ISeB (2005) 3: 127–150
DOI 10.1007/s10257-005-0053-y

ORIGINAL ARTICLE

Andreas Wombacher · Bendick Mahleko · Erich Neuhold

IPSI-PF. A business process matchmaking engine based on annotated finite state automata

Abstract Success of Web services mainly depends on the availability of tools facilitating usage of technology within the addressed B2B integration problems. One severe problem in loosely coupled systems is service discovery including a sufficient matchmaking definition. The concept for service discovery in web service architecture is UDDI providing limited querying functionality and not being capable to deal with the multiple dimensions of a service, like for example semantic, workflow, or Quality of Service aspects. The IPSI Process Finder (IPSI-PF) realizes service discovery by extending the capabilities of UDDI by matchmaking of service descriptions. The extension is realized such that an integration of additional extensions can be added quite easily.

Key words Service discover · Web services · Service orientated architecture

1 Introduction

Web services and related technologies promise to facilitate the efficient execution of B2B e-commerce by integrating business applications across networks like the Internet. A lot of effort has been expended to define standards, e.g., to model business processes and describing interfaces (BPEL4WS, WSDL, etc.) as well as specifying the technical infrastructure for carrying out business transactions (e.g., SOAP, UDDI). If web services are to be as successful as predicted, tools that aid companies, organiza-

A. Wombacher (✉) · B. Mahleko · E. Neuhold
Fraunhofer Gesellschaft, Integrated Publication and Information Systems Institute, Darmstadt, 64293 Germany.
E-mail: andreas.wombacher@ipsi.fraunhofer.de

tions and individuals in their day-to-day work must be available. Applicability of web service technologies in real life scenarios will thus dependent to a large scale on the availability and usefulness of such tools, and what impact they have on the business community.

Web services are advertised as a technology for implementing loosely coupled business processes, that is a dynamic and flexible binding of services. To date, web services are often used as stateless components accessible via a single request-response remote procedure call (RPC). One reason for this limited usage is the missing support for searching and finding state-maintaining/complex web services, that is, the lack of efficient service repositories with expressive query capabilities.

Within this paper we present a tool that allows searching and finding service providers by an automated comparison of business processes for consistency based on an existing UDDI infrastructure. Such a tool takes as input a business process description belonging to one party and returns references to service providers offering services that are consistent, that is, guarantee a successful business interaction with the input business process. The idea is to allow companies and organizations to automatically determine from a business process viewpoint, whom they can do business with. Alternatively, if the organization is already aware of who their potential business partners are, the tool allows them to check whether or not they fulfill their requirement in terms of business process consistency also called matchmaking of business processes. Of course, the underlying assumption is that the different parties have a common semantics on message names. This assumption reduces the complexity of the problem, since no semantic reasoning on messages is required. However, this assumption is applicable since several standards exists (like e.g. OTA, RosettaNet, IOTP) where different organizations agreed on the semantics of a set of messages. We have implemented a tool checking consistency of service provider's workflow description with the service requesters one. The aim of this paper is to describe its implementation and how it can be used within the web service framework.

The paper is organized as follows: Sect. 2 describes an exemplary application, Sect. 3 explains the used approach and Sect. 4 presents the architecture and implementation. Evaluation of IPSI Process Finder **IPSI-PF** matchmaking engine is done in Sect. 5, followed by a discussion on related work in Sect. 6. Summary and future work are presented in Sect. 7. Parts of this work have been published in (Wombacher et al. 2004)

2 Sample application

The exemplary scenario used for further discussion is a simple procurement workflow within a virtual enterprise incorporating a buyer, an accounting department, and a logistics department. The accounting department approves an order (*order* message) sent by a buyer and forwards the order to the logistics department (*deliver* message) to deliver the requested goods. The logistics department confirms the receipt (*deliver_ conf* message) and forwards it to the buyer extended by the expected deliver date and the parcel

tracking number using the *delivery* message. Further, the buyer may perform parcel tracking (*get_ status* and *status* messages) of the shipped goods, which is forwarded by the accounting department to the logistics department. The overall scenario is depicted in Fig. 1.

The scenario sketched above is used for explaining the concepts within the IPSI Process Finder (IPSI-PF) matchmaking engine. In the following the local workflow of the accounting department denoted according to the BPEL specification is described in more detail. To keep the example simple, the specification of the messages is neglected and the names of the messages are simplified. Concrete message types could be exemplary taken from the RosettaNet Partner Interface Processes (PIPs) 3A4 (Request Purchase Order), 3A7 (Notify of Purchase Order Update), 3B2 (Notify of Advanced Shipment) (RosettaNet). As stated in the introduction, due to the usage of these commonly agreed semantics of messages, using the same message name implies semantic equivalence of messages.

Within Web service specification one or more messages specify an operation representing a potential message exchange. If a porttype contains only a single input, than the operation is asynchronous, otherwise it is synchronous. A porttype contains a set of operations provided by a service provider, which is specified in the corresponding WSDL files. Figure 2 gives an overview of the operations and the related messages used in the exemplary accounting department BPEL description, where the messages are labeled in accordance to the high level description of the scenario. The *accounting* and the *logistics* porttypes represent the operations provided by the corresponding department service, that is contain messages that are received by the accounting and logistics department respectively. Consequently, the *accountingCallback* and *logisticCallback* porttypes contain operations provided by the buyer and the logistics department respectively. All operations are asynchronous except the *getStatusOP* operation provided in the *logisticCallback* porttype, which is synchronous.

The description of the local workflow is based on these porttype definitions by directly referencing them. The accounting department local workflow denoted as BPEL process is depicted in Fig. 3. The process starts by

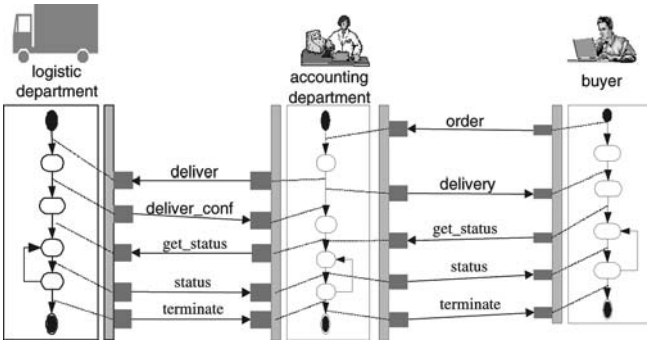

Fig. 1 Global procurement scenario

```
<portType name="accounting">
    <operation name="orderOp">
        <input message="tns:order"/>
    </operation>
    <operation name="getStatusOp">
        <input message="tns:getStatus"/>
    </operation>
    <operation name="terminateOp">
        <input message="tns:terminate"/>
    </operation>
</portType>
<portType name="accountingCallback">
    <operation name="deliveryOp">
        <input message="tns:delivery"/>
    </operation>
    <operation name="statusOp">
        <input message="tns:status"/>
    </operation>
</portType>
```

```
<portType name="logistic">
    <operation name="outOfStockOp">
        <input message="tns:delivery"/>
    </operation>
    <operation name="deliver_confOp">
        <input message="tns:deliver_conf"/>
    </operation>
</portType>
<portType name="logisticCallback">
    <operation name="getStatusOp">
        <input message="tns:getStatus"/>
        <output message="tns:status"/>
    </operation>
    <operation name="deliverOp">
        <input message="tns:deliver"/>
    </operation>
    <operation name="terminateOp">
        <input message="tns:terminate"/>
    </operation>
</portType>
```

