In Proc. of the 5th International Workshop on Engineering Service-Oriented Applications (WESOA'09)

- EXTENDED TECHNICAL REPORT -

Service Customization by Variability Modeling

Michael Stollberg and Marcel Muth

SAP Research
CEC Dresden, Germany
{michael.stollberg},{marcel.muth}@sap.com

Abstract. The establishment of service orientation in industry determines the need for efficient engineering technologies that properly support the whole life cycle of service provision and consumption. One challenge is adequate support for service consumers for employing complex services in their individual application context, which becomes particularly important for large-scale enterprise technologies where generic services are designed for reuse in several business scenarios. This paper presents an approach for service customization by model-driven variability management. The variable aspects of the services are explicitly described on the basis of a metamodel. Upon this, service consumers can easily create personalized service variants that properly suit their specific context while the consistency for service invocation is maintained.

1 Introduction

In the last years, service orientation has become the dominating design principle for modern ICT technologies in industry as well as in the public sector. The aim is to exploit the enormous potential of services for enhancing the interoperability among systems and the reuse of implementations. In consequence, a steadily growing number of available services and service-based applications can be observed. The design, development, usage, and management of such solutions requires sophisticated engineering technologies that support the life cycles of service provision and service consumption in an efficient and integrated manner.

This is subject to the emerging discipline of service engineering: Numerous efforts in academia and industry have developed a wealth of techniques, methodologies, and tool support for this. However, existing solutions focus mainly on support for service providers, i.e. for the design, development, publication, and management of services. The consumption side – i.e. the support for service consumers for finding suitable services and integrating them into the specific target application – is often neglected. Existing technology support for this is mostly limited to low-level technical details, leaving the major part of the analysis and integration task for actually consuming services to manual inspection.

The limitations become obvious when considering the consumption of more complex services that commonly occur in real-world business applications. For example, consider the Enterprise Services that form the basis of SAP's modern service-based enterprise technology. These are designed in a generic manner and cover several usage options, therewith becoming reusable in various business scenarios. On the other hand, their interfaces and usage conditions are considerably complex. Typically, a customer merely requires a subset or a specific flavor of the provided features. Hence, the Enterprise Services need to be configured and integrated in order to properly fit the customer's needs. This is a non-trivial task that requires both technical knowledge and business expertise. Due to the limited tool support, the customization requires massive human involvement and thus becomes a highly cost-intensive and error-prone task.

To overcome this, we present an approach for service customization by the creation of simplified variants that only expose those features of the service that are relevant for the usage scenario of an individual consumer. These are defined on the basis of variability specification models that explicitly describe the usage conditions and constraints of the original service. To support model-driven engineering, we define a metamodel for describing the variable aspects of services, i.e. the mandatory and optional operations, properties of message types as well as their dependencies. We define an engineering process for service customization, provide tools that firstly support the creation of service variability models, and secondly for service variant creation via intuitive user interfaces. Finally, a technical interface is generated from the service variant model. This usually is significantly less complex than the interface of the original service, while the consistency for the correct invocation is maintained. Furthermore, the simplified service description can serve as the basis for other service engineering techniques, e.g. for mash-up techniques that are mostly limited to services of limited complexity.

The paper is structured as follows. At first, Section 2 provides a concise overview of the approach and defines the engineering process for service customization. Section 3 specifies the metamodel for service variability modeling, and Section 4 defines the tool-supported procedures for service customization. Section 5 illustrates the techniques and tools for customizing an Enterprise Service. Section 6 positions our approach within related work, and finally Section 7 concludes the paper and outlines future work.

2 Overview

The following provides an overview of the approach for service customization. We define the central artifacts and roles involved in the process of providing and consuming customized services, outline the technical solution that is presented in detail in this paper, and motivate the need for such technologies.

Figure 1 provides a comprehensive overview, identifying the involved roles, phases, and relevant artifacts for the engineering process of providing, preparing, and consuming customized services. We distinguish three roles: the *Service Provider* develops and publishes services, the *Domain Expert* prepares them for customization by defining the variability specification model along with pre-configurations for respective user groups, and the *Service Consumer* customizes and personalizes the service in order to fit it into the specific application context.

