Closing the User-Centric Service Coordination Cycle by Means of Coordination Services

Hans Weigand[1], Paul Johannesson[2], Birger Andersson[2],
Maria Bergholtz[2], and Jeewanie Jayasinghe Arachchige[1]

[1] Tilburg University, P.O. Box 90153,
5000 LE Tilburg, The Netherlands
{H.Weigand,J.JayasingheArachchig}@uvt.nl
[2] Royal Institute of Technology
Department of Computer and Systems Sciences, Sweden
{pajo,ba,maria}@dsv.su.se

Abstract. In the future vision of an Internet of Services, users take an active role in service selection and composition. In this context, web services are mostly interfaces to real services and can be classified as coordination services with respect to the latter. To enable users to perform service composition, the effect of the coordination services must be described in such a way that users are not only able to discover services but also to detect and prevent possible conflicts in their composition. To meet these requirements, a service description language for coordination services is proposed based on the REA business ontology.

Keywords: Internet of Services, service design, REA, IOPE.

1 Introduction

In spite of considerable progress that has been made in the area of Service Oriented Computing, the impact on society has still been limited. There is not yet such a thing as an Internet of Services that would allow users to integrate the services they want to use easily and seamlessly. It has been acknowledged that users must play a more active role in service composition, if only because of the long tail of specific and heterogeneous services around [AC06] that simply cannot be handled all by the IT departments. Enterprise mashups may provide an instrument to realize this service co-creation effort of users and developers [HS09]. In this paradigm, software resources such as *web services* are embedded in *widgets* that provide simple user interaction mechanisms to these resources; these (visual) widgets are combined by the user himself to create *mashups*.

However, users are not interested in composing web services as such. To them, these are merely interfaces to "real" services such as traveling, meeting support, child care, entertainment or car maintenance. Users have a need to *plan* and *coordinate* the services they use (cf. [Be04]).

Fig. 1 depicts the envisioned user-centric service coordination cycle: users compose mashups and interact with the widgets in them to access web services. The

P. Soffer and E. Proper (Eds.): CAiSE Forum 2010, LNBIP 72, pp. 267–282, 2010.

web service typically supports the coordination with a service provider who offers a real-world service as part of a service bundle. The service affects a resource that concerns the user (the resource could be the user himself, for instance in the case of a hotel reservation). That web services themselves may be composite software entities is left out of this figure as being less relevant to the user, but is of course relevant to the software developer.

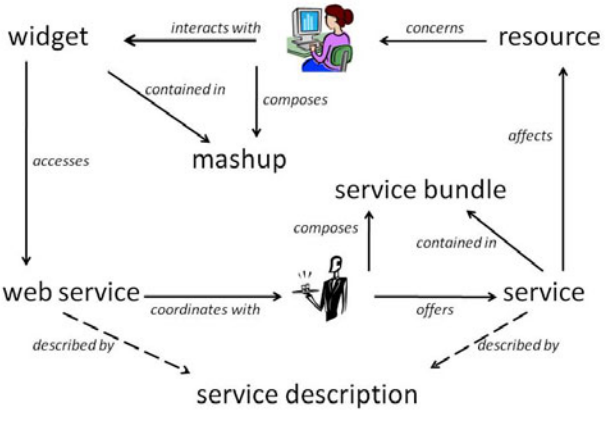

Fig. 1. User-centric service coordination cycle

Both web services and services need a description, but what should be in this description? In composing web services, a major challenge is to reconcile incompatible data representations. In composing services in the real world, a major challenge is to meet the constraints imposed by the fact that resources are scarce, can only be in one place at a time and often cannot be shared. For that reason, [PT09] argues convincingly that "asset-driven" service modeling will be a central concern in developing an Internet of Services and claims that "novel methodologies and tools are needed to support the modeling of the key assets of services". In our view, this modeling should support at least conflict prevention and conflict detection.

In order to make conflict prevention and conflict detection possible at all, web services must provide more information than input and output requirements such as we find in a WSDL document. What we need is a generic language to describe services, the resources they use as well as planned and actual events on the type level, Web services can use this language to represent the preconditions and effects of the real services they connect to as well as their own semantics. A mashup environment can collect and combine this information, integrate it with other sources such as the user's agenda (that should be represented in the same format) in order to provide the user with means for conflict prevention and conflict detection. On the basis of the service description and after instantiating the formulae with actual data, the user immediately knows the effect of a successful service invocation.

In this paper, we propose to ground the service description language in the REA ontology [Mc82] where we concentrate on coordination services as being of most interest to the user. An advantage of REA is that it has a very small set of basic

concepts, and therefore is relatively easy to understand. For conflict detection and prevention we propose a small set of coordination services and supporting concepts. By conflict detection and prevention we mean the following. Let *s* be a service that a user U intends to consume and let M be the set of resources and actors involved in the execution of *s*. Each *m* in M has a time-based context A(E,C) where E is a set of events planned for *m* and C a set of constraints on E. The goal of *conflict prevention* is to ensure that when *s* is added to the planning of U, all context constraints are still met, for all *m* in M. Typical events that stem from the planning of *s* are the start of the service execution and its ending. The goal of *conflict detection* is to check context constraints when an event *e* is added. Typical events are contingencies such as a flight being delayed. We can assume that in a future Internet of Services and Internet of Things, most of these events are generated without active user involvement. If *s* is a composite service, then the check should be done on all the services involved individually and jointly.

