Flexible Service Specification and Matching Based on Feature Models

Muhammad Naeem and Reiko Heckel

University of Leicester {mn105, reiko}@mcs.le.ac.uk

Abstract. We propose to use variability techniques from the realm of product lines to help make service specifications more flexible. Feature diagrams provide a high-level model of the essential and optional aspects of services in combination with detailed models of service's semantics based on visual contracts specified by graph transformation rules. In this way we hope to provide a precise, yet flexible specification of requirements towards as well as provided services, which is amenable to automation while being visual and user-friendly.

1 Introduction

The use of the web as a platform for application integration provides the technology to react to today's rapidly changing business climate [1]. However, while technically web services can be discovered, selected and bound to at runtime [2], the necessary automation of these tasks continues to pose major challenges. One of them is the level of flexibility required to find a service satisfying the specified requirements. Human programmers, if they cannot find exactly what they asked for, will be happy enough to adapt their demands to a sufficiently similar service and change their implementation accordingly. But any automated selection and binding requires a detailed specification not just of the signatures and data types of the services required but also of their actions and protocols. Such a detailed description will, however, be less likely to be matched by any existing service.

Let us consider a scenario where a requester (potential client) finds a service satisfying most of the requirements. For example, while requesting a service for hotel and flight reservation, requester may also be interested in transport from the airport to the hotel, a guide to visit holy or historical places, while having a preference to pay via bank transfer. The provider may offer to book flight, hotel and transport, and allow payment via bank transfer or credit card. Comparing our requirements with the offer we find a mismatch in the fact that no guide services are provided. To resolve this mismatch we need information about the relevance of the missing feature, e.g., whether it is optional or compulsory for the success of our application. In addition, there will be dependencies between features as well as with other elements of our models.

In this paper we are proposing to combine visual techniques for the modelling of product lines, specifically *feature models* [3–5], with a modelling approach using *visual contracts* (graphical pre- and post conditions for operations [8, 9]) for

specifying and matching service descriptions as produced by the service provider against requirements as expressed by the requester. The approach is based on an easy integration with mainstream software UML notations, provides the level of formality and flexibility required to realistically allow automated matching of services, and supports the adaptation of provider and requester interfaces and implementations to the chosen variant.

The remainder of the paper is organised as follows: Section 2 discusses some related work, Section 3 introduces the specification and matching of services by visual contracts using a small case study. Section 4 extends this approach by feature models and Section 5 describes the matching and adaptation of these extended models. Section 6 concludes the paper.

2 Related Work

A number of approaches address flexible matching of services with semantic descriptions. Paolucci et al. in [11] have described an engine that allows matching of advertisements and requests on the bases of the capabilities. Several matchmaking frameworks are developed in [12–16], which operate on service descriptions written in RDF, DAML+OIL or DAML-S. Wu in [17] proposes a similarity-based approach, which grounds the matching process on a comparison of signature specifications in WSDL, but not on semantic descriptions. Matchmaking approaches ranking the similarity between advertisement and request are developed in [18] and [19]. In [20], Yongley adopted the ranking technique to service matching based on semantic descriptions.

We believe that variability needs to be built into the service specification, so that automated decisions can be made in the case of imperfect matches. This is different from ranking matches according to their degree of similarity. Moreover, our approach, while being inspired by standard methods for specifying and matching service semantics, does so on the basis of a visual notations which allows integration into mainstream software modelling languages.

3 Specifying and Matching of Services

In this section we describe the basic approach to visual semantic modelling and matching of services by means of a case study of an online travel agent. We use graph transformation rules to represent visual contracts.

Graph Transformation Systems (GTS) provide a modelling language where graphs model (data) states and rules specify state changing operations. The class of admissible states is specified by a *type graph* [9]. Fig. 1 shows a possible state graph of the case study where bookings have been generated by a client for hotel, flight, and transport. The types of nodes and their associations are represented by the type graph in the left of Fig. 1.

In the context of semantic web services we can think of the type graph as a representation of an ontology and of the instance graph as a data confirming

to it. Transformations of instance graphs are due to the application of graph transformation rules, shown in Fig. 2.

