UML as Semantic Configuration Web Service Description Language

Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Zanker

Institut für Wirtschaftsinformatik und Anwendungssysteme, Produktionsinformatik,
Universitätsstrasse 65-67, A-9020 Klagenfurt, Austria,
email: {felfernig}@ifit.uni-klu.ac.at.

Abstract. Web services are an upcoming approach to effectively support application integration in Web-based environments. UDDI, WSDL, and SOAP as state-of-the-art technologies in this area do not support the idea of intelligent application integration - an idea, which is included in the long term vision for the Model Driven Architecture (MDA). Using the application scenario of distributed Virtual Private Network configuration we motivate the idea of Semantic Configuration Web Services which enable intelligent integration of configuration systems. We show how semantic service descriptions can be supported by UML/OCL.

1 Introduction

The Model Driven Architecture (MDA) [3] is the result of ongoing standardization efforts of the Object Management Group (OMG[1]). The goal is to provide an integrated architecture supporting system interoperability on the application level based on the sharing of metadata. The overall strategy for sharing and understanding metadata consists of the automated development, publishing, management, and interpretation of models [3]. The long term vision of MDA includes applications capable of automatic discovering capabilities/properties of other applications. This is a step from metadata based application integration towards a knowledge-based application integration. In the following we sketch how automatic services discovery can be supported by using UML/OCL [4,6] as basic representation languages for the semantic description of Web services. We show how such descriptions can be integrated into the Web services architecture proposed by [2] and give a simple example how semantic service descriptions can be used to support the intelligent integration of configuration systems.

The idea behind the concept of Web services is to alleviate the task of application integration by providing a basic infrastructure which is based on state-of-the-art Web technologies. Examples for Web services are hotel booking services, translation services, payment services, or configuration services - the latter can be interpreted as a complex Web service. Basically, the application of Web services is based on three aspects (see Figure 1).

[1] For further details see *www.omg.org*.

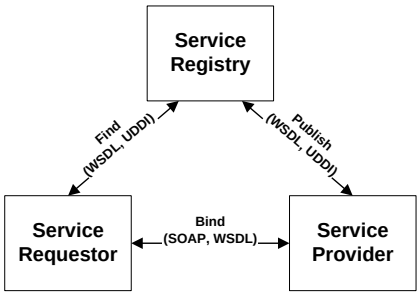

Fig. 1. IBM Web Services Model [2]

1. Service publishing: services are published by a service provider in a service registry which provides the basic infrastructure for service registration on the one hand and service identification on the other.
2. Service identification: using the service registry, an application developer is enabled to retrieve the needed services.
3. Service execution: having retrieved the needed services, the application developer (service requestor) can directly establish connections to service providers and utilize the provided services.

How this basic Web Services Model can be applied to the task of intelligent application integration of product (service) configurators will be shown in the following sections.

2 Integration scenario for configuration systems

Product (service) configuration can be seen as a special kind of design activity [5], where the configured product is built of a predefined set of component types and attributes, which can be composed conform to a set of corresponding constraints. Example applications can be found in various domains such as the telecommunication industry, automotive industry, financial services, or furniture industry. Configuration systems have proven their applicability in these domains. However, today's economy is exhibiting a growing trend towards highly specialized solutions providers cooperatively offering configurable products and services. This requires the extension of state-of-the-art configuration technology with knowledge sharing and distributed problem solving capabilities.

The basic scenario for the integration of configuration systems is shown in Figure 2. An integration platform for configurable products and services supports the provision of integrated solutions to customers. An example for this integration scenario is the distributed configuration of virtual private networks (VPNs). A VPN is an extension of an enterprise's private network which provides network services based on public network infrastructures such as the Internet. A secure communication environment is provided for defined communities of

interest. The major advantage of such networks is that no expensive maintenance for company WANs is necessary. The infrastructure for VPNs is provided by specialized solution providers who offer different sub-components (services) for the VPN (e.g. telephony services, leased lines, firewalls, or computers). Sub-components provided by specialised solution providers are integrated by resellers (integrated solution providers), who in the following offer complete VPN solutions to their customers. In many cases sub-components of specialised solution providers are configurable, i.e. the integration of such a set of components can be interpreted as a distributed configuration task. Consequently, distributed configuration problem solving capabilities are needed in order to assure the consistency of an offered solution. The result of a distributed configuration problem solving process is an integrated solution (customer-specific VPN), where suppliers guarantee to provide configurations which can be integrated into the overall solution.