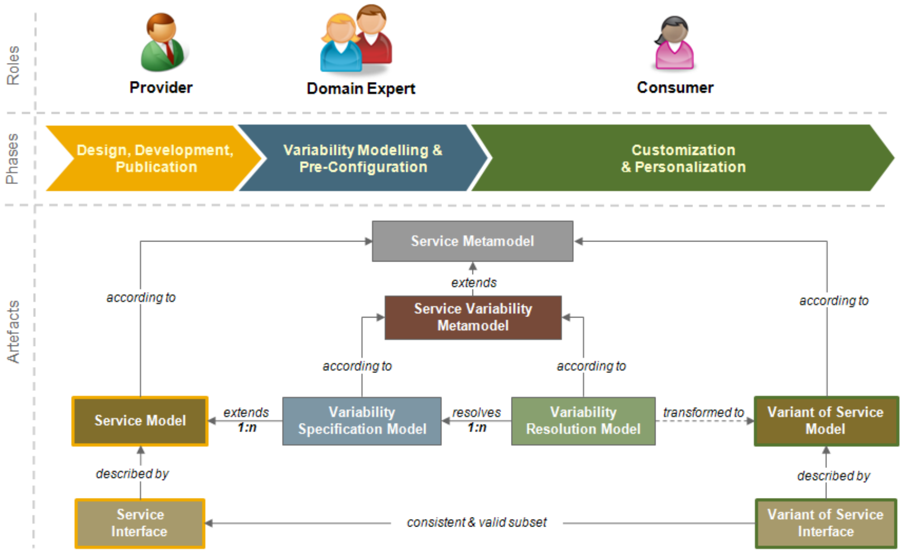

Roles
Provider
Domain Expert
Consumer
Phases
Design, Development, Publication
Variability Modelling & Pre-Configuration
Customization & Personalization
Artefacts
Service Metamodel
extends
according to
according to
Service Variability Metamodel
according to
according to
Service Model
extends 1:n
Variability Specification Model
resolves 1:n
Variability Resolution Model
transformed to
Variant of Service Model
described by
described by
Service Interface
consistent & valid subset
Variant of Service Interface

Fig. 1. Service Customization – Roles, Phases, Artifacts

Note that these are abstractions from the several sub-roles that can be found in real-world service engineering processes.

In the first phase, the Service Provider develops a service and publishes it in a repository. In the context of model-driven engineering, the service interface that defines the operations, messages, and endpoints is described by a Service Model on the basis of a metamodel (e.g. a WSDL metamodel [6] or SoaML [7]). In the second phase, the domain expert prepares the service for customization. For this, he creates a *Variability Specification Model* that describes the variable aspects of the service (i.e. the mandatory and optional operations, messages, and message types as well as the dependencies). This is defined in accordance with the *Service Variability Metamodel* that defines the necessary constructs for modeling the variability of services. The Domain Expert might define multiple variability specification models for a service where each one is pre-configured for a particular application scenario (e.g. for specific industry sectors or geographical usage contexts). In the third phase, the Service Consumer adapts the service to the individual consumption context by defining a *Service Resolution Model*. For this, the variable aspects defined in the Variability Specification Model are resolved by selecting the desired features and defining concrete values for the parameters that are not changed dynamically during the invocation. Finally, a Service Interface for the variant of the Service Model is generated: this only contains the selected features and represents a valid subset of the original service interface, while the explicit variability modeling and the validation of the usage conditions throughout the customization process ensure the correct invocation of the service.

The technical solution for supporting service customization presented in this paper encompasses the specification of the Service Variability Metamodel and tools for supporting the creation of Variability Specification Models by Domain Experts as well as for Variability Resolution by Service Consumers. The overall idea for enabling service customization by variability modeling is adopted from works on variability management in Software Product Line Engineering (SPLE, e.g. [2,15]), which however deal with different elements and thus employ different models and techniques (see Section 6 for a more detailed discussion). In order to ensure the efficiency of the service engineering process, we consider a model-driven approach where the service and variability models are defined on the architecture level (i.e. on the PIM level in terms of MDA [8]). Our prototype implementation works on SoaML [7] a metamodel for describing services; however, our Service Variability Metamodel is defined orthogonally to the base model, so that it can also be applied to other service metamodels.

Before presenting the technical solution in detail, let us discuss the motivation and business relevance of such a service customization technology. As outlined above, the need arises from the growing complexity of services, which particularly arises in the context of business applications. For illustration, consider a business service for creating and managing sales orders. A sales order is a relatively complex data object (consider the standard definitions in RosettaNet or ebXML), and, furthermore, its detailed structure is defined differently within the standards for different industries. In order to be reusable in various applications within several industries, a general purpose business service needs to support all options and specific features that are relevant for the targeted usage industries. In consequence, its interface becomes very complex regarding the number operations, the size of input- and output objects, and the conditions and constraints for proper and consistent consumption. A specific consumer typically only requires a subset of the provided features for his individual sales order management. However, due to the complexity, the general purpose service is not easy to understand, and its configuration for the individual needs of the consumer is a time consuming task that usually requires external support.

Our approach enables a step-wise reduction of the complexity and improves the technology support for customization. At first, a Domain Expert can define variants that are pre-configured for specific user groups, e.g. one variant for the automotive industry, one for steel production, and another one for the telecommunication sector. Furthermore, the detailed usage conditions for each variant are described explicitly in terms of a variability specification model. On this basis, a consumer can then define a personalized variant by selecting the desired features. The model-driven approach facilitates the abstraction from technical details as well as the provision of intuitive graphical user interfaces for modeling support, and the variability models ensure that the selections by the consumer are compliant with the usage conditions of the service. The generated technical interface for the consumer's variant is naturally significantly less complex than the one of the original sales-order service while it adheres to its usage conditions, so that a correct and consistent invocation is ensured.

3 Service Variability Metamodel

This section presents the metamodel for describing the variability of services. First, we introcuce the main elements and then the constructs for variability modeling on different levels of service descriptions.

