Decentralized decision making protocol for service composition

Andreas Wombacher
University of Twente, Enschede, The Netherlands
a.wombacher@utwente.nl

Abstract

Service composition based on state dependent services is a challenge if it is done in a decentralized way, that is without a centralized coordinating partner knowing all involved parties. In particular, the challenge is the combination of services to a composite service, such that every party involved considers its view of the composite service to be acceptable. In the paper a protocol is proposed which incrementally derives proposals for composite services. These proposals are then evaluated by the involved parties locally and a consensus on the local decisions on accepting or rejecting the proposed composite service is derived.

1. Introduction

Service composition may invoke services provided by a single organization or by different organizations. In particular, service composition is the service oriented way of solving integration problems by setting up a collaboration of services. In classical EDI based systems this integration problem has been addressed by manually setting up a collaboration between the different parties involved. However, Web Services and service composition ease the integration process (compared to classical EDI) because they rely on a common communication infrastructure (XML and SOAP). Although, the specification on which services to compose and how they are composed has still to be negotiated by the involved parties. The result of the negotiation is a composition of services, which represents the agreement of the involved parties forming a collaboration. Still, composition of services, or more general, the establishment of a collaboration is a complicated and expensive process due to its centralization, which is addressed in this paper by proposing a decentralized approach. In particular, decentralized approaches, like for example P2P systems, are considered to be more robust and require cheaper maintenance costs, because they are self-organized.

The decentralization idea of P2P systems are facilitated by the autonomy of services. While the idea of self-organization can easily be applied to stateless services, handling state dependent services is much more complicated. This is because state dependent services have interaction with other services which might interfere during the composition. Within this paper, an approach is presented to establish collaborations in a decentralized way, that is, without having a centralized coordinating party. In particular, a protocol is proposed which generates multi-lateral collaborations based on bilateral collaborations and propagates these multi-lateral collaborations to relevant parties resulting in an agreement / consensus on the completeness of a multi-lateral collaboration. That is, a decision is made whether all party's local workflow have assigned corresponding opponents. The resulting complete multi-lateral collaborations (each representing a service composition) are the basis for further processing to guarantee successful business interaction known as consistency (see also [9]).

The paper is structured as follows: An exemplary scenario and some formalization is introduced to describe the proposed decentralized service composition approach. Then, a four step approach is presented consisting of service discovery, bilateral collaboration establishment, combinations of collaborations, and consistency checking. Afterwards, the implementation is sketched and the related work is discussed. Finally, the paper concludes with future work.

2. Motivation

The scenario illustrating the addressed service composition problem is based on a procurement scenario consisting of a buyer, an accounting and a logistics department. The buyer is ordering a product at the accounting department (further abbreviated with accounting), which approves the order and forwards it to the logistics department (further abbreviated with logistics) to deliver the good. Further, the buyer has the option to do parcel tracking of the order by asking the logistics on the delivery state of the order. All functions are implemented as state dependent services. In the following the behavioral aspects of the states are represented as local workflows.

Local workflows can for example be modeled based on

Workflow Nets [2] or Annotated Finite State Automata [11]. In either case they consist of service invocations resulting in a message exchange. Thus, finally they consist of message sequences, where a message is specified by a sender, a recipient, and a message name itself. Since a concrete sender and recipient of a message at run-time is unknown at design-time of the local workflow, sender and recipient are specified as roles, that is, a placeholder for a concrete service sending or receiving the message. Due to this usage of roles, a local workflow can be partitioned into disjoint parts by calculating an abstraction of the local workflow with regard to a specific role [2, 11]. An abstraction keeps all messages unchanged which are either sent or received by a specified role, while the remaining messages are ignored, that is, they are relabeled by silent messages. Hence, a role can be assigned with a concrete service, if the abstraction of the local workflow with regard to the role is consistent with an opponents abstraction, that is, the represented message sequences have a non-empty intersection. A service fulfilling a role of an opponent's local workflow and vice versa is also known as bilateral matchmaking [11].

With regard to the example, three roles have been used: buyer, accounting, and logistics. However, the different roles do not necessarily represent different services. [1] The procurement scenario is extended by having several services for buyer, accounting and logistics roles. In particular, the scenario consists of two accounting services $A1$ and $A2$, two buyer services $B1$ and $B2$, and three logistics services $L1$, $L2$, and $L3$, where the local workflows of $A2$, $L2$, $L3$, and $B2$ require parcel tracking, and the local workflows of $L1$ and $B1$ do not support parcel tracking at all, while the local workflow $A1$ supports both options. Further, the services of $A1$, $L1$, $L2$, and $B1$ are located in Germany, while the services of $A2$, $L3$, and $B2$ are located in the US. In addition, buyer $B1$ wants to interact only with services located in Germany, while $B2$ does not have this restriction. Further, the logistics services $L1$, $L2$, and $L3$ each follow an internal strategy of only having a business relationship with a single accounting service. An exemplary set of collaborations, which can be constructed based on these requirements are depicted in Figure 1, where each collaboration is depicted as a triangle and annotated with a concatenation of the involved service names.

