Web Service Composition Based on XML Nets

Haiying Che\textsuperscript{1, 2}, Yu Li\textsuperscript{2}, Andreas Oberweis\textsuperscript{2}, Wolffried Stucky\textsuperscript{2}
\textsuperscript{1}School of Software, Beijing Institute of Technology
\textsuperscript{2}Institute AIFB, University of Karlsruhe (TH), Germany
chrissieche@hotmail.com, \{yu.li|oberweis|stucky\}@aifb.uni-karlsruhe.de

Abstract

Service Science, Management and Engineering (SSME) is a growing interdisciplinary research area for studying, designing, implementing and improving service systems. As one of the instantiations of service systems, Web services (WSs) offer a new paradigm for distributed computing and system collaboration. Recently, Web service composition (WSC) has increasingly gained attention from SSME community since it aims to provide value-added WSs by combing existing WSs.

In this paper we firstly survey WSC methods, among which Petri net-based methods are listed. As a variant of high-level Petri nets, XML nets have formal semantics, graphical nature, and the strength in exchanging XML-based structured data. They are very suitable for WSC because messages can be modeled and manipulated as place tokens for message passing, and the labels in arcs can be used to model constraints for WS discovery and selection. Using XML nets for WSC can thus improve WSC models and increase their dynamics.

1. Introduction

On the background that global markets are increasingly service-based economies, Service Science, Management and Engineering (SSME) is growing as an interdisciplinary research area for studying, designing, implementing and improving service systems. To impel service innovation that is a central concern of SSME, efficient and effective collaboration among service systems is needed. Service-oriented computing (SOC) takes on the challenge of heterogeneous system collaboration. As one of the instantiations of service systems and a mature and elaborate implementation of the SOC concept, Web services (WSs) are considered as self-contained, self-describing and modular applications that can be published, located and invoked across the web.

Due to constant changes in today’s service-led economies, flexible value-adding business processes that dynamically combine and arrange service systems as process activities are strongly needed to increase productivity and efficiency. From this point of view, Web service composition (WSC) is an important research field of SSME since it aims to provide value-added WSs by combing existing WSs. Efficiently and effectively selecting, integrating and arranging heterogeneous WSs dynamically to create new value-added inter-organizational WSs is an issue of much current attention. Many researchers and practitioners have contributed a lot in this field.

As one of the proposed WSC techniques, Petri nets are a well-founded process modeling language with formal semantics, graphical nature, high expressiveness, analyzability and vendor-independence [32]. Petri net models can be formally verified and simulatively validated. XML nets [14], a variant of high-level Petri nets, have additional advantages in the description of process objects and inter-organizational exchange of standardized structured data (e.g., XML documents). While some Petri net variants (e.g., CPN [41], Object nets [17]) have been proposed for WSC, the ability and strengths of XML nets in this field were still not adequately exploited.

In this paper we elaborate that XML nets are very suitable for WSC because messages can be structured and manipulated as place tokens for message passing, which is pivotal in WSC, and arc inscriptions, the so-called Filter Schemas that are used in XML nets to filter and manipulate XML-based tokens, can be used to model criterions or constraints for WS discovery and selection. Using XML nets for WSC can thus improve WSC models and increase their dynamics.

The rest of the paper is organized as follows: The next section surveys existing WSC methods with comparison. Section 3 focuses on analyzing various Petri net-based WSC methods. In Section 4, an XML net-based WSC method is proposed. Section 5 elaborates how XML nets are used to facilitate and improve dynamic WSC. Section 6 concludes this paper.

2. Survey of WS composition methods

To compose Web services, researchers have developed diverse methods from different aspects using various theories and tools.
In [28] these methods are divided into categories: One is WSC using workflow techniques and the other using AI planning. The latter generally includes state sets (all possible states, initial states, mediate states and final states or goals), action sets and translation relation or rules that define prediction and effects for the execution of each action.