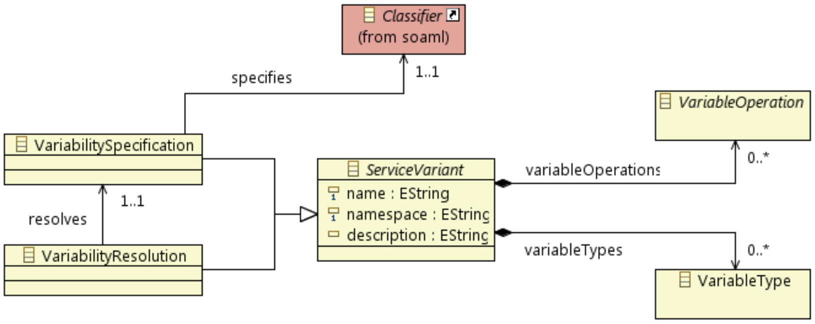


Fig. 2. Service Variability Metamodel – Main Elements

Shown at Figure 2, a *VariabilitySpecification* and *VariabilityResolution* serve as containers for the variability descriptions of the variable artefacts (operations and datatypes) of a specific service (which here is described by a SoaML Classifier [7]). The *VariabilitySpecification* contains the definition of all the variable aspects which are resolved by instances of a *ResolutionElement*. The variability description of a service are modeled with the help of four mechanisms shown at Figure 3:

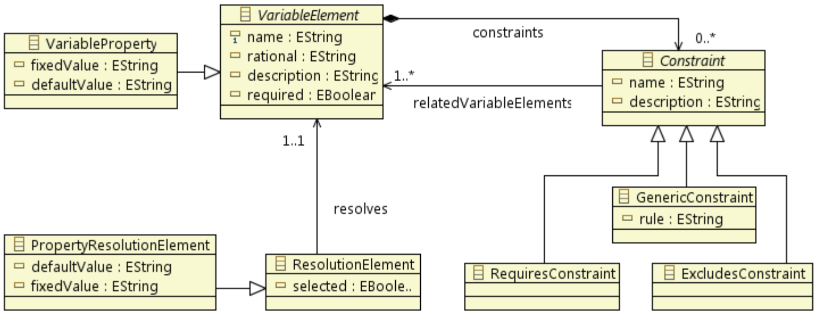


Fig. 3. Service Variability Metamodel – Variability Mechanisms

1. Declaration of mandatory and optional elements, modeled by the boolean `required` property of *VariableElement* which serves as the the superclass for variability modeling on different levels of service descriptions.
2. Definition of dependencies among elements, modelled by the *Constraint* class with direct support for defining excluding and requiring constraints
3. Selection of desired features for a specific application context, modeled by the boolean `selected` property of *ResolutionElement*, and
4. Definition of fixed values that are changed within the application scenario or default values that are used for invocation when no concrete value is provided for, which is only applicable for *VariableProperty* and *PropertyResolutionElement*.

3.1 Operation Level

We now turn towards the actual variability modeling for the various aspects of services. The first level is concerned with operations that define how to consume a service by the exchange of messages.

Figure 4 shows the metamodel elements for this. A *VariableOperation* inherits the mechanisms explained above for defining mandatory operations and their dependencies as well as for selecting the desired ones. The operations defined the base model (i.e. original service description) are bound to a *SimpleVariableOperation*. In addition, *ComplexVariableOperation* enables the grouping of related operations whith the boolean `multiple` property for modeling exclusiveness (`true` if only one of the related operations can be used).

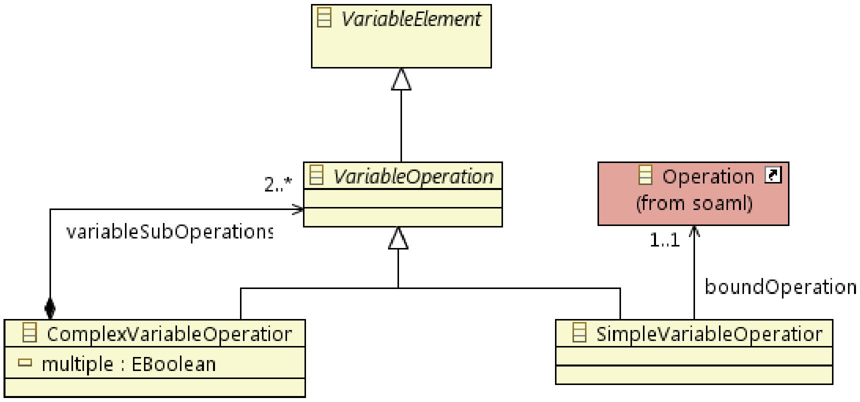


Fig. 4. Service Variability Metamodel – Operation Level Elements

3.2 Data Level

The second level of variability modeling is concerned with the message types, i.e. the data structures used within the messages. Figure 5 shows the metamodel elements for this. A *VariableType* is bound to a data type from the base model. In order to enable precise and flexible modeling, the actual variability is defined on the properties of the types by *VariableProperty*, using the mechanisms introduced above.

