Operating Guidelines - an Automata-Theoretic Foundation for the Service-Oriented Architecture

Peter Massuthe Karsten Schmidt

Humboldt-Universität zu Berlin
Institut für Informatik
Unter den Linden 6
D-10099 Berlin
{*massuthe, kschmidt*}*@informatik.hu-berlin.de*

Abstract

In the service-oriented architecture (SOA), we distinguish three roles of service owners: service providers, service requesters, and service brokers. Each service provider publishes information to the broker about how requesters can interact with its service. Thus, the broker can assign a fitting service provider to a querying requester.

We propose the information published to the broker to be operating guidelines*. Operating guidelines are essentially communication instructions for the service requester. We present an automata-theoretic approach that is centered around operating guidelines and is capable of implementing all tasks arising in the SOA.*

Keywords: Service-oriented architecture, service composition, operating guidelines, matching, automata

1. Introduction

A *service* is an artifact that consists of an interface and internal control. The *service-oriented architecture* (SOA) [1] provides a framework for the interaction of services. It distinguishes three roles of service owners: *service provider*, *service broker*, and *service requester* (see Fig. 1). It postulates a general protocol for interaction: A service provider registers at the service broker by submitting information about how to interact with its service. The service broker manages such information about all registered service providers and allows a service requester to *find* an adequate service provider. Then, the service of the provider and the service of the requester may *bind* and start interaction.

The interaction of services may cause nontrivial communication between a requester and a provider. Consider, as a running example for a provided service, a vending machine as depicted in Fig. 2. If this service is bound to a requester's service, the requester must send a coin (€), press a button (T or C), and finally receive a beverage (B).

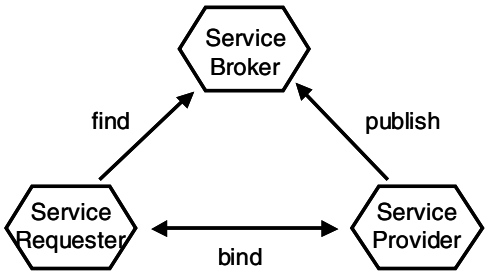

Figure 1. The service-oriented architecture.

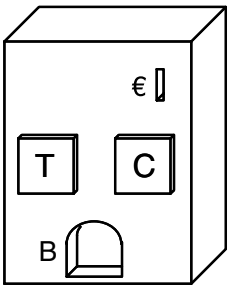

Figure 2. A vending machine that sells, for 1 Euro, either a cup of tea (press T), or a cup of coffee (press C).

It is obviously desirable that a service requester gets assigned only such a service provider that their services do not ill-communicate with each other (such as running into a deadlock or sending unanticipated messages). In our example, the broker must not deliver our vending machine service to a requester that wants to pay in other currencies than

€ or a requester who expects the beverage before paying.

For this purpose, the service broker needs information about the internal control structure of the provider's service – the provider's interface only (as its WSDL specification for example) is not sufficient. Publishing the whole internal control to the service broker would solve the problem. This is, however, not feasible, as the service provider may want to keep its internal structure secret.

We propose the published information to be an *operating guideline* for the provider's service P. The operating guideline for P essentially represents, in a condensed form, the set of *all* well-communicating services R of requesters of the service P. Thus, the operating guideline for the vending machine in our example would cover requesters that pay in €, but not a requester who pays in £, for instance. The concrete formalization of the concept of operating guidelines follows in Sec. 3.

As an alternative to this approach, it has been suggested to condense the internal control of the provider's service to an abstracted version, i.e. a *public view* [2, 3], of that service and to send this public view to the service broker. The essential difference between operating guidelines and public views is that a public view describes the service of the *provider* while an operating guideline describes the services of well-communicating *requesters*, instead.

