

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

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

May 21, 2005

## Abstract

*In the service-oriented architecture (SOA) [1], we distinguish service providers, service requestors, and service brokers. Each service provider publishes information to the broker about how requestors can interact with itself. Thus, the broker can assign a fitting service provider to a querying requestor.*

*We propose that the information published to the broker be operating guidelines. Operating guidelines are essentially communication instructions for the service requestor. 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 artefact that consists of an interface and internal control. The *service-oriented architecture* (SOA) provides a framework for the interaction of services in three roles: the *service provider*, the *service broker*, and the *service requestor* (see Fig. 1). It postulates a general protocol for interaction: A

service provider registers at a service broker by submitting information about how to interact with itself. The service broker maintains such information about all registered service providers. A service requestor queries the service broker for a service provider that fits its needs. The service broker selects, from the registered providers, a fitting one and returns links such that requestor and provider can be bound to each other and the provided service can be invoked by the service requestor.

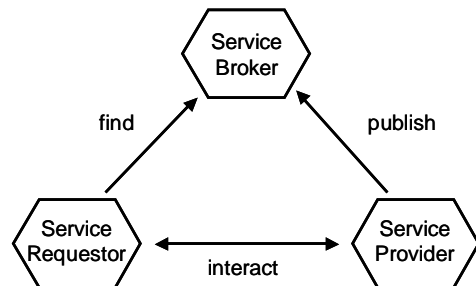


Figure 1: The service-oriented architecture.

Invoking such a provided service may cause non-trivial communication between the requestor and the provider. Consider, as an example for a provided service, a vending machine as depicted in Fig. 2. If this service is bound to a service requestor, the requestor

must send a coin, push a button (T or C), and finally receive a beverage. Fig. 3 depicts the behaviors of some possible service requestors. The requestors (a), (b), and (c) communicate well with the vending machine service, while (d) does not.

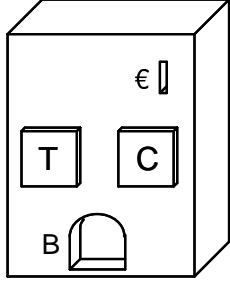


Figure 2: A vending machine that sells, for 1 Euro, either a cup of tea (push "T"), or coffee (push "C").

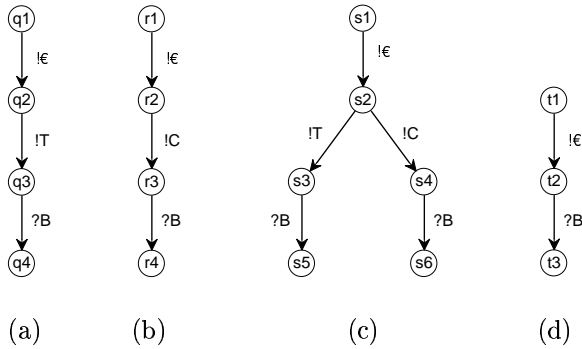


Figure 3: (a) (The behavior of) a service requestor that wants to buy tea. (b) A requestor that wants to buy coffee. (c) A requestor that decides about its requested beverage after payment. (d) An ill-communicating requestor.

It is obviously desirable that a service requestor gets assigned only such a service provider that they do not ill-communicate with each other (such as running into a deadlock or sending erroneous messages). In our example, the broker must not deliver our vending machine service to the requestor from Fig. 3 (d) (but there could be another registered service provider that fits). For this purpose, the service broker needs information about the internal of the ser-

vice provider - just the provider's interface (as its WSDL specification for example) is not sufficient. Publishing the whole internal control of the service provider 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 that the published information be *operating guidelines* for the service provider. Operating guidelines essentially represent, in a condensed form, the set of *all* well-communicating requestors. Thus, the operating guidelines for the vending machine in our example would cover the behaviors in Fig. 3 (a)-(c), but not the requestor (d)'s behavior. The concrete formalization of the concept of operating guidelines can be found in Sec. 3.