Fig. 2 WSDL porttype definition of the exemplary scenario

receiving an *order* message sent by the buyer, which is forwarded to the logistics department via a *deliver* message. The logistics department answers asynchronously with a *deliver_ conf* message, which is forwarded by the accounting process to the buyer via a *delivery* message. Due to the fact that the buyer is allowed to do parcel tracking an undetermined number of times, the parcel tracking must be contained within a non-terminating loop. To enable a termination of the accounting and logistics department processes a *termination* message initiated by the buyer and forwarded by the accounting to the logistics department terminates the corresponding processes. Alternatively, the accounting department may receive a *get_ status* message sent by the buyer, which is forwarded by a synchronous invocation of the

```
<process>
    ......
    <sequence name="accounting department process">
        <receive partnerLink="buyer" portType="tns:accounting" operation="orderOp" variable="order"/>
        <invoke partnerLink="logistic" portType="tns:logisticCallback" operation="deliverOp" inputVariable="order"/>
        <receive partnerLink="logistic" portType="tns:logistic" operation="deliver_confOp" inputVariable="order"/>
        <invoke partnerLink="buyer" portType="tns:accountingCallback" operation="deliveryOp" inputVariable="order"/>
        <while name="parcel tracking" condition="1=1">
            <pick>
                <onMessage partnerLink="buyer" portType="tns:accounting" operation="getStatusOp" variable="getStatus">
                    <sequence>
                        <invoke partnerLink="logistic" portType="tns:logisticCallback" operation="getStatusOp"
                                inputVariable="getStatus" outputVariable="status"/>
                        <invoke partnerLink="buyer" portType="tns:accountingCallback" operation="statusOp"
                                inputVariable="status"/>
                    </sequence>
                </onMessage>
                <onMessage partnerLink="buyer" portType="tns:accounting" operation="terminateOp" variable="terminate">
                    <sequence>
                        <invoke partnerLink="logistic" portType="tns:logisticCallback" operation="terminateOp"
                                inputVariable="terminate"/>
                        <terminate/>
                    </sequence>
                </onMessage>
            </pick>
        </while>
    </sequence>
</process>
```

Fig. 3 BPEL notation of the accounting local workflow

logistics process and reporting the status via a *status* message back to the buyer.

Based on this process description the aim of an accounting department is to find a logistics service provider, which is compatible with the own local workflow, that is finding a complex web service provider guaranteeing a successful business interaction. Due to general Web Service Architecture, the accounting department is querying a service repository to derive a list of potential service providers each fulfilling the constraints on the service definition expressed in the query. With regard to the example, the accounting department issues a query searching services, which

1. Support the role of a logistics department,
2. Are provided by a business entity located in Germany, and
3. Are compatible with the own local workflow specified by a BPEL process document.

3 Approach

Analyzing the above query shows, that the first two constraints can already be served by a UDDI registry being the proposed service repository within web service infrastructure. The last constraint can not be handled by a UDDI registry, because the required comparison operation is more complex than string comparison as provided by a UDDI registry. The IPSI-PF extends a UDDI repository by the feature for matching workflow descriptions. In particular, a query posed to IPSI-PF is decomposed into a UDDI and a local workflow sub-query. The subqueries are processed by a classical UDDI and the IPSI-PF matchmaking engine. The final query result is derived on behalf of the partial results by doing a natural join, where the business service key maintained by the UDDI and being unique for each service instance within the UDDI is used as a primary key. The local workflow subquery uses the business service key as a foreign key. In addition to querying the IPSI-PF provides also a publish functionality, which is not discussed in detail within this paper.

3.1 UDDI sub-query

The UDDI query is based on standard UDDI API, that is a *find_ service* SOAP call (IBM 2002). The call related to the above example is depicted in Fig. 4. In this example, the categorization of the service providers geographical location is realized by the common taxonomy based on ISO 3166-1999 which is predefined in UDDI repository. The taxonomy of roles is implemented as a 'private' taxonomy described in more detail in Sect. 4.4.

```
<find_service generic="2.0" xmlns="urn:uddi-org:api_v2">
    <findQualifiers>
        <findQualifier>caseSensitiveMatch</findQualifier>
    </findQualifiers>
    <name>%</name>
    <categoryBag>
        <keyedReference keyName="de.fhg.ipsi.oasys.ipsi-pf:IOTP:Role" keyValue="logistic'
                        tModelKey="uuid:A035A07C-F362-44DD-8F95-E2B134BF43B4"/>
        <keyedReference keyName="uddi-org:iso-ch:3166-1999" keyValue="de"
                        tModelKey="uuid:4E49A8D6-D5A2-4FC2-93A0-0411D8D19E88"/>
    </categoryBag>
</find_service>
```

Fig. 4 UDDI query

3.2 Local workflow sub-query

BPEL (Andrews et al. 2003) specifies a workflow in terms of tasks (activities in BPEL terminology) representing basic pieces of work to be performed by potentially nested services. The control flow of the BPEL process constraints the performance of tasks by choice, sequence, and parallel execution. In addition a BPEL process may also define the data flow of the business process regardless of concrete implementation for the tasks. Based on this understanding, a workflow model includes activities realizing the interaction with partners represented by exchanging messages. So, the set of message sequences supported by a workflow model can be derived. When searching for potential service providers, it is necessary that the exchanged message sequences of the query process are consistent to potential trading partners message sequences derived from his local workflow. To be able to check consistency, the following aspects must be considered:

- Due to the fact that BPEL lacks a formal model (van der Aalst 2003), a definition of a compatibility operation might be quite vague. It is preferred to have the match operation defined on a solid formal model based on sets of message sequences.
- Testing consistency is a binary operation, that is all elements contained in a message sequence not related to the opponent must be omitted. Thus, a partner specific view on the own supported message sequence must be calculated.
- The match operation compares the set of message sequences provided by a service provider and contained in the database with the relevant set of message sequences related to the query process.

A detailed description of the following three subsections can be found in (Wombacher et al. 2004a, b).

3.3 Formal model

While different formal models representing sets of message sequences exist, in the following, the annotated finite state automata (aFSA) (Wombacher et al. 2004) model is used because it is based on states and transitions like

BPEL is, and it provides preliminary results required to express the proposed approach. Other notations like for example, Petri Net (Jensen 1992, Peterson 1981), Workflow Net (van der Aalst and van Hee 2002), flowcharts (Grefen et al. 2000, Klingemann et al. 1999), statecharts (Harel 1987; Harel and Naamad 1996; Peron 1995), or finite state automata (Hopcroft et al. 2001) could also have been used. In the following the required formal definitions of aFSA and partial aFSA is introduced.

Definition 1 (annotated Finite State Automata (aFSA)) *An annotated finite state automaton A is represented as a tuple* $A = (Q, \Sigma, \Delta, q_0, F, QA)$ *where* Q *is a finite set of states,* Σ *is a finite set of messages,* $\Delta : Q \times \Sigma \times Q$ *represents labeled transitions,* q_0 *a start state with* $q_0 \in Q$, $F \subseteq Q$ *a set of final states, and* QA: $Q \times E$ *is a finite relation of states and logical terms within the set E of propositional logic terms.*

The terms in E are standard Boolean formulas. Adapting the definition in (Chomicki and Saake G 1998):

Definition 2 (definition of terms)
The syntax of the supported logical formulas is given as follows:

- *the constants true and false are formulas,*
- *the variables* $v \in \Sigma$ *are formulas,*
- *if* ϕ *is a formula, so is* $\neg\phi$,
- *if* ϕ and ψ *are formulas, so is* $\phi \wedge \psi$ *and* $\phi \vee \psi$.

