Policy-based Web Service Composition

Soon Ae Chun
CDS Dept.
Seton Hall University
chunsoon@shu.edu

Vijayalakshmi Atluri
CIMIC & MS/IS Dept.
Rutgers University
atluri@cimic.rutgers.edu

Nabil R. Adam
CIMIC & MS/IS Dept.
Rutgers University
adam@adam.rutgers.edu

Abstract

With the proliferation of Web technologies, the need to deliver services via the Web has increased tremendously. More and more, customers now demand one-stop service that calls for multiple services crossing organizational boundaries, which are required to be carefully and dynamically composed in a customized manner. Such a composition should not only select the most appropriate available service, but should also adhere to the policies and regulations governing the organizational services. In this paper, we demonstrate how such a composition can be accomplished to form a coherent service flow by using rules and services expressed as a knowledge base and topic ontology. The description of rules with topic concepts allows the system to easily identify the relevant rules in a certain domain and to identify and select appropriate Web services for composition. We consider different types of compositional rules including syntactic, semantic and pragmatic (contextual), which play a major role in the discovery and selection of Web services. We model the knowledge of rules and of the topic ontology using OWL, DAML-S, RuleML and RDF standards.

1. Introduction

Traditional business and commerce activities have been undergoing unprecedented transformation with the Internet. There has been an increase in intra-organizational business processes to facilitate the coordination among tasks and activities from different work units. An increasing number of corporations has also been expanding their business scope and value chains by forming alliances with other companies. These alliances with partners involve interorganizational business processes that are established based on cooperative agreements between firms, and involve exchange, sharing, or co-development. In addition to these *static, agreed upon cooperations*, recently businesses have started to use the WWW as a vast repository of resources for communication such that businesses are now able to link people and resources across organizational boundaries dynamically, creating innovative enterprises, such as virtual companies, markets and trading communities. The customers' demands in the EC environment, such as customized one-stop shopping for various items or services, also drive the agglutination of services and resources from various organizations, resulting in temporary virtual enterprises. These *loosely coupled, dynamic processes* are also the grand vision of the Semantic Web [4] in which the users' interactions with the static Web resources (documents and services) will change into automated agents' interaction of dynamic services. Web services from multiple autonomous organizations in different locations will be identified, selected and composed together to perform tasks, provide information, transact business, and take action on behalf of users. These dynamic business processes or services that involves multiple organizations' Web services are called *interorganizational Web services.*

Interorganizational Web services are typically dynamic and loosely coupled Web services, created on demand, that are composed of component (atomic) services from participating organizations. The issues in designing inter-organizational Web services include how to discover and locate the "right" component Web services that meet the interorganizational business goals and the customer satisfaction with tailored services. We call this process of defining a customized Web service the *Web composition problem.*

The inter- and intra-organizational processes are subject to various business rules that govern the way a business operates, including the specification of policy or conditions that must be satisfied. Our approach to the composition problem considers these policies and rules.

Overview of our approach to Service Composition: The inter-organizational process is modeled as a composite Web service, a workflow called service flow. Each participating organization has a set of services to offer, called component (or atomic) services. A composite service is made up of these component services from each organization. However, service composition requires to first select the services that are compatible. We consider three levels of service compatibility in order to make sensible compositions: syntactic, semantic and policy level compatibilities. Under syntactic compatibility, the service discovery and selection will verify whether a service is compatible with respect to its specifications, such as matching of the output of one service with the input of the subsequent service.

