A Theory of Service Behavior

Karsten Wolf

Universität Rostock, Institut für Informatik

Abstract. We outline a fundamental approach to behavioral aspects of services. In the center of this approach, we see behavioral models of services, interactions, and finite representations of sets thereof. Several operations and relations can be defined and their implementation on our representations can be studied. Finally, a number of interesting problems can be traced back to our models and operations. On the boundary of our theory, we place interfaces to other aspects of services.

1 Introduction

Services are made for being loosely coupled to larger artifacts. A reasonable coupling (i.e. interaction via message transfer) must take care of various aspects of compatibility, including:

- Semantical compatibility: what does the content of exchanged messages mean?
- non-functional compatibility: how is the exchange of a message organized?
- behavioral compatibility: in which order are messages exchanged?

We target the behavioral aspect of compatibility. The remaining aspects are taken care of by a well-defined interface which enables us to integrate any techniques, approaches, and results regarding semantics or non-functional aspects of service composition.

2 Representing Service Behavior

There seem to be two complementary approaches to the specification of service behavior. In the first approach, we specify the control flow of a single service (end point, peer, participant). This control flow implicitly constrains the order of messages that are transmitted via the middleware. In the second approach, we specify sequences of message transmissions which we want to see in the middleware (a choreography). A theory of service behavior should support both points of view.

For the specification of single services, we propose to use *service automata*. They offer concepts of state and transition for modeling control flow. Transitions can be labeled with primitives like sending or receiving a message thus modeling the interaction behavior of a service. Service automata may be compactly represented as Petri nets or expressions in a process algebra thus inheriting

several results from existing theories. Service automata are well linked to relevant languages like WS-BPEL or BPMN as there exist back-and-forth translations between these languages to Petri nets [1, 2], and back-and-forth translations between Petri nets and service automata.

For the specification of a choreography, we have not yet identified a canonical formalism. On one hand, a set of traces of interaction primitives (like sending or receiving a single message) seems to be a more reasonable starting point than process description formalisms like the pi-calculus. The reason is that the middleware that connects services is not an actor which deliberately takes decisions. It is rather a medium that records the effect of decisions taken elsewhere. On the other hand, some authors insist that it might be essential for a choreography description to record who is in charge for selecting a particular sequence from the space of opportunities given by a set of traces. We conclude that the selection of an appropriate formalism for a theoretical study of choreographies is still a future work task.

In addition to the specification of a single behavior, we consider specifications of sets of behaviors. A meaningful example in the case of single services is an operating guideline, i.e. a finite representation of the set of all compatible partner services to a given service. We believe that many interesting problems can be traced back to simple questions concerning single behaviors or sets of behaviors.

3 Operations and Relations on Service Behaviors

Among the important operations on behavior, there are of course a few trivial ones. These include, for example, the composition of services to larger ones or ecxtracting the set of traces in the middleware that can be realized by a given composition of services.

A next class of operations concerns the synthesis of missing components to an incomplete specification. We already have algorithms for the synthesis of compatible partners to a given services [3], or for synthesizing an adapter to a set of incompatible services [4]. Similar techniques should work for synthesizing end points to a given choreography. In many known scenarios, a synthesis algorithm that calculates a single fitting behavior can be extended to an algorithm that computes a finite representation of all fitting behaviors of some kind [5, 6].

In a third class of algorithms, we investigate standard operations on our core objects. As a starting point, we look for realizations of standard set operations (intersection, union, complement, projections) to representations of sets of service behaviors. The actual challenge is that we start with, and want to arrive at, finite representations of infinite sets (where each element is as complex as the control flow of a service). We have some indication, that we will succeed with a minor extension of the structures used for operating guidelines (characterization of the set of all compatible partners of a given service). Further down in this article, we sketch some useful applications.

In a fourth class of operations, we would transform given service behaviors. Potential applications include the repair of malfunctioning compositions [7], the generation of public views out of private views, or vice versa [8, 9].

Service behaviors need to be compared with each other. A core concept in this regard is the one of equivalence. Several equivalence notions for services have already been proposed. Not all of these notions are very well motivated. For instance, substituting a service with a trace equivalent one may not preserve compatibility with other services. On the other hand, requiring bisimulation equivalence is often too strong a requirement that prevents harmless substitutions.

We believe that reasonable equivalence notions need to be derived from application scenarios. As an example, study a scenario where a service is substituted by another one such that all compatible partners of the old service remain compatible with the new one. The corresponding equivalence holds between all services with the same set of compatibvle partners. It turns out that this equivalence is strongly tied to the process algebraic notion of a should testing equivalence which is a non-trivial entry in the huge zoo of equivalences proposed in process algebra. In future work, we want to identify other application scenarios which may call for different notions of equivalence.