Based on the aFSA definition, an intersection and emptiness operation has been defined in (Wombacher et al. 2004), which is quite similar to the one of standard automata. In particular, intersection combines annotation of states by conjunction, while the emptiness test checks reachability of final states not only via a single transition, but via all transitions contained in a conjunction of an annotation.

An annotated finite state automaton (aFSA) consists of states (circles) representing business tasks and transitions connecting states representing a message exchange. The transitions are labeled with *s#r#msg* representing sender *s* and receiver *r* of the message as well as its message name *msg*. aFSAs contain final states represented by a circle with a solid line within the automaton. The start state is indicated by an input arc of the start state. An annotation of a state is given within a square connected to the state via a line.

The query process of the accounting department as depicted in Fig. 3 can be translated into an aFSA for a particular trading partner by doing the following mapping of BPEL activities:

- represent *send*, *receive*, *pick*, and *invoke* activities as transitions
- *switch* and *pick* activities represent choices, that is modeled as several transitions
- a *flow* activity represents a parallel execution, that is modeled by shuffling the parallel tasks
- data management operations like *assign* are neglected

The annotated FSA of the accounting department derived by this transformation is depicted in Fig. 5.

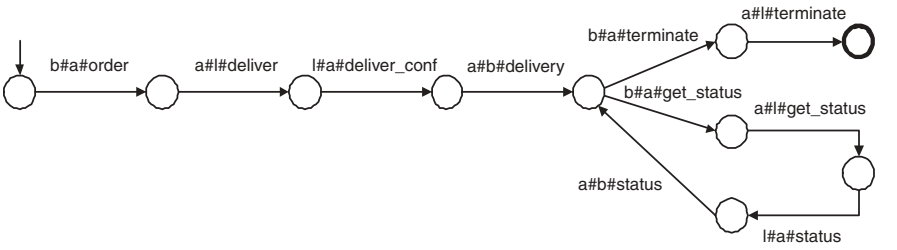

Fig. 5 Annotated FSA of the accounting department local workflow

The process is started by the buyer *b* sending a *b#a#order* message to the accounting department *a*, which forwards the order to the logistics department *l* via a *a#l#deliver* message. The logistics department *l* confirms this request (*l#a#deliver_ conf* message extending the provided information with the planned delivery date and the parcel tracking number) to the accounting department *a*, which forwards the delivery details of the order (*a#b#delivery* message) to the buyer *b*. Afterwards, the buyer *b* is allowed to do parcel tracking with the logistics department *l*, where the accounting department acts as a proxy to the buyer *b*. The corresponding messages are *b#a#get_status* and *a#l#get_ status* for requesting the status being answered by *l#a#status* and *a#b#status*. Alternatively, the processing can be terminated by receiving a *b#a#terminate* message from the buyer, which is forwarded to the logistics department by the *a#l#terminate* message.

3.4 View generation

The accounting department local workflow depicted in Fig. 5 contains messages related to the interaction between buyer and accounting department. Because these messages are irrelevant for the interaction with the logistics department, they must be omitted for calculating the match operation between accounting and logistics workflow. This can be achieved by replacing messages related to the buyer with silent transitions ϵ, which can be removed finally by ϵ removal algorithms similar to those of standard finite state automata. A possible logistics view of the accounting department local workflow is depicted as an annotated FSA in Fig. 6.

3.5 Match operation

Two workflows are consistent, that is, they match, if they guarantee a successful interaction. A necessary condition to achieve successful interactions is that the workflows provide at least a single common message sequence, that is, the intersection of the sets of message sequences supported by the workflows respectively is non-empty. The operands of the match operation are the corresponding views of the local workflows.

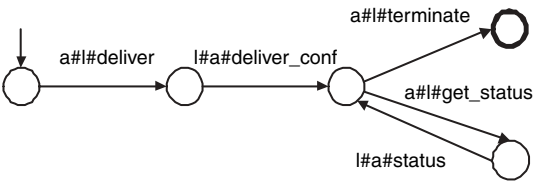

Fig. 6 View of the logistics department on the accounting department formal local workflow

To be able to illustrate the match operation, Fig. 7 depicts an exemplary logistics department local workflow denoted as an annotated FSA. Because all messages within this workflow are related to the interaction with the accounting department, the view of the accounting department on this workflow is equivalent to the original workflow depicted in Fig. 7. The logistics workflow uses the same messages as the accounting department workflow extended by the *l#a#outOfStock* message representing an out of stock information sent by the logistics department *l* to the accounting department *a*. After receiving the *a#l#deliver* message the logistics has to decide to either send a *l#a#deliver_ conf* or a *l#a#outOfStock* message. Since the accounting must support both options, the state is annotated with a logical expression of a conjunction of the two messages representing these two options. The logistics department extends the functionality provided by the accounting department workflow by supporting parcel tracking at any time and providing an option for cancellation of *deliver* messages due to an ordered good being out of stock.

Calculating the match operation of the annotated FSA depicted in Figs. 6 and 7 requires that they have a non-empty set of common message sequences. The set of common message sequence is

$$a\#l\#deliver\#l\#a\#deliver\_confa\#l\#terminate$$

$$a\#l\#deliver\#l\#a\#deliver\_confa\#l\#get\_statusl\#a\#statusa\#l\#terminate\vdots\}$$

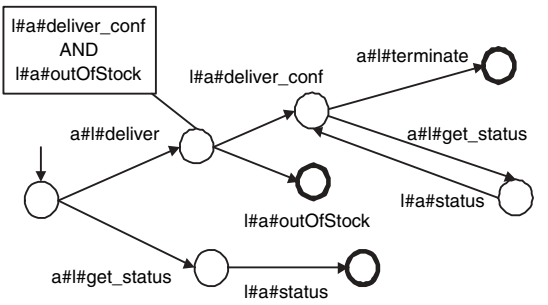

Fig. 7 View of the accounting department on the logistics department formal local workflow

Thus, the necessary condition is fulfilled. But, to be able to guarantee successful interaction the choices represented in each annotated FSA, but not contained in the set of common message sequences must be analyzed. In particular, it must be ensured that all messages that might be sent within the sender's workflow are accepted by the receiving workflow. This sufficient condition is illustrated on behalf of the above example.

While the additional option of the logistics department workflow supporting parcel tracking at any time does not result in message sequences not supported by the accounting department, this does not hold true for the out of stock option. In particular, after receiving a *a#l#deliver* message the logistics department might send a *l#a#outOfStock* message, which is not supported by the accounting workflow causing a failing interaction.

Thus, the views of the accounting and the logistics departments local workflows do not guarantee a successful interaction, thus the match operation fails. If the out of stock option in the logistics department workflow is omitted, the workflow matches the view of the accounting department workflow.

Matching of views of local workflows can be formally specified as non-empty intersection of the corresponding annotated FSA.

4 Architecture and implementation

This section describes an architecture and implementation of the IPSI Process Finder (IPSI-PF), realizing a service discovery for statefull web services supporting UDDI queries extended by workflow descriptions. As stated in the previous section, the input query has two parts: (i) the BPEL (Andrews et al. 2003) part and (ii) the UDDI (IBM 2002) part. The BPEL part contains process-related descriptions of the query and the UDDI part provides information that is traditionally provided via the UDDI registry, e.g., business service categories.