In [2] the authors distinguish between syntactic (XML-based) and semantic (ontology-based) WSC. In syntactic WSC, two main approaches are mentioned: The first one is WS orchestration, which combines available WSs by adding a central coordinator (the orchestrator) that is responsible for invoking and combining sub-activities. The second approach is WS choreography, which instead does not assume a central coordinator but rather defines complex tasks via the definition of conversation that should be undertaken by each participant. The overall activity is then achieved as composition of peer-to-peer interactions among the collaborating WSs.

In [16], the classification is more concrete. The authors describe the techniques as template-based, interface-based, logic-based, ontology-driven, quality-driven, automata-based or Petri net-based techniques.

In [22], the methods such as BPEL, Semantic Web (OWL-S), Web components, Algebraic Process Composition, Petri nets, Model Checking, and Finite-State machine are listed.

Table 1. Classification of WSC methods

<table>
<thead>
<tr>
<th>WSC using WF techniques</th>
<th>Orchestration</th>
<th>Methods or Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSC using AI planning</td>
<td>Choreography</td>
<td>BPEL [24]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Petri nets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Automata-based (e.g. Finite State Machine, SPIN)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Web components [38]</td>
</tr>
<tr>
<td></td>
<td>Conversation Specification [41]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WSCI [35], WS-CDL [34]</td>
<td></td>
</tr>
<tr>
<td>Logic-based</td>
<td>WSC using AI planning</td>
<td>OWL-S [33]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WSMO [37], WSML [36]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Algebraic Process Composition, Pi-calculus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Situation Calculus [20][21][23]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PDDL [7]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rule-based planning - CSSL (Composition Service Specification), SWORD [25]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HTN (Hierarchical Task Network) planning, e.g. SHOP2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Theorem Proving, e.g. Linear Logic (re. [29][30])</td>
</tr>
</tbody>
</table>

The classifications are made according to different criterions. Certainly, some methods overlap each other in some way. In order to give a clear view on most of the methods and standards, we list them in a classification table (see Table 1). In this paper we distinguish static, dynamic and automatic WSC. In static WSC, processes and activities for WS invocation are predefined and unalterable at run-time. In dynamic WSC, activities can be abstract and used to call different WSs at run-time according to process context. Automatic WSC has reasoning ability and mostly uses AI planning aiming at automatic generation of process models without human intervention. In dynamic WSC, we distinguish further between semi-dynamic and fully dynamic composition. The semi-dynamic one means that the structure of a composition process is determined at development-time in form of an abstract process including a set of abstract activities. Each abstract activity contains a query clause used to find, select and integrate appropriate real WS to fulfill the task. In this case, only the selection and binding of WS is done dynamically. In fully dynamic WSC, not only Web services are bound to composition process dynamically, but the process structure can also be modified at runtime. In this paper we focus on semi-dynamic WSC.

WSC should satisfy several fundamental requirements: Connectivity, support for non-functional quality of service metrics, correctness, scalability, and in the desiderative situation, automatization. A table (see Table 2) is worked out with respect to these important properties. In order to make the table compact, following abbreviations are made: Con (connectivity), Non (non-functional metrics), Cor (correctness), Sca (scalability), Auto (automatization), Ave (average), H (high), L (low), Semi (semi-automatic), and Full (fully automatic).

