Applying Variability Modeling Concepts to Support Decision Making for Service Composition

Kai Petersen, Nadine Bramsiepe, Klaus Pohl
Software Systems Engineering
Institute for Computer Science and Business Information Systems (ICB)
University of Duisburg-Essen
45117 Essen; Germany
(kai.petersen|nadine.bramsiepe|klaus.pohl)@sse.uni-due.de

Abstract

Service oriented architectures consist of loosely coupled services that can be quickly composed to support flexibility in business processes. The flexibility requires alternative service compositions to fulfill a customer's business process. However, customers are often not aware of their options and thus cannot make good decisions on how to compose their services. Therefore, we propose to support the decision making of the customer by modeling the different alternatives explicitly in a variability model and communicating the alternatives to the customer.

1. Introduction

Business processes have to be flexible because of unstable business environments [5]. Therefore, IT-infrastructures for supporting flexible business processes have to be easily adaptable to changing business processes. The Service oriented architecture pattern (SOA) helps achieving the desired degree of flexibility (cf. [2][4][5]). To adapt to new or changed business processes, SOA allows a flexible re-composition of loosely coupled services [5].

Service oriented architectures consist of different layers. Common layers are business process layer, service composition layer, service application layer and implementation layer [5]. The business process layer consists of process activities that require services to be executed. In order to execute a business process activity, not just one service is required, but a combination of services. A suitable combination of services, which fulfills the requirements of the business process activity, is determined in the service composition layer. All available services from which a service composition can be created reside within the service application layer. A service within the service application layer is an abstraction of an underlying implementation, e.g. a legacy system within the implementation layer. Service composition within the service composition layer becomes necessary, when the relation between services and business process activities is characterized as follows:

- It is not reasonable to consider all possible service compositions because not all of them contribute to the goals of a process activity that they should support [7].
- Services are usually more fine-grained than business process activities, i.e. one business process activity is realized by several services.
- To fulfill the goals of a business process activity, several service compositions might be possible to fulfill the goals of the process activity.

If business process activities and services have the same level of granularity, then they can be mapped one to one and no composition is necessary. However, if business process activities and services have different levels of granularity, business process activities and services cannot be mapped one to one. Therefore, we require support to choose suitable service compositions to achieve the goals of the business process activity. Even though, service discovery can be done automatically [2], the optimal selection of a services composition might not be achieved because an automatic selection of services does not imply that the customer is aware which alternative compositions are available to him. If the customer is not aware of alternative service compositions, he might overlook much better ones.

Consequently, we need to communicate alternative service compositions to the customer in an understandable way. That leads to the following question: *How do we support the communication to the customer considering that in a service oriented* *architecture many business process activities can be fulfilled by more than one combination of services?*

Our approach is to explicitly model the alternative service compositions that are suitable to achieve the goals of a business process activity and to present them to the customer as a basis for decision making.

The contribution of this paper is to show, how the orthogonal variability modeling language (OVM), which we have introduced for managing the variability of software product lines [6], can be utilized to explicitly document and communicate variability [3] in service composition.

In the context of software product line engineering, variability is used to document and manage what varies between the members (applications) of a product line. For instance, applications in an e-shop product line can have many commonalities (e. g. authentication), but may vary in terms of the supported language, e.g. 'English', 'Spanish', and 'German'. Requirements engineering in product line engineering is on the one hand responsible for the definition of the variability and on the other hand for the selection of variants to define a specific product.

2. Supporting the Communication of Alternative Service Compositions with Variability Models

For the documentation of alternative service compositions we propose to use our orthogonal variability modeling language (OVM) we introduced in [1]. Figure 1 shows an example of an OVM variability model.

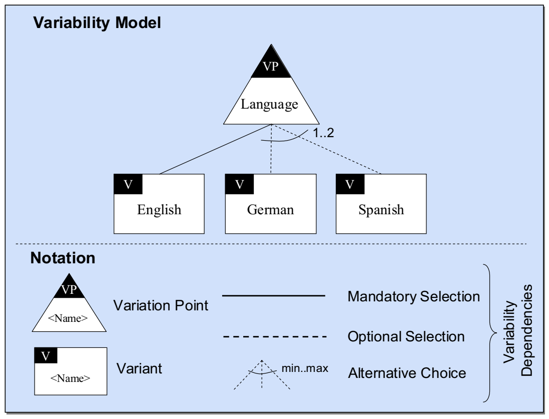

Figure 1 The Orthogonal Variability Model

A variation point ("what does vary") is represented by a triangle (see fig. 1, Language). The variants ("how does it vary") represent functionality and quality that can be chosen with regard to a specific variation point (see fig. 1, English, German and Spanish). The continuous line connecting the variation point 'Language' and the variant 'English' defines that the selection is mandatory. The dotted lines connecting the variation point 'Language' and the variants 'German' and 'Spanish' specify that the selection is optional. The semicircle that connects the dotted lines defines that at least one of the two variants has to be selected (for further details on the OVM language can be found in [6]).

