Variability-Driven Selection of Services for Service Compositions^{*}

Kai Petersen, Johannes Maria Zaha, and Andreas Metzger

University of Duisburg-Essen, Software Systems Engineering
Schuetzenbahn 70, 45117 Essen, Germany

{kai.petersen,johannes.zaha,andreas.metzger}@sse.uni-due.de

Abstract. In order to deliver services that realize their requirements at low cost, in short time, and with high quality, service engineers reuse existing services for building composite services. For each service that is part of such a composite service and which is offered by a service provider, a service level agreement has to be established and the quality of service has to be monitored. Therefore, in order to keep service management controllable, the overall number of services across all service compositions that are maintained by an organization should be as small as possible. However, currently there exists no technique that would support service engineers in selecting such a minimal set of services when building composite services. By drawing on research results from software product line engineering, we define a service selection process (SeVAR) that exploits the similarities in the requirements in order to select the minimal set of services that achieves the best coverage of those requirements.

1 Introduction

In order to deliver services that realize their requirements at low cost, in short time, and with high quality, service engineers reuse existing services for building composite services. For each service that is part of such a composite service and which is offered by a service provider, a service level agreement (SLA) has to be established and the quality of service (QoS) has to be monitored. Therefore, in order to keep service management controllable, the overall number of services across all service compositions that are maintained by an organization should be as small as possible.

The following example illustrates how the total number of overall services can be kept small: Let us assume that an organization composes services to offer value added *e-shop services* on the service market. The consumers of the e-shop service might share common requirements, but typically will also demand customized solutions. For example, all consumers of a composite e-shop service might require payment by credit card (common requirement). Additionally, different consumers of the e-shop service might require different forms of payment as they, for instance, target different kinds of (end-)customers. These different

^{*} Parts of this work have been sponsored by DFG under grant PO 607/1-1 PRIME.

kinds of (end-)consumers might result in some of the consumers of the composite e-shop service requiring 'payment by pay pal' while others might require 'payment by debit card' (variable requirements).

In order for the e-shop service provider to keep the total number of services that he needs to manage at a minimum, he should thus consider the requirements of all potential service consumers when choosing the services he will compose into the composite e-shop service. This means instead of selecting the services to be composed into the e-shop service on a consumer by consumer basis, the e-shop service provider should pro-actively select a set of services that is able to fulfill the requirements of the potential e-shop service consumers while keeping the number of services that need to be composed to a minimum. As an example, if there exists a 'payment' service on the market that is able to handle all 'payment' options that are required by the different e-shop service consumers, the e-shop service-provider might select this service instead of a multitude of other services which provide a different combination of payment options.

To support the composite service creator in selecting a minimal set of services that will fulfill the requirements shared by different service consumers, we propose a service selection process that builds on previous work in software product line engineering and component-based software engineering. Drawing on the experiences that have been gained in software product line engineering (cf. [6]), we expect a significant payoff from using the same sets of services across different service compositions, i.e. from the reuse of services across the compositions. Figure 1 illustrates the potential benefit of this approach. If the selection process is done in such a way as to achieve the minimal number of services to realize the requirements of the potential service consumers, the number of different services that need to be managed by an organization can be kept considerably smaller than when selecting the services on a composition by composition (i.e., customer by customer) basis.

So far, research in the area of service oriented systems has not exploited the benefit of systematic, pro-active reuse of services between service compositions. However, similar problems were identified in software product line engineering when selecting COTS components for a software product line. The COTS

Fig. 1. Reduced Number of Different Services through Reuse (cf. [6], p. 10)

components were selected under consideration of reuse of these COTS components in different products of the product line. Ulfat-Bunyadi and Pohl (cf. [6], pp. 285-301) proposed an abstract process called CoVAR (Component Selection considering Variability, Architectural Concerns, and Requirements) for the selection of COTS components in software product line engineering to increase the reuse of COTS components between different products.