The collaboration $A1 - L1 - B1$ does not support parcel tracking and consists of services all located in Germany. Contrary, the remaining two collaborations $A1 - L2 - B2$ and $A2 - L3 - B2$ support parcel tracking, while the location of the services is irrelevant. However, due to the constraints stated above, also alternative collaborations could have been created.

[1] An example where the accounting service provides the accounting role and the buyer role is given, if a person from the accounting orders some goods.

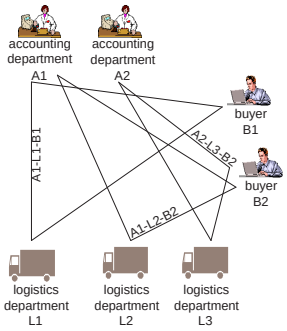

Figure 1. Concurrent Example

As stated above, a multi-lateral collaboration consists of a set of services forming a service itself. The aim of multi-lateral collaboration establishment is to find a set of services, where all roles contained in the local workflows are assigned exactly one consistent opponent. If a multi-lateral collaboration fulfills this property it is called complete. The lack of a centralized coordinating party makes establishing a complete multi-lateral collaboration a challenge.

Decentralized establishment of a multi-lateral collaboration must therefore be based on bilateral matching services, which can be checked locally. Further, it can locally be decided whether a local workflow is complete, that is, whether each role used within the local workflow is assigned exactly one bilateral matching service. As a consequence, if all local workflows involved in a multi-lateral collaboration are complete, the multi-lateral collaboration is complete.

Due to the lack of a centralized coordination the establishment of a multi-lateral collaboration based on bilateral matching local workflows requires that potential combinations have to be created by any party while all parties have to agree on the completeness of a proposed multi-lateral collaboration. The later one is known as consensus problem in distributed systems [5]. In this paper an approach addressing this issue is presented.

3. Formalization

In the following the definition made in the previous section are summarized and some formalisms are introduced. A service is given by its local workflow WF, which is based on service invocations observed as messages using roles to specify sender and recipient of the message. In particular, a role is a placeholder for a concrete service used at run-time. The set of roles used within a local workflow *WF* is specified as *R(WF)*. Further, an abstraction π_r with regard to a role r of a local workflow WF denoted as $\pi_r(WF)$ is derived by replacing all messages, which are neither sent nor received by a role r, by a silent message.

A service s is specified by the tuple $s := (WF_s, \nu_s)$ containing a service's local workflow WF_s and a mapping ν_s of the contained roles $R(WF_s)$ to a set of concrete bilaterally matching services provided by other parties. The set of all services is denoted as S. Two services s_1 and s_2 with $s_1 := (WF_1, \nu_1)$ and $s_2 := (WF_2, \nu_2)$ match bilaterally, if there exist abstractions of the local workflows which have a non-empty intersection, that is, $\exists r \in R(WF_1), r' \in R(WF_2).\pi_r(WF_1) \cap \pi_{r'}(WF_2) \neq \emptyset$. A service $s = (WF, \nu)$ is complete, if and only if each role is assigned exactly one service, that is, $\forall r \in R(WF).|\nu(r)| = 1$.

A multi-lateral collaboration C is specified as a set of n services $C := \{s_1, \ldots, s_n\} := \left(\bigcup_{i=1}^{n} WF_i, \bigcup_{i=1}^{n} \nu_i\right)$ forming a service on its own, where

$$\forall s_i \in C, r \in R(WF_i). \quad \exists^1 s_j \in C \setminus \{s_i\}.\exists r' \in R(WF_j). \; \pi_r(WF_i) \cap \pi_{r'}(WF_j) \neq \emptyset$$

Dependent on the used workflow model a formal definition of union and intersection operations can be found in [2, 11]. Under the assumption that the roles in the different local workflows are named uniquely, the union of the mapping relation can be simply merged resulting in single mapping relation ν_C.

