On interoperability and conformance assessment in service composition

Dick Quartel\textsuperscript{a,b} and Marten van Sinderen\textsuperscript{a}

\textsuperscript{a}Centre for Telematics and Information Technology
\{D.A.C.Quartel, M.J.vanSinderen\}@utwente.nl
\textsuperscript{b}Telematica Instituut
\{Dick.Quartel\}@telin.nl

Abstract

The process of composing a service from other services typically involves multiple models. These models may represent the service from distinct perspectives, e.g., to model the different roles of systems involved in the service, and at distinct abstraction levels, e.g., to model the service’s capability, interface or the orchestration that implements the service. The consistency among these models needs to be maintained in order to guarantee the correctness of the composition process. Two types of consistency relations are distinguished: interoperability, which concerns the ability of different roles to interoperate, and conformance, which concerns the correct implementation of an abstract model by a more concrete model. This paper discusses the need for and use of techniques to assess interoperability and conformance in a service composition process. The paper shows how these consistency relations can be described and analysed using concepts from the COSMO framework. Examples are presented to illustrate how interoperability and conformance can be assessed.

1. Introduction

In previous work [18] we introduced a conceptual framework for the modelling of services, named COSMO. This framework presents a set of concepts that allows one to model services (i) at different abstraction levels, e.g., the service’s goal or capability, its interface and the orchestration that implements the service, and (ii) from different perspectives, e.g., the role (participation) of the user or the provider in the service.

In this paper we show how these concepts and models can be applied for service composition. We consider service composition as the process of composing a (new) service from multiple existing services. In particular, we are interested in the consistency among the various models that are used in the composition process.

Two types of consistency relations are considered: interoperability and conformance. The interoperability relation relates two models representing different system roles. It defines that these roles must be able to interoperate, such that they can execute the service in cooperation and establish some effect. The conformance relation relates two models at different abstraction levels. It defines that a model at a lower abstraction level must be a correct implementation of a model at a higher abstraction level.

To guarantee the correctness of the service composition process, one needs to be able to assess the interoperability and conformance relation. This holds irrespective of the composition approach that is followed, e.g., ranging from manual composition to automated techniques involving service discovery.

The goal of this paper is twofold. First it discusses the use of interoperability and conformance relations in the service composition process, and the need to be able to assess these relations. And second, it explains how the concepts from the COSMO framework can be used to model these consistency relations. Based on these concepts, techniques are described to assess interoperability and conformance between service models.

This paper is further structured as follows. Section 2 briefly introduces the COSMO framework. Section 3 describes the models involved in a service composition process, including the interoperability and conformance relations that must hold between these models. Sections 4 and 5 explain how the interoperability and conformance relation can be described using concepts from the COSMO framework, respectively, and discuss techniques to assess these relations. Section 6 illustrates our ideas with an example. And section 7 discusses related work and presents our conclusions.

\textsuperscript{1} This work is part of the Freeband A-MUSE project (http://a-muse.freeband.nl), which is sponsored by the Dutch government under contract BSIK 03025.
2. Service modelling

We define a service as the establishment of some effect through the interaction between two or more systems. The COSMO framework defines concepts for the modelling of a service at different abstraction levels and from the perspective of the different system roles involved in a service. Furthermore, the framework considers different categories of service properties, denoted as service aspects. In this paper we focus on the behaviour and information aspects of services.

2.1. Abstraction levels

We consider three generic abstraction levels to model services: as a single interaction, as a choreography of multiple related interactions, and as an orchestration of multiple (sub-)services.

Single interaction. At a high abstraction level a service can be modelled as a single interaction between two or more systems. This interaction represents an activity in which the involved systems produce some common result in cooperation. At this abstraction level, we are only interested in what result(s) can be produced, and not in how this is done. Consequently, an interaction is considered an atomic activity that either occurs and establishes the same result for all involved systems, or does not occur for any of the systems and therefore does not establish any result.

The interaction result represents the effect of the service. Each system may have different expectations on this effect, and therefore impose different constraints on the interaction result. This is modelled by defining an interaction as the composition of two (or more) interaction contributions, one for each involved system. An interaction contribution represents the participation of a system in the interaction, by defining the constraints this system has on the possible interaction result, and thereby its responsibilities in performing the interaction.

