Matchmaking for Business Processes*

Andreas Wombacher

Peter Fankhauser

Bendick Mahleko

Erich Neuhold

Fraunhofer Gesellschaft, Integrated Publication and Information Systems Institute
64293 Darmstadt, Germany
firstname.lastname@ipsi.fhg.de

Abstract

Web services have a potential to enhance B2B e-commerce over the Internet by allowing companies and organizations to publish their business processes on service directories where potential trading partners can find them. This can give rise to new business paradigms based on ad-hoc trading relations as companies, particularly small to medium scale, can cheaply and flexibly enter into fruitful contracts, e.g., through subcontracting from big companies by simply publishing their business processes and the services they offer. More business process support by the web service infrastructure is however needed before such a paradigm change can materialize. A service for searching and matchmaking of business processes does not yet exist in the current infrastructure. We believe that such a service is needed and will enable companies and organizations to be able to establish ad-hoc business relations without relying on manually negotiated frame contracts like RosettaNet PIPs. This paper gives a formal semantics to business process matchmaking and an operational description for matchmaking.

1. Introduction

Web services have a potential to enhance B2B e-commerce over the Internet by allowing companies and organizations to publish their business processes on service directories where potential trading partners can discover them. This can give rise to new business paradigms based on ad-hoc trading relations as companies, particularly small to medium scale, can cheaply and flexibly enter into fruitful contracts, e.g., through subcontracting from big companies by simply publishing their business processes and the services they offer.

*This work has been carried out in the framework of the IST-project eBroker (Electronic Brokering Services for Open Trading Infrastructures, IST/1999-11060).

To date, business process descriptions have been mainly investigated as a basis for flexibly coupling processes. This includes approaches for dynamic process composition [16], modeling of cross-organizational workflows [13, 19], and support of external process views [20, 8]. But the information published by means of business processes can also significantly facilitate service discovery. Existing infrastructures to service discovery, fail to utilize this information. UDDI [2], for example, only provides for simple string comparisons. This is not sufficient for business processes, especially if there do not exist any pre-negotiated and uniquely named frame contracts published by standardization organizations as, for example, RosettaNet.

Other service based infrastructure face the same issue. In particular, within the ebXML framework business partners can express their business capabilities (including their business processes) using trading partner profiles (CPPs) without providing any means to match these.

This paper presents an approach to more precise service discovery using business process descriptions rather than individual messages. The next section illustrates limitations of existing approaches to service discovery by way of simple examples of compatible and incompatible business processes. Section 3 formalizes a more precise notion of business processes matching based on finite state automata. Section 4 discusses at which level the introduced techniques can be deployed in existing service description frameworks. Finally, Section 5 concludes and outlines future work.

2. Example

Figure 1 depicts two business processes involving two trading parties: a vendor v and a customer c . Nodes represent the states of a business process; the end states are identified by a double circle. Edges represent state transitions, which are labeled with messages denoted as *from#to#message_name*, where *from* represents the message sender, *to* represents the message recipient, and *message_name* is the name of the message.

Figure 1(a) shows the vendor business process, where the vendor expects to receive a purchase order ($c\#v\#PO$) message, followed by a credit card payment ($c\#v\#ccPay$) and finally sends back a delivery ($v\#c\#Delivery$) message to the respective customer. The customer process depicted in Figure 1(b) also initiates the process with a purchase order request ($c\#v\#PO$). But then it insists on delivery ($v\#c\#Delivery$) before payment by credit card ($c\#v\#ccPay$) or by invoice ($c\#v\#invoicePay$).

Figure 1. (a) Vendor Message Sequence. (b) Customer Message Sequence.

At the level of individual messages these two business processes match. However, because they require a different order of payment and delivery, they are incompatible, that is, they can not successfully interact. In order to avoid matching incompatible business processes we thus need to take into account message sequences rather than individual messages.