A bilateral collaboration represents a set of two services $\{s_1, s_2\}$, where $s_1 := (WF_1, \nu_1)$ and $s_2 := (WF_2, \nu_2)$ with $\nu_1(r) = \{s_2\}$ and $\nu_2(r') = \{s_1\}$ and $\pi_r(WF_1) \cap \pi_{r'}(WF_2) \neq \emptyset$. The bilateral collaboration ID is $(ID(s_1, r), ID(s_2, r'))$ being equivalent to $(ID(s_2, r'), ID(s_1, r))$, where $ID()$ is a bijective function generating globally unique IDs. Thus, from the knowledge of a bilateral collaboration ID $(ID(s_1, r), ID(s_2, r'))$ the two services s_1 and s_2 can be derived forming a bilateral collaboration $C = \{s_1, s_2\}$.

4. Decentralized Collaboration Establishment

The approach proposed in this paper is based on four steps: First, a list of potential trading partners is derived, which provide a bilateral match with a role used in the local workflow. This step is also known as service discovery. Second, the derived partner list is used to form bilateral collaborations with selected partners being the basis to set up multi-lateral collaborations. Third, any party may combine several bilateral collaborations to form multi-lateral ones and propose them to services involved in the used bilateral collaborations. This task also contains the consensus making on the completeness of the proposed multi-lateral collaboration. Finally, the derived complete multi-lateral collaborations are evaluated whether they guarantee successful execution, that is, whether the collaboration is consistent. The different steps are discussed in detail next.

In the following it is assumed, that the underlying communication is asynchronous but reliable, while services are always available. These assumptions ease the approach, while it remains applicable for example in virtual enterprise scenarios. More general requirements result quite fast in unsolvable consensus problems (see also [5]).

4.1. Service Discovery

The first step is performed by each individual party to find relevant parties, which provide bilateral matching services with regard to a certain role used in the own local workflow as introduced in Section 3. Within the domain of Web Services this is known as service discovery. Service discovery can be applied using all kind of classification criteria for the service discovery, like for example QoS [4], choreographies [11], Semantic Web classifications [8], or simple key word based approaches (e.g. UDDI). Since the knowledge of the local workflow and the partial ones of the corresponding parties is required in the final consistency checking of a derived multi-lateral collaboration, the service discovery can also use this workflow information to improve the precision of the service discovery results, as for example proposed in [13]. The derived list of potential services providing a particular role of a local workflow are the basis for establishing bilateral collaborations.

With regard to the example introduced in Section 2, the buyer $B1$ requires to interact only with services located in Germany. As a consequence, $A1$ can be used as assignment for the accounting role, and $L1$ and $L2$ can be used as assignment for the logistics role. In addition, $B1$'s workflow does not support parcel tracking, which limits the potential services to $A1$ for the accounting role, and $L1$ for the logistics role. With regard to buyer $B2$, no regional constraint is specified, but the local workflow requires parcel tracking, which is supported by $A1$ and $A2$ fulfilling the accounting role, $L2$ and $L3$ fulfilling the logistics role.

In particular, buyer $B1$ and $B2$ have to perform bilateral service discovery for each role involved in the local workflow, that is, the accounting and logistics role respectively. Using the service discovery implementation described in [13] returns for $B1$ and $B2$ the above described values. Applying this tool to all services results in a list of bilateral matching services for each local party and role as depicted in Table 1. This service discovery result is the basis for establishing bilateral collaborations.

4.2. Bilateral Collaborations

A bilateral collaboration is an agreement of two parties with bilateral matching local workflows to cooperate in future, where this bilateral collaboration is characterized by a globally unique identifier agreed between the two parties. The quite simple handshake protocol used to establish the bilateral collaboration between two parties uses a connect

party	role: services	role: services
B1	accounting: A1	logistics: L1
B2	accounting: A1,A2	logistics: L2,L3
A1	buyer: B1,B2	logistics: L1,L2,L3
A2	buyer: B2	logistics: L2,L3
L1	accounting: A1	buyer: B1
L2	accounting: A1,A2	buyer: B2
L3	accounting: A1,A2	buyer: B2

Table 1. Bilateral Service Discovery Results

message and a preceding accept message sent by the trading partner. The protocol described below from the view point of a service i is given in Process IO-Automata notation [5]. In particular, the **states** paragraph introduces the variables used to maintain the local state of the service, which is effected by the incoming, outgoing, and internal transitions. Transitions represent state changes, where internal state changes are differentiated from transitions caused by incoming messages (input transitions) and outgoing messages (output transitions). The signatures of transitions are specified in the **input**, **output**, and **intern** paragraphs. The effects and the preconditions of the transitions are specified in the **transitions** paragraph. Finally, the **tasks** paragraph specifies in which order and for which parameters the output and intern transitions are executed. The transitions are annotated with *send_* and *receive_* indicating whether the message is sent or received.