Selecting an adequate service provider for a querying requester is a tedious task for the broker. Our operating guidelines approach reduces the complexity of that task: In the public view setting, the broker must perform a *compliance check*, i.e. a check whether the composed system of public view and requester's service behaves well, e.g. does not contain a reachable deadlock. The complexity of the compliance check is in the order of the *product* of the sizes of the requester and the provider. In the operating guideline approach, the broker must solve a matching problem, i.e. a check whether the requester's service "follows" the operating guideline. The complexity of matching is basically in the order of the size of the requester's service only. Due to space limitation, we cannot compare operating guidelines with public views in detail and refer to [4] for such a discussion. For further supporting our approach, we argue, however, that operating guidelines are a well-established approach in everyday life: Instructions that can be found at a real vending machine describe, step by step, the *customer*'s intended behavior and not at all an abstract description of the internal behavior of the *vending machine*.

The rest of the paper is structured as follows: In Sec. 2, we present our formal foundation for services and their interaction via asynchronous communication. Section 3 is devoted to operating guidelines. We give a formal definition for operating guidelines and show how they can be used to derive all well-communicating service requesters for a provider. In Sec. 4, we state the existence of unique operating guidelines for a given provider and show how its operating guidelines can be automatically generated. Section 5 sketches extensions of our approach and, finally, Sec. 6 concludes this paper.

2. Formalization of services

In our approach, we assume that services are essentially automata. In this paper, we restrict ourselves to finite and acyclic automata. These automata consist of states and transitions. Transitions are labeled with letters preceded by a question mark or an exclamation mark. Label $!x$ represents generation (sending) of an item x (which can represent a message or a real trade item). Label $?x$ represents consumption (receiving) item x. We require that, inside one and the same automaton, a letter occurs either everywhere with question mark, or everywhere with exclamation mark. A formal model of our vending machine could thus look like Fig. 3. Please note that, in this example, we do not distinguish the delivering of tea from the delivering of coffee.

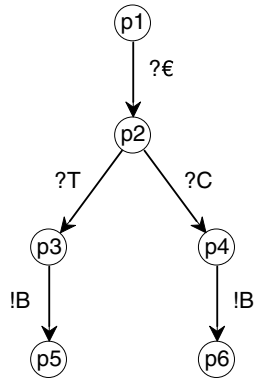

Figure 3. A formal description of the vending machine service. First, the automaton requires an Euro. Then, in state p2, it accepts the buttons T or C. Finally, it generates the beverage B.

The service of a requester can be modeled in the same way. Figure 4 depicts some services of possible requesters of our vending machine. The automata (a), (b), and (c) communicate well with the vending machine service, while (d) does not.

Two services fit syntactically, if each letter preceded by exclamation mark in one of them, occurs preceded by question mark in the other, and vice versa. This way, communication channels between the two services are modeled. As introduced in the previous section, we assume that all communication between services is asynchronous. This com-

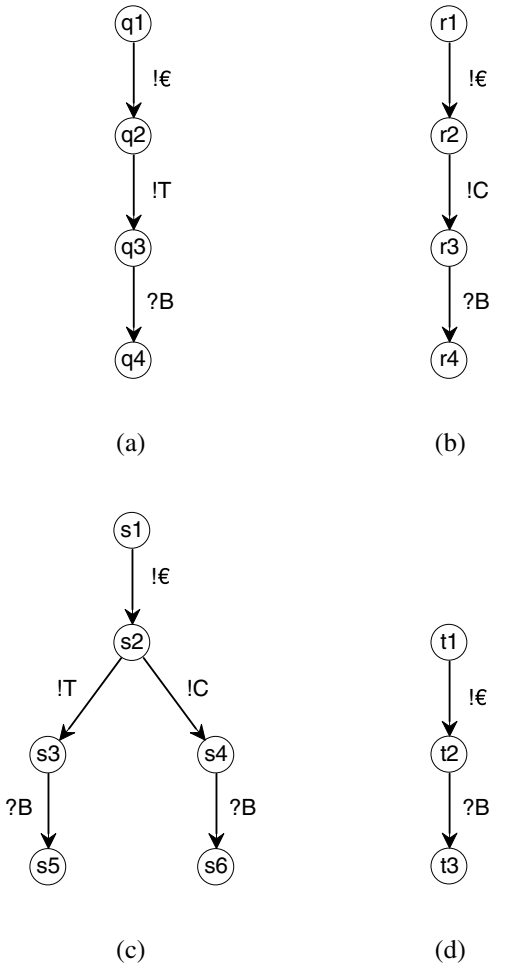

Figure 4. (a) A service requester that wants to buy tea. (b) A requester that wants to buy coffee. (c) A requester that decides about its requested beverage after payment. (d) An ill-communicating requester.