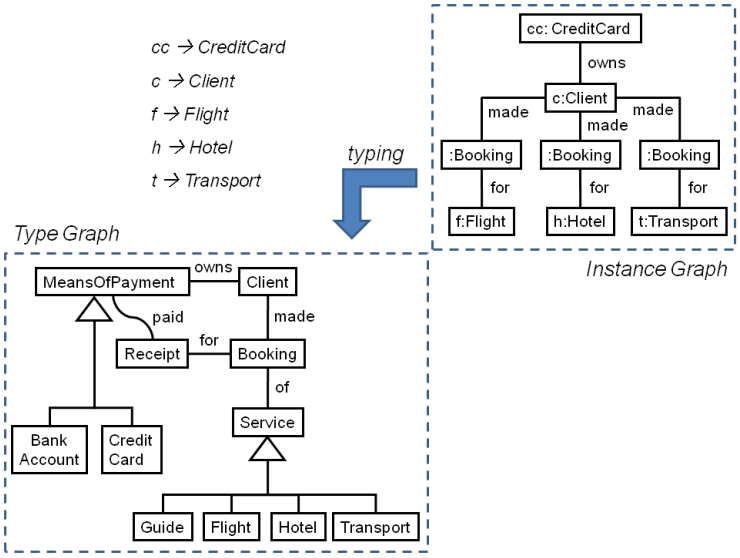


Fig. 1. Type and instance graphs

Service models are given in two version: from the requester's point of view describing desired functionality and from the provider's perspective specifying the services that are actually implemented.

The rule in the top part of Fig. 2 represents the requirements of a requester who is looking to reserve guide services and is willing to pay by credit card. The left side of the rule shows the preconditions of the operation while the right-hand side shows its postcondition / effect [9]. The rule expresses the demand for an operation which allows a client to book a guide payed for by a credit card owned by the client. Similar operations could be requested for the booking of hotel, flight, and transport, or to allow payment via bank transfer.

A rule describing the operation as implemented by the provider is in the bottom of Fig. 2. It offers to book any service using any available payment method.

It should be clear that, in this case the requirements of the requester are satisfied by the operations as specified by the provider because of the subtyping relation between *MeansOfPayment* and *CreditCard* as well as *Service* and *Guide*. These relations are captured in the ontology that both rules are based on, and which we assume to be standardised.

However, while we clearly have to rely on such standardisation to allow matching of services at all, we would like to have the possibility for example

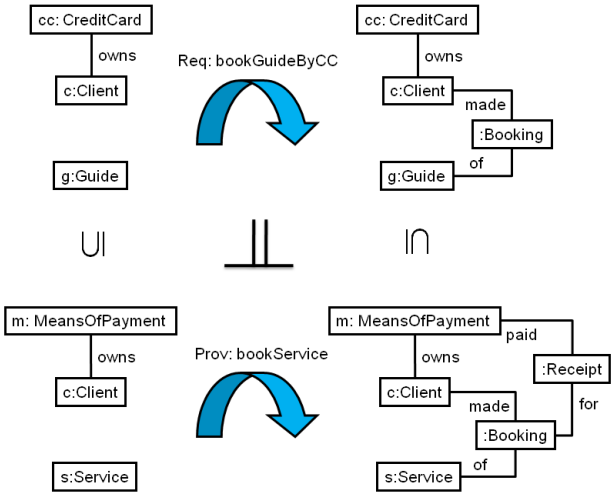


Fig. 2. Visual contracts specifying booking operations from requester (top) and provider (bottom) point of view

to specify a provider that does not offer guide services. We could do so at a moment by replacing the generic rule shown in the bottom of Fig. 2 by a number of more specific ones for booking hotel, transport and flight. In this case, the requester's requirement would not be satisfied by this service. In the next sections we will introduce a more economical way of representing this and other provider models based on features. We will also have to extend our notion.

Formally, a (provider) rule satisfies another (requester) rule if the preconditions of the first entail the preconditions of the second and the postconditions / effects of the second entail those of the first. Entailment in this case boils down to subgraph matching, allowing for the specialisation of types, i.e., the right-hand side of the provider rule entails the right-hand side of the requester rule because the former is a supergraph of the latter with more general types.