Interaction results can be represented using aforementioned information modelling concepts. For this purpose, one or more so-called information attributes are associated with an interaction. An information attribute has a type and will be assigned a value when the interaction occurs. The value is an individual that represents (part of) the interaction result constraint can be defined on an information attribute to constrain its possible values. This result constraint is a predicate (post-condition) that states the properties that have to be
satisfied by the individual that represents (part of) the interaction result. For example, the result constraint of interaction contribution reserve in Figure 1 specifies that the customer wants to eat Italian at 20.00h in a restaurant within 10 distance units (e.g., km) from his location. Furthermore, he wants to pay for the reservation not more than 5.00 euro, which should be charged via his ISP (Internet Service Provider). Constraints of the provider are that a booking should consist of at least one reservation and each reservation should be a reservation for a dinner or a taxi. Furthermore, payments can be made via ISP or credit card, but should be via ISP in case the amount is smaller than 2.50 euro.

The result constraints define the goal of the user and provider for participating in the service. This interpretation of goal corresponds to definitions found in requirements engineering literature ([19], [27]). Therefore, we also denote the single interaction model of a service as a goal model.

**Choreography.** In general, a service cannot be implemented as a single interaction and we have to refine the abstract interaction into a structure of multiple smaller more concrete interactions. Figure 3 depicts a possible refinement of the booking interaction into multiple related interactions: rvRest represents the reservation of a restaurant, payISP/CC represents the payment for the reservation, and cancel represents the cancellation of the reservation(s). Interaction pay is defined as two alternative interactions: one involving interaction contributions pay and payISP and the other involving interaction contributions pay and payCC. The provider offers the possibility to pay via the ISP through payISP, or by credit card through payCC. In addition, the provider offers the possibility to reserve a taxi through rvTaxi.

![Figure 3. Booking choreography](image)

Multiple related activities are represented by a behaviour, which is graphically expressed as a rounded rectangle. An activity can be an interaction contribution or action (see section 2.2). Relations between activities can be modelled in different ways, e.g., in terms of state transitions or temporal relations. We define relations in terms of causality. A causality relation links each activity to its causality condition, which defines how this activity causally depends on other activities. An activity is enabled, i.e., allowed to occur, if its causality condition is satisfied (i.e., a causality condition resembles a pre-condition). Three basic conditions of some activity a are distinguished (see Figure 4): (i) enabling condition b represents that activity b must have occurred before a can occur; (ii) disabling condition ~b represents that activity b must not have occurred before nor simultaneously with a to enable the occurrence of a; (iii) the start condition represents that activity a is enabled from the beginning of the behaviour and is independent of any other activity. These basic conditions can be combined using the conjunction and disjunction operators to represent more complex conditions. For example, workflow operators such as and-join, and-split and or-split can be represented using a combination of enabling and disabling conditions.

![Figure 4. Relations between activities](image)
don’t need this refinement (yet), since the orchestration abstractions from their implementation. The response interactions in bold correspond to the original interactions, since they model the same interaction result. The other interactions are inserted to model intermediate results in performing the original interactions.

The model in Figure 5(ii) is an abstraction of the orchestration in Figure 5(i), and will be explained later.

Figure 5. Booking orchestration

2.2. System roles

The service models in section 2.1 define the involvement, or role, of the user (customer) and provider in the service. The user and provider roles define two complementary perspectives on a service. The user perspective defines the participation of the user in the service, representing the expectations the user has on the effect, and thus on the service provider. This partial definition of the service is also called the requested service. The provider perspective defines the participation of the provider role, representing the expectations it has on the user. This partial definition of the service is also called the offered service.

A third perspective is the so-called integrated perspective, which defines the joint behaviour of the offered and requested service, abstracting from the distinction between a user and provider role. This more abstract definition of the service is called the integrated service. For this purpose, the action concept is introduced. An action represents an activity that is performed by a single system. Similar to the interaction concept, an action either occurs and establishes a result, or an action does not occur and establishes no (partial) result. Constraints can be attached to an action defining the possible results.

An action is used to represent an integrated interaction (service), by considering the systems that perform the user and provider roles as a single system. Furthermore, the constraints of the action are defined by the conjunction of the constraints defined by the (integrated) interaction contributions. Figure 6 depicts the integrated perspective of the booking interaction in Figure 1. The constraint in gray originates from the provider, but is implied by the constraints of the user.