The protocol is described from the point of view of a service i specified by the corresponding local workflow WF_i. The service discovery results of service i derived from the previous step are provided to the protocol in $match_i(r)$ for all roles $r \in R(WF_i)$. However, not all derived services may be considered as a bilateral collaboration. This may depend on internal strategies. The derived bilateral collaborations are stored in $bilateral_i$ being a set of tuples, each containing the role and the service as well as the two IDs specifying the bilateral collaboration.

states:
service i: WF_i
service discovery result: $\forall r \in R(WF_i).match_i(r) \subseteq \{j \neq i \mid \exists WF_j, r' \in R(WF_j).\pi_r(WF_i) \cap \pi_{r'}(WF_j) \neq \emptyset\}$
set of already established bilateral collaborations:
$bilateral_i$ = set of (Role, Partner, id1, id2)
input: $receive\_connect(id1)_{j,i}, receive\_accept(id1.id2)_{j,i}$
output: $send\_connect(id1)_{i,j}, send\_accept(id1, id2)_{i,j}$
intern: none
transitions:
$send\_connect(id1)_{i,j}$
precondition: $\nexists r, id1, id2.(r, j, id1, id2) \in bilateral_i$
effect: $bilateral_i := bilateral_i \cup \{(r, j, id1, null)\}$ where $\exists r.j \in match_i(r)$
$send\_accept(id1, id2)_{i,j}$
precondition: none
effect: $bilateral_i := \{(r, j, id1, id2)\} \cup (bilateral_i \setminus \{(r, j, id1, accept)\})$
$receive\_connect(id1)_{j,i}$
precondition: none
effect: if $(\exists r.j \in match_i(r))$ then $bilateral_i := bilateral_i \cup \{(r, j, id1, accept)\})$
$receive\_accept(id1, id2)_{j,i}$
precondition: $\exists r.(r, j, id1, null) \in bilateral_i$
effect: $bilateral_i := \{(r, j, id1, id2)\} \cup (bilateral_i \setminus \{(r, j, id1, null)\})$
tasks:
$\forall r \in R(WF_i).\forall j \in match_i(r).send\_connect(new\ id)_{i,j}$
$\forall r \in R(WF_i).\forall (r, j, id1, accept) \in bilateral_i.$ $send\_accept(id1, new\ id)_{i,j}$

In the following, the protocol is explained by means of one potential execution sequence between a service i and bilateral matching service j. Due to the first task, service i sends a connect message (*send_connect*) containing a unique identifier provided by the service i to a bilaterally matching service j, where no bilateral collaboration exists already. Exchanging this message inserts a tuple in $bilateral_i$ as well as in $bilateral_j$ although, the value of the second ID is different. Due to the second task, service j sends an accept message (*send_accept*) to service i, which updates the second ID of the corresponding entries in $bilateral_i$ and $bilateral_j$ by the $id2$ value provided by service j. The current specification of the protocol guarantees the establishment of a bilateral collaboration, although an internal strategy could be applied to limit sending accept messages, as required for the logistics services within the example. In this case, no further message is exchanged. In case of acceptance, a bilateral collaboration ID is constructed by concatenating the unique identifier contained in the corresponding connect message with an unique identifier provided by the accepting service. The bilateral collaboration ID is combined from the two partial ones to prevent a service to send an accept message, which has not been initiated via a connect message from the corresponding service.

Table 2 provides for each party the bilateral collaboration IDs in accordance with the different roles. However, the bilateral collaboration IDs are represented as a tuple containing the service name for better readability like for example $(A1, B1)$ being equivalent to $(B1, A1)$. The logistics $L2$ and $L3$ as stated in Section 2 have constraints on the bilateral collaborations to select exactly one service offering for the accounting role based on internal criteria like for example a trust relationship. Thus, the service $L2$ and $L3$ selected $A1$ and $A2$ as service for the accounting role respectively.

party	role: bilateral collaboration IDs	role: bilateral collaboration IDs
B1	accounting: (A1,B1)	logistics: (B1,L1)
B2	accounting: (A1,B2),(A2,B2)	logistics: (B2,L2),(B2,L3)
A1	buyer: (A1,B1),(A1,B2)	logistics: (A1,L1), (A1,L2),(A1,L3)
A2	buyer: (A2,B2)	logistics: (A2,L2),(A2,L3)
L1	accounting: (A1,L1)	buyer: (B1,L1)
L2	accounting: (A1,L2)	buyer: (B2,L2)
L3	accounting: (A2,L3)	buyer: (B2,L3)