4.1 Framework

Processing the query is initiated by submitting a form as exemplary depicted in Fig. 8. Triggered by this page the data flow of the architecture depicted in Fig. 9 is started, which is realized as an Apache Cocoon pipeline. We decided for Apache Cocoon, because the data flow between the different processing steps is XML based and thus provides good support by Cocoon. In particular, the pipeline realizes the query decomposition as well as the result list merge component depicted in Fig. 9. A query decomposition component separates the three parts as follows: BPEL part is sent to the matchmaking engine via a transformation component, BPEL $\rightarrow$ formal model, while the UDDI part is sent to the category matchmaking component being realized by a Cocoon LogicalSheet in the pipeling calling the web services based UDDI API. Finally, the merging of the results is also part of this pipeline and is realized by using XSLT.

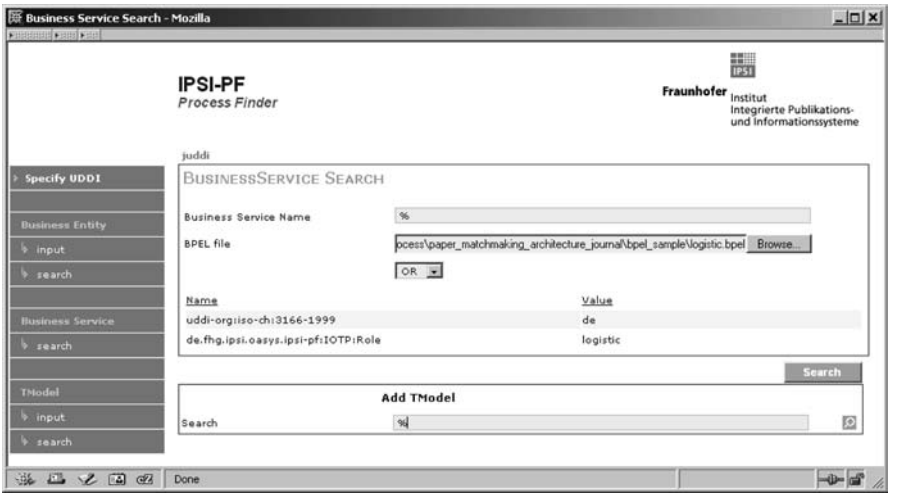

Fig. 8 Exemplary Query Form

We decided not to use a more expressive framework like e.g. Struts or a workflow engine due to the fact that the remaining user interactions within the IPSI-PF are much simpler and using the before mentioned concepts seem to be not appropriate.

The implementation is based on Apache Cocoon 2.1.2. The above described pipeline uses *cinclude*, *XSLT* and *session* transformers. A *cinclude* transformer allows to load content of different web resources into a single document. A *XSLT* transformer applies a specified XSLT stylesheet on the intermediate version of the document passed through the pipeline and a *session* transformer grants access to data contained in a HTTP request and stores intermediate results in a session container.

4.2 BPEL $\rightarrow$ formal model component

The role of the BPEL $\rightarrow$ formal model component is to transform the BPEL description to a formal model suitable for calculating process matches as described in Sect. 3.2. The results of the transformation are used as input to the matchmaking engine for deciding the match of the processes. This component has been implemented in Java being part of the process matchmaking engine component described next.

4.3 Process matchmaking engine component

This component is in charge of performing the comparisons of the query workflow with the workflows stored in the IPSI-PF as introduced in Sect.

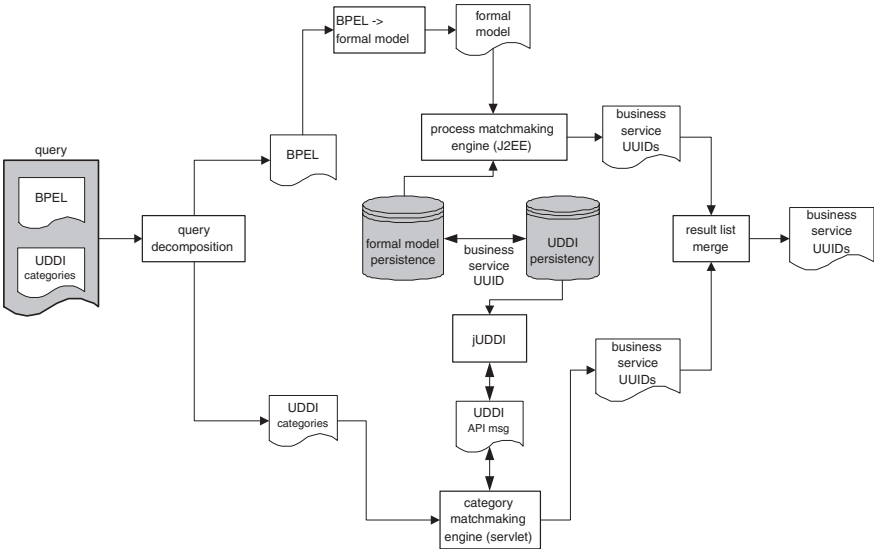

Fig. 9 Architecture

3.2. The formal model used for calculating business process matches is annotated finite state automata 2004.

Each business process in the collection is associated with category data in the UDDI repository via a business service UUID used as a foreign key (see also Fig. 9) to join the partial results as described above.

The matchmaking engine is implemented based on J2EE architecture. The used application server is JBoss. The main matchmaking component is an Enterprise Java Bean (EJB) and a web client (not included in Fig. 9) is used for administering the server. Persistency is achieved by J2EE container managed persistency relying on a MySQL database system.

4.4 UDDI category matching component

The Category Matchmaking component relies on a UDDI repository as introduced in Sect. 3.1. Most of the information queried by UDDI subquery are maintained in a UDDI registry within categorization bags. A categorization is a *name-value* pair assigned to a service entity, where the *name* is the name of the used taxonomy and the *value* is a taxonomy value. The UDDI specification provides a number of predefined, common taxonomies, like for example ISO 3166-1999, that is, a taxonomy for geographical locations. To establish such a taxonomy within a UDDI repository, a publication process including approval of the registrar to the taxonomy has been specified (IBM et al. 2002) to ensure that only those taxonomies are published, which are relevant to the user group of the UDDI repository.

With regard to the example, a taxonomy representing the roles within the exemplary procurement scenario will not get the approval for a global taxonomy, because it is used only by a specific virtual enterprise. To allow users

to introduce private categorization of services within a local UDDI registry the UDDI specification provides an alternative approach named ''General Keywords taxonomy''. Similar to the general taxonomy approach *name-value* pairs are maintained, where the *names* must be prefixed with the organizations own namespace and the supported *values* must be published in the virtual organization internally. In the prototype implementation we used jUDDI as an open source implementation of the UDDI repository.

4.5 Administration tool

The administration tool (Fig. 10) was developed as part of the IPSI-PF project to support the testing and evaluation of the IPSI-PF. The functionality provided by this tools is

- the construction of exemplary data set is based on a general specification of potential message sequences and assigned probabilities on different branches,
- the analysis of the constructed data sets,
- the performance of a bunch of queries, measuring the individual results, and providing different analysis of the response times, and
- the maintenance of the database, that is, the bulk up-load of data into IPSI-PF, the clean-up of IPSI-PF, and providing status information on the number of objects persistently contained in IPSI-PF.

5 Evaluation

The architecture described above has been implemented and first tests have been performed so far. In the following, a discussion of the applicability of the concept and the implementation is presented as well as initial results of the performance testing.