Table 2. Comparing WSC requirements

<table>
<thead>
<tr>
<th>WSC</th>
<th>Con</th>
<th>Non</th>
<th>Cor</th>
<th>Sca</th>
<th>Auto</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPEL</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>Ave</td>
</tr>
<tr>
<td>WSCI, WS-CDL</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>H</td>
</tr>
<tr>
<td>OWL-S</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Ave Full</td>
</tr>
<tr>
<td>WSMO &amp; WSML</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Ave Full</td>
</tr>
<tr>
<td>Petri nets</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
<td>Semi</td>
<td></td>
</tr>
<tr>
<td>Automata-based</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
<td>Semi</td>
<td></td>
</tr>
<tr>
<td>Web components</td>
<td>✓</td>
<td>✓</td>
<td>H</td>
<td>Semi</td>
<td></td>
</tr>
<tr>
<td>Conversation Specification</td>
<td>✓</td>
<td></td>
<td></td>
<td>Ave</td>
<td></td>
</tr>
<tr>
<td>Algebraic Process Composition</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
<td>Full</td>
<td></td>
</tr>
<tr>
<td>Situation Calculus</td>
<td>✓</td>
<td>✓</td>
<td>Ave Full</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDDL</td>
<td>✓</td>
<td></td>
<td>L</td>
<td>Full</td>
<td></td>
</tr>
<tr>
<td>Rule-based planning</td>
<td>✓</td>
<td></td>
<td>L</td>
<td>Full</td>
<td></td>
</tr>
<tr>
<td>HTN planning</td>
<td>✓</td>
<td></td>
<td>Ave Full</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Theorem Proving</td>
<td>✓</td>
<td>✓</td>
<td>Ave Full</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As shown in table 2, only AI-based WSC methods support automatic WSC. Among them, Algebraic Process Composition, Situation Calculus and Theorem Proving satisfy the correctness requirement. OWL-S and WSMO & WSML support in some degree the requirement of non-functional metrics. Nevertheless, the main disadvantage of AI-based WSC methods is the...
gap between their conceptual modeling ability and their insufficient suitability for simulation and execution because they focus on generating implementation skeletons that need to be refined into formal specification for execution [28]. Therefore we focus on WSC methods using workflow techniques. As emerging mechanisms with relatively short history, choreography methods do not yet satisfy the correctness requirement. In contrast, orchestration methods have already gained rich experience due to the development of process-oriented information systems. For example, a lot of research efforts have been devoted to workflow patterns that are helpful in the analysis, design and simulation of WSC models.

Many enterprises adopt BPEL for integrating and assembling Web services. However, BPEL lacks formal semantics, and BPEL models can therefore not be verified directly.

An automaton is a mathematical model for a finite state machine (FSM). Automata-based methods, as shown in table 2, have in many aspects similar abilities as Petri nets, and both of them have graphical representation. However, Petri nets are in our opinion more suitable for modeling workflows as they provide inherent ways for describing states and behaviors of multiple activities and their interactions, while automata are mainly used to describe states and behavior of a single activity. In addition, using variants of high-level Petri nets, process metrics like time, cost and resource consumption that are used to validate, monitor and improve process models can be easily modeled.

Web components methods encapsulate composite-logic information inside a class definition that represents a Web component, which can be specified in two isomorphic forms: a class definition and an XML specification described in Service Composition Specification Language [38]. Therefore, Web components methods are specification methods and lack the ability for formal verification.

In the next section we will examine further Petri net-based WSC methods.

Many papers use more than one method to solve the WSC problem in order to combine advantages of individual methods. For example, combining BPEL and formal verification methods, the authors in [19] transform WSC models in BPEL into Petri net models for formal verification. In [5], a framework is presented that translates BPEL models first into intermediate representations, and then into verification models using a verification language called Promela, input language of the model checker SPIN which is a finite-state verification tool. In [4] similar work has been done: BPEL models and Message Sequence Charts models for message passing are both transformed into FSP (Finite State Process notation) to concisely describe and reason about concurrent programs.

3. Petri net-based WSC approaches

Petri nets are a well-founded process modeling language with formal semantics and graphical nature. A Petri net is a directed, connected and bipartite graph in which nodes represent places and transitions, and tokens occupy places. The main attraction of Petri nets is the natural way of identifying basic aspects of concurrent systems, both mathematically and conceptually. Using Petri nets in WS modeling, each Web service has an associated Petri net that describes its behavior and contains exactly one input place and one output place. At any given time, a service can be in one of the following states: Not-Instantiated, Ready, Running, Suspended or Completed [8]. Petri net variants are also used to model WSC, or to model in other methods (e.g., MSCs, BPEL) and then transform them to Petri nets for verification. Here we address four WSC techniques based on Petri nets:

1. Transforming MSCs into Petri nets: In [13] [31], the authors transform communication protocol model in MSCs (Message Sequence Charts), which is regarded as a popular and intuitive specification language for describing messaging scenarios, into Petri net model for analysis.