The contribution of this paper is a process that allows evaluating different sets of services with respect to their coverage of variable requirements, called service selection considering variability (SeVAR). Our work uses the abstract process CoVAR as a basis and adopts as well as concretizes this process for the service context. This instance allows selecting a minimal set of services to be used in service compositions which achieves the best coverage of requirements shared between customers of a market segment.

The remainder of this paper is structured as follows. In Section 2, we provide an overview of state of the art in service selection. In Section 3, we present a detailed description of our extension of the CoVAR process. Thereafter, we provide an application scenario in Section 4. Section 5 presents the conclusions and proposes future research work.

2 Background and Related Work

In the area of matchmaking between services and specifications, extensive work has been done to support the selection of services and components to match functional requirements and quality requirements. Benatallah et al. [2] provide a matchmaking algorithm that receives a service request or query (including the requirements that need to be covered) and a set of ontologies that describe services. Their algorithm allows determining the "best cover" of the requested requirements. In [9] and [1] the focus lies on the selection of service components based on quality requirements. They assume that certain services fulfill the same functional requirements, but differ regarding their quality factors. In this case, the goal is to select services that best suit the quality requirements of the customer considering global quality constraints, that means performance constraints are defined with respect to the overall composition. However, as the problem is NP-complete the optimal solution cannot be guaranteed in a reasonable computing time. Therefore, a "good-enough" solution is aimed at.

Software product lines are defined as "a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment of mission and that are developed from a common set of core assets in a prescribed way" [4]. A variety of software product line success stories showed that the pro-active reuse in product line engineering leads to less development cost, shorter time to market, and higher quality in comparison to single system development [6]. The development process of software product lines is separated into two development processes called domain engineering and application engineering [8][6]. In domain engineering, common and variable development artifacts are created like common and variable requirements

(development for reuse). In application engineering, the artifacts developed in domain engineering are reused (development with reuse). Common artifacts are part of each product of the software product line while variable artifacts are selectable. The selection of artifacts and constraints concerning this selection are determined by variability models, i.e. variability models describe how the products of the software product line can differ.

In earlier work, the CoVAR process was presented to support the selection of components in the context of product line engineering (cf. [6], pp. 285-301). The CoVAR process consists of the activities component screening, detailed component evaluation, and component selection. The three activities have common and variable requirements as well as domain architectural concerns as input, including the desired variability of the software product line. In the first activity components are screened, i.e. the description of the components is used to determine whether the components fulfill the required component features. Thereby, a large set of components can be already excluded to save evaluation effort in the detailed component evaluation activity. This detailed component evaluation is performed as second activity, whereby a detailed evaluation is performed on the components themselves. The result of this test of the components under consideration of variability results in a value that is assigned to each component. This is the basis for the actual selection of components in the last activity.

3 Service Selection Considering Variability (SeVAR)

3.1 Abstract Description of the SeVAR Process

The SeVAR process includes the same abstract steps as the CoVAR process (see Figure 2). In analogy, the first step is to conduct a market study of existing services based on service descriptions. These descriptions could use standards like the Web Services Description Language (WSDL) or Universal Description, Discovery and Integration (UDDI). The results are different sets of services ranked according to their degree of reusability in different service compositions. In the second step, the sets of services discovered during screening are invoked and monitored. The data recorded during monitoring forms the basis for the detailed evaluation of the services. In the third step, the final selection takes place by prioritizing different alternatives identified during the detailed evaluation. One possible technique that can be applied for prioritizing available services is for instance the Analytic Hierarchy Process (AHP) [7].

The three activities are very abstract and there are different alternatives to instantiate the activities. In this paper, we propose a detailed instantiation of the screening activity. The other activities are subject of future work.

3.2 The Screening Activity

During the screening activity, services are evaluated in terms of how well a combination of services can be reused to fulfill requirements posed on different service compositions. In order to conduct the evaluation it is necessary to make

Fig. 2. Overview of the SeVAR Process

the differences between service compositions explicit. In product line engineering, variability models can be used to document differences between products. In this instantiation we use the orthogonal variability model (OVM) [3] to describe differences between service compositions. In the following, we present the role of variability in service selection and thereafter the steps taken in the screening activity.