4 Targeted Problems

Of course, the objective of our theory is to provide useful solutions to interesting problems. In this section, we demonstrate that the outlined theory of service behavior yields approaches to a number of interesting problems.

Verify and validate a service

Through the synthesis of a compatible partner [3], we may prove the principal wellformedness of a service. Some initial approaches suggest that even the construction of diagnosis information for a given malfunctioning service involves operating guidelines, i.e. a core element of our theory [10]. Using the set of all compatible services [5], we may compare the external effect of a service to a specification. In particular, we may verify whether or not certain targeted partners are among the compatible ones. The characterization of all partners may yield useful (positive) test cases in some scenarios [11] while the complement of that characterization might include negative test cases.

Construct a service

Due to an already existing link from service automata via Petri nets to abstract WS-BPEL, we may support the automatic generation of services for various purposes. These services are compatible by construction. Our theory enables a flexible selection of services to be generated [12]. We may, for instance, translate various requirements into finite representations of sets of services and then use intersection as an instrument for filtering some desired behavior out of the set of all compatible ones.

Compose services

Composition may be supported for instance by synthesizing missing components (adapters) [4], by transforming participants of a malfunctioning composition [7], by exchanging components (for instance, public views with private views in contract scenarios) [8]. We may support the selection of services from repositories. For finding a compatible partner of a given service R in a repository of services $P_1, \ldots, P_n$, we may check containment of R in one of the sets OG_i $(1 \leq i \leq n)$ where OG_i is the (finite representation of the) set of compatible partners of P_i. Even more efficiently, we may precompute the unions $OG_1 \cup \ldots \cup OG_{ndiv2}$ and $OG_{ndiv2+1} \cup \ldots \cup OG_n$ to more quickly reduce the search space. Using this idea, we may select a suitable P_i with $\log n$ containment checks instead of n such checks.

Replace a service

Using the right equivalence notions, replacement of services can be done without harming compatibility. Our existing substitutability notion can be traced back to basic set operations as follows. Let OG_X be the set of compatible partners of service X. Then P can be exchanged with R iff $OG_P \subseteq OG_R$ which is equivalent to $OG_P \cap \overline{OG_R} = \emptyset$. The remaining emptyness check should be easy. Moreover, a nonempty $OG_P \cap \overline{OG_R}$ canonically provides examples which prove non-substitutability. Such an example may help in providing diagnostic information and is not available in existing approaches to checking substitutability.

Verify and validate a choreography

There exist notions of realizability of a choreography. We believe that it is possible to translate the realizability problem into a partner synthesis problem. In the future, we may see more interesting problems concerning choreographies.

5 Problem Parameters

Most problems and solutions can be formulated in several settings. So far, we have identified the following parameters which more or less influence all approaches stated so far.

Compatibility notion

There are several reasonable notions of compatibility. The one occurring most frequently is deadlock freedom in the composed system. One may also require that it should always be possible to reach a designated terminal state, or to have a composed system that is sound (as defined for workflow models). Instead of possible termination one can also require eventual termination. The latter notion requires that the model contains information about fairness of decisions in the control flow. Additional user definable constraints may parametrize compatibility.

Nature of message passing

In a canonical setting, message passing is thought of being asynchronous. Existing approaches do or do not allow overtaking of messages. In the literature, synchronous communication is frequently studied, too. Further, we may or may not consider constraints that are implied by the semantics of messages. For instance, semantics may identify message type a as "empty form" and b as "filled form" which implies that it would not make sense to send b before having received a. Further down, semantical constraints are discussed in more detail.

Distribution of partner

Many services have interfaces to more than one partner. For synthesizing partners, we may or may not assume the capability of those partners to be coordinated during run-time or during build-time. This leads to different results concerning well-formedness [13].

6 Interface to Other Aspects

For being applicable, our solutions must be in line with the remaining aspects of service compatibility. As an example, we sketch an interface to semantics which we already found useful in the context of adapter synthesis.

We already mentioned that the semantics of messages may imply constraints on the behavior of a synthesized service. Examples of such constraints include

- Do not send a filled form before having received an empty one (while the order of unrelated messages does not matter)
- Do not send a message containing somebody else's password without having received it in another message (wihle you may send your own password without having received any message)

We claim that most relevant semantical constraints can be expressed in terms of transformation rules such as empty_form $\rightarrow$ filled_form, $\rightarrow$ own_password, meter $\rightarrow$ feet, or street + zipcode + name $\rightarrow$ address. The rules specify the semantically implied effect of message contents on the behavior, without implying any particular approach to represent or discover semantics as such. In fact, the rules may be specified manually, synthesized from semantic web approaches, etc. We already showed that it is possible to trace back synthesis problems in presence of semantical constraints to plain synthesis problems [4, 14].