2. Transforming BPEL into Petri nets: In [27] all BPEL control flow constructs are mapped into labeled Petri nets (including control flows for exception handling and compensation). The output is verified by BPEL2PNML and WofBPEL (including reachability analysis). In [9] a BPEL2PN parser is used to automatically transform BPEL models into Petri net models.

3. Using Nets within Nets: In [1], Nets within Nets is adopted in modeling coordination protocols and workflows. The authors first propose a Petri net model for WS peer, which is able to run any workflow and to dynamically interpret protocols required for coordination. The execution of these protocols allows the peer to integrate functionalities offered by external peers. The Linda communication model is used to support the integration among peers. In [6], Nets within Nets is also used to model inter-organizational software processes based on mobile agent systems. The authors translate Nets within Nets into flat nets for analysis. To enhance the flexibility of execution, a weak synchronous concept is introduced. In [17], the authors translate inter-organizational communication MSCs models into Nets within Nets models under consideration of protocol consistency. In [3] WSC is similarly modeled as Object Petri nets that are a variant of Nets within Nets.

---

1 http://www.workflowpatterns.com/
4. Using Colored Petri nets: In [41] CPN model for WS conversation protocols is proposed. The authors model conversation specifications with CPN and verify the WSC process and its conformance to component services conversation protocols. In [39] [40] hierarchical CPN are used to model the interface of each WS and the choreography among them. In [10] a CPN-based model for WSC is built, and an algorithm for translating BPEL models to CPN models is introduced. In [26] the authors decompose messages into Meta message, which is the smallest unit of message. They define a message-oriented activity-based CPN model according to relationship of messages and WS behaviors to describe WSs, and use an automatic WSC algorithm to automatically create a specification for composition processes.

Compared with the above Petri net-based WSC methods, XML nets have additional strengths in describing process objects and exchanging XML-based structured data. In the next section we will explain in detail XML nets and their suitability for WSC.

4. WS composition based on XML nets

4.1 XML nets

XML nets [14] represent a variant of high-level Petri nets, in which places are interpreted as containers of XML documents that must conform to XML Schemas typifying the places. Transitions can be inscribed by predicate logic expressions, whose free variables must be contained in the inscriptions of adjacent edges, the so-called Filter Schemas used to read or manipulate XML data objects.

Based on [15], we define XML net as a tuple \( \XiN = \langle P, T, A, \Psi, IP, IA, IT \rangle \), where

(i) \( \langle P, T, A \rangle \) is a Petri net with a finite set \( P \) of places, a finite set \( T \) of transitions, and a finite set \( A \) of arcs connecting places and transitions.

(ii) \( \Psi = \langle D, FT, PR \rangle \) is a structure consisting of an non-empty and finite individual set \( D \) of \( \sum \), a set of formula and term functions \( FT \) defined on \( D \), and a set of predicates \( PR \) defined on \( D \).

(iii) \( IP \) is the function that assigns an XML Schema to each place.

(iv) \( IA \) is the function that assigns a valid Filter Schema to each arc. The Filter Schema must conform to XML Schema of the adjacent place. Filter Schemas of incoming arcs of a transition may have selective or manipulative filters to read or (partially) delete XML documents. Filter Schemas of outgoing arcs of a transition must have manipulative filters to modify or create XML documents.

(v) \( IT \) is the function that assigns a predicate logical expression as inscription to each transition. The inscription is built on a given structure \( \Psi \) and a set of variables which are contained in the Filter Schemas of all adjacent arcs. The inscription must be evaluated to true in order to enable transition occurrences.