On the other hand, semantic compatibility considers domain-specific guidelines, and checks whether the service composition follows the "correct" combination specified in a standard operating procedure, such as manufacturing a specific drug that requires right kind and amount of ingredients and their combinations. The policy level compatibility requires that the policies and rules that each company imposes should be compatible with the policies of other services. For instance, the delivery day for one service is on every Thursday, thus, the composition needs to select other services that conform to this company policy. The policy level compatibility considers both organization-internal policies as well as regulatory rules that are imposed on the industry level business practices. Figure 1 illustrates these different levels of compatibility rules in a service composition. Prior work [7, 8] on Web service composition considers syntactic and semantic compatibilities, whereas, in this paper, we consider

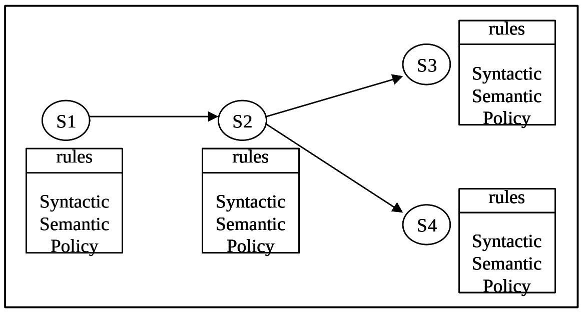

Figure 1: Service composition based on Compatibility rules

all the three types of compatibilities.

The standard XML markup languages such as WSDL, UDDI, RDF, DAML-S, and OWL, facilitate the Web service discovery and composability problem. They allow the discovery and composition based on syntax and semantics of described Web services. We argue in this paper that there should be more than these for fully *automated Web composition*. More pragmatic rules need to be considered, such as geographic location of Web services, quality of service, delivery policy, product cost structure, trust of hosts (e-Vendors), and relationships among hosts (competitors), regulatory constraints, etc. Many pragmatic rules are specified in company policies. In composition, not only the user's requirements are considered for customization of Web services, but also the organizational policies, because many policy rules specify how the actual business contracts and negotiations need to be handled within the organization as well as with other organizations and customers. We focus on company policy rules and their role in composition. We discuss how to represent these policy-related pragmatic rules, and user models to specify complex requests. These rules affect not only the proper selection of services and service providers but also the composition.

This paper is organized as follows. In Section 2, we discuss the Web service composition rules and policy ontology. Section 3 presents policy-based composition model. In Section 4, we present our user model followed by our system architecture in Section 5. The relevant work and the conclusions are presented in Section 7 and 8, respectively.

2. Policy Rules for Service Composition

The inter- and intra-organizational processes are subject to various business rules that govern the way a business operates, including the specification of policies and conditions that must be satisfied. Business rules include various policies and procedures internal to a particular unit or organization as well as various business protocols between units and among organizations. For example, "Pay a supplier invoice only if it has been approved." "Only good customers may obtain credit orders." "Overdue invoices occur 30 days after statement." "Many payments can be made per invoice." "Sum of payments should be equal to order value less of sum of credit notes." "One invoice should be generated for one order." "Credit balance should be greater than or equal to order value to accept order, otherwise reject." In addition, certain business processes also need to follow the inherent technical and semantic constraints and rules to achieve a proper task. For example, a semi-conductor chip making process or an auto manufacturing process needs to follow certain technical specifications. They also are subject to external regulations and codes of business conduct imposed by laws and auditing authorities.

These rules and policies govern the composition of services, by restricting certain combinations of services or by imposing constraints on selection of a service over another services. We have defined three levels of rules to consider:

Syntactic (operational) Rules: The composability of services depends on the preconditions, input, and output requirements. For instance, the output of a Web service *s1* needs to be compatible with the input of service *s2*. The syntactic rules ensure that the composition of the services yields a composite service that is operationally or syntactically possible.

Semantic Rules: The service composition should make sense according to rules and the standard business practices. The semantic composition rules often require domain expert knowledge, such as business law, trade laws, State legislations, federal environmental regulations, formal company policies, experimental procedures and environments etc. For example, according to New Jersey law, any land development close to the coastal area forces the agent to select *the coastal permit application service*. The selected service is required to be part of the service flow. These rules enforce the expertise knowledge in a domain. Thus, the application of semantic constraints (rules) plays a significant role in service composability.