Figure 2(a) shows a purchase order business process provided by a vendor. The process starts with a purchase order ($c\#v\#PO$) message, followed by a delivery ($v\#c\#Delivery$) message, and either a credit card payment ($c\#v\#ccPay$) or an invoice payment ($c\#v\#invoicePay$) message. In case the ordered product is not on stock, the vendor may reject a purchase order by sending a no stock available ($v\#c\#noStock$) message. The vendor process involves two kinds of choices. On the one hand, it insists on the availability of both, the $v\#c\#noStock$ and the $v\#c\#Delivery$ message (conjunctive choice, depicted by an arc connecting the two choices). On the other hand, it supports the two payment options as genuine alternatives (disjunctive choice).

Figure 2(b)¹ depicts a customer business process. While this process matches the vendor process with respect to the delivery payment order, it can not handle the required $v\#c\#noStock$ message. Therefore, the two business processes can not reach the end state, if an ordered product is not on stock.

¹This process is equivalent to the one depicted in Figure 1(b) and described above.

Figure 2. (a) Vendor Message Sequence insisting on $v\#c\#noStock$ and $v\#c\#Delivery$ Messages. (b) Customer Message Sequence. (c) Customer Message Sequence with optional $v\#c\#noStock$ Message.

Conversely, the business process in Figure 2(c) supports $v\#c\#noStock$ and $v\#c\#Delivery$ messages, whereas it supports only one payment option. This process now satisfies all conjunctive and disjunctive choices of the vendor process. Thus, the vendor process and the customer process are compatible.

In summary, the two examples in Figure 1 and 2 illustrate that (1) message sequence and (2) conjunctive choices need to be taken into account to determine the compatibility of business processes.

3 Approach

Finite state automata [12] constitute a suitable starting point to model business processes for the purpose of matchmaking. They can represent (possibly infinite) sets of message sequences without considering branching conditions and parallel execution capabilities as provided by more expressive approaches such as Petri Nets [18].

Formally, finite state automata can be represented as follows:

Definition 1 (Finite State Automata (FSA))

A finite state automaton A is represented as a tuple $A = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, $\Sigma \subseteq P \times P \times M$ is a finite set of messages in M sent by a sender in P to a receiver in P , $\delta \subseteq Q \times \Sigma \times Q$ is a set of transitions, q_0 a start state with $q_0 \in Q$, and $F \subseteq Q$ a set of final states.

The only difference to the standard definition of FSAs is that the alphabet Σ consists of triples rather than of atomic tokens. However, for the purpose of matchmaking business processes, these triples can be treated like atomic tokens: Two message triples are equal, if their sender, their receiver, and the message (with its parameters) are equal.

An FSA A generates a language $L(A)$ which enumerates the (possibly infinite) set of all message sequences supported by a business process. The languages for the example business processes depicted in Figure 2 are:

the vendor language of Figure 2(a):

$$L(\text{vendor}) := \{c\#v\#PO \rightarrow v\#c\#noStock, \\ c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#ccPay, \\ c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#invoicePay\}$$

the simple customer language of Figure 2(b):

$$L(\text{customer}) := \{ \\ c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#ccPay, \\ c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#invoicePay\}$$

the customer language with $v\#c\#noStock$ message of Figure 2(c):

$$L(\text{customer}_{noStock}) := \{c\#v\#PO \rightarrow v\#c\#noStock, \\ c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#ccPay\}$$

Two FSAs match, if their languages have a non-empty intersection. The intersection of two FSAs is again an FSA, which can be determined with the usual cross product construction:

Definition 2 (Intersection of two FSAs)