![Figure 5](image)

Figure 6. Integrated booking interaction

As a second example, Figure 7 depicts the integrated perspective of the booking choreography in Figure 3, omitting the interaction attributes. The causality conditions of the actions are defined by the conjunction of the causality conditions of the corresponding interaction contributions. Pseudo-action rvTaxi is colored gray to represent that the user does not participate in this interaction, and therefore it can never occur. The integrated model clearly shows the consequence: the payment and cancel interactions can not occur because they depend on the taxi reservation interaction.

![Figure 7](image)

Figure 7. Integrated booking choreography

3. Service composition

We consider service composition as the process of composing a service from existing services to satisfy a request for some service. During the composition process, the abstraction levels and system roles that were presented in the previous section may have to be considered explicitly or implicitly. Figure 8 illustrates a structure of service models that represent these abstraction levels (vertical axis) and system roles (horizontal axis), including the relationships between these models.

The presented structure is independent of the approach that is followed to perform the service composition process. A possible scenario for service composition is illustrated in Figure 9. The scenario starts with a request for a service, which has as parameter a model of the goal and choreography of the requested service, where the choreography should be a correct refinement of the goal (condition 1). The purpose of the goal model is to facilitate the discovery of an offered service that satisfies the requested goal (condition 2). However, goal satisfaction does not imply that the requested service can interoperate with the offered service. Therefore, an additional check is needed to determine if the choreographies of both services match (condition 3). Optionally, one may also check if the offered
choreography is a correct refinement of the offered goal (condition 4), in case one can not trust this has been done before, e.g., upon registration of the service.

Figure 8. Models in service composition

If conditions 2 to 4 are satisfied, the offered service can be returned as an answer to the request. If not, or in case no offered service could be found, the offered service has to be composed based on the requested service models. We abstract from how a composition is created, and assume some technique exists to construct an orchestration for the offered service ([6],[21]). Such a technique may apply the composition process recursively when it decomposes the offered service into sub-services.

Ultimately, the resulting orchestration model must consist of matching sub-services (condition 5) and must conform to, i.e., be a correct refinement of, the offered choreography model (condition 6). The offered goal and choreography model can be derived in essentially two ways: (i) from the requested goal and choreography models, e.g., by taking their ‘complement’ behaviour, or (ii) from the orchestration model by abstracting from internal and interface behaviour (see section 5). Consequently, besides conditions 5 and 6, conditions 2 to 4 may need to be checked as well. If all these conditions are satisfied, the offered service can be returned as an answer to the request. Otherwise, an alternative (improved) orchestration model should be composed.

This paper focuses on the relationships between the service models that are used in the composition process. The consistency of these relationships need to be assessed in order to guarantee the correctness of some service composition. These relationships exist irrespective of the particular composition technique that is followed. We present in this paper concepts and techniques to describe and analyse these relationships independently of how the models are used by composition algorithms, thereby facilitating re-use of these concepts and techniques. Two types of relationships are distinguished: an interoperability and a conformance relation.

The interoperability relation defines whether some requested and offered service match, i.e., can interoperate in a meaningful way, or not. To establish if the interoperation is meaningful, different criteria can be chosen (see section 4). The integrated service model facilitates one to analyse these criteria, since it defines the actual interactions and effects that are made possible by both the requested and the offered service (as illustrated by the right-most column in Figure 8). The interoperability relation relates goal-to-goal and choreography-to-choreography models.

Figure 9. Composition scenario

The conformance relation defines whether a more concrete service model MC is a correct refinement of a more abstract service model MA. Similar to the interoperability relation, different conformance criteria can be chosen. The conformance relation relates goal-to-choreography, choreography-to-orchestration, and (indirectly) goal-to-orchestration models.

4. Interoperability relation

This section explains how interoperability requirements at goal and choreography level can be modelled using the concepts presented in section 2. In addition, techniques are discussed to assess interoperability using the resulting models.

4.1. Goal interoperability

The requirements that have to be satisfied to achieve interoperability at goal level, can be represented directly through the interaction concept. Interaction contributions and their associated result constraints represent the intended effect, or goal, of the user and the provider. The term capability is also used to denote the provider goal.

An interaction between a user and a provider is possible, if a common effect can be established that satisfies the constraints of both the user and provider. This means that the result constraints of the user and provider
must overlap. For example, the booking interaction of Figure 1 can occur if a booking can be established that satisfies the constraints of the user and the provider. In this case, the provider must e.g. be able to find an Italian restaurant in the vicinity of the user. Interaction would e.g. not be possible if the provider does not support payments via the ISP of the user.