5.1 Conceptual discussion

The conceptual approach sketched in this paper can be applied to testing compatibility of bilateral interaction of services. In case of multi-lateral collaborations, the approach is limited to collaborations, where from a global point of view the interactions between the different parties represent a tree structure rather than a graph. The reason for this limitation is that dependencies between different interactions maintained by a single party may result in cyclic dependencies (Wombacher and Aberer 2004), which could not be recovered by the presented approach. This limitation currently is less restrictive due to the fact that current usage of web services rarely addresses this issue in current real world applications.

A further limitation is the assumption of robust local workflows, that is, local workflows which are unconstraint in receiving messages. This

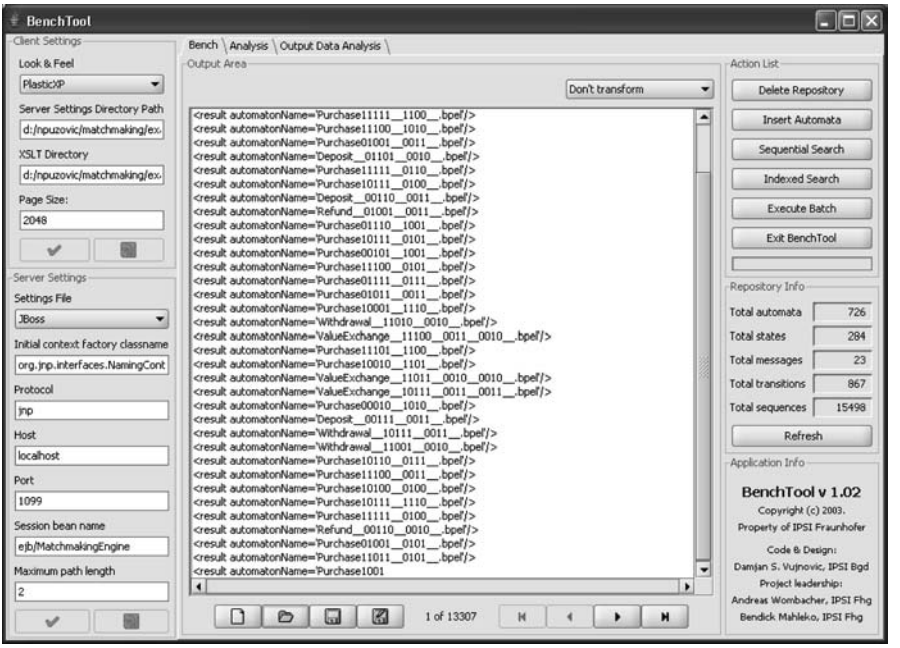

Fig. 10 Benchmark tool

limitation ensures that no explicit handling of conditions has to be established for checking the consistency of bilateral interaction of services. Finally, the approach is based on a subsumption relation on message names. In particular, the subsumption relation used in the proposed approach is the string equivalence although other approaches are possible, as e.g. addressed by the semantic web community.

An advantage of the concept is that it relies on existing UDDI infrastructure extended by querying of workflow models. In particular, the provided match operation increases the precision of the query results and avoids false matches - that is finding service providers which claim to be compatible without being it - and without having false misses - that is not returning service providers, which are compatible but have not been found by the match operation.

5.2 Implementation discussion

The current implementation has been realized to evaluate the concept and gain experience in doing querying of workflow models. Thus, it has some limitations, which will be addressed in the next release. In particular, the BPEL $\rightarrow$ formal model component currently does not support *links* and *join conditions* being part of the BPEL control structure specification, which limits the expressiveness of supported BPEL workflows. An analysis of the overall BPEL expressiveness can be found at (Wohed et al. 2002) being

useful to understand the concrete effects of the before mentioned limitation on workflow modeling. While doing experiments with the implementation, we didn't come across a concrete real example, which couldn't have been modeled considering this limitation.

As stated in Sect. 4.3 the persistence implementation of the process matchmaking engine is based on container managed persistence. This design decision has been made for ease of use and by the knowledge that we nevertheless have to re-implement the persistence model after gaining first experience with the process matchmaking component. We plan that the next release will be based on a bean managed persistence model.

5.3 Performance measurements

The main goal of the evaluation was to find out the major factors influencing performance.

5.3.1 System parameters

Measurements were conducted on a Dell machine, with a Pentium 4 processor 2.00 GHz clock speed and 512 MB RAM. The total disk space was 74 GB. The machine was running under Windows XP operating system. MySQL server version 4.0 was the used database engine. The machine was also running the JBOSS 3.2.3 application server which provided the J2EE environment for the IPSI-PF process matchmaking engine. The matchmaking engine is implemented on the same data model and level of abstraction using container managed persistency. We allowed no buffering/caching of results; all tests were run under cold start conditions.

5.3.2 Specification of data set

Performance measurements were based on a data set of local workflows in accordance to the Internet Open Trading Protocol (IOTP) (Burdett 2000) specification. We used IOTP to ensure realistic workflows as well as to achieve an appropriate structural complexity of workflows. Figure 11 shows the message exchange structure for IOTP data from which valid workflows were generated.

The circle labeled with **S** represents the start state, while that labeled with **E** represents the end state. A valid workflow has at least one path starting from the start state and terminating in an end state. Each exchange consists of at least two messages being exchanged, where

- the authentication exchange supports the authentication of one trading partner,
- the brand dependent/independent offer exchange represents signing an offer with/without a predefined payment method,

– the payment exchange and delivery exchange represent exchange of payment and delivery information respectively, and
– the payment and delivery exchange is a shortcut for the payment exchange followed by a delivery exchange.

These exchanges are used to construct characteristic business patterns like purchase, authentication, deposit, refund, withdrawal and value exchange by combining the exchanges along different path depicted in Fig. 11.

A total of 726 workflows was generated based on the IOTP specification. The number of transitions, states, messages, cycles and their respective objects in the database were measured and recorded.

Table 1 summarizes results of these measurements. The information in Table 1 gives an idea about the complexity of the input data set. In particular, the data comprised a mixture of workflows with cycles and without cycles.

Figure 12 shows the distribution of transitions in the data set. The figure shows that most workflows had between 13 and 40 transitions, with the simplest workflow having 5 transitions and the most complex having between 65 and 70 transitions. The highest number of workflows having the same number of transitions was 38, with 14 transitions.

Figure 13 shows a plot of the number of states against number of transitions. The plot shows an almost linear relationship between the number of states and number of transitions. Two lines with different slopes can be observed. The line with a steeper slope represents those workflows without cycles, because there are more transitions for some states (due to the reuse of states by cyclic transitions). The line whose slope is less steep represents workflows with cycles. This is because some states are reused by cyclic transitions, thus resulting in less overall number of states for a given number of transitions. The length of the cycles in the used IOTP data set is always 1 or 2, accounting for the small difference in the slopes.

Figure 14 shows a plot of the number of workflows against the number of messages. Most workflows had between 9 and 20 messages. The workflow with the least number of messages had 5 messages, while that with the highest number of messages had 23. A total of 104 workflows comprised 13 messages. From Table 1, we can see that a total of 10 506 message objects were created from the overall total of 23 message entities. This indicates a high-level of message reuse in the workflows.

5.3.3 Results

During the evaluation of the proposed system, we figured out that the highest measured response time for the UDDI part has been about 2,7 s, while the matchmaking engine component required 40 s. As a consequence, we consider the UDDI part as less critical and thus focused on the evaluation on the matchmaking engine component. The results presented next are related to this new component only.