The intersection $A_1 \cap A_2$ of two automata

$A_1 = (Q_1, \Sigma_1, \delta_1, q_{10}, F_1)$, and $A_2 = (Q_2, \Sigma_2, \delta_2, q_{20}, F_2)$ is $A = (Q, \Sigma, \delta, q_0, F)$, with: $Q = Q_1 \times Q_2$, $\Sigma = \Sigma_1 \cap \Sigma_2$, $\delta = \{((q_{11}, q_{21}), \alpha, (q_{12}, q_{22})) \mid (q_{11}, \alpha, q_{12}) \in \delta_1, (q_{21}, \alpha, q_{22}) \in \delta_2\}$, $q_0 = (q_{10}, q_{20})$, and $F = F_1 \times F_2$

If the resulting automaton does not contain at least one path (possibly of zero length) between the start state and an end state, its language is the empty set $\emptyset$. In this case, the business processes modeled by the FSAs are incompatible, because there does not exist any common message sequence. For example, the intersection of the two business processes in Figure 1, does not contain any path between its start and its end state.

The intersection of the vendor process in Figure 2(a) and the customer process in Figure 2(b) is equivalent to the customer process. However, even though this intersection is not empty, it does not contain the required transition $v\#c\#noStock$ of the vendor process, and thus is not a suitable basis for matching the two processes.

The reason for this false match is that standard FSAs can not distinguish between disjunctive and conjunctive choices. Usually, a choice in a standard FSA is regarded as a disjunction. Consequently, the language of an FSA denotes a disjunction of all possible message sequences. However, the intended meaning of the vendor process

requires both, disjunction and conjunction. In conjunctive normal form, this meaning can be expressed by the following logical expression on message sequences:

$$(c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#ccPay \vee \\ c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#invoicePay) \\ \wedge c\#v\#PO \rightarrow v\#c\#noStock$$

This logical expression can be represented by the conjunction of two FSAs depicted in Figure 3.

Figure 3. Vendor Message Sequence from Figure 2(a) represented as conjunction of disjunctive FSAs.

Business processes represented by two conjunctions of individual FSAs can be matched as follows. Intuitively, two such business processes match, if for each FSA in one business process there exists at least one FSA in the other business process with a non-empty intersection. As can be easily verified, a sufficient condition for this is that the intersection of each individual FSA in one business process with the union of all FSAs in the other business process is non-empty. The following definition formalizes this:

Definition 3 (Intersection of two conjunctive FSAs)

With $C_1 = A_{11} \wedge \dots \wedge A_{1m}$, $C_2 = A_{21} \wedge \dots \wedge A_{2n}$ two conjunctive FSAs, and

$$C'_1 = A_{11} \cup \dots \cup A_{1m}, C'_2 = A_{21} \cup \dots \cup A_{2n}$$

the FSAs resulting from the union of the individual FSAs in C_1 , and C_2 , the intersection $C_1 \cap C_2$ of two conjunctive FSAs is: $\bigwedge_{1 \leq i \leq m} (A_{1i} \cap C'_2) \wedge \bigwedge_{1 \leq j \leq n} (A_{2j} \cap C'_1)$

This definition can be easily implemented with an algorithm that is linear in the number of the individual automata A_{1i} and A_{2j} . Note however, that computing the intersection of two (disjunctive) FSAs is quadratic in the number of states. One possible approach to a more scalable algorithm for this intersection may be the deployment of appropriate index structures for regular expressions (e.g. [4]).

If the intersection of two conjunctive FSAs contains an empty automaton, two business processes do not match, because at least one required message sequence of one process is not supported by the other process. Otherwise, they match.

For the compatible business processes depicted in Figures 3 and 2(c) their intersection generates the following

logical expression on message sequences (Note that this can be further simplified by discarding the redundant third term in the conjunction.):

$$\begin{aligned} & c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#ccPay \\ \wedge & c\#v\#PO \rightarrow v\#c\#noStock \\ \wedge & (c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#ccPay \\ & \vee c\#v\#PO \rightarrow v\#c\#noStock) \end{aligned}$$

Because this does not contain the empty choice, the two business processes match. In contrast, the intersection of the incompatible business processes depicted in Figures 3 and 2(b) contains an empty choice:

$$\begin{aligned} & (c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#ccPay \\ \vee & c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#invoicePay) \\ \wedge & \emptyset \\ \wedge & (c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#ccPay \\ \vee & c\#v\#PO \rightarrow v\#c\#Delivery \rightarrow c\#v\#invoicePay) \end{aligned}$$

Therefore, these two business processes do not match.

