

Towards a UML Profile for Service-Oriented Architectures¹

Reiko Heckel, Marc Lohmann, and Sebastian Thöne

*Faculty of Computer Science, Electrical Engineering and Mathematics
University of Paderborn, Germany*

1 Introduction

Application development naturally starts with functional requirements given by the business that shall be supported. The model capturing these requirements is then refined taking more and more aspects of the technology and target platform into account. To support and automate in particular the later steps of this process is the objective behind the OMGs model driven architecture (MDA), summarized in Fig. 1.

The development of distributed applications is supported by platforms like Web services [2] or Jini [3]. The aspects of the technology shared by these platforms, like the roles of *service providers*, *requesters*, and *registries* as well as their *publish*, *find*, and *bind* operations, are conceptualized in the service-oriented architectural style (SOA). To support the model-driven development of SOA applications at the platform-independent Level 2 of Fig. 1, a notation has to be defined to support the representation of SOA concepts. Moreover, these concepts have to be given a semantics which allows to interpret the behavior of applications modeled in that style.

In this paper, we fulfill the demand for a suitable syntax for this domain by sketching a UML profile for SOA by means of an example. Once the profile is properly defined, its semantics can be given in terms of a graph transformation

Email addresses: reiko@upb.de (Reiko Heckel), mlohmann@upb.de (Marc Lohmann), seb@upb.de (Sebastian Thöne).

¹ Research partially supported through the European Research Training Network *SegraVis* (on *Syntactic and Semantic Integration of Visual Modeling Techniques*) and the International Graduate School *Dynamic Intelligent Systems* at the University of Paderborn

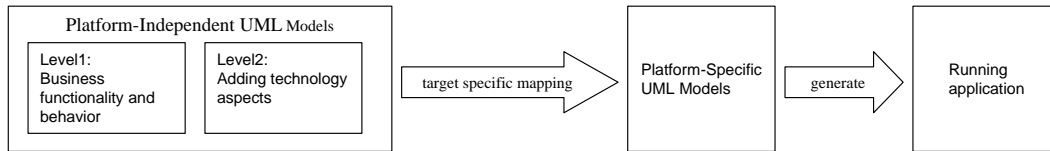


Fig. 1. OMG's outline of MDA

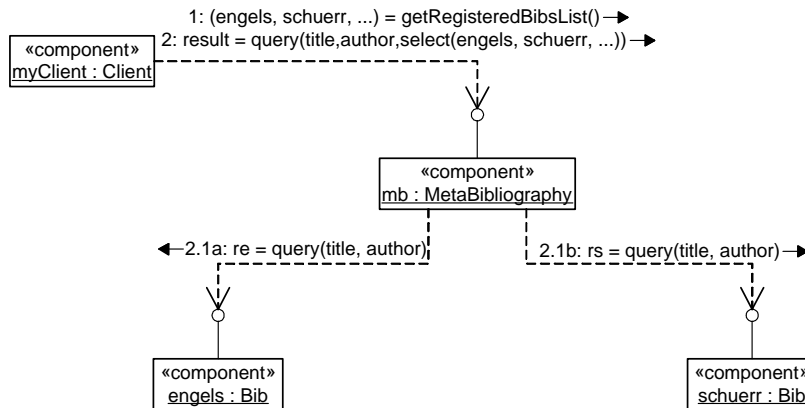


Fig. 2. Collaboration diagram modeling execution of a query

model for the SoA style introduced in [1], expressed here for conciseness using collaboration diagram notation.

The paper is organized as follows: Below, we introduce the platform-independent business architecture of our sample application (cf. Level 1 of Fig. 1), a meta-bibliography service integrating an open number of local bibliographies. Then, in Section 3, we sketch the SoA profile and its semantics given by the behavior of the SoA operations. Following the MDA outline, Section 4 provides the SOA model of the publication service based on the profile (cf. Level 2 of Fig. 1), and Section 5 concludes the paper.

2 Business Architecture

In this section, we introduce the running example of the paper. The objective behind the meta-bibliography is to provide an integrated bibliography service for the *SegraVis* network (see www.segravis.org) by loose coupling of individual solutions that are already present at the 12 member sites. Each site has to wrap its local bibliography data source (a relational data base, BibTeX file, etc.) inside a Web service interface. The meta-bibliography is accessed over the web by a browser and an incoming request is forwarded to all known bibliography services. The application is also used as a running example for exercises in a lecture on *Model-Based Development of Web Applications* at the University of Paderborn (see www.upb.de/cs/ag-engels/).