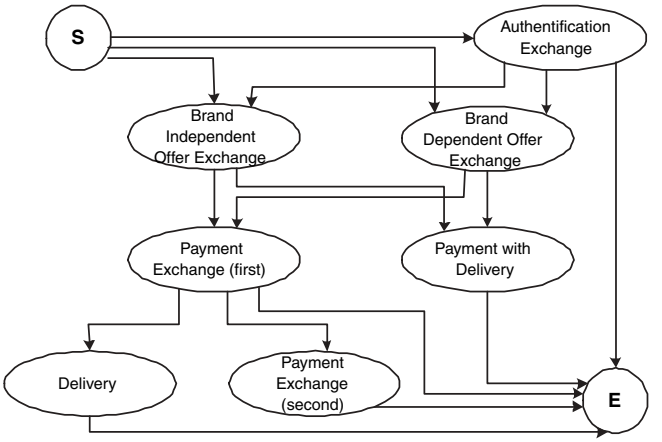

Fig. 11 IOTP message exchange structure

Table 2 summarizes the response times obtained during the experiments. The minimum response time for the data set was about 3 s, the maximum 40 s, and the mean response time was almost 18 s.

Figure 15 shows the distribution of response times on workflows. The figure shows that most workflows had a response time between 6 s and 32 s. The pattern of this distribution is very similar to the distribution of number of transitions on workflows (Fig. 12), suggesting a possible influence of the number of transitions in a workflow on the response time. We did another experiment to verify this.

Figure 16 shows a plot of response time against number of transitions. The figure shows a linear dependency of response time on the number of transitions in a workflow. This behavior can be explained as follows: intersection is computed in quadratic time$O\left(|\Delta_q| * |\Delta_D|\right)$ where $|\Delta_q|$ is the number of transitions in the query workflow and $|\Delta_D|$ is the number of transitions for the entire workflow database. But $|\Delta_D|$ is constant since only the query workflow is varied for a given experiment. The response time is therefore $O(|\Delta_q| * constant)$ which is $O(|\Delta_q|)$, thus explaining the linear behavior.

Another characteristic of Fig. 16 is the splitting into two of the line as the number of transitions increases. The reason for the splitting can be explained in terms of the structure of IOTP data (Fig. 11) and how the emptiness test is performed on the intersection workflow. From Fig. 11 we can determine that the **Authentification Exchange** block is preceded and succeeded by a start and end state respectively. As a consequence, the Authentication Exchange represents a short message sequence from start to final state of a workflow, which supports a fast emptiness checking in case of both workflows

Table 1 Data set description

#wf	#states	#mesg	#trans	#cycles	#state obj	#mesg obj	#trans obj
726	284	23	867	640	20178	10506	19366

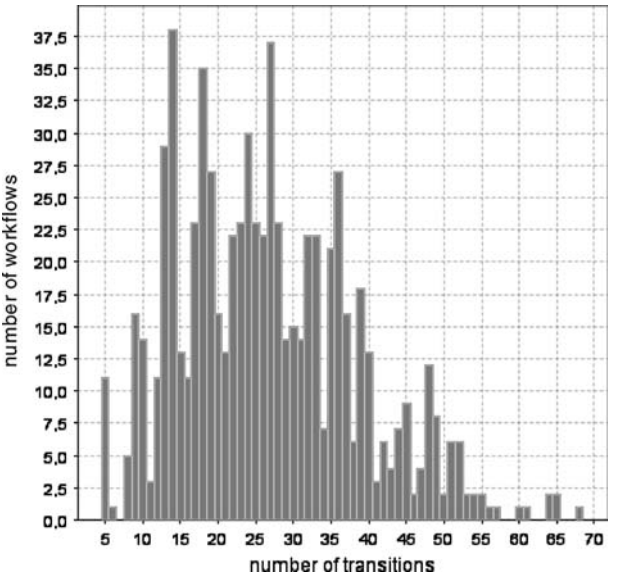

Fig. 12 Plot of number of automata versus number of transitions

containing this path. However, in case of more complex business patterns, the length of generated message sequences is significantly longer resulting in a higher complexity of the emptiness test. Thus, we found that workflows

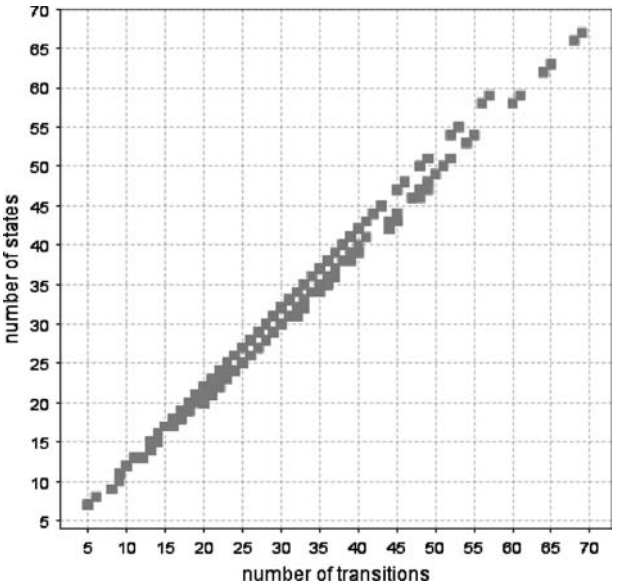

Fig. 13 Plot of number of states versus number of transitions

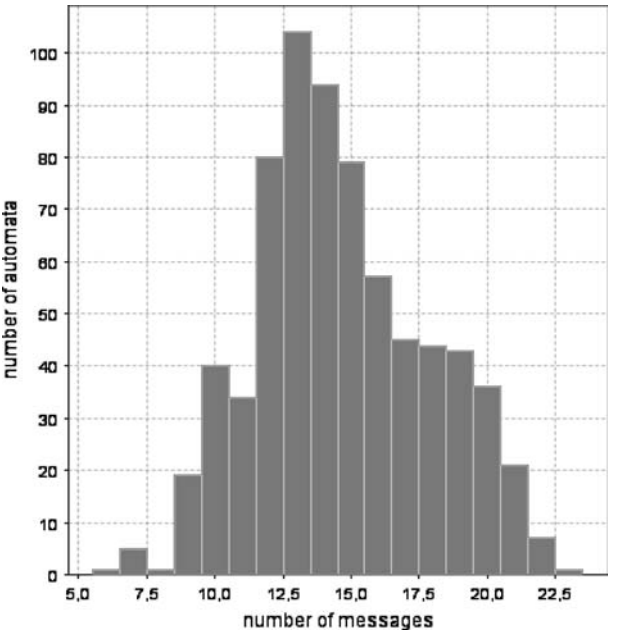

Fig. 14 Plot of number of automata versus number of messages

that do not contain the **Authentification Exchange** building block characterize the steeper line in the Fig. 16. This is because they require more computation time for emptiness testing i.e., to determine if the intersection workflow is empty or not. The line underneath (in the figure) characterizes those workflows containing the **Authentification Exchange** building block, which are taking less time to do emptiness testing.

This performance evaluation shows that the performance of the matchmaking engine is not sufficient for online service discovery and requires an indexing mechanism to speed-up the response time. We started this task already by examining existing indexing mechanisms.

6 Related work

Service discovery is a popular research topic addressed by different communities focusing on different aspects. All proposals have in common to extend the limited functionality as provided by UDDI.