Table 2. Bilateral Collaboration Results

4.3. Combination of Collaborations

Based on the determined bilateral collaboration IDs, a complete multi-lateral collaboration has to be established in a decentralized way. In particular, a multi-lateral collaboration is complete, if all local workflows are complete, that is, each role is assigned exactly one service. Since multi-lateral collaborations are constructed from bilateral ones, they are represented as a set of bilateral collaboration IDs. Thus, each party can construct a proposal for a multi-lateral collaboration by combining bilateral collaboration IDs one for each role used in the local workflow. To determine whether the proposed multi-lateral collaboration is complete the remaining parties involved have to be asked via a *ready* message containing the set of bilateral collaboration IDs. The following reactions are possible:

- under specification (*cancel* message): a proposed collaboration is under specified, if the local workflow contains a role which can not be associated with a bilateral collaboration ID contained in the set of IDs. That is, $\exists r \in R(WF_i).|\nu(r)| = 0$ Due to this under specification, the proposed multi-lateral collaboration can be extended by adding a bilateral collaboration ID for each role not already covered by the set of IDs forming a new proposed multi-lateral collaboration. The process starts with the new collaboration again.
- over specification (*cancel2* message): a proposed collaboration is over specified, if the local workflow contains a role which can be associated with more than one bilateral collaboration ID contained in the set of IDs. That is, $\exists r \in R(WF_i).|\nu(r)| > 1$
- consider proposal to be complete: if the set of IDs allows to associated each role of the local workflow with exactly one service, then it is considered to be complete, although the proposed multi-lateral collaboration has to be forwarded to parties reachable via the role assignment. That is, $\forall r \in R(WF_i).|\nu(r)| = 1$

In case a single party considers a proposed multi-lateral collaboration to be incomplete this decision is backward propagated until all parties are reached. A multi-lateral collaboration is finally complete, if neither a corresponding *cancel* nor *cancel2* message is send or received. Again, the protocol is specified as a Process IO-Automaton, where messages are annotated by *send_* and *receive_* indicating whether they are sent or received.

The local state of service i consists of the local workflow WF_i, the bilateral collaborations $bilateral_i$ derived from the previous protocol, the set of multi-lateral collaborations $collabs$, the set of under specified collaborations $partial$, a history of exchanged messages $history$, and the three lists $(initReady, initCancel, initCancel2)$ maintaining $ready$, $cancel$, and $cancel2$ messages to be send respectively.

states:
service i: WF_i
set of bilateral collaborations:
$bilateral_i$ = set of (Role, Partner, id1, id2)
set of cancel messages to be send: $initCancel$
set of cancel2 messages to be send: $initCancel2$
set of ready messages to be send: $initReady$
set of collaborations: $collabs$
history of exchanged messages: $history$
set of under specified collaborations: $partial$
input:
$receive\_ready(Collaboration\ C)_{j,i}$
$receive\_cancel(Collaboration\ C)_{j,i}$
$receive\_cancel2(Collaboration\ C)_{j,i}$
output:
$send\_ready(Collaboration\ C)_{i,j}$
$send\_cancel(Collaboration\ C)_{i,j}$
$send\_cancel2(Collaboration\ C)_{i,j}$
interncreate_collaboration()
transitions:
$receive\_ready(Collaboration\ C)_{j,i}$
precondition: none
effect:
$history := history \cup \{(j,' ready', C)\}$
$B := \{(r, j, id1, id2) \in bilateral_i \mid (id1, id2) \in C\}$
if $(\forall r \in R(WF_i).\exists^1(r, j, id1, id2) \in B)$
then \\ accept
$collabs := collabs \cup \{C\}$
else if $(\exists r \in R(WF_i).\ \nexists(r, j, id1, id2) \in B)$
then \\ under specified
$initCancel := initCancel \cup \{(j, C)\}$
$partial := partial \cup \{(r, j, C)\}$
else if $(\forall r \in R(WF_i).$
$\exists(r, j, id1, id2), (r, j', id1', id2') \in B.$
$(id1, id2) \not\equiv (id1', id2'))$
then \\ over specified
$initCancel2 := initCancel2 \cup \{(j, C)\}$
$receive\_cancel(Collaboration\ C)_{j,i}$
precondition: none
effect: $collabs := collabs \setminus \{C\}$

$history := history \cup \{(j,' cancel', C)\}$
$B := \{(r, j, id1, id2) \in bilateral_i \mid (id1, id2) \in C\}$
$\forall (r, j, id1, id2) \in B.$
$initCancel := initCancel \cup \{(j, C)\}$

$receive\_cancel2(Collaboration\ C)_{j,i}$
precondition: none
effect: $collabs := collabs \setminus \{C\}$
$history := history \cup \{(j,' cancel2', C)\}$
$B := \{(r, j, id1, id2) \in bilateral_i \mid (id1, id2) \in C\}$
$\forall (r, j, id1, id2) \in B.$
$initCancel2 := initCancel2 \cup \{(j, C)\}$