The precondition for triggering such a distributed configuration problem solving process is the identification of those configurators which are capable of providing the sub-components needed for an integrated solution. This identification is a kind of *matchmaking* process which is sketched in Section 4. The software components needed to support distributed configuration problem solving were developed within the scope of IST project CAWICOMS[2].

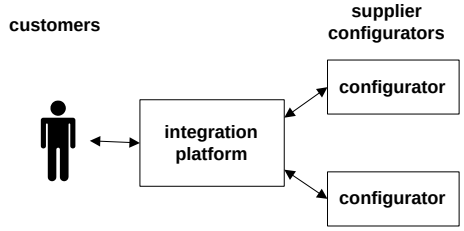

Fig. 2. Integration platform for configurable products and services

3 Semantic Configuration Web Services

The inter-relationship between the Web Service Architecture depicted in Figure 1 and Semantic Configuration Web Services is shown in Figure 3.

The *Service Provider* (supplier of sub-components) introduces functional descriptions (based on UML/OCL) of provided products and services into the *Service Registry*. With respect to state-of-the-art standards available for the definition of Web services, no mechanisms are available to support such a (semantic) functional description of product (service) capabilities. In order to support a

[2] CAWICOMS is the acronym for Customer-Adaptive Web Interface for the Configuration of products and services with Multiple Suppliers (EU-funded project IST-1999-10688).

standard description of configurator capabilities we propose the application of UML/OCL (see Section 4).

The *Service Requestor* (integrated solution provider) receives a set of functional requirements from the customer and forwards those requirements to the *Service Registry*. Provided with such requirements and the service descriptions of service providers, the service registry can start a *matchmaking* process with the goal of identifying those providers which are capable of providing subcomponents needed in order to fulfill the given customer requirements. Having identified a set of potential suppliers, a distributed configuration process is triggered. A simple example will be given in the following in order to illustrate the application of UML/OCL in this context.

Following this approach the identification of a configuration Web service is based on the matching between available semantically annotated configuration services[3], and a given set of customer requirements. In addition to mechanisms supporting a semantic service description, *model generation* mechanisms are required which support the mentioned matchmaking, i.e. check the consistency between the given customer requirements and the existing semantic service descriptions.

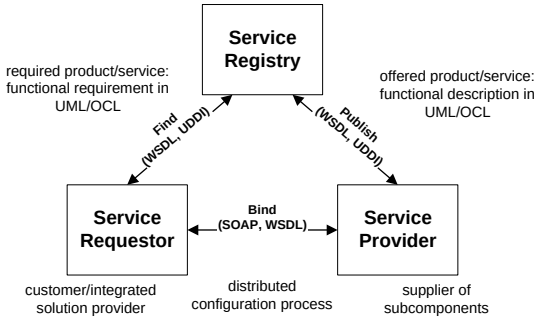

Fig. 3. Configuration Web Services

Within CAWICOMS a hierarchical approach for configuration knowledge representation is applied (see Figure 4). Using UML/OCL as knowledge representation language one can define a configuration specific ontology (profile) which includes those modeling concepts useful for designing configuration models (e.g. component, function, or resource). Using those concepts (from the *generic level*) one can define a product-specific ontology (e.g. a VPN ontology). In order to define the capabilities of their products, suppliers now use the product-specific ontology (*product domain level*); additional restrictions are defined on the set of provided products, e.g. supplier ABT only provides AccessPoints with a bandwidth between 256 and 2048 (*supplier-specific product domain level*).