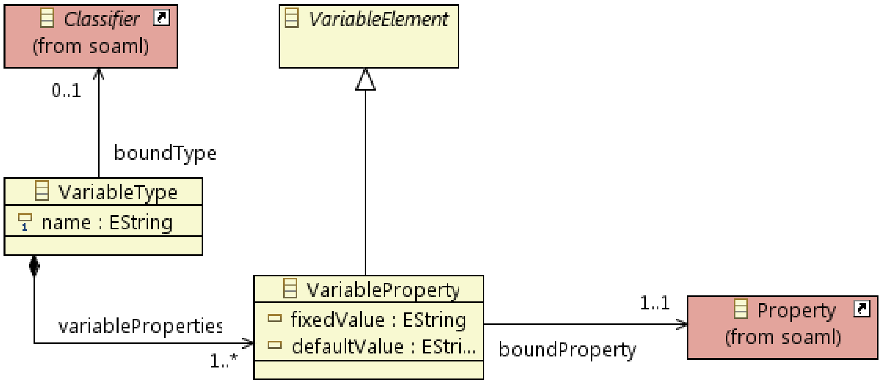


Fig. 5. Service Variability Metamodel – Data Level Elements

4 Service Customization Procedures and Tool Support

This section explains the techniques for service customization that work on the metamodel presented above, refining the overall engineering process introduced in Section 2. We here focus on the methodological aspects, while presenting the tooling support in the context of the illustrative example below in Section 5.

4.1 Service Variability Specification

As outlined above, the first step for enabling the customization of services is the creation of a variability specification model. This is performed by a domain expert in the second phase of the overall engineering process (*cf.* Figure 1 above). For this, the domain expert analyses the description of the base service, and creates a variability specifcation along with pre-configurations for the foreseen application context.

Figure 6 shows the procedure for this, which is supported by engineering tools. In order to reduce the manual modeling effort, at first the *Extractor* creates a skeleton of the variability specification model by extracting the already

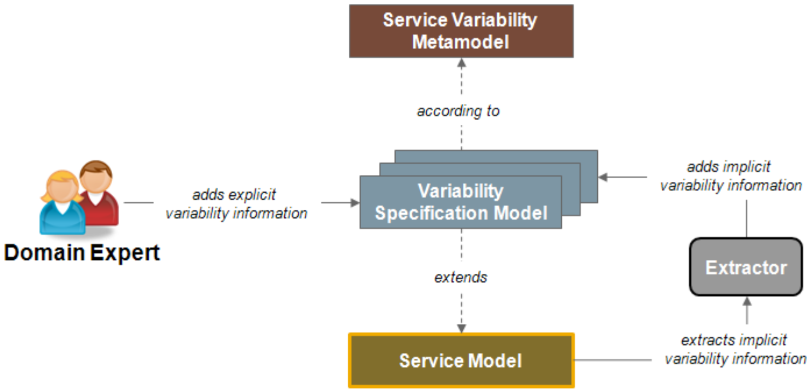

Service Variability Metamodel
according to
adds explicit variability information
Variability Specification Model
adds implicit variability information
Domain Expert
extends
Extractor
Service Model
extracts implicit variability information

Fig. 6. Variability Specification Modeling

existing variability information from the original service description. This means that the elements of the variability specification model for the operations and message types are generated. Furthermore, the basic usage conditions can be extracted; as a pre-requisite for this, sophisticated SOA governance process that are commonly employed in industry ensure that information such as mandatory operations and data fields is given in the service description [4]. Then, the domain expert can refine the skeleton by adding further variability information, using the mechanisms supported by our metamodel. Note that there can be several variability specification models for a service, where each one is pre-configured for a specific application scenario. The diverse variability specification models may define different usage conditions and distinct pre-selected sets of mandatory and optional fields, as e.g. one industry standard requires certain fields which are irrelevant in another standard.

4.2 Service Variability Resolution

In the last phase, the consumer creates a service variant that suits the individual application context. For this, he chooses a suitable variability specification model of the service that has been previously prepared by a domain expert, resolve this by selecting the desired features and setting predefined values. From this the consumer generates a variant of the original service model which describes the interface for invoking the service.

Figure 7 shows the detailed procedure for this, which is supported by a graphical engineering tool that we shall present below. At first, the chosen variability specification model along with the bound orginal service model is imported. Then, the variability is resolved by selecting the desired features and defining default and fixed values for type properties and message parameters that will remain unchanged during the actual service invocation. The tool ensures that the user inputs comply with the conditions defined in the variability specification model. Finally, a variant of the original service model is generated from the

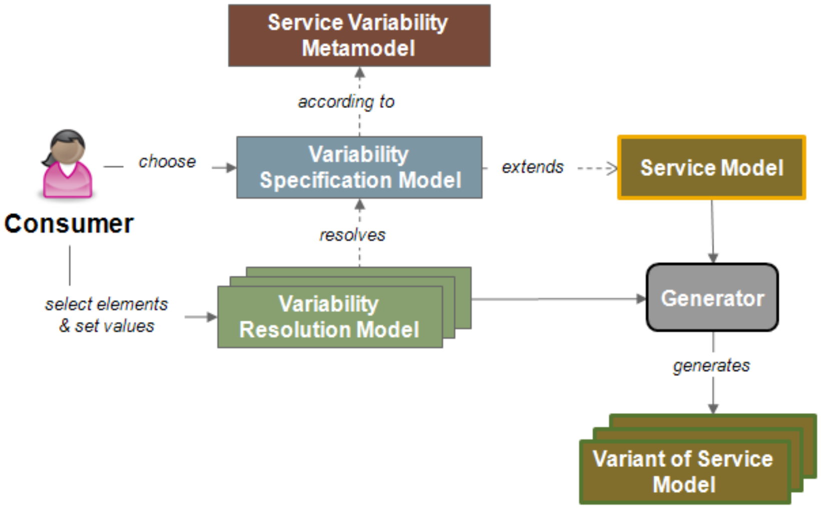

Service Variability Metamodel
according to
choose
Variability Specification Model
extends
Service Model
Consumer
resolves
select elements & set values
Variability Resolution Model
Generator
generates
Variant of Service Model