To arrive at rigorous and relevant research results, we use Peffers' design science phases [PT08]. The *problem identification and motivation* has been stated. Our *solution objective* is to develop a coordination service description language based on REA. In section 2, we position coordination services as the "glue" between software services and real services. In section 3 we work how to represent services and the coordination of services in REA. We identify a couple of very common coordination patterns (*design*). On the basis of that we show in section 4 how service descriptions can be developed that enable the required conflict detection (*design and development*). This is applied to the well-known hotel reservation case (*demonstration*); we adopted this case to facilitate comparison with other solutions.

2 Coordination as a Service

According to the OASIS reference architecture foundation for SOA, it is essential that participants can use a SOA-based system to realize actual effects in the world [OA09]. However, when talking about the real world, OASIS makes a sharp distinction between the social world and the physical world (in line with the Language/Action Perspective tradition and the communicative theory of Habermas [Di06]). Many, if not most, effects that are desired in the use of SOA-based systems are actually social effects rather than physical ones. For example, opening a bank account is primarily about the relationship between a customer and a bank – the effect of the opened account is a change in the relationship between the customer and the bank. For that reason, OASIS talks about social actions that result in social facts. "A social fact is an element of the state of a social structure that is sanctioned by that social structure". Social facts include policies and commitments where "a commitment is a social fact about the future: in the future some fact will be true and a participant has the current responsibility of ensuring that that fact will indeed be true". A completed business transaction establishes a set of social facts relating to the exchange; typically to the changes of ownerships of the resources being exchanged.

What remains a bit out of the OASIS picture is that social facts refer to physical world events, such as the delivery of a product. For a full account of service effects, this relationship between social facts and the real world must be made explicit (Fig.2).

It is widely recognized that input and output descriptions of web services, or its operations, are not sufficient for capturing the semantics that users need. Precondition and Effect descriptions have been added. Although WSDL-S provides a mechanism to include these attributes, it does not give guidance on how to do specify their contents. The OASIS reference model views web services as coordination mechanisms and emphasizes the social effects. How these are to be represented, and how these social facts relate to real-world business events is still to be worked out. In the following, we address this research gap by proposing the REA ontology for coordination service description.

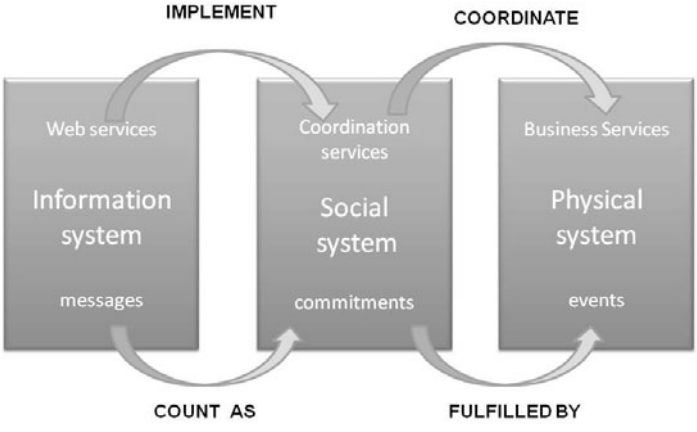

Fig. 2. Coordination services are the glue between web services and business services

3 REA - Background and Modifications

3.1 The REA Business Ontology

The Resource-Event-Agent (REA) ontology was first formulated in [Mc82] and has been developed further, e.g. in [UM03, GM99, Hr06]. The following is a short overview of the core concepts of the REA ontology based on [WJAB09].

A *resource* is any object that is under the control of an agent and regarded as valuable by some agent. This includes goods and services. The value can be monetary or of an intangible nature, such as status, health state, and security. Resources are modified or exchanged in processes. A *conversion process* uses some input resources to produce new or modify existing resources, like in manufacturing. An *exchange process* occurs as two agents exchange (provide, receive) resources. To acquire a resource an agent has to give up some other resource. An *agent* is an individual or organization capable of having control over economic resources, and transferring or receiving the control to or from other agents [GLP08].

The constituents of processes are called *economic events*. An economic event is carried out by an agent and affects a resource. The notion of stockflow is used to specify in what way an economic event affects a resource. REA identifies five

stockflows: produce, use, consume, provide and receive, where the first three occur in conversion processes and the latter two in exchange processes. REA recognizes two kinds of duality between events: conversion duality and exchange duality.

Events can be assigned to a *location*. Sometimes the acronym REAL is used for REA plus location [OL99] (Fig. 3).

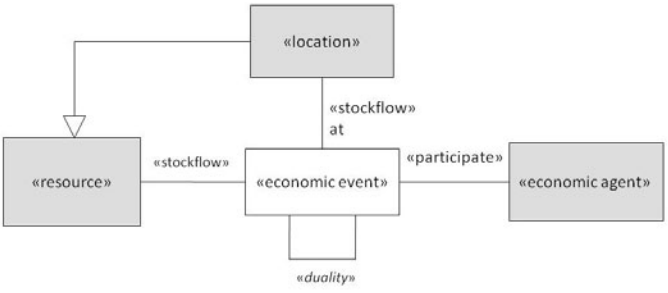

Fig. 3. REA basic categories including location. Events are rendered in white, the other objects in grey.