Policy rules: In real life it is often the case that several services possess the same profile and provide the same functionalities. The automated selection of one service among these functionally equivalent services may require policy level compatibility. For example, when buying a book and flowers as a combined gift, the delivery of the book and flowers need to be coordinated. The service provider who can deliver the book and flowers at the same day is a preferable choice over others. The policy of the bookstore and the policy of the flower shop in terms of the delivery day at the same destination should be compatible for this particular case. The service and provider selection are further restricted by the policy compatibility requirements. Several vendors may offer the services, but the system needs to find the vendors whose policies on delivery are compatible

In summary, syntactic, semantic and business policy rules allow an agent to restrict the selection and composability of services. These rules allow the service composition to be customized for the user's service requirements as well as the organizational policy level composability.

2.1. Policy Ontology

Our approach for building an agent for automatic Web service composition is based on the rules/policy ontology. The rules ontology describes organizational policies with tags from the classification (is-a) ontology as well as with terms from the topic (thematic) ontology. The concepts in the ontology help agents easily identify the applicable rules for composition, and the identified rules, in turn, help to find the Web services that obey the rules. It also allows an organization to specify the resources of the rules ontology definitions that are used for specifying the rules, thus allowing various domain rules and policy ontologies independently developed by various organizations to be used.

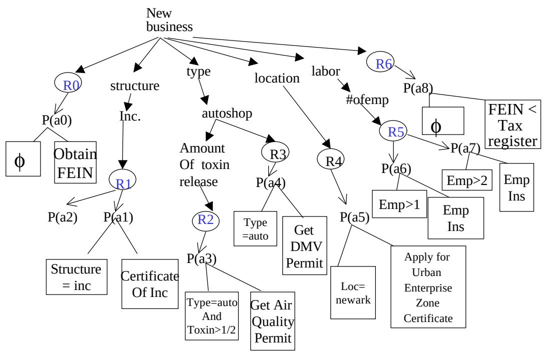

Figure 2: Topic ontology for government regulations for business registrations.

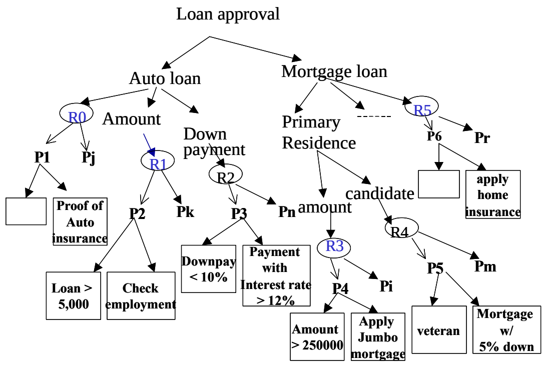

Figure 3: Topic ontology for loan approval policies

For example, the knowledge about regulations is modeled as regulatory topic ontology (Figure 2) in the domain of new business registration. Figure 3 shows a topic ontology of a loan approval policies.

This topic-subtopic hierarchy allows the identification of regulatory and policy rules based on topic concepts. Each node can be associated with a rule. Each rule node, e.g. R1, R2,…, has concrete instances, specific to company policy rules, P1, P2,…. Different companies or different government agencies can implement a rule with their own policy rules, imposing conditions and terms particular to the organization.

Each policy rule has a Condition-Action pair, where the Condition is a logical expression and an Action specifies the type of rules (insert rule, ordering rule, etc.) and a link to Web services that implement it (i.e. an *implemented-by* relationship). We formally define the policy rules as follows:

Definition 1: [Policy Rule] *Policy rule P is defined as a tuple <C, o, c, a, m, N>, where C denotes the concept P belongs to in a policy ontology, o denotes the organization P belongs to, c the conditions, a the actions, m Î {negotiable, non-negotiable} the mode of the policy, and N a negotiation procedure.*

Rules are either negotiable or non-negotiable, *m*, and there is a negotiation procedure *N* on how the negotiation should proceed to relax the conditions *c* of the policy P.