The possible common effects are defined by the conjunction of the user and provider constraints. This is represented directly by the integrated service perspective. For example, Figure 6 models all possible effects (results) for the booking interaction of Figure 1.

Varying criteria for interoperability can be chosen. A minimum criterion is that at least a single common effect can be established. Alternatively, a more stringent criterion one might want to use is that the provider should support multiple, or even all effects requested by the user. Techniques that can be used and developed to assess such interoperability criteria depend on the languages that are applied to express the result constraints. In this paper, we use description logics, in particular OWL-DL ([11]), to represent these constraints. An alternative could be e.g. UML class diagrams in combination with OCL.

A basic reasoning function supported by OWL-DL is subsumption checking, which allows one to check if a class A (the subsumee) is a sub-class of class B (the subsumer). Figure 10 illustrates how this can be applied to assess interoperability. Classes A and B are defined to represent the set of individuals that satisfy the constraints of the user and provider, respectively. Now by checking if B subsumes A, one can assess if the provider supports all effects requested by the user; see Figure 10(i). To assess if the provider supports some or at least one of the requested effects, first a class is defined that represents the intersection of classes A and B, and subsequently it is checked if this class is satisfiable, i.e., can have individuals; see Figure 10(ii). Checking the (non-)satisfiability of some class is a special case of conjunction checking, by checking if the class is subsumed by the empty class.

![Figure 10. Reasoning in OWL-DL](image)

We have experimented with Protégé, Jena and SPARQL to represent and query information models in OWL-DL, and used Racer as a reasoning engine to perform subsumption checks ([16],[8],[17],[22]). Although we were able to verify our ideas using several (simple) examples, we believe practical applicability of these tools to realistic cases is still limited.

4.2. Choreography interoperability

Compared to goal interoperability, an additional concern to be considered for choreography interoperability is formed by the relationships that are defined between the interaction contributions of the user and provider. These relationships define constraints on the possible ordering of (and dependencies between) the interactions during an execution.

A common execution of the service is possible if the ordering (and result) constraints of both the user and provider can be satisfied. For example, considering the booking choreography in Figure 3, both the user and provider define that payment is only possible after one or more reservations have been made. However, the user allows a booking to be cancelled from the beginning, whereas the provider allows this only after the reservations have been made.

Similar to the goal level, the integrated service perspective allows one to define the possible common executions (behaviour) of a service. For example, Figure 7 depicts the integrated perspective of the booking choreography. This model shows that the ordering constraint on the cancellation of a booking is the same as the one defined by the provider, since it is more strict than the one defined by the user.

Again, varying interoperability criteria can be chosen at choreography level. A minimum criterion is that at least a single common execution is possible. However, as an alternative criterion, one could demand that multiple or all possible orderings requested by the user should be supported by the provider. Different techniques can be used to assess these interoperability criteria. For example, using the integrated perspective one may check if the conjunction of user and provider constraints result in impossible (conflicting) conditions for some of the interactions, such as cyclic dependencies (deadlocks) and dependencies on the occurrence and non-occurrence of the same (interaction). Some of these conflicts can be identified by calculating the transitive closure of causality conditions ([20]). Alternatively, one may simulate the integrated service model. For this purpose, we use the simulator for ISDL ([19],[7]), which is a language that supports the concepts from the COSMO framework.

Another technique we applied is based on the mapping of services modelled in ISDL onto Petri Nets ([23]). In this case existing tools can be used for reachability analysis to check if some service execution is possible.

5. Conformance assessment

This section explains how conformance requirements can be modelled using the concepts presented in section 2. In particular, we consider conformance of a choreography w.r.t. some goal, called goal conformance, and
conformance of an orchestration w.r.t. a choreography, called choreography conformance. Techniques are discussed to assess these types of conformance.

5.1. Goal conformance

When refining a goal model into a choreography model, the following types of refinement can be distinguished: information refinement, activity refinement and causality refinement.

Information refinement. Information refinement is used to model interaction results (and therefore the effect or goal of a service) in more detail. Two elementary forms of information refinement are specialization and property aggregation.

Specialization allows one to refine an abstract result of abstract type TA into a concrete result of concrete type TC, such that TC is a sub-type of TA. In terms of the associated information model this means that class TC is defined as a sub-class of class TA. For example, the provider goal in Figure 1 defines as interaction result as a specific type of Booking, namely a DinnerBooking.