3.2 Commitments in REA

Commitments were added to the REA ontology in [GM99] as "important economic phenomena", and modeled as the pair-wise connection of requited commitments. The pair-wise connection is similar to the *duality* relationship between actual exchanges or conversions but as it is not between events, REA calls it a *reciprocal* relationship. In the following, we refine and extend the commitment concept of REA by adding explicit commitment events and by rethinking the "reserve" relationship. Starting point is that we consider a commitment as a special type of resource, so that it can be handled in the same way, that is, be manipulated and used in exchange and conversion events using stockflow relationships.

A commitment is a promise regarding the future. Commitments are formalized as clauses in contracts and those commitments are subsequently fulfilled through economic events. A distinction can be made between increment commitments (assets in the agent's perspective) and decrement commitments (liabilities in the agent's perspective) [Hr06]. Figure 4 illustrates this: in an economic event, a provider gives a decrement commitment that is received by a customer in the same event. He has given a promise to e.g. deliver a service in the future. Depending on the commitment type (decrement vs. increment) the relationship to the commitment is characterized as a give or take stockflow. A customer can, in a *decommit* event, take a d-commitment that is received by a provider in the same event. This represents an absolving of a commitment. The provider did commit to deliver something in the future and this promise is now considered void by the customer.

A structure involving increment commitments can be constructed as well (not illustrated here) for the customer's part of the contract, but still from the provider's point of view. In a commit event a provider becomes the receiver of an i-commitment (increment) through a take stockflow. The customer owes the provider. The provider

can, in a decommit event, give the customer an i-commitment back, thereby cancelling the debt. Note that the customer cannot cancel this debt himself, but he can request for it. The exchange reciprocity between commitments reflects an exchange duality between commitment events.

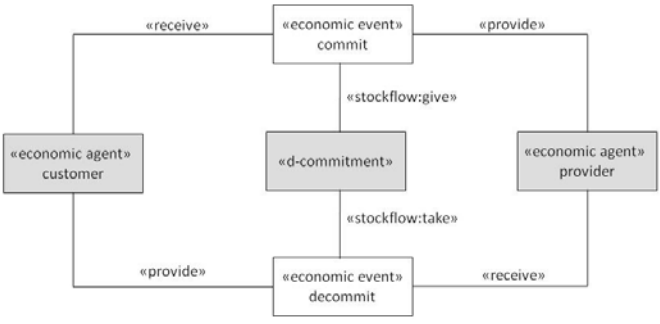

Fig. 4. REA commitment pattern for decrement commitment

Thus contract formation can be thought of as giving and taking corresponding d-commitments and i-commitments (Fig 4).

Commitments can also be returned in a *decommit* event. Two main types of decommit can be distinguished that maintain the duality axioms of REA. In the case of *canceling*, the commitment is returned in exchange with the reciprocal commitment being returned. For instance, a purchase order is cancelled and the payment is cancelled at the same time (of course, the contract may specify a penalty for the one who requests cancelation or even forbid cancelations altogether). In the case of *fulfillment*, the commitment is returned in exchange with some other economic event being provided, being the content of the commitment. For instance, when a delivery is made, the purchase order commitment is returned. In the following, we will adhere to the standard REA fulfill relationship as an abbreviation for this duality.

Committing is modeled as an economic event, rather than as a system event as in standard REA. An economic event is defined as a change in the value of economic resources of the enterprise. We claim that this also applies to taking or giving a commitment, e.g. the commitment to a future payment.

Commitments are most often about resource *types* (e.g. a non-smoking hotel room, or a certain book title to be delivered), whereas the business transaction itself is about a resource instance, that is, a specific hotel room or a specific copy of the book. In some cases, the commitment is about the resource instance itself, e.g. in the sales contract of a house. In order to handle both commitments in a unified way, we apply the notion of resource group [GM06]. Let the object of the commitment be a resource group of a certain resource type. Cardinality of the set/quantity of the resource is the most important attribute of resource group, and additional constraints can be specified. The relationship between resource group and resource type is a *policy* relationship [GeMc06]. It specifies the type of resources that may go into the resource group. Fig. 5 presents our refinement of the REA commitment ontology using the grouping concept. It works as follows. When a commitment is created, the commitment always refers to a resource group that is created at that time. However,

the *grouping* need not be filled in. In the case that a particular resource is to be reserved (e.g. a specific house), the *grouping* relationship is made at commitment time. In all other cases, it is specified later when the purchase contract is being executed. For example, a commitment to reserve two hotel rooms is formalized as a clause in a contract. The object of the commitment is a group. The policy for this group says that it must contain "double non-smoking rooms", and 2 of these. In the economic event that fulfills the commitment two double rooms (resources of the specified type) are allocated to this group.