The Role of Variability in Service Selection. The elements of the OVM are variation points (What can vary?), variants (How can it vary?) and variability dependencies and constraint dependencies (How can a variant be selected at a variation point?) (cf. [6], pp. 57-88). In Figure 3, the notation of the OVM is shown. Variation points (VP) are represented by triangles and variants (V) are represented by boxes. The dashed lines connecting variation points and variants document that the selection of a variant is optional while a solid line documents that the variant has to be selected. Furthermore, constraint dependencies between variants document that if one variant is selected (V3) then another variant has to be selected (V4) or that variants are mutually exclusive, i.e. one variant cannot be selected when another variant is selected. Variants are related to requirements, documenting that if a variant is selected, then all requirements related to the variant have to be fulfilled by a set of services.

In order to achieve a high degree of reuse between different service compositions, the set of services that constitute the different composite services built by an organization should provide a high coverage of the elements of the variability model, i.e. variation points and variants. If a variation point is covered by a set of services, this means that the different requirements can be realized by this set of services. For instance, if VP1 in Figure 3 is completely covered by a set of services, this implies that all related variants are covered and thus all requirements associated to V1, V2 and V3 are fulfilled. Therefore, this set of services can be reused in different compositions, i.e. a composition that either includes V2 or V3.

Fig. 3. Orthogonal Variability Model Notation

Process Steps of the Screening Activity. To determine how well different sets of services cover the variability describing the difference between service compositions three steps are executed: market study (1), matchmaking between requirements and components (2) and determining the degree of variability coverage for a combination of components (3). An overview of the different steps is given in Figure 4.

Step 1 - Market Study: In the first step in Figure 4, a market study is conducted to identify a set of potential services used for the detailed evaluation in the proceeding activities. The inputs of this initial evaluation are the common and variable requirements of the service compositions. Based on these requirements, services are searched for in service registries. For a detailed description of how to conduct a market study we refer to [5]. As a result of the market study, services on the market are discarded as they do not fulfill the requirements (e.g. because they belong to a different domain). This leads to a set of potential services which is a subset of the services available on the market.

Step 2 - Matchmaking between requirements and services: In the second step, the services identified during the market study are examined in terms of which sets of these services fulfill the common and variable requirements to a predefined threshold. E.g., combinations of services that fulfill at least 70% of the requirements. Alternative thresholds are the acceptance of sets of services that fulfill 80% of the requirements prioritized as very important, 60% of the requirements considered as important and 10% of the requirements considered as less important or even separate thresholds for specific variants (e.g. because one variant is used more often than other variants when deriving concrete applications). These thresholds should be adjusted in case the number of discarded services is too high or too low. One might argue that it is enough to consider only the coverage of requirements. However, this is not the case because a variant can be related to several requirements. If only a small subset of the requirements related to a variant is covered, this variant is not realizable if selected. Therefore, the variability model has to be taken into consideration when calculating the coverage in step 3.

Step 3 - Determining the Degree of Coverage for a Combination of Services: In step three the different sets of services are evaluated in terms of how well they

Fig. 4. Screening of Services

cover the predefined variability. To determine the coverage, we first define what degree of coverage of a variability model means. The check marks next to the variation points or variants show the coverage of the elements (see Figure 5). A variant is covered, if all related requirements are covered by a set of services, e.g. variant 'b' is covered since 'VR₃' and 'VR₄' are covered. A variation point is covered, if all related variants are covered. In our example, variation point '2' is covered as variants 'c', 'd' and 'e' are covered. The overall OVM is covered, if all variation points belonging to the OVM are covered. The requires and exclude dependencies are not considered when calculating the coverage of the OVM because of complexity reduction. Considering the dependencies would lead to high computational complexity and is subject of further work.