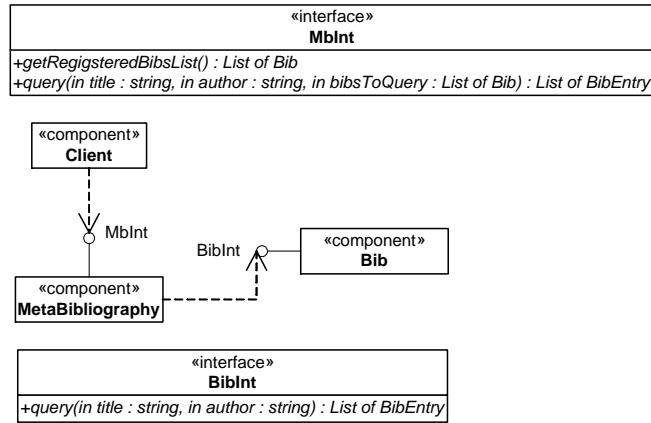


Fig. 3. Business architecture derived from collaboration diagram

Figure 2 shows a collaboration diagram modeling a sample scenario for executing a query on the meta bibliography. Before clients can send a query, they have to select those local bibliographies where the query should be executed. Therefore, clients retrieve a list of registered bibliographies, i.e., all local bibliographies that are registered with the meta bibliography. Then, the client selects the bibliographies of the research groups of Gregor Engels in Paderborn and Andy Schürr in Darmstadt. The search string that is entered afterward is forwarded concurrently to all selected bibliographies. From the scenario, we can derive the UML component diagram in Fig. 3 to identify the (types of) components and interfaces used in the application.

3 Service-Oriented Architectures in UML

The roles and artifacts of the service-oriented style are represented as follows (see upper left of Fig. 4). Services are components with stereotype `<<service>>`. Provider and requester are interpreted as roles of components implementing and using an interface, respectively. (We ignore here the business perspective of, e.g., the provider as organization owning the service.) A service registry is a service implementing a special interface (marked by “?” as shorthand for a stereotype `<<registry>>`) where service descriptions can be published by providers and queried by requesters.

A service description is represented as a UML package marked by a stereotype `<<desc>>`, see upper right of Fig. 4. The models contained in this package specify the service and its interface(s) as indicated by the `<<specifies>>` dependencies. Requesters store their requirements for a service in packages marked by a stereotype `<<req>>`. We assume a conceptual relation `<<satisfies>>` to represent the fact that all required properties are guaranteed by a description. Further, we have to represent the fact that a component (requester or registry)

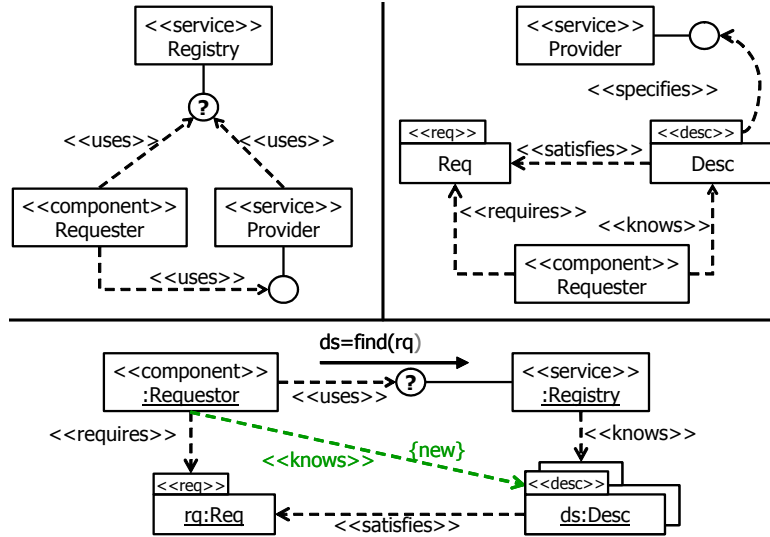


Fig. 4. Syntax of UML for SOA and semantics of the *find* operation

has access to a certain description by a <<knows>> relation.

The SoA style provides three distinguished operations, to *publish* and *find* service descriptions, and to *bind* to a service. In the bottom of Fig. 4 a collaboration diagram is provided to specify the semantics of the *find* operation, by which a requester obtains all service description known to a registry that satisfy the requester’s requirements.

Similarly, there are collaboration diagrams defining the operational semantics of the other two operations. These collaboration diagrams can be further refined into graph transformation rules, as expressed in [1]. The operational semantics covers only the communication and architectural reconfiguration of components and services. Other aspects, like the contents of service descriptions, including operation signatures, data models, etc. are left open to further refinement in the platform-specific model.