As can be easily verified, the (normalized) conjunctive FSAs as introduced above are closed under intersection. However, FSAs, which contain a conjunctive choice at some inner node rather than at the top level, can not necessarily be transformed into conjunctive normal form. For example, the simple FSA in Figure 4 can not be unfolded into conjunctive normal form, because the underlying required set of message sequences is an infinite conjunction of purchase order sequences.

Figure 4. Example for an infinite conjunction.

Whether and how the intersection of unnormalized conjunctive FSAs can be determined without normalization is an open issue. Also, the support of explicitly excluded message sequences is an open issue. However, because FSAs are closed under complement, at least support of top level exclusions appears to be straight forward.

4 Application to Service Definition Languages

This section illustrates how the presented approach to matchmaking for business processes can be applied to exist-

ing service definition languages. Services are typically described at three major levels: Messages, abstract processes, and execution processes.

(1) Message descriptions such as WSDL and EDIFACT describe the syntax and structure of messages. The Web Service Definition Language (WSDL) [5] uses XML Schema to describe the input and output of operations supported by a service. These operations can be associated to roles, which correspond to sender and receiver of message descriptions used in this paper. Thus, WSDL descriptions may be used as one concrete form to encode and match individual message descriptions. Alternatively, web based EDI like EDIFACT [7] can be used for this purpose. Such syntactic message descriptions can be matched by component wised comparison. A more ambitious approach is addressed by DAML-S profiles [1]: these profiles describe messages by means of ontological concepts such that semantic reasoning can be used to more flexibly match messages.

(2) Abstract processes describe the sequences in which messages may be exchanged. There are several proposals for specifying abstract processes, including WSCL, cpXML, the abstract part of BPEL, and ebXML BPSS. WSCL [3] uses finite state automata to model abstract business processes. As depicted in Figure 5, WSCL descriptions can be easily extended to accommodate top level conjunctions of finite state automata, by introducing a top level element *automataset*, that consists of one or more *conversation* sub-selements.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xsd:id="WSCL" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsd:targetNamespace="http://www.w3.org/2002/02/wscl10">

  <xsd:element name="Automataset">
    <xsd:complexType>
      <xsd:element name="Conversation" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Conversation">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="ConversationInteractions"/>
        <xsd:element ref="ConversationTransitions"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

Figure 5. WSCL Extension

Conversation Policy XML (cpXML) [11, 9, 10] extends finite state automata with hierarchical states, which encapsulate again a finite state automaton. Conjunctive hierarchical states also constitute an option to model conjunctive finite state automata. Thus the techniques presented in this

paper can be readily applied to match business processes. BPEL [6] (synthesized from XLANG [17] and WSFL [15]) and ebXML BPSS [14] also allow for the specification of parallel recursive business processes, which can not be described by finite state automata. Therefore, for the purpose of matchmaking, parallelism needs to be abstracted away, which may introduce false matches.

(3) Execution process description extend abstract process description with information necessary to execute a business process. This includes the binding of the abstract process to internal processes, constraints on message parameters and on time. This additional information is usually confidential and therefore not advertised publicly. Nevertheless, especially constraints may be deployed for improving the precision of matchmaking.

5 Summary and Future Work

This paper has introduced an approach to match business process descriptions described by means of conjunctive finite state automata. By explicating message sequence and required messages such descriptions allow for more precise matches than current approaches matching only individual messages. Thereby, existing service description languages based on finite state automata can be readily deployed as a basis for more precise service discovery.