munication behavior is formalized in the following.

The system composed of two automata is a transition system. A node of the transition system is comprised of three components. The first two components correspond to states of the involved service automata, and the third component represents the state of the message channels, i.e. a multiset (bag) of all (pending) messages. A transition in the composed system corresponds to a transition in one of the involved services. If there is a transition from state x to state y labeled $!a$ in the first service, then there is a transition labeled $!a$ from $[x, z, M]$ to $[y, z, M \oplus a]$ in the composed system. If there is a transition from x to y labeled $?a$ in the first service, and M contains a, then there is a transition labeled $?a$ from $[x, z, M]$ to $[y, z, M \ominus a]$ in the composed system. Transitions in the second service define transitions in the composed system accordingly.

Figure 5 shows the vending machine service of Fig. 3 composed with two requesters: Figure 5 (a) shows the composed transition system of Fig. 3 and Fig. 4 (a), and Fig. 5 (b) shows the composed system of Fig. 3 and Fig. 4 (d).

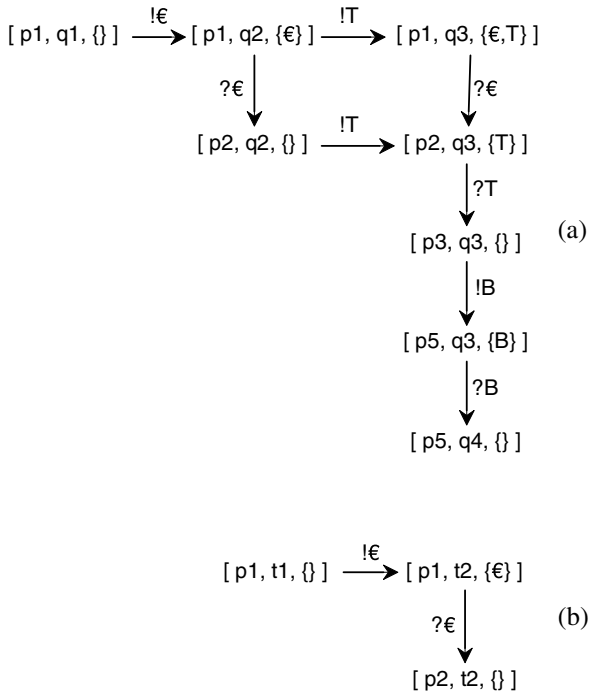

Figure 5. System composed of the vending machine (Fig. 3) and (a) the automaton in Fig. 4 (a), and (b) the automaton in Fig. 4 (d).

The node [p1, q1, {}] in the transition system in Fig. 5 (a) means that the first component, i.e. the vending machine, is in its state p1, the second component, i.e. the tea requester in Fig. 4 (a), is in its state q1, and there are no messages pending. Since it was possible for the requester to send a coin in state q1, it is possible in [p1, q1, {}], too. Thus, a new node is reached, where the requester is in state q2 and there is an Euro pending to be consumed by the machine.

The example Fig. 5 (a) shows that, in the composed system, one can, from every node, reach a node where both involved services are in their respective end states, and there are no pending messages left. Each pair of services exhibiting this property is called *well-communicating*, otherwise it is called *ill-communicating*. Figure 5 (b) shows the composed system of two ill-communicating services.

In the following, we show how a *set* of services can be represented. The representation of the set of *all* well-communicating requesters is then called *operating guideline* for the provider.