7 Tools

There is already a family of tools which provide some of the discussed functionality:

- LoLA for the exploration of state spaces and thus for the investigation of complete compositions;

– Fiona for the synthesis of compatible partners and partner sets as well as for the synthesis of adapters, checking compatibility and a few other applications;
– BPEL2oWFN and oWFN2BPEL for the translation between WS-BPEL and Petrio nets;
– Rachel for a repair of a malfunctioning choreographies
– and certainly a number of tools developed in other groups.

They prove that, to the degree innplemented, operations can indeed be applied to realistic service specifications.

8 Conclusion

We propose a reasonable set of objects and operations to constitute a theory of service behavior. We have already identified a number of interesting problems which all can be traced back to a small number of operations on recurring kinds of objects like sets of service behaviors (also known as operating guidelines). We believe that a further consolidation of the theory would yield additional insights into the nature of services and their composition. Moreover, tracing back many interesting problems to a few operations helps us to strengthen the tool support for a large variety of problem settings.

References

1. Lohmann, N.: A feature-complete Petri net semantics for WS-BPEL 2.0. In Dumas, M., Heckel, R., eds.: Web Services and Formal Methods, Forth International Workshop, WS-FM 2007, Brisbane, Australia, September 28-29, 2007, Proceedings. Volume 4937 of Lecture Notes in Computer Science., Springer-Verlag (2008) 77–91
2. Lohmann, N., Kleine, J.: Fully-automatic translation of open workflow net models into simple abstract BPEL processes. In Kühne, T., Reisig, W., Steimann, F., eds.: Modellierung 2008, 12.-14. März 2008, Berlin, Proceedings. Volume P-127 of Lecture Notes in Informatics (LNI)., GI (2008) 57–72
3. Wolf, K.: Does my service have partners? LNCS ToPNoC **5460**(II) (2009) 152–171 Special Issue on Concurrency in Process-Aware Information Systems.
4. Gierds, C., Mooij, A.J., Wolf, K.: Specifying and generating behavioral service adapter based on transformation rules. Preprint CS-02-08, Universität Rostock, Rostock, Germany (2008)
5. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In Kleijn, J., Yakovlev, A., eds.: 28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, ICATPN 2007, Siedlce, Poland, June 25–29, 2007, Proceedings. Volume 4546 of Lecture Notes in Computer Science., Springer-Verlag (2007) 321–341
6. Stahl, C., Wolf, K.: Deciding service composition and substitutability using extended operating guidelines. Data Knowl. Eng. (2008) (Accepted for publication in December 2008).
7. Lohmann, N.: Correcting deadlocking service choreographies using a simulation-based graph edit distance. In Dumas, M., Reichert, M., Shan, M.C., eds.: Business Process Management, 6th International Conference, BPM 2008, Milan, Italy, September 1–4, 2008, Proceedings. Volume 5240 of Lecture Notes in Computer Science., Springer-Verlag (2008) 132–147

8. Aalst, W.M.P.v.d., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: Multiparty contracts: Agreeing and implementing interorganizational processes. Comput. J. (2008)
9. König, D., Lohmann, N., Moser, S., Stahl, C., Wolf, K.: Extending the compatibility notion for abstract WS-BPEL processes. In Ma, W.Y., Tomkins, A., Zhang, X., eds.: Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21–25, 2008, ACM (2008) 785–794
10. Lohmann, N.: Why does my service have no partners? In Bruni, R., Wolf, K., eds.: Web Services and Formal Methods, Fifth International Workshop, WS-FM 2008, Milan, Italy, September 4–5, 2008, Proceedings. Lecture Notes in Computer Science, Springer-Verlag (2008)
11. Kaschner, K., Lohmann, N.: Automatic test case generation for interacting services. In Feuerlicht, G., Lamersdorf, W., eds.: Service-Oriented Computing – ICSOC 2008, 6th International Conference, Sydney, Australia, December 1-5, 2008. Workshops Proceedings. Lecture Notes in Computer Science, Springer-Verlag (2008) (to appear).
12. Lohmann, N., Massuthe, P., Wolf, K.: Behavioral constraints for services. In Alonso, G., Dadam, P., Rosemann, M., eds.: Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24–28, 2007, Proceedings. Volume 4714 of Lecture Notes in Computer Science., Springer-Verlag (2007) 271–287
13. Wolf, K.: Does my service have partners? LNCS ToPNoC **5460**(II) (2009) 152–171 Special Issue on Concurrency in Process-Aware Information Systems.
14. Wolf, K.: On synthesizing behavior that is aware of semantical constraints. In Lohmann, N., Wolf, K., eds.: 15th German Workshop on Algorithms and Tools for Petri Nets, AWPN 2008, Rostock, Germany, September 26–27, 2008, Proceedings. Volume 380 of CEUR Workshop Proceedings., CEUR-WS.org (2008) 49–54