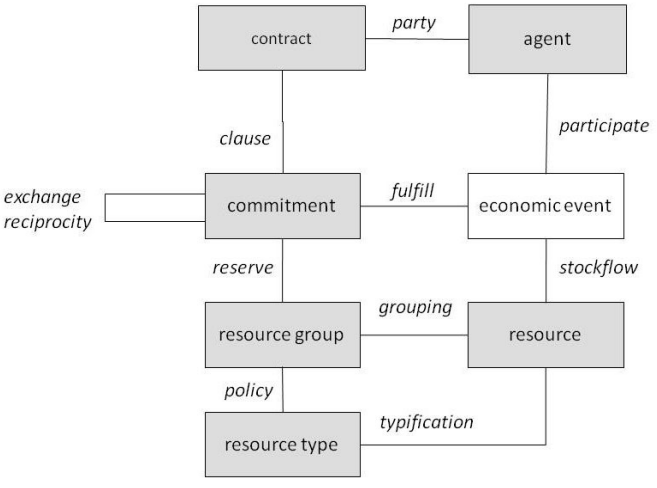

Fig. 5. REA commitment "reserve" revisited

3.3 Capacity Planning

Using the REA model, we can define the notions of *capacity* and *availability*. We take the perspective of the resource manager *a* (e.g. hotel manager) who has received or reserved certain resources from another agent *x* (e.g. hotel owner). He can commit resources of a certain resource type to another agent *x* for a certain date. In that case, there is a specify relationship between the reservation and the resource type. The commitment/reservation has a cardinality indicating the number of resources reserved. The actual allocation of resources (instances) to a certain reservation is usually done later. If we assume the Capacity is stable over time, the following definitions suffice:

$$\text{Capacity}(a,t) = \text{card}(R)$$
$$R = \{r\text{: resource} \mid \text{typification}(r,t) \wedge (\exists x\text{:agent receive}(a,x,r) \vee \exists s\text{:reservation (provide}(x,s) \wedge \text{receive}(a,s) \wedge \text{specify}(s,t) \wedge \text{reserve}(s,r))\}$$
$$\text{Reserved}(a,t,d) = \sum \text{card}(s),\quad s \in RS(a,t,d) \text{ where}$$
$$RS(a,t,d) = \{s\text{: reservation} \mid (\exists x,a\text{:agent provide}(a,s) \wedge \text{receive}(x,s) \wedge \text{specify}(s,t) \wedge \text{date}(s,d)\}$$
$$\text{Available}(a,t,d) = \text{Capacity}(a,t) - \text{Reserved}(a,t,d)$$

The capacity for a resource type *t* is what the agent has received or that is made available to him (and that is of the resource type *t*). To calculate the availability at some date/time *d*, we first sum up the commitments, and detract this number from the capacity.

3.4 Services in REA

In REA, a service is a kind of resource as it is viewed as valuable by some agent and can be transferred between agents [WJAB09]. As such, it inherits all features of resources, in particular that it can be exchanged between agents, that it is governed by a contract and that it is part of a conversion process chain. As depicted in Fig. 6, the service is exchanged between agents in return for money (top right cluster). All the coordination services that can be used within an exchange process apply to service exchanges as well; we will use this feature below. At the same time, the service is a resource produced in a conversion process by the provider (top left cluster), and consumed in a conversion process by the customer (bottom cluster). REA usually renders only one agent perspective, but for the understanding of the service interfacing between the provider and customer, we have included both perspectives (indicated by dotted rectangle) in one figure. Note that Fig. 6 is simplified in order to reduce clutter. In particular, the usual agent boxes are not present in top left or bottom.

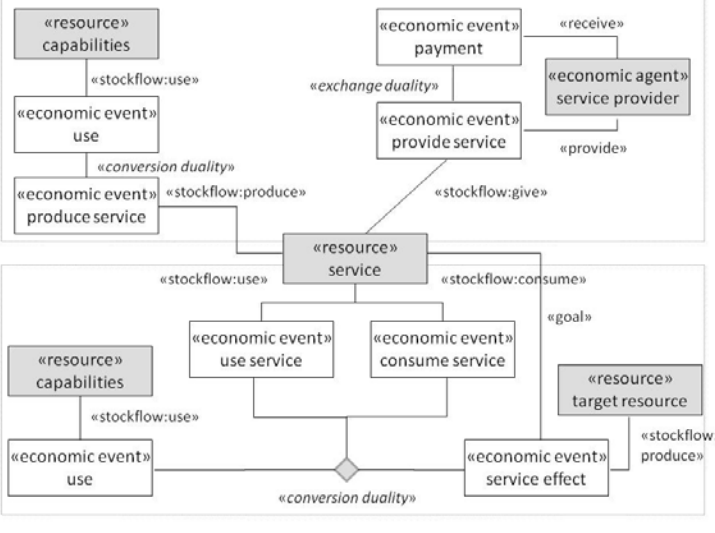

Fig. 6. REA application pattern for Service Exchange. It is assumed that the «goal» stereotype is defined in the REA meta-model

The economic increment event for creating the service stands in conversion duality to one or more resource use events. For example, a hotel service is realized by *using* the hotel room resources. At the customer side, we distinguish between service use and service consumption. Both can add value (production event) to some target

resource, typically in combination of some effort of the customer himself – that is why we also include a resource use event here. However, in the case of service consumption, the service is no longer available after the event, whereas service use draws on the existence of the service without changing its status. The service consumption may be conceived of as an atomic event or as a process over time. The latter is especially the case when the service is offered for a certain period. Then for economic purposes, the amount of service consumption is typically linear on the time having passed by.

