Abstract Large-scale service-oriented computing is based on the idea that services from various servers are combined into one distributed application. Referring to a collection of services on one server as a “service cloud”, the problem investigated in this paper is to define formal high-level specifications of such distributed applications and to enable the location of suitable services for them. Based on the language-independent model of Abstract State Services (AS²s), which serves as a universal integrated model for data and software as services, we extend AS²s by high-level action schemes called “plots” as a means to specify permitted sequences of service operations. On these grounds, we develop a model for service mediators, that is, specifications of composed services in which service slots have to be filled by actual services, and investigate matching conditions for slots of mediators and services. For a services to match a slot in a mediator, a (generalised) projection of the mediator must comply with the plot of the service. Furthermore, the service must be semantically adequate, which requires the use of a service ontology.

Keywords Service cloud · Abstract state service · Service-oriented computing · Service mediation · Service ontology

1 Introduction

A common slogan in cloud computing claims that “everything becomes a service”, which suggests to consider cloud computing as an umbrella for service orientation in the large with the world-wide web as the key medium. In fact, it would be possible to consider clouds as repositories of software services that are built on top of platforms and infrastructure services. Key differences arise with respect to ownership and usage rights. For instance, in the “software as a service” model (SaaS), everything is owned by the service provider, whereas in the “infrastructure as a service” model (IaaS), only the basic hardware infrastructure is owned by the provider, but used by the client, whereas the software is owned by the client. Whether this software is again offered as a service and thus used by others is a decision of the client who could at the same time become a software service provider. Therefore, in this paper, we adopt a loose use of the term “service cloud” to refer to a repository of data and software services well knowing that there is more to cloud computing than only a functional view of services.

Despite this big interest in the area, and the many ideas and systems that have been created, many fundamental questions have still not been answered. For instance, a web service could be almost anything, a simple function, a data warehouse
or a fully functional Web Information System, as long as it is made available via the world-wide web. The unifying characteristic is that content, functionality and sometimes even presentation are made available for use by human users or other services. However, the commonly used notion of web service would not capture all of these.

1.1 Our contribution

In this article, we first extend the formal model of Abstract State Services (AS2s) by formalising the notion of plot of a service, which specifies algebraically how a service can be used. This was left implicit in the original work on AS2s [21]. The notion of plot is a term adopted from the movie business that has already been used for long time in the context of Web Information Systems [25]. It actually captures the possible sequencing of service operations, which is only implicitly present in the AS2 model. For this, we exploit Kleene algebras with test, which are known to be the most expressive formalism to capture propositional process specifications. So, adding plots to the AS2 model is a little new extension.

The key contribution of this paper, however, is the introduction of service mediators, which mediate the collaboration of services. For this, we exploit plots with open slots for services to specify intended service-based applications on a high level of abstraction. The novel idea is to specify service-oriented applications that involve yet unknown component services. We then formally define matching criteria for services that are to fill the slots. A problem in finding such matching criteria is the fact that we would like to be able to skip component operations of services and change their order.

This enhances the work on service composition, which is already a well-explored area in service computing with respect to services that are understood functionally. In the AS2 model, this corresponds to the service operations rather than the services as a whole. More precisely, what we actually need to compose are “runs” of services that are determined by the plot. What makes our contribution even more interesting is that we investigate conditions, under which particular service operations can be removed or their order can be changed. This leads to the rather complicated matching conditions between services and slots in mediators.

Finally, slots in mediators only make sense if the services that could match them are located somehow. Here, we adapt the idea of a service ontology, which is already omnipresent in the area of the semantic web, also in our previous work in [22]. In the paper, we exploit a variant of DL_LITE, but other description logics could be used in the same way.

1.2 An application scenario

Suppose we want to develop an integrated complete service for conference trip organisation. To make it simple, we anticipate only three parts: registration for the conference, booking of accommodation and booking of the travel—for simplicity let this be only a flight. We can imagine that for all these components, web-based services exist and how these are organised. A rough picture would be the following:

- For registration, we would have to provide personal data, author information and possible discounts, if applicable plus payment information.
- For flight booking, we have to search for flights, select the most suitable one and provide again personal data and payment information, say credit card details.
- For accommodation booking, we have to search for available hotels, select a hotel and a room and provide again personal data and payment information.

We observe some particularities here. For flight and accommodation booking, there may be several competing services. As we do not care which one should be used, we would have to access these services in parallel for the search, combine the results and continue with one of the services only, once a selection has been made. Furthermore, all the services involved contain redundant parts for entering personal data and credit card details, but we only want to provide such data once. A solution might be to collect these data locally and to push them through to the corresponding services when required.

Next, we have the choice to either adopt a bottom-up or a top-down approach to system specification. In the former case, we would search for suitable services and compose them. In the latter case, we try to specify the composed system with slots that are to be filled in by not yet known services. Let us concentrate on the top-down approach.

In this case, we specify a mediator consisting of local components and yet unknown services, each of which providing some service operations. In our scenario, we would need one conference registration service and several services for flight and accommodation booking. The mediator would have to specify the flow of data in and out of the services. Figure 7 illustrates this idea for our application scenario. It can be seen that ideally it will be necessary that services can interact with each other and that they employ several service operations users or other services can interact with.

The next step would be to search for services that match the “slots” in the mediator. For matching, we have to fulfil functional conditions regarding the input and output, conditions that refer to the application domain (so we really get flight booking services) and conditions that align the flow of data in the mediator with those in the individual services. Figure 4 illustrates the flow of data within the individual services, so the projection of the data flow from the mediator must be compatible with this.

In addition to matching slots in the mediator with suitable services, we have to make a selection, as there may