Based on our experience with variability modelling, we propose to use the OVM to communicate alternative service compositions to the customer. This has the following significant benefits:

- The model explicitly documents and visualizes alternative options for service composition.
- The model allows us to show the customers dependencies between different variants within the model, i.e. they become aware of what is feasible and what is not.
- The explicit documentation of alternatives makes the customer aware of what options he has to select service compositions. That helps him to make informed decisions.

The relationships between variability modeling and service composition are shown in the meta-model in figure 2, which is an extension of the OVM meta model that is presented in [6].

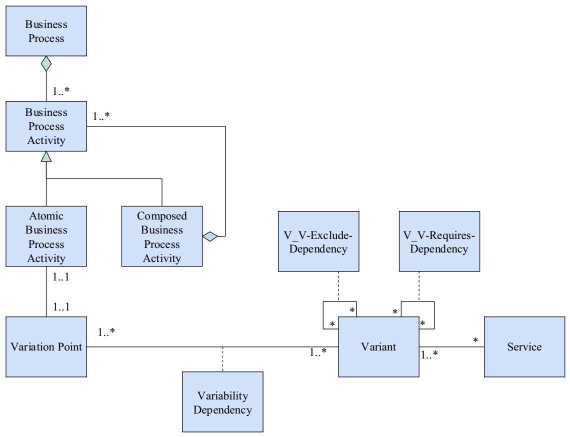

Figure 2 Meta Model of the OVM for Service Composition

The meta-model documents the fact that that a business process consists of one or many business process activities. A business process activity can be either a composition of process activities or an atomic process activity which is not further composed into sub activities.

On the lowest abstraction level of sub-activities (i.e. atomic activities), one variation point is attached to exactly one process activity.

Variation Points: Variation points are used to visualize and communicate to the customer, that there exist alternative service compositions to fulfil the goals of the business process activity.

One or several variants are associated with a variation point. A variant represents a composition of services which can or must be selected with respect to a specific variation point. A variant is associated with one or many variation points while a variation point is associated to one or many variants.

Variants: A variant is used to visualize and communicate to the customer one alternative to fulfil the goals of a business process activity.

Variant dependencies define whether a variant connected to the variation point has to be selected (mandatory variant) or can be selected (optional variant). Moreover, we can define how many variants have to be selected (e.g. at least two and at most four) with regard to a specific variation point.

Variant Dependencies: Variant dependencies are used to discuss with the customer whether services have to be selected or if the selection is optional.

Variant to variant dependencies (V_V) can be documented through requires and exclude dependencies. Exclude dependencies define that one variant does not allow another variant to be selected. Include dependencies define that one variant requires another variant to be selected.

Variant to Variant Dependencies: Variant dependencies are used to communicate to the customer what is feasible. For instance, it is not beneficial for the customer when he selects one variant without selecting another one.

A variant is realized by one or several services while a service can be used by no variant (the service is available, but not needed) or several variants. Therefore, the selection of variants determines for the selection of services.

In the next section we present an example where we use a variability model to represent alternative service compositions explicitly to support the communication of these alternatives to the customer.

3. Example

In our example, a customer runs a shop with the business activities ‘order product’, ‘bill customer’ and ‘distribute product’. He wants to support his process activity ‘bill customer’ with services. Therefore, services have to be discovered that support payment activities in general (e.g. automatically or manually).

The services application layer provides two possible combinations of services to support the process activity ‘bill customer’ (see Table 1).

Table 1 Alternative Compositions of Services

Composition 1			
Services	calculatePrice	payDebitCard	printReceipt
Composition 2			
Services	calculatePrice	payCashReg.	printReceipt

The first composition realizes the payment by debit card, the second composition the payment by cash.