$send\_ready(Collaboration\ C)_{i,j}$
precondition: $(j,' ready', C) \notin history$
effect: $initReady := initReady \setminus \{(j, C)\}$
$history := history \cup \{(j,' ready', C)\}$

$send\_cancel(Collaboration\ C)_{i,j}$
precondition: $(j,' cancel', C) \notin history$
effect: $initCancel := initCancel \setminus \{(j, C)\}$
$history := history \cup \{(j,' cancel', C)\}$

$send\_cancel2(Collaboration\ C)_{i,j}$
precondition: $(j,' cancel2', C) \notin history$
effect: $initCancel2 := initCancel2 \setminus \{(j, C)\}$
$history := history \cup \{(j,' cancel2', C)\}$

$create\_collaboration()$
precondition: none
effect:
$let\ B^* := 2^{bilateral_i}$
$\forall B \in B^*.$
if $(\forall r \in R(WF_i).\exists^1(r, j, id1, id2) \in B)$ then
$C := \{(id1, id2) \mid (r, j, id1, id2) \in B\}$
$collabs := collabs \cup \{C\}$
\\ combination with partial collaborations
$\forall (r, j, id1, id2) \in B.\forall (r, j, C^*) \in partial.$
$B' := \{(r', k, id3, id4) \in bilateral_i \mid (id3, id4) \in C^*\}$
if $(\forall r' \in R(WF_i).$
$\exists^1(r', k, id3, id4) \in B \cup B')$ then
$C' := C \cup C^*$
$collabs := collabs \cup \{C'\}$
$\forall C \in collabs.$
$B := \{(r, j, id1, id2) \in bilateral_i \mid (id1, id2) \in C\}$
$\forall (r, j, id1, id2) \in B.$
$initReady := initReady \cup \{(j, C)\}$

tasks:
$\forall (j, C) \in initCancel.send\_cancel(C)_{i,j}$
$\forall (j, C) \in initCancel2.send\_cancel2(C)_{i,j}$
$\forall (j, C) \in initReady.send\_ready(C)_{i,j}$
$create\_collaboration()$

All input and output transitions add the exchanged message to the history. The *receive_cancel* and *receive_cancel2* transitions propagate the information to other services involved in the multi-lateral collaboration by adding corresponding messages to the *initCancel* and *initCancel2* list respectively. The preconditions in $send\_ready$, $send\_cancel$, and $send\_cancel2$ transitions minimize the number of exchanged messages by avoiding sending the same information several times. The $receive\_ready$ transition checks completeness of the proposed multi-lateral collaboration C with regard to the local workflow WF_i. In particular, the collaboration is accepted by adding it to $collabs$, or rejected by initiating the sending of $cancel$ or $cancel2$ in case of under or over specification. In case of under specification, the proposed collaboration is added to the set of under specified collaborations $partial$. The intern transition $create\_collaboration$ constructs in the first part complete multi-lateral collaboration proposals derived by combining the bilateral collaboration IDs (power set construction) and adding complete collaborations to $collabs$. In the second part, the constructed collaborations are extended by received collaboration proposals, which have been under specified. The resulting set of collaboration proposals is added to $collabs$ in case they are complete. Finally, for all constructed collaborations corresponding $ready$ messages are initiated.

The approach applied in this protocol is similar to the distributed construction of a spanning tree. Partial intermediate results are combined with local knowledge to come up with a global solution, that is, a complete multi-lateral collaboration. For each of these proposed collaborations, a consensus has to be achieved on whether all services consider it to be complete. In particular, the consensus is realized by forwarding local decisions similar to a simple flooding approach.

With regard to the example, the bilateral collaboration of accounting $A1$ and logistics $L1$ is represented as $\{(A1, L1)\}$ being a tuple of the two services being equivalent to $\{(L1, A1)\}$. Based on bilateral collaborations accounting $A1$ can combine the two bilateral collaborations $\{(A1, L1)\}$ and $\{(A1, B1)\}$ taken from Table 2 containing the bilateral collaborations to the collaboration $\{(A1, L1), (A1, B1)\}$, which can be extended by buyer $B1$ via the bilateral collaboration $\{(B1, L1)\}$ to the multi-lateral collaboration $\{(A1, L1), (A1, B1), (B1, L1)\}$ which has been represented in Figure 1 as $A1 - L1 - B1$. This collaboration has exactly one service assigned to each party's roles.