4 SOA Model

Based on the notation given in the previous section, we can specialize our business model to the SoA style. This requires to add to the initial configuration a registry which holds the list of all local bibliographies participating in the network together with appropriate service descriptions as well as requirements. Moreover, the operations of the original application scenario have to be interleaved with *find* and *bind* operations. One possible such specialization is shown in Fig. 5.

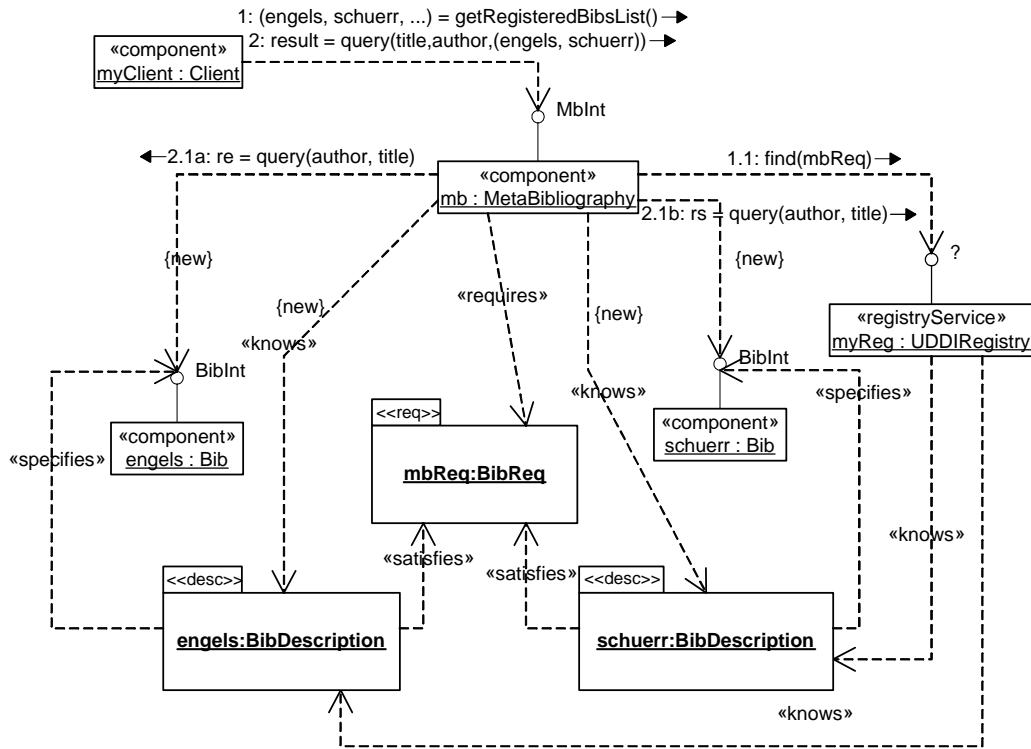


Fig. 5. SoA-specific collaboration diagram

5 Conclusions

The UML profile for service-oriented architectures sketched in this paper requires further refinement, in particular, if platform-specific details, e.g., of XML-based Web services shall be added. This aspect is, however, separated from the purely architectural view of the notation so far, as it is concerned with the contents of service descriptions and queries.

Still at the architectural level, other types of diagrams are involved, like class diagrams with interfaces to define signatures and data types of operations and sequence diagrams as alternative presentation of interactions. They are all left out here because of space limitations.

Based on the formal semantics of the architectural style, which is expressed in [1] by means of a graph transformation system, we are also planning support for validation of SoA models by simulation using a graph transformation tool like Fujaba (see www.fujaba.de) or by model checking using one of the emerging approaches to model checking graph grammars [5,4].

References

- [1] L. Baresi, R. Heckel, S. Thöne, and D. Varró. Modeling and validation of service-oriented architectures: Application vs. style. In *Proc. ESEC/FSE 2003*, Helsinki, Finland, September 2003. To appear.
- [2] M. Champion, C. Ferris, E. Newcomer, and D. Orchard. *Web Service Architecture, W3C Working Draft*, 2002. <http://www.w3.org/TR/2002/WD-ws-arch-20021114/>.
- [3] Sun microsystems. *Jini Architectural Overview - Technical White Paper*, 1999.
- [4] A. Rensink. Model checking graph grammars. In *Proc. Workshop on Automated Verification of Critical Systems (AVoCS 2003)*, Southampton (UK), April 2003.
- [5] D. Varró. Towards symbolic analysis of visual modelling languages. In Paolo Bottoni and Mark Minas, editors, *Proc. GT-VMT 2002: International Workshop on Graph Transformation and Visual Modelling Techniques*, volume 72 of *ENTCS*, pages 57–70, Barcelona, Spain, October 11-12 2002. Elsevier.