Figure 4 shows an extension of OWL (Web Ontology Language) [14, 11] with proposed tags to represent the policy rules and ontology. The actual body of the rule is linked by the resource link in the description and is represented in RuleML [15] as shown in Figure 5.

```
<policy rdfID="#P2">
   <owl:Topic rdf:ID="AutoloanAmount">
     <rdfs:resource="http://www.myloan.com/bizrules" />
     <rdfs:SubtopicOf Auto Loan />
   </owl:Topic>
   <hasRestrictRule
rdf:resource=http://www.myloan.com/amount_rule.ruleml   />   (see
figure 8)
  <hasProperty ref:resource="# non-negotiable"/ >
  <hasNegotiation  rdf:resource=http://service.com/negotiate"/>
   <belongTo  rdf:resource="#ABC Insurance, Inc."/>
 </policy>
```

Figure 4: Proposed OWL extension for Policy Ontology

```
<damlRuleML:ind>amount_rule</damlRuleML:ind>
   </damlRuleML:_rlab>
<damlRuleML:_body
       <damlRuleML:atom>
          <damlRuleML:_opr>
            <damlRuleML:rel>Less than $2000  <damlRuleML:rel>
          </damlRuleML:_opr>
        <damlRuleML:var>LoanAmount</damlRuleML:var>
       </damlRuleML:atom>
   </damlRuleML:_body>
<damlRuleML:_head>
   <damlRuleML:atom>
     <damlRuleML:cterm>
        <damlRuleML:_opc>
          <damlRuleML:ctor>Insert<damlRuleML:ctor>
  </damlRuleML:_opc>
  <damlRuleML:var>VerifyEmpolyment Status
  </damlRuleML:var>
   </damlRuleML:cterm>
   </damlRuleML:atom>
  </damlRuleML:_head>
<damlRuleML:imp>
```

Figure 5. Example of RuleML representation for a policy rule: If the amount of loan is greater than $2000, then invoke a service to verify the applicant's employment status.

3. Policy-based Service Composition Model

3.1. Service Model

A Web service s is represented by a tuple *<A, Pre, I, O, Post, Prop>*, where *A* denotes service provider, *Pre* preconditions, *I* input, *O* output, *Post* post-conditions, and *Prop* properties of s. This representation is implemented using standard DAML-S ontology notation [10]. While the standards of WSDL provide syntactic annotations for Web services that include tags mostly limited to network connection, input/output specifications, data types and how they bind to concrete ports, DAML-S provides semantic descriptions of Web services, using an ontology to describe service capabilities and properties. Web services in DAML-S are described with profiles, models, and groundings. The service profile describes information on what organization provides the service, what function the service computes, and other characteristics of the service. It includes contact information of a service provider, input and output of the service, and features such as the category of a given service, quality rating of the service, an estimate of the maximum response time, the geographic radius of a service, etc. The service profile provides the type of information about "what the service does." The profile is therefore used for advertising, registry, discovery, and matchmaking for the agent to find the service that meets its purpose. The service model tells "how the service works," guiding a service-seeking agent to compose a service as a process, using inputs, outputs, preconditions, effects, and sub-processes. This information is used to perform a specific task, to coordinate the activities of the different participants, and to monitor the execution of the service. A service grounding specifies the concrete details of how an agent can access a service, such as communications protocol (e.g., RPC, HTTP-FORM, SOAP), and port numbers used in contacting the service.

```
 <damls>
  <damls:class  openCellphoneAccount
    <damls SubclassOf  openPhoneAccount > </damls:class>
    <damls Provider   uri:http://www.cardissue.com/ >
    <damls:function>
     < precondition:exp validCreditcard >
     < input:var  #Address, Name>
     < output:var  #cellphonenumber> </damls:function>
     <damls:prop
       <damls:var #quality-of-service  >
       <damls:var  #max-response-time  >
       <damls:var #geographic-area-covered  > </damls:prop>
    </damls >
```

Figure 6: DAML-S description of a Web service

The Service profile is used for advertising, registry, discovery, and matchmaking. Once an agent has located a service appropriate for its need, the service model and service grounding give enough information for an agent to make use of the selected service. Figure 6 is an example of a service description using DAML-S. The service “openCellphoneAccount” is a subclass of “openPhoneAccount,” it has a provider, and it computes a cell phone number as its output. It also specifies properties such as quality of the service, maximum response time, as well as geographic area covered by the service.