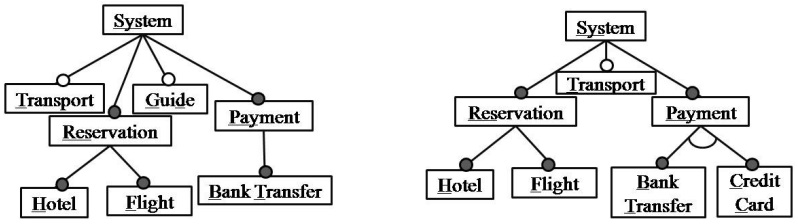
4 Visual Contracts with Features

In order to allow for a more fine-grained specification of the functionality on offer as well the desired flexibility in the matching of provisions to requests, we propose to use feature models. A feature is "a distinguishable characteristic of a concept that is relevant to some stakeholders" [4] while feature diagrams can be used to show the variability of features in a hierarchical form, including different types of features (such as optional, mandatory, alternative, etc.) and their interdependencies [3–7]. A feature model consists of a feature diagram and

other associated information, in our case given by the type graph (ontology) and visual rules (visual contracts) of requester and provider models.

Semantically, a feature diagram describes a set of instances, each representing a permissable subset of features. By taking the intersection between the sets of subsets on the requester and provider side we can identify the feature sets agreeable to both parties. Each such set describes a particular selection of features which can be used to derive a corresponding variant of the underlying service models.

Feature diagrams for requester and provider are shown in Fig. 3(a), 3(b). For example, the requester diagram declares Transport and Guide as optional features whereas Hotel and Flight Reservation and Payment by Bank Transfer as mandatory features. We have used *System* for the concept node of both requester and provider feature trees for ease in comparison.



(a) Requester (b) Provider

Fig. 3. Feature Diagrams of Requester and Provider

The connection between feature diagrams and visual contract models is provided by labelling the model elements (node types, rules, etc.) by the features they are part of. This is shown for our example in the type graph in Fig. 4 by small gray boxes with dashed borders placed at the corners of classes. Semantically, this means that since the Guide is not available as a feature of the provider (as seen from the provider's feature diagram), the corresponding class in the type graph is projected from the provider's view. In this way, the provider's booking rule in Fig. 2 describing the booking for all services does not promise subsume the booking of Guide services. For brevity we have used underlined characters of feature names (from Fig. 3) as labels in Fig. 4.

Similarly, operations and their visual contracts are labelled, e.g., rule *Req :: bookGuideByCC* would be labelled *Guide* and *CC* while *Prov :: bookService* would carry all labels except for *Guide*.

5 Matchmaking and Adaptation

In order to find out if a provider description matches the requirements expressed by a requester model, we proceed in three steps.

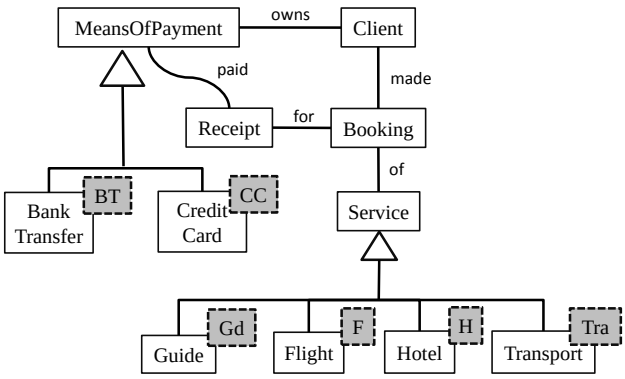


Fig. 4. Type Graph labelled by features: BT for Bank Transfer, CC for Credit Card, Gd for Guide, F for Flight, H for Hotel, Tra for Transport

1. Compute intersection of feature sets of requester and provider feature diagrams.
2. For each feature set in the intersection, derive the corresponding variant of provider and requester model.
3. Check compatibility of each derived pair of models.

For Step 1, feature trees can be converted into propositional formulas [3, 6, 7]. The set of solutions of the conjunction of the propositional formulas derived from the two feature models provides us with the desired intersection, i.e., the set of all subsets of features that are admissible according to both models. The result can be visualised as a feature diagram again, such as in Fig. 5 representing the intersection of feature diagrams of Fig. 3(a), 3(b). The largest admissible features set is $\{Sys, Res, F, H, Tra, Pay, BT\}$, but also $\{Sys, Res, F, H, Pay, BT\}$ is in the intersection.