A transition in an XML net is enabled (i.e. it may occur) for a given marking and for a given instantiation of the variables in the transition’s environment if the following three conditions hold: Each place in the pre-set of the transition contains at least one valid XML document that conforms to the Filter Schema inscribing the edge from the place to the transition. Additionally, places in the post-set of the transition have to be checked: If the transition modifies existing XML documents in post-set places, the existence and validity of these documents must be ensured. If the transition creates new XML documents in post-set places, then the places must not contain the documents to be created. Finally, the transition inscription must be evaluated to true for the given instantiation of the variables, in order to enable the transition. If an enabled transition occurs, XML documents of pre-set places are...
read or (partially) deleted for the given instantiation of variables, and in the marking of post-set places, new XML documents are inserted or existing documents are modified.

Figure 1 shows a simplified XML net for hotel reservation with XML Schema and Filter Schema diagrams assigned to places and edges, respectively. The Filter Schema FS1 contains element filters (e.g., “hotel”, “price”, “roomAvailable”) and attribute filters (e.g., “id”, “name”) that are regarded as variables. The instantiation of the attribute filter “name” and the element filter “roomAvailable” ensures that only information of hotel “Ibisco” with available rooms is selected. The inscription “$single Room <= 50” of the transition “route hotel” restricts that a single room of the selected hotel should cost no more than 50 Euro per night. The black bars in the Filter Schema diagrams represent manipulation filters used to create or delete XML documents in this context. The rectangles with an inscribed “A” stand for element placeholders of the XML data type anyType and can be instantiated by elements of any type.

If the transition “route hotel” occurs, XML documents are deleted from the places “hotel information” and “customer information” according to FS1 and FS2, respectively. Because of the element placeholders in FS1 and FS2, the name and the structure of the elements “address”, “stars”, “double Room” and “with Breakfast” that are defined in the XML Schema XS1, and the descendants of the root element “customer” defined in XS2 are not checked. According to the Filter Schema FS3 and the XML Schema XS3, XML documents are created which may contain element or attribute values coming from the deleted documents, if the document to be created and the corresponding deleted documents contain common elements (i.e., same-named simple type elements) or attributes (i.e., same-named attributes that belong to same-named elements).

4.2 XML nets modeling WS composition

Basically, the behavior of a Web service is a partially ordered set of operations. XML nets make it particularly easy to model the flow of (semi-)structured data and to specify operational semantics of control flow. Thus, the definition and description of Web services can be visualized by XML nets. Compared with other Petri net variants, XML nets have the native ability that is suitable for WSC, as tokens in XML nets are XML documents, and Filter Schemas defined on arcs can conveniently and directly control the flow of XML documents, manipulate the XML documents and consequently facilitate message filtering, message passing, and dynamic WS discovery and selection.

Here we divide WSC into two categories: One is based on control flow and the other is based on conversation protocols or to say messages passing. In our opinion control flow-based composition is mainly built within organizations. Inter-organizational WSC is mostly based on messages passing, because it is hard to model control flow without knowing related process details of other organizations. It has been proved by many works that CPN have strong ability to model workflow, so have XML nets as a special variant of CPN the same strengths in this field. The composition of control flows is normally modeled using control flow operators such as sequence, choice, unordered sequence, iteration, parallelism with or without communication, discriminator, selection, and refinement. In [8] Petri net-based and especially in [15] XML net-based formal description of WSC based on control flow operators is elaborated.

In [15] a primal contribution about using XML nets for WSC is made, but the authors mainly focus on interpreting WS composition operators using XML nets. In this paper we will deeply analyze advantages of XML nets for WSC and how XML nets support for dynamic WSC.

According to [15], an XML service net (XSN) describes the logical behavior of a Web service with the following properties: (i) XSN has exactly one input place “P” with no incoming arcs. An XML document (with a unique identifier) must be assigned to this place as initial marking. (ii) XSN has exactly one output place “O” with no outgoing arcs. An XML document inserted to this place cannot be deleted or manipulated by any transition of the XML net. Thus, an XSN is an XML net with an input place variable (defined by XML elements) that invokes the Web service. In the output place we define the call for the return value. The output place provides information by sending messages as XML documents.