3.2. Service Flow Model

A Web service flow is a composite service, consisting of individual atomic Web services. The service flow specifies the coordination among these component Web services. We adopt the formal workflow model developed in [3, 9] to represent a Web service flow that can be implemented with BPEL4WS [5]. We use the Business Process Execution Language for Web Services (BPEL4WS or BPEL for short), an XML-based workflow definition language, to implement the composite services. BPEL documents are executable scripts that can be interpreted by business process engines to implement the described process. Each step in the process corresponds to a Web service provided by a business organization. It also provides tags to specify Web service coordination with the information necessary to link the various tasks in a process. Its transaction specification provides a framework to monitor the success or failure of each coordinated activity, and ways to cancel the process in case of failure. Figure 7 shows the BPEL representation of the business process of a credit information provision service[1].

```
<process name="CreditInfoProcess"
   targetNamespace="http://acme.com/simpleloanprocessing"
   xmlns="http://schemas.xmlsoap.org/ws/2002/07/business-process/"
   xmlns:lns="http://loans.org/wsdl/creditinfo"
   xmlns:creditdef="http://cimic.rutgers.edu/services/creditservice"
   xmlns:apns="http://cimic.rutgers.edu/services/creditinfo">
 <partners>
   <partner name="customer"
            serviceLinkType="lns:creditinfoLinkType"
            myRole="client"/>
   <partner name="provider"
            serviceLinkType="lns:creditInfoLinkType"
            partnerRole="provider"/>
 </partners>
 <containers>
   <container name="request"
                messageType="creditdef:SSNinfoMessage"/>
   <container name="creditInfo"
                messageType="apns:creditInfoMessage"/>
 </containers>
 <sequence>
   <receive name="receive1" partner="customer"
            portType="apns:creditInfoPT"
            operation="provide" container="request"
            createInstance="yes">
   </receive>
   <invoke name="invokeprovider"
           partner="provider"
           portType="apns:creditInfoPT"
           operation="provide"
           inputContainer="request"
           outputContainer="CreditInfo">
   </invoke>
   <reply name="reply"partner="customer"
            portType="apns:creditinfoPT"
            operation="provide" container="creditInfo">
   </reply>
 </sequence>
</process>
```

Figure 7. BPEL4WS representation of a service flow of obtaining credit information process

3.3. Policy-based Selection of Service Provider

We have developed an automatic workflow composition algorithm and prototype in the E-government domain [7,8,9]. In the e-government domain, the services discovered based on the rules are unique. In the electronic commerce domain, the services with the same functionalities can be provided by many service providers. The problem is to select the one service for composition. Our approach is to automatically generate a service flow with candidate service sets, $S= \{S_1, S_2, \ldots\}$ and D, a set of dependency relations among services in S. Unlike our previous approach, now each candidate service S_i in S is a set of functionally equivalent services, $S_i = \{s_1, s_2, \ldots\}$ where

[1] We assume the services are described according to WSDL standards as shown in the URI links.

each s_j in S_i is a functionally equivalent service to s_k, but s_j and s_k are provided by different vendors and implements different policy rules. In order to select one appropriate service s_j from each S_i, we use the compositional compatibility in syntactic, semantic as well as policy level among s_k in S_i and s_l in S_j.