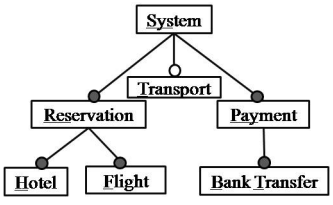


Fig. 5. Intersection of feature diagrams of Figures 3(a), 3(b)

In order to define the variant of requester and provider models in Step 2, we have to delete from these models all elements labelled by features not in the relevant feature set, and then recursively all the elements dependent on those deleted. For example, deleting the *Guide* class results in deleting the corresponding subtyping relation as well as all the operations and rules containing instances of this class. They are therefore disregarded in the next step. Further, if a subclass is removed from a superclass which occurs in a rule, this rule is removed as well and replaced by all its specialisations where the superclass is replaced by all permissable subclasses. In this way, from rule *Prov :: bookService* in Fig. 2 we obtain three specialisations, one of which (for Service → Transport) is shown in Fig. 6 together with the variant of the type graph.

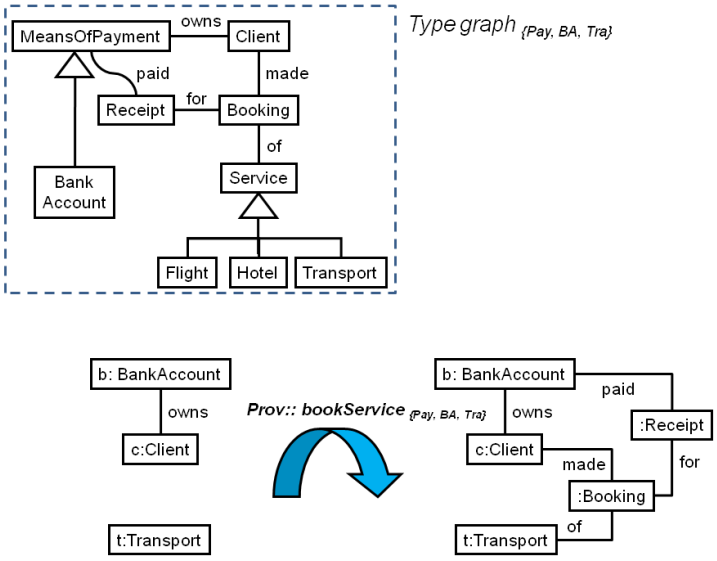


Fig. 6. Variant of type graph and rule

Since as a result of Step 2 we will obtain a pair of ordinary models (without features) for each subset of features selected in Step 1, we can apply the standard notion of matching as explained in Section 3 to check that they are indeed compatible.

Thus, there could be two reasons for a requirement not to be matched by a service description: An empty intersection of their feature diagrams (e.g., if a mandatory feature of the requester is not provided), or an incompatibility in the semantics of the actual operations. For the latter, consider a rule like the one in Fig. 2 but without the links between *c:Client* and *cc:CreditCard* in the left- and right-hand sides. In this case, the client would try to pay with a credit card not

owned by the person who has made the booking, which would contradict the requirement of the provider rule.

Notice that we take for granted here the fact that all models are specified over a shared ontology. Thus rules use the same classes for the same concepts and naming of features is consistent.

6 Conclusion

We proposed an extension by feature models of a visual approach to semantic web services based on graph transformation. As feature models can be rephrased in terms of propositional logic and visual contracts map to simple description logics, the entire approach could be handled in a purely logical framework. However, we believe that the visual presentation of models is as important for their usability as the explanation of the matching procedure at the same level.

Future work will focus on evaluating the approach, developing tool support, and extending it towards prioritising of features to be able to rank different feature sets for their level of satisfaction before going on to check the consistency of the semantic descriptions.