The difference between service consumption and service use can be illustrated by the example of [FG09] of a fire brigade. This could be a service hired by a municipality for a certain period. Service consumption is here a matter of time: at the end of the period, it is completely consumed. During the period, the fire brigade may become active in the case of an emergency, as stipulated in the service contract. This is service use. The effect of service use is a particular house (resource) being rescued, whereas the effect of the service consumption is the increased security of all houses in town (resource).

For the user-centric description of a service, the "*goal"* is important [WJAB09]. A service aims to produce an effect on resources of the customer in such a way that the value increases. If the effect is not reached, this may cause the transaction to fail. Formally, the goal relationship can be seen as an extension of the REA meta-model. However, as it can also be seen as a derived relationship, since it is defined as "the production events at the customer' side that stands in conversion duality with the service use and consumption". When also the *consumption* events at the customer and provider side are relevant, we could add a "source" relationship, analogous to the "goal" relationship. Together, source and goal provide a reference to all resources affected by the service execution. As the description of all kinds of failures and exceptions is never exhaustive, we refrain from including that in the effect. It can be specified in the contract.

For web services and similar software artifacts to deserve the label "service", the service model elements should be clear. What is the goal of the web service, that is, what resources does it create or affect that have value to the client? Who are the actors involved in the exchange process? In the next section, we will consider coordination services as one important subclass of web services.

3.5 Coordination Services and Coordination Objects

Coordination services are defined in [WJAB09] as services supporting an exchange process (a set of events) for a good or a service. Processes like identification, negotiation, order execution and after-sales take place in a good exchange as well as a service exchange. We introduce the notion of coordination *object* for the object of these processes: what *is* negotiated and executed? The central coordination object is the purchase order that is first negotiated, then created, and then fulfilled by the exchange event. In complex business processes there are more coordination objects. The following two also reoccur often, especially when services are concerned: *reservation* and *appointment*. The rationale for the latter (appointment) is that the delivery of a service requiring resources from both the provider and customer to be present at the same time and place requires more coordination than the delivery of a good.

The rationale for reservation is the following. Note first that in standard REA, a reservation is a relationship between a commitment and a resource or resource group (section 3.2). In the context of coordination, we use the term "reservation" more specifically for a commitment that precedes the purchase order, which obliges a provider not to sell a resource to any other agent than the customer for whom the reservation is created. From an economic point of view, the main objective of this kind of reservations is to reduce uncertainty about the business transaction – to mitigate the risks involved, such as items being out of stock or functionality not available, and to reduce the need for slack [WH06]. So although the reservation has some costs in the form of less operational discretion, it increases the total value for both customer and provider.

Drawing on the REA ontology, coordination objects can be specified in terms of commitments. Therefore, another way of characterizing coordination services is to say that these services manipulate commitments: their goal is to provide, receive and fulfill commitments.

For a better understanding of the three coordination objects and their commitments, is it good to see how they are related. Although they are applied differently in different domains, there exist logical dependencies between them. The reservation always precedes the purchase order. The conversation around the reservation addresses questions such as: Is the resource available? If so, can I reserve it? How can I cancel the reservation, if needed? Do I have to pay for the reservation or for the cancellation? The seller commits himself to make sure that the necessary resources will be available. Usually, the customer has fewer commitments, but he may commit himself to the availability of necessary resources (in his case, financial resources) as well. This can be achieved in practice by providing credit card details, for instance.

The purchase order follows after the reservation logically and in time. A contract is being formed with terms and conditions. Now the commitments go much further: the seller promises to deliver the good or service at some moment or period in time, and the customer promises to pay an agreed upon price. These are firm commitments that can only be left unfulfilled in exceptional circumstances as specified in general law or in the contract itself.

We assume that for all coordination objects (so not only for the Purchase Order) there is an agreement process first followed by an execution and evaluation process, that is, the coordination process per coordination object takes the form of a "Conversation for Action" [Di04, Go06]. The message exchange in these conversations is not in the scope of this paper, but what is important is the effect of these conversations, since that is directly relevant for a user composing and using a certain mashup application.

The REA application model in Fig. 7 focuses on the coordination object purchase order but also specifies how it relates to the coordination object reservation. The reservation is a commitment that specifies a resource type and there is a "reserve" relationship with resource, being all resources involved in the fulfillment of the commitment and set apart for that purpose. In line with [Hr06] we distinguish between d-commitments (decrement) and i-commitments (increment), for commitments by or to the service provider, respectively. The fulfill relationship is one between commitment and economic event. The fulfillment of the reservation is the accept-order event by which the purchase order is created. The fulfillment of the

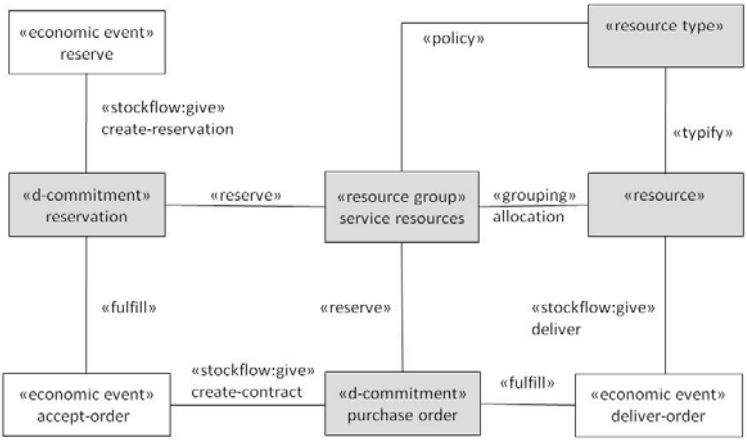

Fig. 7. REA Application Model for purchase order including relationship with reservation

purchase order is the service exchange event. The actual service exchange can be seen as the ultimate objective of the reservation as well. So in certain sense, this exchange fulfills the reservation, although indirectly. To capture this notion of fulfillment, we define a fulfill* relationship being the recursive closure of fulfill-relationships. So the actual exchange *fulfils* the purchase order, and *fulfils** the reservation.