Currently, the approach is being implemented for a business process repository supporting matchmaking and discovery based on business process descriptions. In addition, future work will be devoted to the following issues: (1) How can bilateral matching for business processes be extended to multilateral processes? (2) How can the syntactic match of individual messages be extended to take into account more semantics? (3) How can the rather expensive intersection of finite state automata scale to large service description repositories by means of appropriate index structures? (4) How can the intersection of automata be used as a basis for effectively coupling matching processes?

References

- [1] D.-S. C. A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. McDermott, D. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. Daml-s: Web service description for the semantic web, 2002. <http://citeseer.nj.nec.com/ankolekar02damls.html>.
- [2] I. Ariba, I. Corporation, and M. Corporation. Universal description, discovery and integration, September 2000. <http://www.uddi.org/>.
- [3] A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K. Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, S. Sharma, and S. Williams. Web services conversation language (WSCL) 1.0 W3C note, March 2002. <http://www.w3.org/TR/wscl10/>.
- [4] C. Chan, M. Garofalakis, and R. Rastogi. Re-tree: An efficient index structure for regular expressions. In *Proceedings of the 28th VLDB Conference*, 2002.
- [5] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1 W3C note, March 2001. <http://www.w3.org/TR/wsdl>.
- [6] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana. Business process execution language for web services, version 1.0, July 2002. <http://www.ibm.com/developerworks/library/ws-bpel/>.
- [7] EDIFACT. United nations directories for electronic data interchange for administration, commerce and transport. <http://www.unece.org/trade/untid/welcome.htm>.
- [8] P. Grefen, H. Ludwig, and S. Angelov. A framework for e-services: A three-level approach towards process and data management. Technical Report RC22378 (W0203-061), IBM Research Division, Thomas J. Watson Research Center, March 2002.
- [9] J. E. Hanson, P. Nandi, and S. Kumaran. Conversation support for business process integration. In *Proceedings 6th IEEE International Enterprise Distributed Object Computing Conference (EDOC-2002)*, pages 65–74. IEEE Press, 2002.
- [10] J. E. Hanson, P. Nandi, and D. W. Levine. Conversation-enabled web services for agents and e-business. In *Proceedings of the International Conference on Internet Computing (IC-02)*, pages 791–796. CSREA Press, 2002.
- [11] J. E. Hanson. cpxml: Conversation policy xml version 1.0. <http://www.research.ibm.com/convsupport/papers/index.html>.
- [12] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2001.
- [13] E. Kafeza, D. K. W. Chiu, and I. Kafeza. View-based contracts in an e-service cross-organizational workflow environment. In F. Casati, D. Georgakopoulos, and M. Shan, editors, *TES 2001 LNCS 2193*, pages 74–88. Springer, 2001.
- [14] P. Levine, J. Clark, C. Casanave, K. Kanaskie, B. Harvey, J. Clark, N. Smith, J. Yunker, and K. Riemer. Business process specification schema. www.ebxml.org/specs/ebBPSS.pdf.
- [15] F. Leymann. Web services flow language (WSFL 1.0), May 2001. <http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>.
- [16] M. Mecella, B. Pernici, and P. Craca. Compatibility of e-services in a cooperative multi-platform environment. In F. Casati, D. Georgakopoulos, and M. Shan, editors, *TES 2001 LNCS 2193*, pages 44–57. Springer, 2001. Lecture notes in Computer Science 2193.
- [17] S. Thatte. XLANG web services for business process design.
- [18] W. van der Aalst Kees van Hee. *Workflow Management - Models, Methods, and Systems*. MIT Press, 2002.
- [19] W. v.d. Aalst. Interorganizational workflows: An approach based on message sequence charts and petri nets. *Systems Analysis - Modelling - Simulation*, 34(3):335–367, 1999.
- [20] A. Wombacher and B. Mahleko. Finding trading partners to establish ad-hoc business processes. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE LNCS 2519*, pages 339–355. Springer, 2002. ISBN: 3-540-00106-9.