Based on the definitions of coverage for variants, variation points, and the variability model we are able to define functions to calculate the coverage of the overall variability model. Thereby, the decision whether a variant is covered or not is not binary. The variant might also be selectable if 90 of 100 associated requirements are fulfilled. For example, a performance requirement is not fulfilled because the response time exceeds a predefined threshold. The same holds for variation points. A variation point is e.g. partially fulfilled if two of its variants are fulfilled by 90%.

Fig. 5. Coverage of Variability Model, Variation Point, and Variant

Formally, we define the variability model OVM as the tuple of variation points VP and variants V .

$$OVM := (VP, V), V := \{v_1, \dots, v_n\}, VP := \{vp_1, \dots, vp_m\} \quad (1)$$

To calculate the coverage of variants, variation points and the orthogonal variability model we further define mappings between the elements that are relevant during the calculation of the coverage including services, variants, variation points and requirements. We define a mapping $VMap$ (see Formula 2) of variants to a power set of requirements. The function $VMap$ delivers all requirements related to a given variant. For the lower variability model in Figure 5, $VMap(a)$ delivers the variable requirements VR_1 and VR_2 .

$$VMap : V \rightarrow \wp(R), R := \{r_1, \dots, r_p\} \quad (2)$$

The Mapping Variability Dependency VD delivers all variants related to a variation point. In the example, $VD(2)$ delivers the variants c , d and e .

$$VD : VP \rightarrow \wp(V) \quad (3)$$

The mapping $SMap$ delivers all requirements that are covered by a given service. For instance, if a service s_1 covers the variable requirements VR_1 and VR_2 then $SMap(s_1)$ delivers those variable requirements.

$$SMap : S \rightarrow \wp(R), S := \{s_1, \dots, s_q\} \quad (4)$$

The mappings describing the relations between services, variation points, variants and requirements can now be used to calculate the coverage of the OVM. The mapping $VCover$ calculates the coverage of a variant v by a set of services S . The result of this calculation is a value between 0 and 1 for the degree of coverage. Based on this mapping, we define the function $VCover(v, S')$ as an arithmetic average, calculated as the the number of elements within the intersection set of the set of variable requirements covered by services in S' and the set of requirements related to a variant, divided by the number of requirements related to the variant.

$$VCover : V \times \wp(S) \rightarrow [0..1] \quad (5)$$

$$VCover(v, S') = \frac{|\bigcup_{s \in S'} SMap(s) \cap VMap(v)|}{|VMap(v)|} \quad (6)$$

The mapping $VPCover$ calculates the coverage of a variation point vp for a given set of services S' . For this mapping, we define a function which calculates the sum of the coverage of all variants associated to a given variation point (determined by $VCover(v, S')$) divided by the number of variants associated to the variation point.

$$VPCover : VP \times \wp(S) \rightarrow [0..1] \quad (7)$$

$$VPCover(vp, S') = \frac{\sum_{v \in VD(vp)} VCover(v, S')}{|VD(vp)|} \quad (8)$$

Finally, the mapping $VMCover$ calculates the coverage of the OVM by a set of services S' . The function for this mapping calculates the sum of coverage of all variation points that are part of the OVM divided by the number of variation points belonging to the OVM.

$$VMCover : \wp(S) \rightarrow [0..1] \quad (9)$$

$$VMCover(S') = \frac{\sum_{vp \in VP} VPCover(vp, S')}{|VP|} \quad (10)$$

In Section 4, we show how the calculations are applied on an e-shop service. Additionally, we use the example to show how the calculations could be adapted based on the context in which they are used, e.g. different priorities associated to variants. However, the possibility to adjust the calculation is not included in the formula in its current version.

4 Application Scenario

In this section, we apply the screening activity to an e-Shop example. Therefore, we first present the e-shop variability model, the variable requirements and the candidate services identified during the market study. With this information as input, we apply the screening activity presented in Section 3.2.

4.1 eShop-Example

The e-shop's variability model is shown in Figure 6. The variability model consists of three variation points and seven variants. When choosing the variant 'card' the optional variation point 'card type' must be considered, documented by the requires dependency connecting the variant 'card' and variation point 'card type'. Furthermore, the selection of a card type requires a secure connection, documented by the requires dependency connecting the variation point 'card type' and the variant 'secure'.