It should be noted that although the meaning of reservation and purchase order is quite stable over different domains, these two coordination objects are not always applied in the same way. This can be confusing, for example when we talk about hotel reservations and flight reservations in one breath. In the case of a hotel, the purchase order is created when the customer checks in. At that moment, the reservation, if any, is fulfilled. In the case of a flight ticket, the purchase order is made when the ticket is sold, typically long before the check-in at the airport. What happens at the check-in is the allocation of a specific resource (a chair with a number). Sometimes, it is possible to take an option on a flight ticket for a few days before buying it. That option is a case of reservation, but the ticket itself is not.

The complete *reservation pattern* is represented in Fig.8. It shows the reciprocity relationships with other commitments that are grouped together in a contract.

An *appointment pattern* is used when two or more agents want to meet at a specific location. Appointments can be made for their own sake, but can also be part of a purchase contract, for example, when customer and provider have to agree on where to deliver the service or good.

Fig. 9 shows an application model for show-up appointments where the commitment is from the side of the customer (so it is an i-commitment), typically reciprocal to an appointment of the other party to be there as well. Since the appointment includes at least a resource (the customer himself, or some resource related to the customer; and the location) there are two “reserve” links. In accordance with our “reserve” ontology, these links point to groups that specify the reservation on an abstract level and that are populated at some time with specific instances.

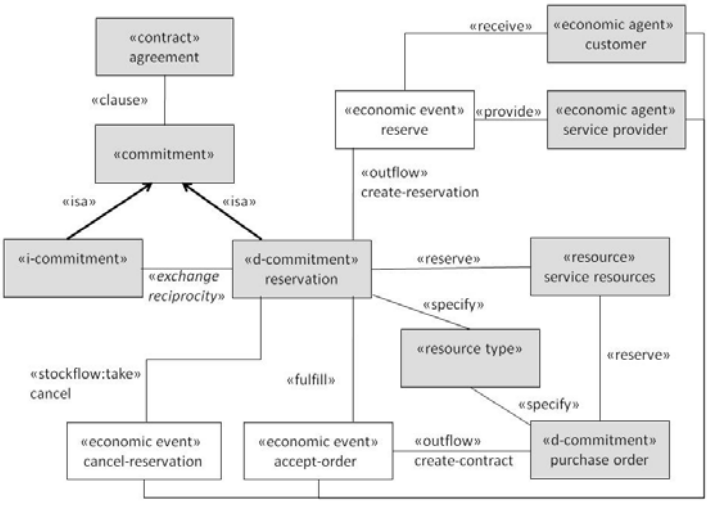

Fig. 8. REA Application Model for reservation

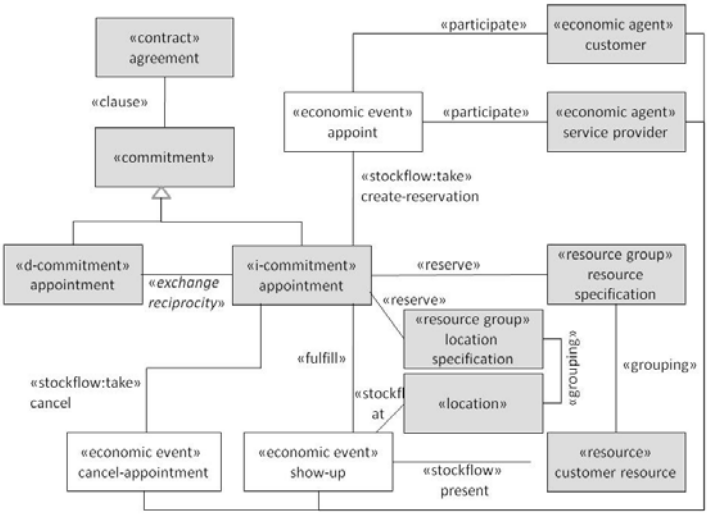

Fig. 9. REA Application Model for (show-up) Appointment

The appointment can be applied throughout the process (for instance, an appointment to negotiate the purchase order via Skype at some specific hour), but most prominently at the fulfillment of the purchase order when this fulfillment is not immediate, as noted above.

We have described three coordination objects corresponding to three coordination services – reservation, purchase contract, and appointment. On the basis of our

experience and considering standards such as [UM03] and the set of patterns identified by [Hr06], we claim that these are the most prominent ones, but the question whether more coordination objects exist should be addressed further by both formal and empirical research.