As an alternative to this approach, it has been suggested to condense the internal behavior of the service provider to an abstracted version, i.e. the *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 view is that a public view describes the behavior of the service *provider* while operating guidelines describe the behavior of the service *requestors*. Due to space limitation, we cannot compare operating guidelines with public views in detail and refer to [5] for such a discussion. For 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 itself.

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. Sec. 3 is devoted to operating guidelines. We give a formal definition for operating guidelines and show how they can be used to match well-communicating service requestors and providers. 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. Sec. 5 sketches extensions of our approach and 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 labelled 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). We require that, inside one and the same automaton, a letter occurs either everywhere with question mark, or everywhere with exclamation mark. Label  $?x$  represents consumption (receiving) item  $x$ . A formal model of our vending machine could thus look like Fig. 4. Please note that, in this example, we do not really distinguish the delivering of tea from the delivering of coffee.

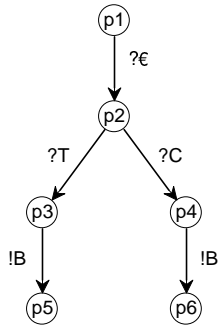


Figure 4: A formal description of the vending machine's internal behavior. First, the automaton requires a coin. Then, in state  $p_2$ , it accepts the buttons "T" or "C". Finally, it generates the beverage.

A service requestor can be modelled in the same way (as we have already done in Fig. 3).

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 modelled. As introduced in the previous section, we assume that all communication between services is asynchronous. This communication behavior is formalized in the following.

The system composed of two automata is again an

automaton. The nodes of the composed system are comprised of three components. The first two components are nodes of the two involved services, and the third component is 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  $x$  to  $y$  labelled  $!a$  in the first service, then there is a transition labelled  $?a$  from  $[x, z, M]$  to  $[y, z, M \oplus a]$  in the composed system. If there is a transition from  $x$  to  $y$  labelled  $?a$  in the first service, and  $M$  contains  $a$ , then there is a transition labelled  $!a$  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.

Fig. 5 shows the vending machine service of Fig. 4 composed with two requestors of Fig. 3: Fig. 5 (a) shows the composed automaton of Fig. 4 and Fig. 3 (a), and Fig. 5 (b) shows the composed automaton of Fig. 4 and Fig. 3 (d).

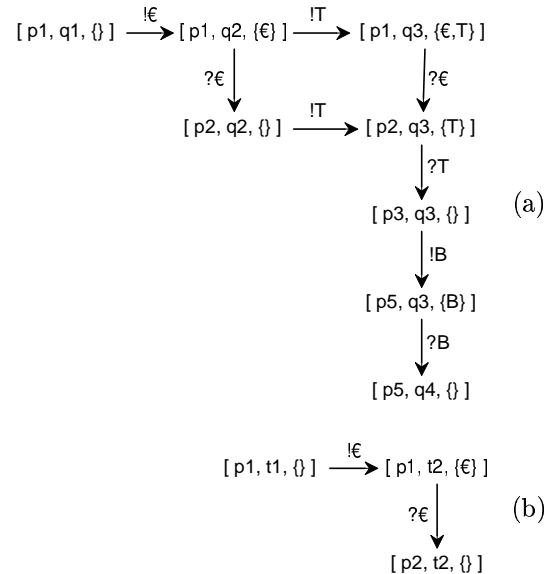


Figure 5: System composed of the vending machine (Fig. 4) and (a) the automaton in Fig. 3 (a), and (b) the automaton in Fig. 3 (d).

The node  $[p_1, q_1, \{\}]$  in the composed automaton 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 requestor in Fig. 3 (a), is in its state  $q1$ , and there are no messages pending. Since it was possible for the requestor to send a coin in state  $q1$ , it is possible in  $[p1, q1, \{\}]$ , too. Thus, a new state is reached, where the requestor is in state  $q2$  and there is an Euro pending to be consumed by the machine.

The example (a) shows that, in the composed system, we can, from every state, reach a state 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*. Fig. 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 requestors is then called *operating guidelines* for the provider.