Overhage and Thomas (Overhage and Thomas 2002) propose a comprehensive framework for web services discovery to supercede the current UDDI framework. The framework, named WS-specification, categorizes web services into white, yellow, blue and green pages. The terms, white, yellow, blue and green pages are adopted from UDDI, but extended and sometimes redefined in WS-specification. White pages for example contain information about general and technological (architecture specification, performance, security) infor-

Table 2 Response time

best case [ms]	average case [ms]	worst case [ms]
3,058	17,940	40,189

mation about services. Yellow pages contain classification information (like in UDDI), blue pages contain conceptual information (process-related, semantics) and green pages contain interface information. Among the observations made in (Overhage and Thomas 2002) is the deficiency of UDDI specification for not describing process-related information of services. Process-related information is thus provided for in the blue pages of WS-specification. They propose BPEL (used in this paper) as a possible formal definition language for business process descriptions. We are however not aware of any implementation of this specification so far and no clue about querying web services based on process information is provided.

Logic based approaches addressing service discovery are Web Service Request Language (WSRL) and Product Lifecycle Management PLM$_{flow}$. WSRL (Papazoglou et al. 2002, Aiello et al. 2002) addresses planning of an orchestration and composition of services to fulfill user requirements. While WSRL performs service discovery on behalf of temporal and linear constraints, PLM$_{flow}$ (Zeng et al. 2002) is based on rule inferencing using the specified business rules rather than a fixed workflow. Thus, PLM$_{flow}$ is characterized as a rule-based non-deterministic workflow engine aiming to establish cooperation based on local decidability of the trading partners of their involvement. These approaches are based on the fact that the local workflow model/rules are provided to the trading partners and do not

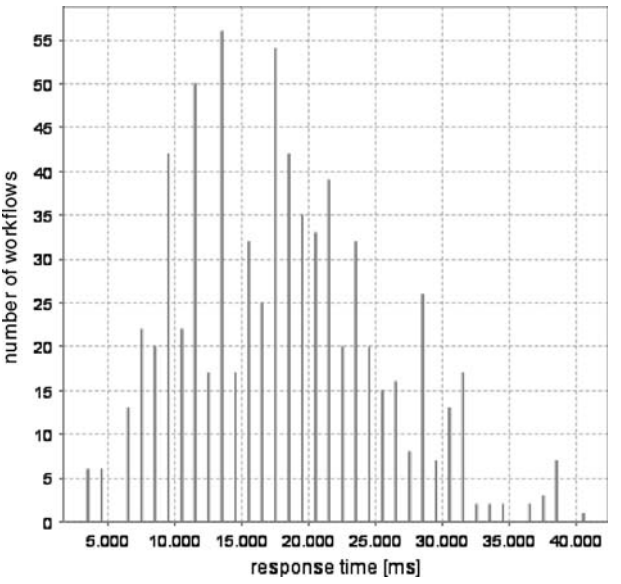

Fig. 15 Query response time

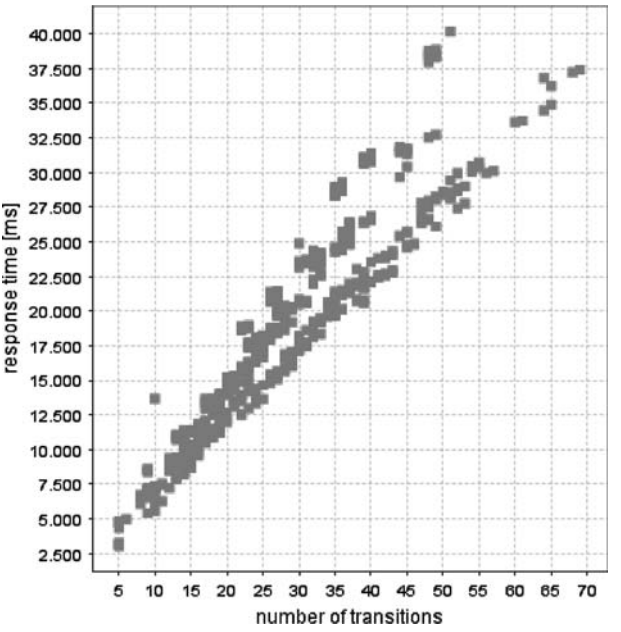

Fig. 16 Query response time versus number of transitions

consider the need of hiding business critical information as is the case with abstract business processes which are addressed in this paper.

In (Mecella et al. 2001), two services are regarded as compatible if every possible trace in one service has got a compatible one in the second one. This approach is similar to that used in this paper, but unfortunately, the description how to do the compatibility check of the traces is not given.

Matchmaking of services is also being addressed in the semantic web community (Berbers-Lee et al. 2001, McIlraith et al. 2001). The used approach is to add semantic annotations to service descriptions. Services are searched on the basis of what they can do, i.e., what features and capabilities they have. Standards such as DAML-S (DAML-S Coalition: Ankolekar et al. 2002) are used to describe such services. This is the approach taken in (Kawamura et al. 2003), where the UDDI search functionality is extended by adding semantic-based search functionality, to search for services, in addition to what UDDI provides. The concept of semantic annotation of web service descriptions for matchmaking is also used in (Paolucci et al. 2002; Sycara et al. 1999; Sycara et al. 1999; Trastour et al. 2002; Gonzalez-Castillo et al. 2001). The main draw back of semantic annotation is the necessity of a common ontology used for annotating and querying services. Unfortunately, no such ontology currently is in place.

AGFlow is an agent-based cross-enterprise workflow management system for discovering services. Service discovery is based on their location and predefined properties (Zeng et al. 2001). The approach does not describe how process-related information can be encoded into the agent, and used as a possible criteria for service discovery.

In (Georgakopoulos et al. 1999, Casati and Discenza 2001) business process modeling approaches are described to integrate processes within and outside the enterprise. The approach presented in (Casati and Discenza 2001) is based on extending workflow systems with the ability to register to events, and to receive events at specific points within the workflow. In (Georgakopoulos et al. 1999), a comprehensive framework is given how multiple virtual enterprises can integrate their processes to work together. The proposed approaches do not however describe how to check for compatible business processes in a dynamic environment as is done in this paper.

The European funded openXchange project aimed to develop a business process matchmaking tool that matches two profiles (collaboration protocol profiles) belonging to different business partners to automatically create an agreement (collaboration protocol agreement) (Folmer and Krukkert 2003) under the ebXML framework. Preliminary results are reported in (Folmer and Krukkert 2003), but the used approach is not clearly explained and no reference implementation is reported.

In (ShaiklAli et al. 2003), UDDIe is presented as an extension to UDDI, and its implementation is provided as open source at (uddie 2003). UDDIe extends UDDI in three main ways: (i) service leasing support, (ii) introducing properties for describing services and the ability to search on these properties and (iii) extending the UDDI *find* API with the ability to support numeric and logical AND/OR operators. This extension provides a more expressive way to describe and query services than originally provided by UDDI, but does not consider the business process aspect of the services as is done in this paper.

Opposed to the previous approach and the one presented in the paper an extension of the search capabilities via an open interface is proposed in (Colgrave et al. 2004). Although the paper discusses the architecture and the interface without proposing an extension on its own.

7 Summary and future work

The paper presented the concept, architecture and implementation behind the **IPSI-PF** process matchmaking engine, including an evaluation. The **IPSI-PF** process matchmaking engine readily fits to the existing web service framework, and it complements UDDI service discovery by extending it with the ability to search for compatible business partners through comparison of their business processes. Future releases of this engine will include an index to efficiently find compatible partners where a large amount of business process data is involved. Finding a suitable index for business process matching is complicated by the fact that, unlike flat key search which is easily indexed using standard techniques such as B^+-trees and hashing, this type of data does require comparison operations interpreting a structure rather than allowing to compare structures.