Fig. 7. Service Variability Resolution

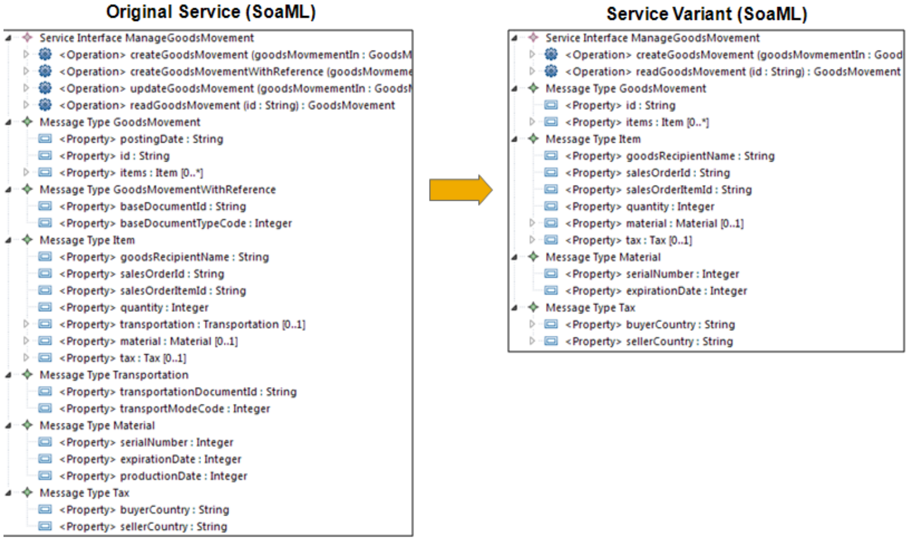
resolution model. This is described in terms of a conventional service model, and thus can be used invoking the service. The explicit variability modeling and the validation of the usage conditions throughout the customization process ensure that the generated service variant is valid for properly and correctly invoking the original service.

5 Illustrative Example

The following illustrates the modeling techniques and procedures introduced above within an example for customizing an existing service for managing goods movement. We first explain the scenario setting, and then demonstrate the definition of a variability specification model as well as the creation of a service variant within our prototype which is implemented as a set of plugins based on the Eclipse Modeling Framework (EMF, see `www.eclipse.org/modeling/emf/`).

5.1 Customizing the Goods Movement Enterprise Service

The example is based on the Enterprise Service "Goods Movement" that provides basic business facilities for managing the movement of goods and is publicly available via the SAP Developer Network (`www.sdn.sap.com`). It offers four operations: two for creating goods movement objects (one with references to related documents like purchase order, in- and outbound delivery, and one without), and one operation each for reading and updating goods movement objects. The message types are defined by business objects for goods movement along with standard objects like item, tax, transportation, and material.



Original Service (SoaML)

```
Service Interface ManageGoodsMovement
    <Operation> createGoodsMovement (goodsMovmementIn : GoodsM
    <Operation> createGoodsMovementWithReference (goodsMovmeme
    <Operation> updateGoodsMovement (goodsMovmementIn : GoodsM
    <Operation> readGoodsMovement (id : String) : GoodsMovement
Message Type GoodsMovement
    <Property> postingDate : String
    <Property> id : String
    <Property> items : Item [0..*]
Message Type GoodsMovementWithReference
    <Property> baseDocumentId : String
    <Property> baseDocumentTypeCode : Integer
Message Type Item
    <Property> goodsRecipientName : String
    <Property> salesOrderId : String
    <Property> salesOrderItemId : String
    <Property> quantity : Integer
    <Property> transportation : Transportation [0..1]
    <Property> material : Material [0..1]
    <Property> tax : Tax [0..1]
Message Type Transportation
    <Property> transportationDocumentId : String
    <Property> transportModeCode : Integer
Message Type Material
    <Property> serialNumber : Integer
    <Property> expirationDate : Integer
    <Property> productionDate : Integer
Message Type Tax
    <Property> buyerCountry : String
    <Property> sellerCountry : String
```

Service Variant (SoaML)

```
Service Interface ManageGoodsMovement
    <Operation> createGoodsMovement (goodsMovmementIn : Good
    <Operation> readGoodsMovement (id : String) : GoodsMovement
Message Type GoodsMovement
    <Property> id : String
    <Property> items : Item [0..*]
Message Type Item
    <Property> goodsRecipientName : String
    <Property> salesOrderId : String
    <Property> salesOrderItemId : String
    <Property> quantity : Integer
    <Property> material : Material [0..1]
    <Property> tax : Tax [0..1]
Message Type Material
    <Property> serialNumber : Integer
    <Property> expirationDate : Integer
Message Type Tax
    <Property> buyerCountry : String
    <Property> sellerCountry : String
```

Fig. 8. Goods Movement Service Scenario

In our example, we will create a variant of the service that only contains the operations and necessary message types for the simple creation and reading of goods movement objects. Figure 8 provides an overview with the original service on the left and the service variant on the right hand side, which obviously is simpler and thus easier to consume. Our prototype uses SoaML as a platform independent modeling language for the basic service descriptions [7]. Sufficient for demonstration purposes, we here work with already simplified data structures for the message types; the actual business objects used within SAP applications contain more than 100 nodes.