[3] The annotation is based on UML/OCL.

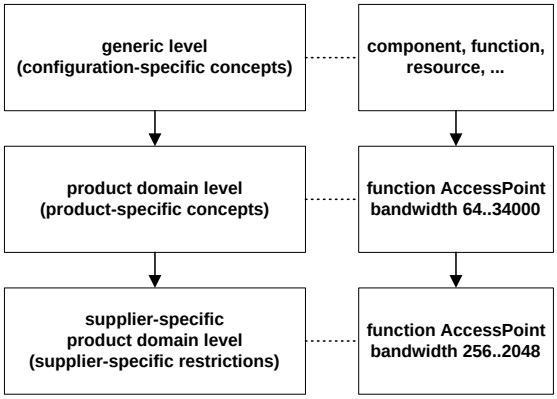

Fig. 4. Knowledge representation: hierarchical approach

4 Example

All the levels (generic, product domain, supplier-specific product domain level) shown in Figure 4 can be represented using UML/OCL. Let us assume that there exists a set of computer supplier entries in the service registry. These providers offer additionally needed hardware, e.g. for the extension of the currently available VPN infrastructure. A part of a computer product domain ontology (product domain level) could be represented as follows[4].

```
package computer
        class HDUnit extends generic.component[5]
                capacity: Integer
        end;
        class IDEUnit extends computer.HDUnit
                inv
                "capacity IDEUnit"
                capacity = 25 or capacity = 30 or capacity = 40
        end;
        ...
end
```

Using such product domain ontologies, suppliers can impose additional restrictions concerning their products (supplier-specific product domain level). Such a restriction for our computer domain ontology could look like as follows: supplier "A" only provides IDEUnits with capacity equal 30; supplier "B" only provides IDEUnits with capacity greater than 25.

[4] We use MML (Meta-Modelling language[1]) for ontology representation.
[5] Concept defined on the *generic level*.

```
package computer_supplierA extends computer
        class IDEUnitA extends computer.IDEUnit
                inv
                "capacity IDEUnitA"
                capacity = 30
        end;
end

package computer_supplierB extends computer
        class IDEUnitB extends computer.IDEUnit
                inv
                "capacity IDEUnitB"
                capacity > 25
        end;
end
```

If we assume that "IDEUnit with capacity equal 30" is part of a given set of customer requirements, then both suppliers (A, B) are selected as potential contributors. The identified set of suppliers can now be contacted within the scope of a distributed configuration problem solving process, where one of the potential suppliers is selected to provide the needed additional hardware components.

5 Conclusions

In this paper we have sketched how UML/OCL can be employed as standard languages for Semantic Configuration Web Service representation. The major precondition for allowing a semantic service description is the inclusion of a corresponding description language (our proposal is to use UML/OCL) into state-of-the-art Web service standards. One of the major tasks in this context is to provide model generation mechanisms for UML/OCL in order to efficiently support matchmaking between requirements (from service requestors) and corresponding service descriptions (from service providers).

References

1. T. Clark, E. Evans, St. Kent, and P. Sammut. The MMF Approach to Engineering Object-Oriented Design Languages. In *Workshop on Language Descriptions, Tools and Applications*, 2001.
2. H. Kreger. Web Services Conceptual Architecture (WSCA 1.0). 2001.
3. J.D. Poole. Model-Driven Architecture: Vision, Standards And Emerging Technologies. In *Workshop on Metamodeling and Adaptive Object Models, ECOOP 2001*, 2001.
4. J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual.* Addison-Wesley, 1998.

5. D. Sabin and R. Weigel. Product Configuration Frameworks - A Survey. In B. Faltings and E. Freuder, editors, *IEEE Intelligent Systems, Special Issue on Configuration*, volume 13,4, pages 50–58. IEEE, 1998.
6. J. Warmer and A. Kleppe. *The Object Constraint Language - Precise Modeling with UML*. Addison Wesley Object Technology Series, 1999.