4 Service Description and Conflict Detection

4.1 Service Description Using REA

Using the REA ontology, service descriptions can be developed for coordination services either in the form of REA events or REA relations. Table 1 specifies the basic predicates.

Table 1. Basic REA predicates

RELATIONS	EVENTS	TERMS
At(Agent,Location)	*Commit(Id,Agent,Agent, e(Resource Type,Time))*	*contract*
Fulfill(Event,Commitment)	*Cancel(Id,Commitment)*	*commitment*
Clause(Commitment, Contract)	*Purchase(Id,Agent,Agent, Resource)*	
Available(Agent, ResourceType,Time): Number	*Pay(Id,Agent,Agent,Money)*	
Capacity(Agent , Resource Type):Number	*Move(Id,Agent,Location)*	
PlannedCapacity(Agent, Resource Type, Time):Number	*Move(Id,Agent,Resource, Location)*	

The relations and terms have a direct counterpart in REA or have been defined in section 2. We use some shorthand for the events. *Commit* stands for create commitment, *Cancel* for withdraw commitment. *Purchase* and *Pay* stand for the standard exchange events. *Move* stands for the event of changing the location of the agent or some resource. In both Commit and commitment we make use of an embedded functor *e(x,t)* where *e* is an Event Type, *x* can be a any object (and there may be more than one argument *x*) and *t* is a time reference. Expressions of this form are called *i-events* and are used in the same way as actions in the situation calculus [MH69], where they can be the object of a *do*-action.

Using these predicates, we define the following list of coordination services (table 2). Note that they are services in terms of [WJAB09]: their goal is an event that affects a relevant resource. Being coordination services, they manipulate commitments. Table 2 presents the IOPEs (Input/Output/Precondition/Effect) for hotel services but in a quite general way. As such it can be applied to a flight service or theater service as well. However, the way the coordination services are bundled in web services may differ. In the typical hotel case, the Create_Contract and Check_In are one transaction: at the moment the customer shows up, according to his reservation, a contract is set up and a specific resource is allocated. In the typical flight case, the Create_Contract is performed long time before the Check_In.

Table 2. Generic coordination services

Coordination Service	Input	Output	Precondition	Effect (Goal)
Check_Availability	ResrcType R Time T User U	Bool A	A= (Available(Self,R,T)>0)	*Not a social fact*
Create_Reservation	Customer C Time T ResrcType R	Id Res	Available(Self,R,T)>0 At(Self,L)	commit(i,Self,C, e(R,T)) and i=Res and commit(j,C,Self, move(C,L,T.start))
Cancel_Reservation	Customer C Time T ResrcType R Id Res	-	commitment(i, Self,C, e(R,T)) and i=Res and not exist p: fulfill(p,i)	cancel(j,i) and forall j: commitment(j,C,Self, move(C,L,T.start)) implies cancel(j)
Create_Contract	Customer C Time T Id Res	Id PO Amount F	commitment(i,Self,C, e(R,T)) and i=Res	commit(j,Self,C,e(Rs,T)) and j=PO and typeof(Rs,R) and exist contract(CT) and clause(PO,CT) and clause(Inv,CT) and commitment(Inv,C,Self, pay(F,T2)) and fulfill(PO.Res)
Check_In	Customer C Time T Id PO	Id Ri	commitment(j,C, Self, move(C,L,T.start) and j= LRes and at(C,L) and commitment(i, Self,C, e(Rs,T)) and i=PO	commit(i,Self,C,e(Ri,T)) and realize(Rs,Ri) and forall m: move(m,C,L) implies fulfill(m,LRes)
Check_Out	Customer C Id Ri	Id S	commitment(i,Self,C, e(Rs,T)) and i=PO and realize(Rs,Ri)	purchase(j,Self,C,Ri,T) and i=S and fulfill(S,PO)
Receive_Payment	Customer C Id PO	Id P	exist contract (CT) and clause(PO,C) and clause(Inv,C) and commitment(Inv,C,Self, pay(F,T2))	pay(j, C,Self, F) and j=P and fulfill(P,Inv)
Cancel_Contract	Customer C Time T Resource Rs Id PO	-	commitment(i, Self,C, e(Rs,T)) and i=PO and exist contract(C) and clause(PO,C)	cancel(j,i) and forall j: commitment(j,C,Self, Q,T') implies cancel(j)

4.2 Conflict Detection

As said in section 1, each resource or agent is assumed to have a time-based context A(E,C) where E is a set of events planned and C a set of constraints on E. To support conflict detection and conflict prevention, we should be able to check whether E meets the constraints C. This applies both to the service provider and the resources that he manages and to the service user and the resources that she manages. In the following, we take the perspective of the user and focus in particular on her agenda.