The syntactic compatibility checks the input, output, and preconditions between individual services from different sets. When they are compatible, we say the services are *syn-compatible,* denoted by s_i ¤ s_j. The semantic compatibility check ensures the quality of services that are required in the domain is met or the user requirements are met. The semantically compatible services are *sem-compatible*, denoted by s_i ˘ s_j. The policy level compatibility ensures the rules imposed by one organization allow its service to be composable with another service organization's policy. The policy compatible services are called *p-compatible*, denoted by $s_i \circlearrowright s_j$. These compatibility checks look at the relationships of policies and rules to filter out any incompatible services in the composite service. The selected services in each set S_i are considered compatible for composition.

Definition 2. [Service Compatibility] *A service s_i is compatible (composable) with s_j, denoted by $s_i \sim s_j$, iff s_i ¤ s_j, s_i ˘ s_j and $s_i \circlearrowright s_j$.*

Consider an e-catalogue company's business policy on delivery to satisfy the customers: P1 states that if purchase items are books and/or CDs and the delivery destination is within NJ, then use (insert) a delivery service that can deliver goods within 1 business day. This policy seeks for a delivery service that could deliver goods in NJ within 1 day. Let's assume that mail service providers A and B are competitors in delivery. The delivery policy of A states that the merchandise can be delivered in NJ within 1 day, while B can deliver in NJ within 2 days. A's delivery service is compatible with the e-catalog company's delivery policy, while B's is not.

The task of identifying p-compatible policy rules is done with the help of the semantic class and topic ontology used in describing policy rules. For instance, the rules on the topic node of "newBusiness," need to consider rules *R0* and *R6* in Figure 2, and in the State of New Jersey, the rule *R0* is imposed with a policy *P0*, i.e. all newBusiness registration requires a FEIN (Federal employer ID number) (*P0* in Figure 2); and *R6* with the policy rule, i.e. FEIN should be obtained before tax registration (*P8* in Figure 2). Similarly, the loan amount-related rule *R5* in Figure 3 is instantiated with a policy *P4* in a company but it may be implemented with a policy *Pn* in another company with different conditions. We use *Ri<={P1, P2,…,Pn}* to denote a topic related to a rule *Ri* that is adopted by different policies by different organizations. Thus, when one considers the topic of *primary home mortgage service*, there is a rule concerning the *amount of mortgage*. This rule varies according to different service providers and these variations are realized with policy rules, *P1,P2,…* The choice of a service provider among many compatible service providers depends on the matching of the conditions of a policy with the customer's mortgage amount. If the condition is met, then the policy is considered p-compatible and the service provider that imposes the policy (e.g. *P4* in figure 3) is selected among other service providers, and its service "*buy home insurance*" is added into the composite service flow.

In case that there is a policy whose conditions do not meet the policy currently looking for, but can be negotiable, the negotiation procedure for the policy is initiated. When the negotiation process results in the mutually agreeable conditions, the policy is made to be p-negotiable. We omit the formal algorithm for searching for a p-compatible policy rule and for negotiation due to the space limitations.

4. User Model Specification Language

The semantic Web envisions that the user goals are specified in simple terms, and the software agent figures out the sensible ways of achieving the user goals. However, in reality, the user has various constraints and preferences in selecting particular services. These preferences, constraints and goals are considered as user specified policies, so-called *user's service policies* in selecting, composing and executing Web services. For instance, the user may want to specify his constraints like "if *s1* and *s2* provide same products, but *s1* offers discount, then select *s1*," or "if the service provider of *s1* can deliver its product within 2 days, then also add *s2* to buy a product *d* from the same vendor to be delivered together," etc. Unlike codified rules and policies, the user policies can be vague and ambiguous. For instance, a user can specify that he wants to get some entertainment ticket on Wednesday evening near Washington Square (with likelihood of 50%, if the ticket costs over $20, and with likelihood of 90% if the ticket costs less than $20). Another example would be "I want a service of type X that is reliable (or trustworthy)." In this case, what does the user mean by being reliable (or trustworthy). For a software agent to compose services automatically, the agent should understand what types of activities are considered entertainment. This may depend on personal tastes. It also should know what defines the characteristics of being a reliable or trustworthy service.