However, the collaboration $\{(B1, L1), (L1, A1), (A1, B2), (B2, L3), (L3, A2)\}$ can be constructed, which is still under specified, since the logistics role at buyer $B1$ and the buyer role at accounting $A2$ are still unspecified. Extending this collaboration by an additional bilateral collaboration results in an over specification of a service's role. For example, adding the bilateral collaboration $(A2, B2)$ results in the buyer $B2$'s accounting role being assigned with accounting $A1$ and $A2$ making the collaboration over specified, because the accounting role at the buyer $B2$ is mapped to services $A1$ and $A2$.

Based on this specification all multi-lateral collaborations can be constructed, because partial results are propagated to other services and are combined with local knowledge to new collaborations, which are further propagated. Finally, the construction process terminates due to the finite set of possible combinations that can be created from a finite set of bilateral collaborations.

4.4. Consistency

After complete multi-lateral collaborations have been established, the decentralized consistency checking can be applied on acyclic local workflows as follows: First, parameter constraints are propagated, that is, omitting irrelevant options at a recipient transitions, which can be excluded by the construction of the local workflow of the sending service. This modification of the local workflow effects also the abstractions for other roles, thus, requires a further propagation of these effects until a fixed point has been reached. Second, occurrence graph constraints are propagated, that is, only those message sequences are considered which are supported by the sending and receiving service. Thus, an irrelevant message sequence contained in a local workflow of a receiving party not supported by the corresponding sending service is omitted effecting also the abstraction of the local workflow with regard to other roles. Again, these effects have to be propagated until a fixed point has been reached. In case all local workflows in the fixed point are consistent, than also the global workflow of the multi-lateral collaboration is consistent, that is, guarantees a successful execution. A complete description of the approach can be found in [10].

5. Implementation and Evaluation

The presented approach has been evaluated in the application domain of Web Services. In particular, the modeling of local workflows has been based on the Business Process Execution Language for Web Services (BPEL) which also supports a notion of roles being called *partnerLinkTypes*. In particular, *partnerLinkTypes* are assigned at run-time with concrete services. Further, the notion of consistency is based on semantically and syntactically equivalent messages. In case the used message structures are based on a standard data directory like for example RosettaNet, then two messages contained in a local workflow are considered to be syntactically and semantically equivalent, if the message names are equivalent. However, the BPEL modeling of local workflows is translated into an extended Finite State Automaton model [11] in accordance to [12].

The implementation of the first and forth step are described in [13] and [10] respectively. The implementation of the second and third step follows: The implementation is based on Web Services, where each service is represented by a stand alone Jetty web server with an attached Apache Axis package and a deployed implementation of the protocol described in detail in Sections 4.2 and 4.3. The exchanged messages have been monitored by using tcpmon provided by the Axis distribution and the Jetty log file. This set up of the architecture allows concurrent receiving of messages at any point in time requiring a concurrency controlled first-in first-out message queue, which allows to have consistent local states of services.

	B1	B2	A1	A2	L1	L2	L3
send_connect	2	4	2	1	0	0	0
receive_connect	0	0	2	1	2	2	2
send_accept	0	0	2	1	2	2	2
receive_accept	2	4	2	1	0	0	0
send_ready	24	44	44	24	23	28	24
receive_ready	16	58	57	18	17	27	17
send_cancel	15	27	27	15	15	19	15
receive_cancel	7	41	39	7	6	23	6
send_cancel2	23	34	32	28	26	18	32
receive_cancel2	18	52	53	20	15	13	16
Sum	107	264	260	116	106	132	114

Table 3. Exchanged Messages

The above sketched implementation has been tested at different scenarios. The observations made are that all scenarios terminated and all complete multi-lateral collaborations have been derived correctly. Thus, a decentralized service composition has been derived. In the following, the scenario described in Section 2 is used to give an exemplary evaluation and discuss of the presented approach.

One evaluation criteria of the approach is its efficiency with regard to the number of exchanged messages. Table 3 contains the number of exchanged message per service, while differentiating sent and received messages. It can be observed, that the total number of messages required to instantiate the bilateral collaborations is quite low (18), while the number required to derive the complete collaborations is quite high (537). Further, the sum of messages a service has been involved in indicates the number of different options which have to be considered at a single service. In particular, the service $B2$ and $A1$ exchanged the highest number of messages due to the higher number of proposed multi-lateral collaborations that can be constructed based on the bilateral collaborations as indicated in Table 2. Comparing the number of sent and received messages shows that one *ready*, four *cancel*, and six *cancel2* messages have been lost. This loss of messages is a consequence of an overload situation at a recipient web server resulting in a time-out of a HTTP connection used as the underlying communication protocol implementing the Web Service. The current implementation does not handle this kind of errors correctly. The

issue will be addressed in a later version.