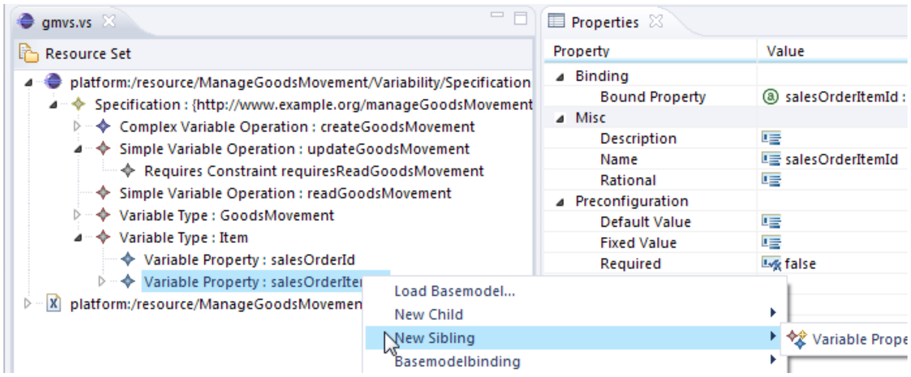
5.2 Variability Specification Modeling

As explained above, the first step in the service customization process is the creation of the variability specification model. Figure 9 shows a screenshot of our prototype tool for supporting domain experts in this task.

The tool provides an editing facility for variability specification modeling with a tree-view representation and context-sensitive editing support for adding dependent variability modeling elements, constraints, and bindings to the base model; the extractor for creating skeletons of variability specifications (*cf.* Section 4.1) is under development at the time of writing.

For our example, the domain expert defines the following explicit variability information. At first, the two operations for creating goods movement objects are grouped into a *ComplexVariableOperation*: conceptually, they present two versions of the same operation which differ in the input- and output parameters; however, they are non-exclusive, thus the `multiple`-property is set to `false`.

Secondly, the *update*-operation requires the *read*-operation, which is modeled by a `RequiresConstraint` (*cf.* Figure 2). Thirdly, the mandatory elements and their dependencies on the data level are defined. For instance, the usage of *salesOrderItem*, which is an attribute of the top-level type *item*, requires the setting of the *salesOrderId*. In addition, default values can be defined, e.g. setting country information to 'Germany' in order to pre-configure the service variant for German customers.



```
Resource Set
  platform:/resource/ManageGoodsMovement/Variability/Specification
    Specification : {http://www.example.org/manageGoodsMovement
      Complex Variable Operation : createGoodsMovement
      Simple Variable Operation : updateGoodsMovement
        Requires Constraint requiresReadGoodsMovement
      Simple Variable Operation : readGoodsMovement
      Variable Type : GoodsMovement
      Variable Type : Item
        Variable Property : salesOrderId
        Variable Property : salesOrderIte
  platform:/resource/ManageGoodsMovemen
```

Property	Value
Binding	
Bound Property	salesOrderItemId :
Misc	
Description	
Name	salesOrderItemId
Rational	
Preconfiguration	
Default Value	
Fixed Value	
Required	false

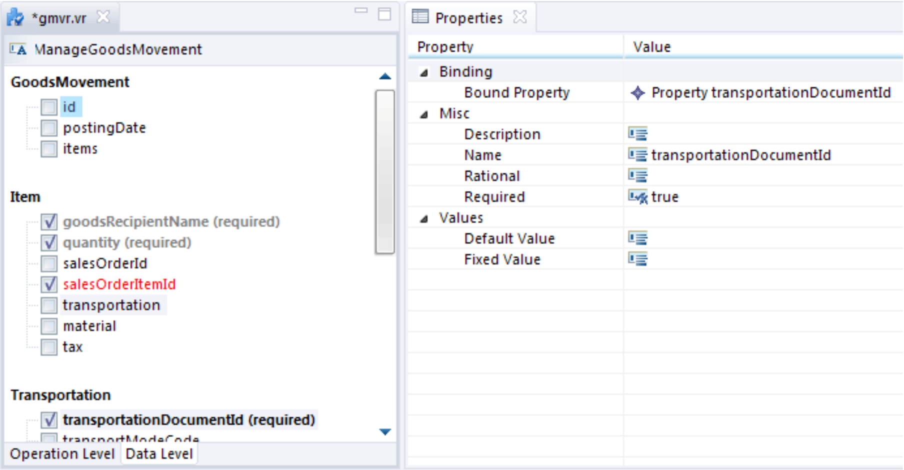
Fig. 9. Variability Specification Tool (Screenshot)

5.3 Variability Resolution Tool

We now turn to the creation of the actual service variant. As explained above, for this the consumer creates a resolution model by selecting the desired operations and data elements as well as defining default or fixed values for static data elements. From this a conventional service model is generated for the variant (*cf.* Section 4.2). In order to support consumers in this task, we provide a tool for selecting the desired features via a graphical user interface along with real-time validation of the dependencies and usage conditions defined in the variability specification model.