So $XSN = \langle P, T, A, \Psi, I_p, I_A, I_T, i, o \rangle$, where $\langle P, T, A, \Psi, I_p, I_A, I_T \rangle$ is an XML net according to 4.1. $i$ and $o$ are the input and output place of XSN, respectively. Web service net (WSN) can thus be defined as $WSN = \langle N, D, L, U, CS, XSN \rangle$ with

- $N$: name of the Web service and unique identifier;
- $D$: Web service descriptions;
- $L$: server where the Web service net is located;
- $U$: address for invoking the Web service net;
- $CS$: set of component Web services;
- $XSN$: XML service net that describes the process of the composed Web service net.

We call a Web service net a basic (non-composite) WSN if the set of component Web services contains only the WSN itself, i.e. $CS = \{N\}$. In this case, $XSN$ describes an atomic Web service $N$, where $T$ contains only “normal” transitions, i.e., it contains no transitions...
for binding or calling other WSs. If $|CS| > 1$, then XSN describes a composite Web service $N$, in which $T$ contains transitions for invoking WSs, each with a corresponding XSN. In addition, $T$ can in this case also contain transitions for chaining and dynamically binding WSs. Different transition types in an XSN describing composite WS will be explained in Section 5. Operations of the algebra of WSs are described in [8].

Due to the fact that the WSN does not only describe the control flow of the service, the description of the WSN concerning behavior, functionality, and interfaces can be directly derived from the XSN [15].

5. Realizing WSC tasks with XML nets

In this section we will explicate and demonstrate how XML nets facilitate dynamic WSC with respect to control flow modeling, data and data flow modeling including message passing, and dynamic WS discovery and selection.

In the context of WSC, XML net elements can have the following additional interpretations:

- **Transitions**: Transition in WSC could be of one of the following types:
  1. *WS chaining transitions* for chaining and coordinating Web services;
  2. *WS invocation transitions* for directly calling Web services;
  3. *Abstract WS transitions* for dynamically binding Web services at run-time.

- **Places**: Containers of input or output messages of Web services.
- **Arcs**: Inputs or outputs of Web services.
- **Tokens**: Messages or descriptions of Web services.
- **Markings**: Local or overall states of composition processes.

5.1. Control flow modeling

As a formal and well-founded process modeling technique with graphical nature, XML nets are very suitable for modeling control flows of WSC processes. Sequence, alternative, iteration and other control flow primitives can be modeled straightforward and graphically. XML nets also support hierarchy modeling by refining transitions or places with sub-models. In contrast to other WSC methods without formal foundations (e.g., BPEL), XML net models can be analyzed at design-time through simulative validation and formal verification to avoid possible structural or non-structural errors (e.g., violation of free choice property, deadlocks, and unreachability), and functional or non-functional drawbacks (e.g., undesirable behaviors, inadequate rules).

In dynamic WSC, we firstly build an abstract process model containing abstract WS transitions whose functionalities are determined depending on concrete scenarios and requirements. The abstract WS transitions are often used to dynamically bind and call appropriate real Web services at run-time.

Figure 2 shows an abstract XML net process model for arranging a travel for a conference. After having received the travel permission, the traveler begins to plan the travel by analyzing his requirements on transport, hotel and conference registration, respectively. In order to gather transport and hotel information from desired information provider, the needed Web services with descriptions stored in a WS repository are bound to the process. Besides abstract WS transitions, WS invocation transitions are also utilized to buy transport ticket, reserve hotel, make registration, and check received ticket and confirmations. The dashed outgoing arcs of the place “WS repository” are read-only connections with selective Filter Schema. The way of how to bind Web services whose descriptions are stored in the repository will be explained in Section 5.3.