Property aggregation allows one to refine an abstract result of abstract type TA into a collection of concrete activity results of concrete types TCi, such that each TCi represents a property of TA. In terms of the associated information model this means that class TCi represents the range of a property relation (different from the subclass relation) between TA and TCi. For example, the user goal in Figure 1 defines as interaction result the combination of a dinner reservation and a payment. This combination is a refinement of abstract type Booking, since a booking has among its properties one or more reservations and a payment.

Activity refinement. Activity refinement is used to model in more detail an activity that is represented by a single abstract action or interaction contribution (called abstract activity). This abstract activity is decomposed into a concrete activity structure, which consists of multiple related, more concrete (sub-)activities. The concrete activity structure makes its result available through the occurrence and associated information attributes of one or more of its final activities. A concrete activity structure can make its result available through the occurrence of (i) a single final activity, (ii) a conjunction of multiple, independent final activities, (iii) a disjunction of multiple, alternative final activities, or (iv) a combination of these options. These final activities are said to correspond to the original abstract activity, since they model the establishment of the result of the same activity. In this way, the final activities constitute reference points for comparing a concrete behaviour to an abstract behaviour to assess conformance, and are therefore also called reference activities.

For example, the choice between interaction contributions payISP and payCC in behaviour Provider of Figure 1 may be considered as a concrete activity structure that refines an abstract interaction contribution pay′, which allows both types of payment as interaction result. Contributions payISP and payCC form a disjunction of alternative final (reference) activities, since the occurrence of one of these activities corresponds to the occurrence of abstract activity pay′.

As another example, the sequential composition of interaction contributions pay and rvRest in behaviour Customer of Figure 3 may be considered as an activity structure with a single final activity that refines an abstract activity reserve′. Contribution pay is a reference activity that corresponds to reserve′, such that both establish the same interaction result. Behaviour Customer establishes this result in two steps, firstly rvRest establishes a reservation and secondly pay establishes the payment. Because pay depends on rvRest, it can refer to the result of rvRest and provide this result together with its own result for further reference to other activities. Figure 11(i) depicts the corresponding abstraction of behaviour Customer, with the information attributes of reserve′ being defined as the combination of the information attributes of pay and rvRest, including the conjunction of their result constraints.

Causality refinement. Causality refinement is used to model the relations between abstract activities in more detail through adding so-called inserted activities. These inserted activities typically represent intermediate results. Abstract activities are not further detailed. Causality refinement should obey the following conformance criteria: (i) an indirect relation between abstract activities defined via an inserted activity in the concrete behaviour must be equivalent to the relation defined directly between the abstract activities in the abstract behaviour;
and (ii) similarly, an indirect relation between attributes must be equivalent to the direct relation.

For example, when considering the orchestration in Figure 5(i) as a refinement of the provider choreography in Figure 3, the interaction contributions in bold-face are reference activities and all other contributions are inserted activities. For instance, reference contributions $\text{rspRest}$ and $\text{rspPayCC}$ correspond to abstract contributions $\text{rvRest}$ and $\text{payCC}$. Inserted contribution $\text{reqPayCC}$ represents an intermediate step between the reservation of a taxi and the payment of this reservation, thereby defining an indirect dependency between $\text{rspPayCC}$ and $\text{rspRest}$. Furthermore, the information attributes of $\text{rspPayCC}$ may refer via the attributes of $\text{rspRest}$ to the attribute values established in $\text{rspRest}$.

**Abstraction rules.** In order to assess the correctness of the refinement of an abstract service model into a more concrete service model, essentially two approaches can be followed: (i) predefining frequently-used types of refinement that are proven to be correct, or (ii) assessing the conformance of a refined model w.r.t. the original model. We currently follow the latter approach, which is more generic since it can be applied independently of the refinement that is made. For this purpose, so-called abstraction rules are used that define how to abstract again from the modelling details that have been added in the refinement step. The abstraction that is obtained this way should be equivalent to the original abstract model, otherwise the refinement is considered to be incorrect. This approach can easily be combined with the first approach, such that the conformance assessment step does not have to be performed each time again for common types of refinement. For a definition of the abstraction rules that have been defined for the types of refinement discussed above, we refer to [20].

### 5.2. Choreography conformance

A typical refinement of a choreography model is an orchestration model that consists of a coordinator (also called orchestrator) and multiple sub-services. The coordinator provides the offered service as defined in the choreography model by requesting services from other providers. These requested services may be defined at goal and (subsequently) at choreography level. Figure 5(i) depicts an example of such an orchestration model.