To each of the variants, variable requirements are related that need to be fulfilled if the variant is selected. In Table 1, the variable requirements for the e-shop are listed. Each of the variable requirements is related to a variant, documented by the reference in the column 'V'. For instance, the functional requirement 'FR.2' is related to variant '1.2 - secure'.

Moreover, services were identified during the market study and composed using, for instance, the techniques for matchmaking and composition of services presented in Section 2. The threshold for creating the set of services used in the composition was that at least 7 out of 15 requirements shall be fulfilled. An overview of the sets of services identified is presented in Table 2.

In the following, we use the variability model, the requirements and the identified sets of services to determine the coverage of the variability model and show how a set of services is selected.

Fig. 6. The e-Shop Variability Model

Table 1. Variable Requirements of the e-Shop

ID	VP	V	type	Description
FR_1	1	1.1	functional	The connection shall be established via HTTP
FR_2	1	1.2	functional	The connection shall be established via HTTPS
FR_3	2	2.1	functional	Payment shall be possible via PayPal
FR_4	2	2.1	functional	The system shall provide the optional security key offered by PayPal
FR_5	2	2.1	functional	The PayPal user agreement shall be displayed prior to the transaction
FR_6	2	2.2	functional	Payment shall be possible via SMS-Message
QR_1	2	2.2	quality	The payment shall be confirmed within less than 1 minute
FR_7	2	2.2	functional	The system shall only accept SMS payment from Europe
FR_8	2	2.3	functional	Payment shall be possible via card
FR_9	3	3.2	functional	Payment shall be possible via credit card
FR_10	3	3.2	functional	MasterCard shall be supported
FR_11	3	3.2	functional	VisaCard shall be supported
FR_12	3	3.1	functional	Payment shall be possible via debit card
QR_2	3	3.1	quality	The verification of the debit card shall take less than 2 days
FR_13	3	3.1	functional	Debit card payment must be possible with debit cards from Germany, France, Italy and UK

Table 2. Identified Sets of Services used in a Composition

Composite Service	No. of services	Requirements Covered
Pay4u	3	FR_6, FR8, FR_9, FR_11, FR_12, QR_2, FR_13
PayAnywhere	4	FR_2, FR_3, FR_4, FR_6, QR_1, FR_7, FR_8, FR_9, FR_10, FR_12, QR_2, FR_13
PayMobile	7	FR_2, FR_3, FR_5, FR_6, QR_1, FR_7, FR_8, FR_9, FR_11, FR_12, QR_2, FR_13

4.2 Application of the Screening Activity

In this section, we present an example for calculating the coverage of the variability model. Furthermore, we show how the sets of services for a composition are selected. The first step is to calculate the coverage of the variants. For instance, variant 'SMS' is related to requirements 'FR_6', QR_1', and 'FR_7'. The coverage of the variant by the set of services called 'Pay4u' is calculated according to the function $VCover$:

$$VCover(SMS, Pay4u)$$

$$\begin{aligned}
 &= \frac{|\{FR_6, FR_8, FR_9, FR_11, FR_12, QR_2, FR_13\} \cap \{FR_6, QR_1, FR_7\}|}{|\{FR_6, QR_1, FR_7\}|} \\
 &= \frac{|\{FR_6\}|}{|\{FR_6, QR_1, FR_7\}|} = \frac{1}{3} = 33,3\%
 \end{aligned}$$

This calculation was conducted for all variants. An overview of the results is provided in Figure 7. In the next step, the coverage of the variation points is calculated. In the following, we calculate the coverage of the variation point 'payment' by the set of services 'Pay4u' using the function *VPCover*:

$$VPCover(payment, Pay4u) = \frac{0 + 33,3 + 100}{|\{paypal, SMS, card\}|} = \frac{133,3}{3} = 44,4\%$$