Figure 10 shows a screenshot of our prototype. It is organized by tabs that support variability resolution on different levels where variability can occur. Currently, the tool contains the operation and data level as the basic functional aspects covered by our metamodel; for the future, this can be extended with additional levels such as non-functional aspects or quality-of-service information.

The tool supports the variability resolution by standard metaphors for graphical user interface design [9]: checkboxes for selecting the desired elements, graphical accentuation of required elements, and colored display of constraint violations. Detailed information, which provides useful guidance for correcting are accessible via the context menu. The SoaML descriptions for the service variant


*gmvr.vr

ManageGoodsMovement

GoodsMovement

- □ id
- □ postingDate
- □ items

Item

- ☑ goodsRecipientName (required)
- ☑ quantity (required)
- □ salesOrderId
- ☑ salesOrderItemId
- □ transportation
- □ material
- □ tax

Transportation

- ☑ transportationDocumentId (required)
- □ transportModeCode

Operation Level | Data Level

Properties

Property	Value
Binding	
Bound Property	Property transportationDocumentId
Misc	
Description	
Name	transportationDocumentId
Rational	
Required	true
Values	
Default Value	
Fixed Value	

Fig. 10. Variability Resolution Tool (Screenshot)

is generated automatically, and stored in the respective folder of the EMF engineering project. With this, we provide easy-to-use tooling support for creating service variants that abstracts from technical details.

6 Related Work

This section positions our approach within related work. As analyzed in [3], a central challenge for industrial scale engineering technologies is the handling of services that deal with large and complex data structures of real-world business objects. This emphasizes the need for sophisticated customization techniques, whose efficient and scalable realization remains a grand challenge [14].

Most existing tool support for the configuration of services is limited to low-level technical aspects, such as the deployment to an execution platform or the configuration of technical parameters for invocation and runtime (e.g. [1]). Although helpful for the service engineers and thus being an integral part of modern industrial service engineering environments, these techniques can only be applied after the service has been adopted to the specific usage scenario, leaving the major part of the customization task to manual inspection.

Only few works address service customization on higher levels of abstraction. Most existing solutions merely provide methodological guidelines with limited tooling support for users (e.g. [16,5]). This appears to be inadequate for reducing the effort and error-proneness resulting from manual customization. [17] presents an approach for user-driven service customization on the basis of usage policies; however, this works with mostly informal policy descriptions that do not allow an automated validation of human customization decisions. More advanced techniques achieve significant improvements by working on formal constraint and user request descriptions (e.g. [12]); however, this requires significantly more effort for creating the additional resource descriptions than our approach.

Our approach adopts concepts of variability modeling developed in the field of model-driven software product-line engineering (SPLE, [2]). We adopt the overall approach of explicitly describing the variable aspects and resolving them by selection and parameter instantiation for a specific usage scenario, as well as the principle of orthogonal design by separating the variability specification from the original service description [15]. However, there are substantial differences: at first, SPLE is concerned with describing the variability of products, while for services we need to consider other aspects like operations, messages, and messages types as supported by our metamodel. Secondly, SPLE variability management considers a closed world where all potential variations are known a priori, and variants are defined by substitution mechanisms that insert the predefined variations into the product description [10]; this conflicts with the nature of service-orientation where we need to consider an open world with unforeseen usage scenarios. A customization technique for complete SaaS applications that employs variability modeling is presented in [13]: the provider defines an application template where the configurable aspects are described by variation points; a consumer binds these to specific services or alternative processes in order to obtain a customized solution. Although the overall approach is similar, our metamodel is significantly more expressive for describing the variability of single services. Thus, the works can be considered to be complementary: the variability modeling and management techniques presented here can be used to enhance customization techniques for service-based applications.

The customization technique supported by our approach is the simplification of complex services. Another possibility is to define extensions to the original service, e.g. by adding supplementing features as suggested in [11]. This appears to be suitable for customizing services that do not encompass support for several application scenarios but merely provide extensible core features; we plan to extend our metamodel and tools to support this technique as well.

7 Conclusions and Future Work

This paper has presented an approach for service customization by the tool-supported creation of personalized variants on the basis of variability models that explicitly describe the variable aspects and usage conditions of services.

The need for such techniques arises from the growing number of complex services that are designed for reuse in various application contexts and deal with extensive data objects, which can be particularly found in service-oriented business applications. These services become too complex to be understandable by humans, and existing service engineering tools appear to not be suitable for supporting their employment in concrete consumer contexts in an adequate and cost-efficient manner. In order to overcome this, we propose a three-phased engineering process: domain experts prepare the services that have been published by a provider for specific usage contexts by defining variability specification models; they explicitly describe variable aspects, upon which consumers can easily create personalized variants that adopt the services to the specific context of the individual application scenarios.