This refinement can be considered as being obtained through the application of causality refinement. The interactions between the coordinator and sub-services are inserted activities that represent intermediate results in the provisioning of the offered (abstract) service. Consequently, to assess whether the orchestration model correctly implements the choreography model, one has to abstract from the inserted interactions and validate whether the obtained abstraction is equivalent to the choreography model. This abstraction is calculated on the integrated orchestration model, since our abstraction rules apply to actions and interaction contributions.

Figure 5(ii) depicts the result of abstracting from the inserted activities, thereby replacing indirect dependencies between abstract activities by direct dependencies. Behaviour Coordinator is not equivalent (assuming strict equivalence) to the provider choreography in Figure 3, since it imposes an ordering among the restaurant and taxi reservations. The refinement has not obeyed the conformance criteria for causality refinement, since with the insertion of contribution $\text{reqTaxi}$ a dependency has been introduced on abstract contribution $\text{rspRest}$. Therefore behaviour Coordinator is considered an incorrect refinement of behaviour Provider in Figure 3.

### 6. Example

This section further illustrates our ideas about the representation and analysis of the interoperability and conformance of service models as explained in sections 4 and 5. This is done by considering part of the service composition scenario in Figure 9. First, we assume a service orchestration is proposed by the composition step and discuss the assessment of its correctness. Second, we consider the (recursive) application and assessment of the discovery and composition step during the creation of such a composition.

#### 6.1. Assessment of an orchestration

Suppose an offered service is requested that should match the user goal and choreography of Figure 1 and Figure 3, respectively. Figure 12 and Figure 13 depict the offered choreography and orchestration models that are returned by the composition step in Figure 9, respectively. This section focuses on the assessment of conditions (5) and (6). For brevity, we omit information attributes.

**Figure 12. Offered choreography**

The orchestration model is defined close to implementation level, modelling operations and their relationships (see also section 6.3). Figure 14(i) and (iii) introduce our shorthand notation for operations and message passing, respectively. Figure 14(iv) represents the interpretation of message passing in terms of our interaction concept, by making the role of middleware explicit. Figure 14(iv) represents a possible interpretation of the operation shorthand notation using message passing. However, our notation supports alternative interpretations. For instance, the catch and fail parts may...
be omitted to abstract from exceptions, or the reply and return part may be omitted to model one-way operations.

Figure 13. Orchestration model

The orchestration model models that first a list of restaurants is obtained through operation listRest. This operation is implemented using a Location service that returns the location of the user, and a Catalog service that allows one to search restaurants based on location and other criteria. Interaction with the catalog service is modelled by two one-way operations using call-back. Furthermore, the call-back should happen within a certain time-out period, otherwise action timeout occurs (the exact time constraints are omitted, but can be modelled using information attributes). In case a list of restaurants is returned in time, operation listRest may be followed by the reservation of a restaurant through operation reserveRest using the Reservation service, unless operation cancel is invoked first. The reservation service requires its user to check first the availability of a table, and secondly confirm the table, if available. The reservation may however be cancelled before and after confirmation through operation withdraw, which is initiated through operation cancel. Operation cancel represents the cancellation of a reservation. Its invocation may disrupt the selection and reservation of a table, and excludes the invocation of the pay(ment) operation. However, once the pay operation has been invoked, cancellation is no longer possible.

Condition (5) concerns the interoperability between the coordinator and the sub-services. Interoperability can be assessed for each sub-service separately, by considering only those operation calls and executions of the coordinator that interact with the sub-service, including their relationships. In this way, the interoperability requirement can be represented and analysed as discussed in section 4. For example, Figure 15 models the interoperability requirement between the coordinator and the reservation sub-service. Observe that to obtain this model one has to abstract amongst others from the indirect dependency between operation check, confirm and withdraw via operation cancel, and replace it by a direct dependency. Reachability analysis of the model learns that common executions are possible, e.g., check followed by withdraw, but also deadlock is possible in case the coordinator only wants to execute operation withdraw (without executing check before).

Figure 15. Interoperability with Reservation service

Condition (6) concerns the conformance of the orchestration model to the choreography model. As a first step (I) in assessing this condition, we construct an integrated orchestration model that is obtained by integrating the interactions of which an operation is composed (see Figure 14) into actions. These actions can be considered as being inserted during the refinement of the offered service choreography. Consequently, as a second step (II) we abstract from these actions using the rules for causality refinement. And as a final step (III), we have to assess the equivalence between the resulting model and the offered service choreography to decide whether the orchestration conforms to the choreography.