### 3 Operating guidelines

Consider a service provider  $P$ , given as an automaton. The purpose of operating guidelines 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], but for different purpose.

An annotated automaton is an automaton as described above, with additional optional labels for nodes. The label of a node  $x$  is a boolean formula with propositions corresponding to labels at transitions leaving  $x$ . Fig. 6 shows an annotated automaton.

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 node  $x$  may only be removed as long as the remaining transitions leaving  $x$  still satisfy the formula annotated to  $x$ . To be more precise, we

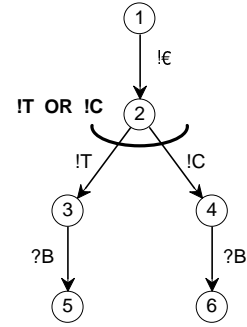


Figure 6: An annotated automaton representing the automata in Fig. 3 (a)-(c).

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 the set of the three automata in Fig. 3 (a)-(c).

Operating guidelines of 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 operating guidelines of  $P$ , 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 guidelines are satisfied.

For example, the requestors in Fig. 3 (a)-(c) are subtrees of the annotated automaton in Fig. 6, whereas the requestor in Fig. 3 (d) is not. Thus, the service broker easily knows that the vending machine will behave well with requestors (a)-(c), but not with requestor (d).

### 4 Justification

The definition of operating guidelines given 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 [4].

In this report, we prove that, for every service  $P$ , there is, up to isomorphism, a *unique* most permissive well-communicating partner service  $R$  of  $P$ . 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. 3, the automaton (c) is more permissive than the ones in (a) and (b). The *most* permissive well-communicating service for our vending machine service (Fig. 4) is even more permissive than Fig. 3 (c) and is depicted in Fig. 7. The possibility to first push a button and afterwards inserting a coin results from the proposed asynchronous communication.

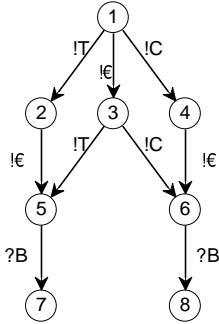


Figure 7: Most permissive well-communicating partner of the service in Fig. 4.

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

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

The operating guidelines of the vending machine show that they actually hide internal behavior of the

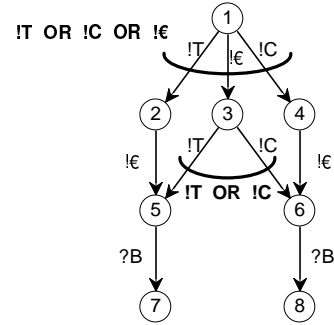


Figure 8: Operating guidelines for the service in Fig. 4.

vending machine. The sequential order of the first two events of the vending machine is not manifested in the operating guidelines.

## 5 Extensions

The approach in this paper is restricted to

- automata without internal, i.e.  $\tau$ -, transitions
- acyclic automata
- services provided to a single requestor

The last restriction has been inserted, however, for better readability. [4] and [5] sketch approaches to the case where a service provider is to be connected to *several* service requestors that act independently of each other. Dropping the remaining restrictions 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 exist most permissive partner services, too.

## 6 Conclusion

We showed a formal approach to the service-oriented architecture that is based on automata. Both service provider *and* service requestor are modelled as such automata, as well as their composition. Then, we

introduced the concept of annotated automata as a condensed form to represent sets of automata. We argued that operating guidelines 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 a service broker. With the help of operating guidelines, the broker can easily match a provider with a querying requestor.

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 requestor. Current research activities concern operating guidelines for multiple partners and systems with cycles.

## References

- [1] K. Gottschalk. *Web Services architecture overview*. IBM developerWorks, Whitepaper, September 2000.
- [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. WiKu-Verlag, Stuttgart.
- [4] K. Schmidt. Controllability of business processes. Techn. Report 180, Humboldt-Universität zu Berlin, 2004.
- [5] P. Massuthe and K. Schmidt. Operating Guidelines - an Alternative to Public View. Techn. Report 189, Humboldt-Universität zu Berlin, 2005.
- [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.