The summary of this evaluation is that the protocol works fine, although it is generating a lot of communication overhead, which might cause significant communication overload situations in large scenarios. The communication overhead can be significantly decrease if strategies for composing potential collaborations are defined as opposed to the current approach of proposing every combination.

6. Related Work

The related work addressing alternative workflow models especially under consideration of decentralized decision making on properties of multi-lateral collaborations has extensively be discussed in [10]. The main result is that the few approaches supporting multi-lateral collaborations allow decisions on properties of multi-lateral collaborations only on a global view of the collaboration ([1, 3]). Further, these approaches assume a pre-existent knowledge of the multi-lateral collaboration and do not address the issue of constructing such collaborations in a decentralized way.

However, a lot of general work has been published on the service discovery step as well as the consensus making in decentralized systems. With regard to the service discovery step a detailed discussion of related work can be found in [13]. A main result of this discussion is that although a lot of approaches exist only a few of them address service discovery using process descriptions as discussed in this paper like for example [6] being based on state machines.

With regard to consensus making, algorithms for a minimal spanning tree like for example GHS [5] differ significantly from the problem addressed in this paper: the parties between whom the consensus has to be made must be known in advance. In particular, all available services are part of the consensus making problem. However, in the problem addressed in this paper, the main challenge is in constructing multi-lateral collaborations, that is, finding all relevant subsets of the set of services and do consensus making afterwards based on a simple flooding approach [5]

Related work in the area of service composition focus mostly on simple services, thus, completely ignores the process aspects of services and requires a global view of the service composition process like e.g. in [7].

7. Conclusion and Future Work

In this paper a protocol for decision making of composed services has been presented. In particular, previous service discovery results and an approach for decentralized consistency checking has been reused, while a decentralized approach for constructing service compositions has been described in detail. Further, an evaluation has been briefly described and its applicability has been illustrated on behalf of an example. This initial step of a decentralized service composition focuses on workflow aspects of services, and does not incorporate additional service description dimensions like for example Quality of Service, security aspects, or semantic issues. Some of these aspects will be considered in future work.

References

[1] W. Aalst. Interorganizational workflows: An approach based on message sequence charts and petri nets. *Systems Analysis - Modelling - Simulation*, 34(3):335–367, 1999.

[2] W. Aalst. Inheritance of Interorganizational Workflows to Enable Business-to-Business E-commerce. *Electronic Commerce Research*, 2(3):195–231, 2002.

[3] E. Kindler, A. Martens, and W. Reisig. Inter-operability of workflow applications: Local criteria for global soundness. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 235–253. Springer-Verlag, 2000.

[4] Y. Liu, A. H. H. Ngu, and L. Zeng. Qos computation and policing in dynamic web service selection. In *Proc. of WWW*, page 66ff, 2004.

[5] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[6] C. Molina-Jimenez, S. Shrivastava, E. Solaiman, and J. Warne. Contract representation for run-time monitoring and enforcement. In *Proc. of Conf. on Electronic Commerce (CEC)*, pages 103–110. IEEE, 2003.

[7] B. Orriëns, J. Yang, and M. P. Papazoglou. Model driven service composition. In *Proc. of the First Intl. Conf. on Service-Oriented Computing (ICSOC)*, pages 75–90, 2003.

[8] D. Trastour, C. Bartolini, and J. Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services. Technical Report HPL-2001-183, Hewlett-Packard, July 2001.

[9] A. Wombacher and K. Aberer. Requirements for workflow modeling in P2P-workflows derived from collaboration establishment. In *Proc. of Intl. Workshop on Business Process Integration and Management (BPIM)*, pages 1036–1041, 2004.

[10] A. Wombacher, P. Fankhauser, and K. Aberer. Overview on decentralized establishment of consistent multi-lateral collaborations. In *Proc. of IEEE Intl. Conf. on E-Technology, E-Commerce and E-Service (EEE)*, 2005.

[11] A. Wombacher, P. Fankhauser, B. Mahleko, and E. Neuhold. Matchmaking for business processes based on choreographies. In *Proc. of Intl. Conf. on e-Technology, e-Commerce and e-Service (EEE)*, pages 28–31. IEEE Computer Society, 2004.

[12] A. Wombacher, P. Fankhauser, and E. Neuhold. Transforming BPEL into annotated deterministic finite state automata enabling process annotated service discovery. In *Proc. of Intl. Conf. on Web Services (ICWS)*, pages 316–323, 2004.

[13] A. Wombacher, B. Mahleko, and E. Neuhold. IPSI-PF: A business process matchmaking engine. In *Proc. of Conf. on Electronic Commerce (CEC)*, pages 137–145, 2004.