References

Aiello M, Papazoglou M, Yang J, Pistore M, Carman M, Serafini L, Traverso P (2002) A request language for web services based on planning and constraint satisfaction. In: Proc. of 3rd Int. Workshop, Technologies for E-Services (TES), LNCS 2444, Springer, pp 76–85

Andrews T, Curbera F, Dholakia H, Goland Y, Klein J, Leymann F, Liu K, Roller D, Smith D, Thatte S, Trickovic I, Weerawarana S (2003) Business process execution language for web services. version 1.1. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbiz2k2/ht%ml/BPEL1-1.asp

Berbers-Lee T, Hendler J, Lassila O (2001) The semantic web. Scientific America 284(5):34–43

Burdett D (2000) Internet open trading protocol - iotp - version 1.0. http://www.ietf.org/rfc/rfc2801.txt

Casati F, Discenza A (2001) Modeling and managing interactions among business processes. Journal of Systems Integration 10(2):145–168. Special Issue: Coordination as a Paradigm for Systems Integration

Chomicki J, Saake G (eds) (1998) Logics for Database and Information Systems. Kluwer

Colgrave J, Akkiraju R, Goodwin R (2004) External matching in UDDI. In: Proc. of IEEE Int. Conf. on Web Services (ICWS), IEEE, pp 226–233

DAML-S Coalition: Ankolekar A, Burstein M, Hobbs JR, Lassila O, McDermott D, Martin D, McIlraith SA, Narayanan S, Paolucci M, Payne T, Sycara K (2002) Daml-s: Web service description for the semantic web, http://citeseer.nj.nec.com/ankolekar02damls.html.

Folmer E, Krukkert D (2003) openxchange as ebxml implementation and validation; the first results

Georgakopoulos D, Schuster H, Cichocki A, Baker D (1999) Managing process and service fusion in virtual enterprises. Information Systems 24(6):429–456

Gonzalez-Castillo J, Trastour D, Bartolini C (2001) Description logic for matchmaking of services. Technical Report HPL-2001-265, Hewlett-Packard

Grefen P, Aberer K, Hoffner Y, Ludwig H (2000) Crossflow: Cross-organizational workflow management in dynamic virtual enterprises. International Journal of Computer Systems Science & Engineering 15(5):277–290, Sep. 2000, CRL Publishing.

Harel D (1987) Statecharts: A visual formalism for complex systems. Science of Computer Programming 8(3):231–274

Harel D, Naamad A (1996) The STATEMATE semantics of statecharts. ACM Transactions on Software Engineering and Methodology 5(4):293–333

Hopcroft JE, Motwani R, Ullman JD (2001) Introduction to Automata Theory, Languages, and Computation. Addison Wesley

IBM, Microsoft, HP, Oracle, Intel, and SAP (2002) Universal description, discovery and integration. http://www.uddi.org/

Jensen K (1992) Coloured Petri Nets. Springer, Berlin Heidelberg New York

juddi juddi. http://ws.apache.org/juddi

Kawamura T, De Blasio J, Hasegawa T, Paolucci M, Sycara K (2003) Preliminary report of public experiment of semantic service matchmaker with uddi business registry. In: Orlowska ME, Weerawarana S, Papazoglou MP, Yang J (eds) Service oriented computing - ICSOC 2003 LNCS 2910, Springer, pp 208–224

Klingemann J, Wäsch J, Aberer K (1999) Adaptive outsourcing in cross-organizational workflows. Proceedings of the the 11th Conference on Advanced Information Systems Engineering (Caise), Heidelberg, Germany

McIlraith S, Son T, Zeng H (2001) Semantic web services. IEEE Intelligent Systems (Special Issue on the Semantic Web)

Mecella M, Pernici B, Craca P (2001) Compatibility of e-services in a cooperative multiplatform environment. In: Casati F, Georgakopoulos D, Shan M, (eds) TES 2001 LNCS 2193, Springer, pp 44–57. Lecture notes in Computer Science 2193

Overhage S, Thomas P (2002) Ws-specification: Specifying web services using uddi improvements. In: Chaudhri AB, Jeckle M, Rahm E, Unland R (eds) Web, Web-Services, and Database Systems: NODe 2002, Web- and Database-Related Workshops, Erfurt, Germany, Vol 2593 / 2003, Springer-Verlag Heidelberg Lecture Notes in Computer Science. pp 100 – 119

Paolucci M, Kawmura T, Payne T, Sycara K (2002) Semantic matching of web services capabilities. In: Proceedings of the First International Semantic Web Conference on The Semantic Web, Springer-Verlag, pp 333–347

Papazoglou M, Aiello M, Pistore M, Yang J (2002) Planning for requests against web services. Bulletin of the Technical Commitee on Data Engineering 25(4):41–46

Peron A (1995) Statecharts, transition structures and transformations. In: Proceedings International Conference Colloquium on Trees in Algebra and Programming (CAAP-TAPSOFT'95, Springer, LNCS 915), pp 454–468
Peterson JL (1981) Petri Net Theory and the Modeling of Systems. Prentice-Hall
RosettaNet. Rosettanet homepage. http://www.rosettanet.org
ShaiklAli A, Rana OF, Al-Ali R, Walker DW (2003) Uddie: An extended registry for web services. In: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT-w03). IEEE Computer Society
Sycara KP, Klusch M, Widoff S, Lu J (1999) Dynamic service matchmaking among agents in open information environments. SIGMOD Record 28(1):47–53
Sycara K, Lu J, Klusch M, Widoff S (1999) Matchmaking among heterogeneous agents on the internet. In: Proceedings AAAI Spring Symposium on Intelligent Agents in Cyberspace, Stanford, USA
Trastour D, Bartolini C, Gonzalez-Castillo J (2002) Semantic web support for business-to-business e-commerce lifecycle. Technical Report HPL-2002-3R1, Hewlett-Packard
uddie (2003) Uddie homepage
vander Aalst W (2003) Dont go with the flow: Web services composition standards exposed. IEEE Intelligent Systems, pp 72–76
van der Aalst W, van Hee K (2002) Workflow Management - Models, Methods, and Systems. MIT Press
Wohed P, van der Aalst WMP, Dumas M, ter Hofstede A (2002) Pattern based analysis of bpel4ws
Wombacher A, Aberer K (2004) Requirements for workflow modeling in P2P-workflows derived from collaboration establishment. Proc. Int. Workshop on Business Process Integration and Management BPIM'04
Wombacher A, Mahleko B, Neuhold E (2004) IPSI-PF: A business process matchmaking engine. Proc. of IEEE Conf. on Electronic Commerce (CEC), pp 137–145
Wombacher A, Fankhauser P, Mahleko B, Neuhold E (2004) Matchmaking for business processes based on choreographies. International Journal of Web Services Research (JWSR), to be published
Wombacher A, Fankhauser P, Neuhold E (2004) Transforming BPEL into annotated deterministic finite state automata for service discovery. Proc. of IEEE Int. Conf. on Web Services (ICWS), pp 316–323
Zeng L, Benatallah B, Nguyen P, Ngu AHH (2001) Agflow: Agent-based cross-enterprise workflow management system. In: Apers PMG, Atzeni P, Ceri S, Paraboschi S, Ramamohanarao K, Snodgrass RT (eds) Proc of 27 international conference. on VLDB, pp 697–699
Zeng L, Flaxer D, Chang H, Jun-Jang Jeng (2002) PLM flow - dynamic business process composition and execution by rule inference. In: Proc. of 3rd Int. Workshop, Technologies for E-Services (TES), LNCS 2444, Springer pp 141–150