5.2. Data and data flow modeling

XML nets provide inherent ways for data and data flow modeling. Data objects that move through XSN can be modeled as XML documents whose structure is typified by XML Schemas. Filter Schemas can be used...
to select, create, modify or (partially) delete XML documents and thus realize data manipulation and flow of the documents. Supported by software tools like INCOME2010 [12], non-flowing process objects, e.g., roles, resources, process metrics, key factors etc., can also be modeled in XML and assigned or bound to process or process elements for validation or monitoring purposes.

In WSC, the flowing data objects are usually messages that serve as input and output of Web services. Messages are encoded in XML format and exchanged over standard XML-based messaging protocols like SOAP. Using XML nets, messages as XML documents can be easily modeled as tokens of places and typified by XML Schemas like SOAP Schema2.

In the context of WSC, message passing usually refers to asynchronous communication among Web services by sending structured data packages over transport protocols. One problem in message passing is that an output message of a Web service could be incompatible with the required input message of another WS. For instance, an output message containing 3 strings as service invocation results cannot be used by a Web service which requires only 2 strings as input parameters. This message incompatibility problem can be solved with XML nets by adding a WS chaining transition as mediator between two WS invocation transitions to adjust their input and output messages. The function of the mediating transition is realized straightforward by Filter Schemas assigned to its incoming and outgoing arcs.

Figure 3 shows an example to illustrate how XML nets are used to solve this problem. A mediating transition “adjust message” is added between two WS invocation transitions for reserving hotel and checking hotel reservation, respectively. The SOAP response of the hotel reservation service contains detailed information about reservation id, hotel, customer, room, price, date, and payment status etc. The SOAP request for the reservation checking service provided by the hotel requires only reservation ID. To preclude this incompatibility, Filter Schema FS1 is used to deliver the SOAP response to the mediating transition, and FS2 to create SOAP request using reservation ID filtered from the response.

The mediating transition can also be used to aggregate or disassemble messages. In the first case, it acts as an AND-join transition, which means it has several incoming arcs and only one outgoing arc. Filter Schema of the outgoing arc contains variables used in Filter Schemas of the incoming arcs. In the latter case, the mediating transition is an AND-split transition that has only one incoming arc and several outgoing arcs. Variables of the Filter Schema assigned to the incoming arc are used to construct Filter Schemas of the outgoing arcs.

5.3. Web Service Discovery and Selection

In this paper we distinguish between Web service discovery and Web service selection. While Web service discovery refers to the process of finding functionally suitable Web services in a set of service descriptions that are unknown a priori, Web service selection deals with comparing and choosing appropriate Web services from a known set of descriptions according to customized preferences, including non-functional constrains or requirements, e.g. quality of services (QoS). WS selection is often used after WS discovery to render more accurate and desirable results, especially when several Web services with the same functionalities are discovered. In WSC, WS discovery and selection are applied to achieve process dynamics by finding and integrating Web services to a predefined abstract service process at run-time.

Taking advantage of Filter Schemas and transition inscriptions that contribute to process rules of XML net models, we can combine WS discovery and selection
with the definition and (re)configuration of XML net process rules. At development-time, Filter Schemas for service filtering can be created according to an XML schema of Web service description, e.g., WSDL Schema$^3$, UDDI Schema$^4$ etc. Transition inscriptions can also be used to define constraints for service selection. At run-time, the workflow engine for XML nets interprets and evaluates Filter Schemas and transition inscriptions to filter and create Web service descriptions as XML documents. With process monitoring and administration tools which are usual components of workflow management systems, new criterions and constraints for WS discovery and selection can be incorporated at run-time by reconfiguring transition inscription statements and variable instantiations in Filter Schemas.