The overall coverage of the variability model by the set of services 'Pay4u' is calculated by the function *VMCover*. The calculated coverage of each variation point can be found in Figure 7.

$$VMCover(Pay4u) = \frac{0 + 44,4 + 83}{|\{connection, payment, cardtype\}|} = \frac{127,4}{3} = 42,5\%$$

The decision which sets of services shall be used in this example is determined by the coverage of the variability model. 'Pay4u' has the lowest priority because of its low degree of coverage (42,5%). However, the variant 'PayPal' is not realizable with 'Pay4u'. Additionally, only 33,3% of the requirements related to variant 'SMS' are covered, indicating that this variant would not be selectable. The sets of services called 'PayAnywhere' and 'PayMobile' have the same coverage of the variability model (74 %). However, 'PayAnywhere' requires less services than 'PayMobile' (see Table 2). Therefore, the services are ranked in the following order: (1) 'PayAnywhere', (2) 'PayMobile', and (3) 'Pay4u'.

It shall be emphasized that the calculations presented here are very simplistic and have to be adjusted based on particular situations. For instance, assuming

Fig. 7. Coverage of the Variability Model

that variants 'PayPal' and 'SMS' have the same priority, from the customers' point of view it is not relevant which one is selected. Therefore, the formula for calculating the coverage of variation point 'payment' can be adjusted in the following way: Instead of considering the coverage of every variant related to 'payment' we only consider the variant with the highest coverage. In the example of 'Pay4u' this is the variant 'SMS', since the variant 'card' is not considered due to its coverage of 100%. Additionally, the sum of coverage of each related variant is not divided by the number of related variants, but by the number of related variants minus one, leading to a higher coverage of variation point 'payment'. Therefore, the contextual situations (e.g. customer priorities) have to be taken into account when using our proposed technique.

5 Conclusion

In this paper, we proposed the service selection process SeVAR which exploits the similarities between the requirements of potential consumers of a composite service. Thereby, a minimal set of services can be determined that – when composed to the composite service – achieves the best coverage of the consumers' requirements. This implies that services will pro-actively be reused between the different service compositions that a service provider offers. We are confident that this reuse of services across the service compositions of a service provider will lead to lower costs and higher quality, because the complexity of managing and monitoring the services that are consumed by the composite service provider can significantly be reduced.

For our future work, we plan to refine the abstract steps of the SeVAR process and provide service engineers with concrete guidelines and techniques to apply our service selection process. Further, we will investigate how SeVAR can be embedded in an overall service-based systems engineering process and how the issue of dynamic adaptation of composite services can be handled.

References

1. Ardagna, D., Pernici, B.: Global and local qoS guarantee in web service selection. In: Bussler, C.J., Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 32–46. Springer, Heidelberg (2006)
2. Benatallah, B., Hacid, M.-S., Léger, A., Rey, C., Toumani, F.: On automating web services discovery. *The VLDB Journal* 14(1), 84–96 (2005)
3. Bühne, S., Lauenroth, K., Pohl, K.: Modelling requirements variability across product lines. In: *Proceedings of the 13th IEEE International Requirements Engineering Conference (RE 2005)*, pp. 41–52 (2005)
4. Clements, P., Northrop, L.M.: *Software Product Lines: Practices and Patterns*. Addison-Wesley Professional, Boston (2001)
5. Kontio, J.: *Otso: A systematic process for reusable software component selection*. Technical report, University of Maryland (1995)

6. Pohl, K., Böckle, G., van der Linden, F.J.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer, Heidelberg (2005)
7. Saaty, T.L.: Multicriteria decision making - the analytic hierarchy process: Planning, priority setting, resource allocation. RWS Publishing, Pittsburgh (1990)
8. Weiss, D.M., Lai, C.T.R.: Software Product Line Engineering - A Family-Based Software Development Process. Addison-Wesley, Reading (1999)
9. Yu, T., Lin, K.-J.: Service selection algorithms for web services with end-to-end QoS constraints. *Journal of Information Systems and e-Business Management* 3(2), 103–126 (2005)