3. Operating guidelines

Consider a provider's service P, given as an automaton. The purpose of the operating guideline for P is to represent, in a condensed form, the set of *all* services R that communicate well with P. This condensed representation is to be published to the service broker.

Our tool for representing sets of services in a condensed form are *annotated automata*. Such automata have already been introduced in [6]. There, the authors propose the annotations are part of a modeling process. In our approach, we compute specific annotations to an already given automaton that represent exactly the set of sub-automata that have the well-communication property.

An annotated automaton is an automaton as described in Sec. 2, with additional labels for states. The label of a state x is a boolean formula with propositions corresponding to labels at transitions leaving x. Fig. 6 shows an annotated automaton.

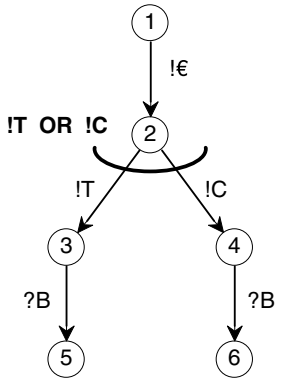

Figure 6. An annotated automaton representing the automata in Fig. 4 (a)-(c). Labels for states with less than 2 successors are skipped. The annotation of a state with one outgoing transition is the label of the transition; the annotation of a state with no successors is *true*.

An annotated automaton represents the set of (no longer annotated) automata that can be obtained by removing transitions according to the following rule: A transition leaving state x may only be removed as long as the remaining transitions leaving x still satisfy the formula annotated to x. To be more precise, we consider an assignment to the propositions that assigns *true* to all present leaving transitions and *false* to all absent transitions and apply this assignment to the formula attached to x.

For example, the annotated automaton in Fig. 6 represents exactly the set of the three automata in Fig. 4 (a)-(c).

The operating guideline for a service P is an annotated automaton, such that it represents *exactly* the set of services that communicate well with P.

It is thus easy to see that, given the operating guideline for P, it is easy to check if a given service R communicates well with P. We just need to check whether it is a subtree such that the annotations of P's operating guideline are satisfied.

For example, the requesters in Fig. 4 (a)-(c) are subtrees of the annotated automaton in Fig. 6, whereas the requester in Fig. 4 (d) is not. Thus, the service broker easily knows that the vending machine will behave well with requesters (a)-(c), but not with requester (d).

4. Justification

The definition of operating guidelines presented in the previous section assumes that it is always possible to represent all well-communicating services in a single annotated automaton. In this section we argue that this is actually the case. We thereby rely on results proven and reported in [5].

In this report, we prove that, for every automaton P, there is, up to isomorphism, a unique *most permissive* partner automaton R, such that R and P are well-communicating. Thereby, an automaton A is more permissive than an automaton B if A's set of states is a superset of B's set of states, and A's set of transition is a superset of B's set of transitions. In Fig. 4, the automaton (c) is more permissive than the ones in (a) and (b). The *most* permissive well-communicating partner automaton for our vending machine service (Fig. 3) is even more permissive than Fig. 4 (c) and is depicted in Fig. 7. The possibility to first press a button and afterwards inserting a coin results from the proposed asynchronous communication.

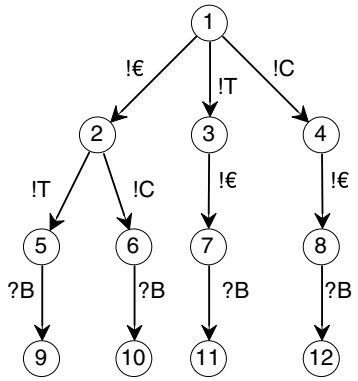

Figure 7. Most permissive well-communicating partner of the vending machine service in Fig. 3.

[5] sketches an algorithm to automatically generate the most permissive well-communicating partner for P.