**Instantiating abstract WS transition**

Figure 4 shows a fragment of the abstract WSC process in Section 5.1 to demonstrate the ability of XML nets in WS discovery. To obtain transport information, Web services whose descriptions are stored in an unknown WSDL repository should be discovered. The token count of the place representing the repository is supposed to be unlimited. The abstract transitions “find transport info service” and “get transport information” are used to bind and call Web services at run-time. The incoming arc of the transition “find transport info service” is a reading connection (represented by a dashed arc) with a selective Filter Schema FS1. A manipulative Filter Schema FS2 used to create Web service descriptions is assigned to the outgoing arc of this transition. FS1 and FS2 are both generated according to WSDL Schema.

**Adapting abstract WS transition at run-time**

Suppose that the transition “get transport information” is used to acquire flight information. Variables in FS1 should be (partly) instantiated to find Web services supplying flight information. As shown in Figure 5a, the variable of the attribute filter “name” of the element filter “service” is instantiated with a regular expression for filtering Web services whose name contains the string “flight”. This regular expression, together with the Filter Schema, is parsed to XQuery/XPath statements in process simulation or execution. If we use a reconfiguration tool to modify the regular expression at run-time as shown for example in Figure 5b, the transition “get transport information” can then be adapted to acquire other transport information, e.g., train information. After a suitable WSDL document has been found, FS2 is used to create a new WSDL document for the place “WS description” by cloning the discovered document. Figure 5c shows the diagram of FS2 containing the WSDL root element “definitions” and an element placeholder, which is used because the internal structure and details of the document to be created are not of interest in the case of document duplication.

**QoS-aware WS selection**

XML nets can also be used to select desired Web services if service constraints like QoS or customized preferences are taken into consideration. As WSDL and OWL-S provide no or limited ways for describing Web service’s QoS, we use in this paper OWL-QoS [43], which complements OWL-S with an ontology to specify QoS metrics for Web services, to demonstrate the ability of XML nets for (QoS-aware) WS selection. Naturally, other Web service ontologies (e.g., MOQ [11]) or formats (e.g., WSLA [18]) that support QoS can also be used for this purpose.

Figure 6 shows another XML net fragment for the acquisition of transport information by selecting OWL-QoS documents which describe quality of Web services from a repository that contains 3 OWL-QoS documents discovered previously. Similar to the example in Figure 2, the Filter Schema FS1 is used to read and select appropriate OWL-QoS documents, and the Filter Schema FS2 to create new OWL-QoS documents for the place “OWL-QoS document”. As Filter Schema provides no means for formulating inequality operators like “>” and “<”, which are usually used to express constraints with parameters whose quantitative values are limited to a certain interval, transition inscriptions

---

$^3$ [http://www.w3.org/TR/wsdl#A4.1](http://www.w3.org/TR/wsdl#A4.1).

can be used in combination with Filter Schemas to formulate QoS constrains.

Suppose that the traveler wants to use only the Web services that cost no more than 100 US Cent. The Filter Schema FS1 should then be created as shown in Figure 7. The variable of the attribute filter “rdfs:resource” of the element filter “owl:ontProperty” is instantiated with a regular expression to filter OWL-QoS documents specifying the property “costUSCent”. To ensure that the maximal cost of the Web service doesn’t exceed 100 US cent, the inscription of the transition “select transport info service” should be formulated as “$owl:maxCardinality <= 100”. In process simulation or execution, the inscription is parsed and integrated into XQuery statements using XPath inequality operators.

6. Conclusions

In this paper we presented a WSC method based on XML nets, which inherit advantages of Petri nets such as formal semantics and graphical expression. XML nets have additional strengths in the description of process and data objects, and the exchange of XML-based structured data. The advantage of using XML nets for WSC is that control flow modeling, data and data flow modeling, and WS discovery and selection can be realized using a uniform powerful modeling language. Some uncomplicated tasks, as illustrated before, can be fulfilled without developing or using additional software components or agents. Naturally, XML nets can also be combined with other Web service techniques to fulfill more complicated and demanding tasks.

7. Acknowledgement

The authors would like to thank the anonymous referees for many valuable comments on an earlier version of this paper.

8. References