In order to support model-driven development, we have defined a metamodel for describing the variability modeling for services on the basis of four mechanisms: the declaration of mandatory and optional elements with their dependencies, furthermore the selection of desired features as well as the definition of default and fixed values for a particular application scenario. We have developed tools for supporting domain experts in the specification of the service variability as well as consumers in the creation of individualized variants, and we have demonstrated them for customizing an Enterprise Service.

The presented technique enables the consumption of complex in individual application scenarios, with the main benefits of minimal modeling effort that abstracts from technical details, and of ensuring the correct service invocation by the explicit variability modeling and thorough resolution validation. We, however consider the presented technical solution as an initial prototype that shall be extended in the future. In particular, we plan to incorporate additional aspects and techniques for service variability (e.g. support for non-functional and quality-of-service aspects, mechanisms for handling conditions on instance data level, and support for service extensions mentioned above). In a longer perspective, the aim is to develop customization techniques for combined service bundles and applications, for which the variability modeling and management techniques for single services presented here shall serve as a basis.

Acknowledgements. This paper is mainly based on works supported by EU funding under the SHAPE project (FP7 - 216408).

References

1. R. Anzböck, S. Dustdar, and H. Gall. Software Configuration, Distribution, and Deployment of Web Services. In *Proc. of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE 2002), Ischia, Italy*, 2002.
2. J. Bayer, S. Gerard, Ø. Haugen, J. X. Mansell, B. Møller-Pedersen, J. Oldevik, P. Tessier, J.-P. Thibault, and T. Widen. Consolidated Product Line Variability Modeling. In T. Käkölä and J. C. Dueñas, editors, *Software Product Lines - Research Issues in Engineering and Management*, pages 195–241. Springer, 2006.
3. J. Beaton, S. Y. Jeong, Y. Xie, J. Stylos, and B. A. Myers. Usability Challenges for Enterprise Service-Oriented Architecture APIs. In *Proc. of IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'08)*, 2008.
4. W. A. Brown, R. G. Laird, C. Gee, and T. Mitra. *SOA Governance – Achieving and Sustaining Business and IT Agility.* IBM Press, 2008.
5. J. Cao, Wang. J., K. H. Law, S.-S. Zhang, and M. Li. An Interactive Service Customization Model. *Information & Software Technology*, 48(4):280–296, 2006.
6. V. de Castro, E. Marcos, and B. Vela. Representing WSDL with Extended UML. *Revista Comlombiana de Computacion*, 5(1), 2004.
7. A. Berre (ed.). Service oriented architecture Modeling Language (SoaML) – Specification for the UML Profile and Metamodel for Services (UPMS). Revised submission, OMG, 2008. OMG document: ad/2008-11-01.

8. D. S. Frankel. *Model Driven Architecture. Applying MDA to Enterprise Computing.* John Wiley & Sons, 2003.
9. W. O. Galitz. *The Essential Guide to User Interface Design – An Introduction to GUI Design Principles and Techniques.* Wiley, 3. edition, 2007.
10. Ø. Haugen, B. Møller-Pedersen, J. Oldevik, G. K. Olsen, and A. Svendsen. Adding Standardized Variability to Domain Specific Languages. In *Proc. of 12th International Conference on Software Product Lines (SPLC 2008)*, 2008.
11. H. Jegadeesan and S. Balasubramaniamm. A Method to Support Variability of Enterprise Services on the Cloud. In *Proc. of the 2nd International Conference on Cloud Computing (CLOUD-II 2009), Beijing, China*, 2009.
12. D. Mandell and S. McIlraith. A Bottom-Up Approach to Automating Web Service Discovery, Customization, and Semantic Translation. In *Proc. of the 12th International World Wide Web Conference, Workshop on E-Services and the Semantic Web (ESSW'03)*, Budapest, 2003.
13. R. Mietzner, F. Leymann, and M. P. Papazoglou. Defining Composite Configurable SaaS Application Packages Using SCA, Variability Descriptors and SaaS Multi-Tenancy Patterns. In *Proc. of 3rd Intl. Conf. on Internet and Web Applications and Services (ICIW 2008)*, Athens, Greece, 2008.
14. L. Peters and H. Saidin. IT and the Mass Customization of Services: the Challenge of Implementation. *International Journal of Information Management*, 20(2):103–119, 2000.
15. K. Pohl and A. Metzger. Variabilitätsmanagement in Software-Produktlinien. In K. Herrmann and B. Brügge, editors, *Tagungsband zur Software Engineering (SE08), München*, LNI. Gesellschaft für Informatik, 2008.
16. Y. Sam, O. Boucelma, and M.-S. Hacid. Web Services Customization – A Composition-based Approach. In *Proc. of the 6th International Conference on Web Engineering (ICWE 2006), Palo Alto, CA, USA*, pages 25–31, 2006.
17. K. Zhang, X Zhang, W. Sun, H. Liang, Y. Hung, L. Zeng, and X. Liu. A Policy-Driven Approach for Software-as-Services Customization. In *Proc. of the 9th IEEE International Conference on E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007), Tokyo, Japan*, pages 123–130, 2007.