Let M be the set of resources relevant to agent *U*. To determine the contents of M, the set of E_u of committed events for *U* is calculated first as follows:

$E_u = \{e\text{: event} \mid \exists c\text{: commitment} \wedge \text{fulfill}^*(e,c) \wedge \text{participate}(e,U)\}$

Then

$M = \{m \mid \exists e \in E_u\text{: stockflow}(e,m) \vee \text{participate}(e,m)\}$ (resources involved and agents participating, as far as known)

For some $m \in M$, the context E_m contains the committed events that involve m. Note that $U \in M$ – by definition, the user herself is also involved and hence the commitment should be put in her agenda. However, not only the context of U, but the context of every $m \in M$ should not violate its constraints. The constraints in the context can be resource-specific, but a very fundamental constraint is that there can be no "agenda conflict":

$$\forall e_1, e_2 \in E_m \; e_1.\text{time} \cap e_2.\text{time} = \varnothing$$

Another general constraint is that a physical resource can be at only one place at a time, and needs time for moving.

$$\forall e_1, e_2 \in E_m : e_1.\text{time.end} = e_2.\text{time.start} \Rightarrow e_1.\text{location} = e_2.\text{location}$$
$$\forall e_1, e_2 \in E_m : \text{next}(e_1, e_2) \wedge e_1.\text{location} <> e_2.\text{location}$$
$$\Rightarrow (\exists e_i \in E_m : e_1 < e_i < e_2 \wedge e_i.\text{type} = \text{move}() \wedge e_1.\text{object} = m \wedge e_i.\text{destination} = e_2.\text{location})$$

where next(e_1, e_2) means that e_2 is the first event after e_1.

To prevent conflicts when considering the use of a service s, the user first adds the commitments produced by s to his context (using the coordination service effect descriptions), and then executes the conflict detection process. If a conflict is detected, the user is informed and/or the coordination service in question is triggered to find an alternative.

5 Concluding Remarks

From an end-user perspective, coordination services form an important service class (cf. [Be04]), as these services allow the user to manage real services that matter to him/her. When using these services, he should be aware of the real-world effects, to detect and prevent possible conflicts with his own agenda (already existing commitments) or the agenda of other resources involved. In this paper, we have explored how REA can be used to describe the effect (IOPE) of a representative set of coordination services.

The focus of this paper has been on the description of coordination services which are the services that support an exchange process (a set of events) of a resource. Creating, executing and evaluating commitments is done in a combination of informational and material processes. The Language/Action Perspective ([Di06,Go06]) has explored a couple of standard micro-patterns on which the informational processes can be based. However, we have not focused on describing processes in this paper.

One line of future research concerns the interpretation of commitments in terms of rights. When an agent commits (d-commitment), he gives away some right on the resources involved, which assumes that he did hold that right before. REA posits a "control" relationship between agents and resources. If control is made more precise in terms of rights (e.g. ownership, access and custody), this can be used in the specification of commitment preconditions and effects.

References

[AC06] Anderson, C.: The Long Tail: How endless choice is creating unlimited demand. Random House Business Book, London (2006)

[Be04] Benatallah, B., Casati, F., Toumani, F.: Web Service Conversation Modeling: A Cornerstone for E-Business Automation. IEEE Internet Computing 8, 1 (2004)

[Di06] Dietz, J.: Enterprise Ontology - Theory and Methodology. Springer, Berlin (2006)

[FG09] Ferrario, R., Guarino, N., Fernandeze Barrera, M.: Towards an Ontological Foundation for Service Science: the Legal Perspective (2009)

[GLP08] Gailly, F., Laurier, W., Poels, G.: Positioning and Formalizing the REA enterprise ontology. Journal of Information Systems 22, 219–248 (2008)

[GM99] Geerts, G., McCarthy, W.E.: An Accounting Object Infrastructure For Knowledge-Based Enterprise Models. IEEE Int. Systems & Their Applications, 89–94 (1999)

[GeMc06] Geerts, G., McCarthy, W.: Policy-Level Specifications in REA Enterprise Information Systems. Journal of Information Systems 20(2), 37–63 (2006)

[Go06] Goldkuhl, G.: Action and media in interorganizational interaction. ACM Comm. 49(5), 53–57 (2006)

[HS09] Hoyer, V., Stanoevska-Slabeva, K.: Towards a reference model for grassroots enterprise mashup environments. In: Proc. ECIS 2009 (2009)

[Hr06] Hruby, P.: Model-Driven Design of Software Applications with Business Patterns. Springer, Heidelberg (2006)

[Mc82] McCarthy, W.E.: The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review (1982)

[MH69] McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence 4, 463–502 (1969)

[OA09] OASIS Reference Architecture Foundation for Service Oriented Architecture 1.0 (2009), http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-cd-02.pdf

[OL99] O'Leary, D.: REAL-D: A Schema for Data Warehouses. Journal of Information Systems 13(1), 49–62 (1999)

[PT08] Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S.: A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems 24(3), 45–77 (2008)

[PT09] Pistore, M., Traverso, P., Paolucci, M., Wagner, M.: From Software Services to a Future Internet of Services. In: Tselentis, G., et al. (eds.) Towards the Future Internet. IOS Press, Amsterdam (2009)

[UM03] UN/CEFACT Modelling Methodology (UMM) User Guide (2003), http://www.unece.org/cefact/umm/UMM_userguide_220606.pdf

[WH06] Weigand, H., van den Heuvel, W.J.A.M.: A conceptual architecture for pragmatic web services. In: Schoop, M., de Moor, A., Dietz, J. (eds.) Proc. 1st Int. Conf. on the Pragmatic Web, pp. 53–66. Springer, Heidelberg (2006)

[WJAB09] Weigand, H., Johannesson, P., Andersson, B.: Bergholtz Value-based Service Modeling and Design: Toward a Unified View of Services. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 410–424. Springer, Heidelberg (2009)