Figure 16(i) depicts an intermediate model, which is obtained after step (I) by
- firstly, considering the actions that belong to a single operation as an action structure, and applying the abstraction rules for activity and causality refinement (see section 5) to abstract each operation into a single action, and
- secondly, repeating this for actions that establish a common abstract result; e.g., operations loc, findReq and findRsp are abstracted into action find, which establishes a list of restaurants, and operations check and confirm are abstracted into action reserve, which establishes a table reservation.
Figure 16(ii) depicts the result of step (II). However, before this model can be compared to the more abstract model in Figure 12, we also have to abstract from operation listRest and from the invocations of the other operations, which can in this case all be considered as inserted activities. Figure 16(iii) depicts the result of this abstraction. The model is not equivalent to the model in Figure 12, since the causality conditions of contributions cancel and reserveRest/rvRest are different. The offered choreography only allows a cancellation after a reservation has been made, whereas the orchestration also allows a cancellation from the beginning. Observe also the alternative 'impossible' condition $X$ of contribution cancel, which is never satisfied. This condition originates (and abstracts) from the possible deadlock situation caused by the partial non-interoperability with sub-service Reservation in Figure 13, as explained before.

6.2. Assessment while creating an orchestration

The integrated model of Figure 16(i) is not only useful in the abstraction from orchestration to choreography, but also in the opposite direction, i.e., in the refinement of the choreography. W.r.t. the choreography model, the actions may be considered as inserted actions, which represent sub-goals in the process of providing the offered service. These sub-goals abstract from who will be responsible for fulfilling these goals, i.e., the coordinator or the sub-services. This model constitutes a useful initial refinement of the choreography, by defining and obtaining insight in what has to be done in the orchestration, while abstracting from how.

In a next refinement step, responsibilities have to be assigned. For example, action timeout may be assigned completely to the coordinator, or only partly. In the latter case, the action has to be decomposed into an interaction between the coordinator and some timer sub-service. Associated with this decomposition, the goal (or effect) requested by the coordinator and, optionally, the goal offered by the sub-services, should be defined.

Based on the assignment of responsibilities, the discovery step may be executed for each sub-goal to find an offered sub-service that fulfils this goal. Assuming this step returns a goal and choreography model for the sub-service, conditions (2), (3) and (4) from Figure 8 should hold.

For example, suppose Figure 17(i) depicts the requested goal associated with the timeout example from above, and Figure 17(ii) depicts three alternative offered goals returned by the discovery process. We assume the simple information model of Figure 17(iii) is used, which distinguishes three different meanings for the timeout value. Using subsumption checking, one can derive that offered goals $timeout_1$ and $timeout_2$ are interoperable with the requested goal, and thus satisfy condition (2).

The assessment of condition (3) requires a model of the requested choreography. One could choose to define this model beforehand, thereby imposing additional constraints on the discovery step, or afterwards, thereby allowing the (manual or automated) alignment of the requested choreography to the returned offered choreography. For example, assume Figure 17(iv) depicts the requested choreography and Figure 17(v) an offered choreography that is returned by the discovery process. These choreographies are not interoperable because the reply part of operation notify does not match a return part of (one-way) operation alarm. However, if notify would be adapted to a one-way operation, a common execution is possible. Reachability analysis also shows that an execution in which the Timer is cancelled is not possible.
The assessment of condition (4) can be done analogously to the examples on conformance assessment that have been presented before.

There is a correspondence between the assessment of condition (3) and condition (5) as explained in the preceding section. Both conditions are equivalent in case a sub-goal can be refined into a sub-choreography through action refinement, such that the sub-choreography can replace the sub-goal independently of the refinement of the other sub-goals. Otherwise, both conditions may need to be considered separately. An example of the latter case is the refinement of sub-goals reserve and cancel, which are offered by the same sub-service. As a consequence the part of the coordinator that interacts with this sub-service involves reservation and cancellation interactions, which are linked via interactions with the user.

6.3. Language and tool support

Figure 18 gives an overview of the languages, language mappings and tools we currently use (and develop) for the modelling and analysis of the interoperability and conformance relation between service models. We aim to build from this an integrated tool set that supports the assessment of these relations during a service composition process at design-time. From this experience, we want to improve our understanding of how to support runtime assessment. For the runtime case, we believe the composition problem should be limited in complexity, e.g., by using predefined composition patterns and simple input-output relations between services.