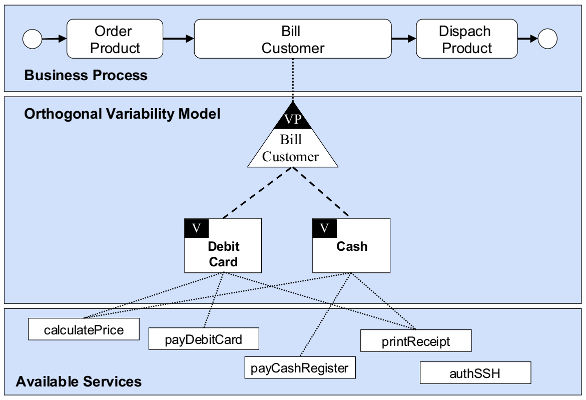

Figure 3 OVM for the Activity “Bill Customer”

The two alternative compositions of services are explicitly modeled in an OVM, represented by the variants ‘Debit Card’ and ‘Cash’ (see figure 3). The explicit modeling enables the requirements engineer to communicate the alternative service compositions to the customer which fulfill the business process activity ‘bill customer’. Thereby, the customer selects one of the alternative variants and he chooses the associated service composition. For instance, when the customer decides to bill the customer by debit card, he would achieve the desired functionality when the variant ‘Debit Card’ and thus the associated services ‘calculatePrice’, ‘payCreditCard’, and ‘printReceipt’ would be selected.

4. Discussion

By explicitly modelling alternative composition of services to fulfil a specific business process activity with OVM, we gain the following benefits:

- The OVM enables the requirements engineer to communicate different alternatives of service compositions to the customer which are suitable to achieve the goals of an atomic business process activity (see e.g. bill customer in our example). Without making the customer aware of alternative options in an easy and understandable way, he might misunderstand alternatives or overlooks better ones.

- The complexity of service composition can be reduced by introducing abstraction in form of variation points and variants.
- The OVM allows us to model exclude and requires dependencies between different combinations. Thereby, the customer becomes aware of what is feasible or not feasible. For instance, one composition of services excludes other services because they do not fit together, logically or technologically.

With the good understanding of the alternative service compositions provided by the model we strongly believe that the customer can make informed decisions and thus he is able to select a service composition that is suitable for his needs. The good understanding is further facilitated by graphical visualization. For instance, it is hard to recognize variation points in a textual description (see cf. [3] for further reasons).

In order to successfully apply variability modelling in the SOA context, we propose a variability modelling process for SOA in analogy to the requirements engineering process of software product line engineering [6].

5. Conclusions and Further Work

In this paper, we illustrate how variability modelling can be used to support the communication with the customer about alternative service compositions. We show that the model visualizes alternative options explicitly to the customer. Thus, it increases the awareness of the customer, what options are available. Last but not least, the customer also notices which combinations of alternatives are feasible.

Open issues and further work are:

- How can the variability model be automatically created to document the actual situation?
- How to define variation points so that they map to business process activities?
- How has the model to be adjusted to fit SOA?
- How does a process for modelling variability in the SOA context look like?

We plan to address these issues as follows:

- Conduct experiments with human subjects to identify drawbacks within the model and adjust it to fit SOA
- Implement a prototype of a tool that allows the automatic binding of variability and thus selection of appropriate service compositions.

6. Acknowledgements

This work was partially funded by the DFG-project PO-607/1-1 VarMod-PRIME.

7. References

[1] Bühne, S., Lauenroth, K., Pohl, K., "Modelling Requirements Variability across Product Lines", *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05)*, Paris, 2005, pp. 41-52

[2] Fickas, S., and Feather, M., "Requirements monitoring in dynamic environments", *Proceedings of the 2nd International Symposium on Requirements Engineering*, 1995, pp. 140-147

[3] Halmans, G.; Pohl, K.: "Communicating the Variability of a Software-Product Family to Customers"; Vol. 2; No. 1; *Software and Systems Modeling*; Springer; Berlin, Heidelberg; pp. 15-36; 2003

[4] Huang, Y., Chung, J., Chao, K., "A Stochastic Service Composition Model for Business Integration", *International Conference on Wireless Networks, Communications and Mobile Computing (NWeSP'05),* 2005, pp. 8.

[5] Krogdahl, P., Luef, G., Steindl, C., "Service-Oriented Agility: An initial analysis for the use of agile methods for SOA development", *Proceedings of the 2005 International Workshop on Service Computing (SCC'05)*, 2005, pp 93-100

[6] Pohl, K., Böckle, G., van der Linden, F., *"Software Product Line Engineering: Foundations, Principles, and Tehniques"*, Springer, Heidelberg, 2005

[7] Schmid, K., Eisenbarth, M., and Grund, M., "From Requirements Engineering to Knowledge Engineering: Challenges in Adaptive Systems", *Proceedings of the 1st International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements (SOCCER'05),* 2005.