Then, operating guidelines can be computed from the most permissive partner by properly annotating its states. This annotation is based on a state-by-state characterization of well-communicating partners proven in [5]. A detailed description of the annotation process can be found in [4]. Figure 8 shows the resulting operating guideline for the service in Fig. 3 after annotating the most permissive automaton. Thus, the three requesters (a)-(c) in Fig. 4 are well-communicating partners for our vending machine, but not the only ones.

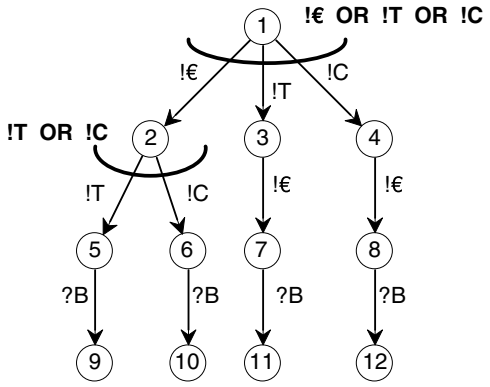

Figure 8. The operating guideline for the vending machine (Fig. 3). Again, annotations for states with less than 2 successors are skipped.

The operating guideline of our vending machine demonstrates that operating guidelines actually hide internal behavior of the vending machine. The sequential order of the first two events of the vending machine automaton is not manifested in the operating guidelines. Instead, one can deduce *all* well-behaving automata for the vending machine: A requester who first presses a button and then inserts a coin may be unexpected, but is well-behaving, too. Therefore, the service broker has not to deny the provided vending machine service to this requester.

5. Extensions

The approach in this paper is restricted to

- acyclic automata,
- deterministic automata without internal, i.e. τ-, transitions,
- services provided to a single requester.

The last two restrictions have been inserted, however, for simplicity only. [5] and [4] sketch approaches to the case where a service provider is to be connected to *several* service requesters that act independently of each other. Current work showed that it is possible to match arbitrary acyclic automata (i.e. automata with τ-transitions and states with multiple outgoing transitions with equal labels) with our deterministic operating guidelines.

Dropping the acyclicity restriction is subject to ongoing research, but our preliminary results show that operating guidelines can as well be defined for automata containing cycles. In particular, there exists some kind of most permissive partner services, too.

6. Conclusion

We introduced a formal approach to the service-oriented architecture that is based on automata. Both service provider *and* service requester are modeled as such automata. The composition of two automata results in a transition system, modeling the asynchronous communication between the services. Then, we introduced the concept of annotated automata as a condensed form to represent sets of automata. We argued that the operating guideline for a service provider, i.e. the annotated automata representing exactly the set of all well-communicating partners, is a suitable and elegant artifact to be published to the service broker. With the help of operating guidelines, the broker can easily match a provider with a querying requester.

We provided an approach on a solid theoretic basis. All constructions and matchings can be performed fully automatically and without giving away the internal structure of the provider.

In this paper, we only studied acyclic systems and restricted ourselves to interaction between one provider and only one requester. Current research activities concern operating guidelines for multiple partners and systems with cycles.

References

[1] Karl Gottschalk. Web Services Architecture Overview. Ibm whitepaper, IBM developerWorks, 01 September 2000. http://ibm.com/developerWorks/web/library/w-ovr/.

[2] F. Leymann, D. Roller, and M. Schmidt. Web services and business process management. *IBM Systems Journal*, 41(2), 2002.

[3] A. Martens. *Verteilte Geschäftsprozesse - Modellierung und Verifikation mit Hilfe von Web Services*. PhD thesis, Institut für Informatik, Humboldt-Universität zu Berlin, 2004.

[4] P. Massuthe and K. Schmidt. Operating guidelines - an alternative to public view. Techn. Report 189, Humboldt-Universität zu Berlin, 2005.

[5] K. Schmidt. Controllability of business processes. Techn. Report 180, Humboldt-Universität zu Berlin, 2004.

[6] A. Wombacher, P. Fankhauser, B. Mahleko, and E. Neuhold. Matchmaking for business processes based on choreographies. *International Journal of Web Services*, 1(4):14–32, 2004.