Besides modelling and analysis, we have developed tools that support the implementation of service models ([4],[5]). In particular, orchestration models can be mapped to BPEL and WSDL specifications by adding stereotyping information to operations, such as references to namespaces and WSDL specifications of the sub-services. Based on this stereotype information we can also simulate and test the orchestration by invoking real instances of the sub-services. The code required to perform the invocations is generated on-demand using JAX-WS and JAXB.

7. Related work and conclusions

Over the past seven years service composition emerged as an active and productive research area. Various approaches and techniques have been presented, such as static vs. dynamic, model-driven, declarative, automated vs. manual, context-based, and workflow vs. planning approaches [6], [21], [12]. However, the applicability of these approaches is still limited considering the assumptions being made. Furthermore, many approaches are defined at a technology level and cannot easily be used with alternative technologies.

To facilitate reasoning and analysis at a more abstract level, several conceptual frameworks supporting service modelling and, amongst others composition, are being developed ([2],[10],[26],[3],[24]). An important aspect covered by these frameworks is the semantics of a service in terms of its input, output, preconditions and effects. This aspect, called here the information aspect, is typically modelled using ontology languages.

Another important aspect to be covered is the interacting behaviour of services. For example, [25] compares the composition problem to the design and specification of computer protocols. To verify the correctness of this protocol behaviour, the application of formalisms like Petri Nets, automata and process algebras are being investigated.

In [18] we have presented our conceptual framework for service modelling, called COSMO. As opposed to the earlier mentioned frameworks, this framework is particularly strong in modelling the interacting behaviour of services and integrates existing (ontology) languages for modelling the information aspect. In addition, our framework allows one to model and relate services at distinct abstraction levels using the same set of concepts.

In this paper we have shown how the COSMO framework can be used to model and analyse requirements on the interoperability and conformance relation between service models. The paper describes the role of these relations during the service composition process and how these relations can be assessed independently of the particular algorithm that is used to synthesize a service composition. We think this constitutes an original and relevant contribution, because we are not aware of other work in which interoperability and conformance have been considered within a single conceptual framework, covering both behaviour and information modelling aspects. Although a lot of work exists on service interoperability (service matching), most of this work focuses on either the information or behaviour aspect ([13],[13]). Related to service conformance we are not aware of work similar to ours.

To facilitate the application of our research, we developed tools that support the modelling and analysis of interoperability and conformance relations. Part of this

<table>
<thead>
<tr>
<th>Modelling</th>
<th>Behaviour</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Languages</td>
<td>ISDL</td>
<td>OWL, Java</td>
</tr>
<tr>
<td>Tools</td>
<td>Grizzle (editing)</td>
<td>Protege (editing)</td>
</tr>
<tr>
<td>Analysis</td>
<td>Languages</td>
<td>ISDL, PetriNets</td>
</tr>
<tr>
<td>Tools</td>
<td>Sizzle (simulation)</td>
<td>OWL, SPARQL, Java</td>
</tr>
<tr>
<td></td>
<td>Abzibble (abstraction)</td>
<td>Jena</td>
</tr>
<tr>
<td></td>
<td>CPNtools (reachability)</td>
<td>Racer (reasoning)</td>
</tr>
<tr>
<td>Mappings</td>
<td>ISDL -&gt; PetriNets</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Languages</th>
<th>ISDL &lt;-&gt; BPEL/WSDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tools</td>
<td>BPEL/WSDL</td>
<td>ISDL-schema, Java</td>
</tr>
<tr>
<td></td>
<td>ISDL &lt;-&gt; BPEL/WSDL</td>
<td>JAXB, XMLBeans</td>
</tr>
</tbody>
</table>

Figure 18. Language and tool support
work comprises the definition of mappings from our concepts to existing formalisms and tools. Typically, these formalisms are either strong in behaviour or in information modelling. Furthermore, multiple behaviour and information formalisms may be used. Our main challenge for the near future therefore is the integration of formalisms and tools to cover both modelling aspects.

Our work aims at the development of methods and techniques to develop mobile and context-aware services. For this domain, no composition techniques exist that construct service composition correctly in a (semi-)automatic way [14]. Therefore, techniques to assess the correctness of service compositions afterwards, as presented in this paper, are of particular interest. At first, we focus on the application of these techniques at design-time. Subsequently, based upon experience gained with design-time analysis, we want to explore the possibilities for applying our techniques to the run-time case.

References