We model the user policies with XML-based semantic tags based on a topic ontology as shown in Figure 8. The user policy uses tags from an ontology so that the terms (e.g. entertainment) correctly refer to specific types of activities. The user constraints expressed in terms of semantic tags help resolving the vocabulary differences between the tags that are used to express user policies and the tags used to express rules or services. We propose to use a user's service policy specification language similar to DAML-S. We call it user service policy language (USPL). This will describe the user policies with the semantic terms derived from an ontology the user picked.

```
<uspl Class:goal="home mortgage loan"
<uspl ref:source="http://www.myontology.com/bizreg">
<uspl Restriction
   <uspl cond:estimatedResponseTime ="#2 weeks" >
   <uspl: ref:source=(some uri where estimated Response
Time is defined)>
   <uspl cond:LoanType="#residential mortage">
   <uspl cond:LoanAmount="#$250,000">
              ….
```

Figure 8: User Service Policy Language (USPL)

5. System Architecture

Figure 9 shows the components of the system architecture for our approach to policy-based service composition system.

Service provider: defines the services and policy rules using service and policy topic ontologies. They advertise their service capabilities along with the rules the service is subject to.

Service request interpreter: processes the user service requests and user policies (constraints) specified in USPL.

Policy rule selection: identifies and discovers p-compatible or p-negotiable rules. The user policies and requests are matched with the policy ontology.

Service and provider selection: uses selected rules to determine appropriate services and their providers.

Service Composition: puts together the selected services, and generates the executable service flow specification in BPEL4WS. The resulting service flow is syntactically, semantically and policy compatible and therefore is sensible to be executed. The composition component requests the specific Web service instances from a service provider, and also requests the specific composition rules identified to use in the composition.

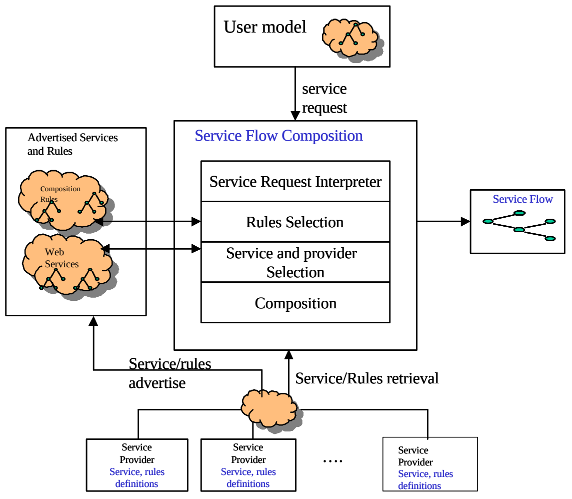

Figure 9: System Components for Automatic Service Flow Composition

6. Related Work

The Web service technologies, such as the XML-based language WSDL to describe the services, UDDI for service registry, and SOAP for a uniform way of passing XML-encoded data, were developed to discover and determine right component services and compose a Web service. WSDL defines a Web service as collections of network endpoints or ports. DAML-S is a semantic markup for Web services for the properties and functionalities of Web services using a set of ontologies, thus capturing the semantics of what the service does (service capabilities), rather than just a collection of the network ports. The Web service standards facilitate service discovery [1, 11], but fall short on automatic composition of services.

Service discovery and composition is an active research area. [13] proposes an approach to compatibility and substitutability relationships among *e*-Services that can be derived by analyzing their interfaces and behaviors. In [2], two-phase composition is proposed. First, a high level analysis uses service descriptors and semantic information and similarity evaluation to obtain compatible services, and second, a structural level analysis measures similarity by matching incoming and outgoing messages and their parameters. Similarly, [16] has three levels of service composition. First the semantic relatedness between the requested service and available services is used to select services. Second, the capabilities of selected services are considered, and finally syntactic analysis is performed to match the interfaces of services. [6] takes a similar approach using syntactic, operational and semantic similarity metrics for service composition. Our approach is different from these and others in that we use not only the syntactic and semantic compatibilities of service, but also use the policies and rules for discovering and finding compatibility to generate sensible service flows.