References

1. Leymann, F.: Choreography for the Grid: Towards Fitting BPEL to the Resource Framework. Journal of Concurrency and Computation: Practice and Experience. **17** (2005).
2. Papazoglou, P.: Service-Oriented Computing: Concepts, Characteristics and Directions. In Proceedings of the Fourth International Conference on Web Information Systems Engineering. IEEE Computer Society Washington, DC, USA. (2003) 3–12
3. Czarnecki, K., Eisenecker, U.: Generative Programming - Methods, Tools, and Applications. Addison-Wesley, Boston, MA. (2000)
4. Kang, K., Cohen, S., Hess, J., Nowak, W., Peterson, S.: Feature-Oriented Domain Analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213. (1990)
5. Kang, K., Kima, S., Lee, J., Kim, K., Shin E., Huh, M.: FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. Annals of Software Engineering. **5** (1998) 143–168
6. Batory D.: Feature Models, Grammars, and Propositional Formulas. In Proceedings of Software Product Lines Conference 2005. Springer Berlin/Heidelberg. LNCS **3714** (2005) 7–20
7. Jong, M., Visser, J.: Grammars as Feature Diagrams. In Proceedings of ICSR7 Workshop on Generative Programming. (2002) 23–24
8. Hausmann, J., Heckel, R., Lohmann, M.: Model-based Development of Web Service Descriptions: Enabling a Precise Matching Concept. International Journal of Web Services Research. **2** (2005) 67–84
9. Engels G., Heckel, R.: Graph Transformation as a Conceptual and Formal Framework for System Modelling and Model Evolution. In Proceedings of the 27th International Colloquium on Automata, Languages and Programming. Springer Berlin/Heidelberg. LNCS **1853** (2000) 127–150

10. Simos, M., Creps, R., Klingler, C., Lavine, L., UNISYS DEFENSE SYSTEMS RESTON VA: Software Technology for Adaptable Reliable System (STARS) Organization Domain Modeling (ODM) Guidebook. Technical Report STARS-VC-A025/001/00, Lockheed Martin Tactical Defense Systems, Manassas, VA. **2** (1996)
11. Paolucci, M., Kawamura, T., Payne, T., Sycara, K.: Semantic Matching of Web Services Capabilities. In Proceedings of the 1st International Semantic Web Conference. LNCS **2342** (2002) 333–348
12. Chiat, L., Huang, L., Xie, J.: Matchmaking for Semantic Web Services. In Proceedings of IEEE International Conference on Services Computing. IEEE SCC. (2004) 455–458
13. Trastour, D., Bartolini, C., Priest, C.: Semantic Web Support for the Business-to-Business E-Commerce Lifecycle. In Proceedings of Eleventh Conference on World Wide Web. ACM New York, NY, USA. (2002) 89–98
14. Gonzales-Castillo, J., Trastour, D., Bartolini, C.: Description Logics for Matchmaking of Services. Proceedings of the KI-2001 Workshop on Applications of Description Logics, Aachen: CEUR Workshop Proceedings. **44** (2001) 89–126
15. Li, L., Horrocks, I.: A Software Framework for Matchmaking Based on Semantic Web Technology. In Proceedings of the Twelfth International Conference on World Wide Web. ACM New York, NY, USA. (2003) 331–339
16. Paolucci, M., Kawamura, T., Payne, T., Sycara, K.: Semantic Matching of Web Services Capabilities. In Proceedings of the 1st International Semantic Web Conference. Springer-Verlag London, UK. LNCS **2342** (2002) 334–347
17. Wu, J., Wu, Z.: Similarity-based Web Service Matchmaking. In Proceedings IEEE International Conference on Services Computing. IEEE Computer Society Washington, DC, USA. SCC **1** (2005) 287–294
18. Noia, T., Sciascio, E., Donini, F., Mongiello, M.: A System for Principled Matchmaking in an Electronic Marketplace. In Proceedings of the Twelfth International Conference on World Wide Web. ACM New York, NY, USA. (2003) 321–330
19. Jaeger, M., Tang, S.: Ranked Matching for Service Descriptions using DAML-S. In Proceedings of the Open InterOp Workshop on Enterprise Modelling and Ontologies for Interoperability Co-located with CaiSE'04 Conference. (2004) 217–228
20. Yao, Y., Su, S., Yang, F.: Service Matching Based on Semantic Descriptions. In Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services. IEEE Computer Society Washington, DC, USA. (2006) 126–131