles, which denote groups of people. Technical parts are represented by data, software and hardware components.

Intersubjective semantics of services are captured by interaction loops, which are able to express the main workflow patterns such as sequence, selection, synchronization and iteration [9]. The objective tradition can be effectively used for defining an internal behavior of objects. An object lifecycle in service-oriented approach is represented by using initial, intermediate and final classes, which are analyzed in the context of interactions between organizational and technical system components. Semantics of objective changes is expressed by using three types of actions: reclassification, creation and termination [9]. A creation action, which is characterized by a missing precondition class, corresponds to a starting point. A termination action, which is characterized by a missing postcondition class, corresponds to the end point in object's lifecycle.

Intersubjective view predefines very basic structure of conceptual representation of service architecture. It is expressed by using service request and response flows into opposite directions, which can be successfully used for separation of concerns in system analysis and design. Typically, a coherent set of interactions are delegated to one independent technical component. All coherent interactions that fit together for the achievement of a common goal are used for breaking down enterprise system into coherent non overlapping subsystems that can be implemented as autonomous services. Since the concept of service is rather well understood in different domains, it has a potential to integrate intersubjective and objective views into one comprehensive notation. In such a way service-oriented diagrams are able to address semantic integrity and consistency problems of business data. It is not sufficient to represent what type of objects are created and terminated when an action is triggered. Service graphical descriptions are capable to express more generic classes, which are referred by using inheritance links. Such classes are typically characterized by an additional set of persistent attributes, on which an action of more specific class has no effect. Composition links are also of a great importance, since they represent related classes of objects, which are synchronously removed or created when an action takes place.

Information flows are reminiscent of arrows in dataflow diagrams [10], because they are representing moving data between enterprise system components, which may be interpreted as data sources and sinks. If a system is implemented without any computer support, then information flows may be understood as moving documents and pre/post-condition classes can be viewed as archived data at rest. Precondition and postcondition classes can be viewed as database files or data stores in the computerised system. It should be noted that the presented modelling approach is useful for graphical description of service architectures, which are not prescribing any implementation details. Semantic constructs follow the basic conceptualization principle by representing only computation neutral aspects.

4. CONCLUSIONS

The understanding of enterprise architecture relies on knowing how different subsystems are interconnected. Semantic relations among enterprise system components and objects define conceptual representations of service architectures. Interplay of intersubjective and

objective views in one service-oriented diagram facilitates better semantic integrity control between the static and dynamic aspects. There are typically many stakeholders involved during the architecture engineering process. For systematic analysis of service architectures, it is crucial to maintain a holistic representation, where external and internal views are visualized together. It is not reasonable to duplicate the same concepts many times in different diagrams just because such separation is required from a technical design point of view. Semantic integrity of static and dynamic aspects of service descriptions is achieved by superimposing the intersubjective and objective perspectives together.

Service-oriented paradigm should open a totally new way for enterprise engineering of service components that span across the organisational and technical system boundaries. Conceptual models of enterprise system architecture can be defined as a set of loosely coupled components. Service-orientation has the potential for organizations to reduce system architecture evolution complexity and to improve learning capacity. A new service-oriented approach for system analysis and design should bring significant benefits including: improved ability for organizations to maintain strategic knowledge in a systematic way, reduced costs for a systematic analysis of new IT solutions before they are implemented, improved integrity and traceability of knowledge within companies by providing comprehensible service architecture descriptions. Our experience in analyzing system specifications by using computation independent notation demonstrates that service-oriented descriptions are more comprehensible for personnel without a technical background. Service-oriented paradigm has no implementation bias and therefore it can be used for bridging a communication gap among system designers and business analysis experts.

REFERENCES

[1] Zachman, J. A., “Enterprise Architecture: The Issue of the Century”, *Database Programming and Design Magazine, 1996*.

[2] Object Management Group Architecture Board (2003), “MDA Guide”, version 1.0.1, 2003, (Ed.) Miller J., Mukerji J., www.omg.org/ docs/omg/03-06-01.pdf, November20, 2006.

[3] Gustas, R. and Gustiene, P., “Pragmatic – Driven Approach for Service-Oriented Analysis and Design”, *Information Systems Engineering - from Data Analysis to Process Networks*, Idea Group Inc., 2007.

[4] Harel, D., & Rumpe, B., “Meaningful Modeling: What’s the Semantics of ‘Semantics’?”, IEEE Computer, October, 2004, pp. 64-72.

[5] Dietz J. L. G., Enterprise Ontology: Theory and Methodology, Springer, Berlin, 2006.

[6] Lankhorst, M. et al., Enterprise *Architecture at Work*, Springer, Berlin, 2005.

[7] Hull, R., Christophides, V., & Su, J., “E-services: A look Behind the Curtain”, *ACM PODS*, San Diego, CA, 2003.

[8] Dori, D., Object-Process Methodology: A Holistic System Paradigm, Springer, 2002, Berlin.

[9] Gustas, R. and Gustiene, P., “Service-Oriented Foundation and Analysis Patterns for Conceptual Modelling of Information Systems”, International Conference on Information System Development, Springer, 2007.

[10] Hoffer, J. A., George, J. F. and Valacich J.S., Modern System Analysis and Design, Pearson Prentice Hall, New Jersey, 2004.