7. Conclusions and Future work

Many real-life business processes (i.e. Web services) have to live by various rules and constraints. In this paper, we have classified these rules into syntactic, semantic and policy rules that play a major role in discovery and selection of Web services. These rules incorporate the knowledge that is necessary to select and to compose Web services into a coherent service flow. Our approach to Web service composition is based on building the knowledge of policy rules using OWL, DAML-S, RuleML and RDF standards. The description of rules with topic concepts allows the system to identify easily the relevant rules in a certain domain and to identify and select appropriate Web services for composition. We introduced syntactic, semantic, and policy compatibility of services to yield correct composite service.

Future work includes how to handle conflicting rules that are distributed and heterogeneous, described by different organizations; there may be precedence orders among organizational policies that need to be addressed; how to build the policy rules base (semi-) automatically from diverse documents where rules are stated; how to maintain the updates of rules and their effects on the composite web services; and how to use the rules description for e-negotiations. The rules ontology and the ontology for user policy specification need to match for correct identification of applicable rules. The ontology interoperability also needs to be addressed.

8. References

[1] A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. DAML-S: Semantic markup for Web services. In Proceedings of the First International Semantic Web Working Symposium (SWWS), pp. 411-430, Stanford, 2001.

[2] V. D. Antonellis, M. Melchiori, and P. Plebani, An approach to Web Service compatibility in cooperative process, in *Proceedings of Workshop on Service Oriented Computing (SOC*), Orlando, 2003.

[3] Vijayalakshmi Atluri, Soon Ae Chun and Pietro Mazzoleni, Chinese Wall Security for Decentralized Workflow Management Systems, *Journal of Computer Security*, (in press).

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. Scientific American, 284(5), pp. 34-43, May 2001.

[5] BPEL4WSWebsite:http://www-106.ibm.com/ developerworks/library/ws-bpel/

[6] J. Cardoso and A. Sheth. Semantic e-workflow composition. Journal of Intelligent Information Systems, (to appear). (also availalable as Technical report, LSDIS Lab, Computer Science, University of Georgia, July 2002.)

[7] S. A. Chun, V. Atluri and N. R. Adam, Domain Knowledge-based Automatic Workflow Generation, in the *proceedings of the 13th International Conference on Database and Expert Systems Applications (DEXA 2002),* September 2-6, 2002, Aix en Provence, France.

[8] S. A. Chun, V. Atluri, and N. R. Adam, Dynamic Composition of Workflows for Customized eGovernment Service Delivery, in the *proceedings of The Second National Conference on Digital Government (dg.o 2002)*, May 19-22, 2002, LA, CA.

[9] Soon Ae Chun, Decentralized Management of Dynamic and Customized Workflows, Ph.D. Dissertation, Rutgers University, 2003.

[10] DAML-S:http://www.daml.org/services/daml-s/0.7/ CongoService.daml

[11] J. Hendler and D. L. McGuinness. DARPA Agent Markup Language. IEEE Intelligent Systems, 15(6):72-73, 2001.

[12] M. Klein and A. Bernstein. Searching for services on the semantic web using process ontologies. In Proceedings of the International Semantic Web Working Symposium (SWWS), July 2001.

[13] M. Mecella, B. Pernici, and P. Craca. Compatibility of e-Services in a cooperative Multi-platform Environment. Procs. Of the 2nd VLDB-TES Workshop, Rome, 2001.

[14] OWL Website: http://lists.w3.org/Archives/Public/www -webont-wg/2002May/att-0173/01-owl.html

[15] RuleML Website: http://userpages.umbc.edu/~mgandh1 /2002/06/DamlRuleML/

[16] Jian Yang, Mike P. Papazoglou: Web Component: A Substrate for Web Service Reuse and Composition. CAiSE 2002: 21-36