Web Service-based Business Process Automation Using Matching Algorithms

Yanggon Kim[1] and Juhnyoung Lee[2]
1 Computer and Information Sciences,
Towson University, Towson, MD 21252, USA,
ykim@towson.edu
2 IBM T. J. Watson Research Center
Yorktown Heights, New York 10598, USA,
jyl@us.ibm.com

Abstract. In this paper, we focus on two problems of the Web service-based business process integration: the discovery of Web services based on the capabilities and properties of published services, and the composition of business processes based on the business requirements of submitted requests. We propose a solution to these problems, which comprises multiple matching algorithms, a micro-level matching algorithm and macro-level matching algorithms. The solution from the macro-level matching algorithms is optimal in terms of meeting a certain business objective, e.g., minimizing the cost or execution time, or maximizing the total utility value of business properties of interest. Furthermore, we show how existing Web service standards, UDDI and BPEL4WS, can be used and extended to specify the capabilities of services and the business requirements of requests, respectively.

1 Introduction

A *business process* refers to a process in which work is organized, coordinated, and focused to produce a valuable product or service. Business processes comprise both internal and external business partners and drive their collaboration to accomplish shared business goals by enabling highly fluid process networks. A *business process solution* consists of a model of the underlying business process (referred to as a *process model* or a *flow model*) and a set of (flow-independent) business logic modules. The abstractions of the elementary pieces of work in a flow model are called *activities*; the concrete realizations of these abstractions at process execution time are referred to as *activity implementations*. The prevalent technique for creating business process solutions follows a manual and tedious approach involving assimilation of varied process design and vendor specifications and writing vast amount of code that produces a tight inflexible coupling between processes. *Web*

services provide a set of technologies for creating business process solutions in an efficient, standard way. The promise of Web services is to enable a distributed environment in which any number of applications, or application components, can interoperate seamlessly within an organization or between companies in a platform-neutral, language-neutral fashion. From the perspective of business process solutions, a Web service could represent an activity within a business process, or a composite business process comprising a number of steps [7]. A Building a business process solution by using Web services involves specifying the potential execution order of operations from a collection of Web services, the data shared among the Web services, which business partners are involved and how they are involved in the business process, and joint exception handling for collections of Web services. A basis for these specification tasks is the discovery, composition, and interoperation of Web services, which are primary pillars of automatic process integration and management solutions. In this paper, we focus on the following two problems of the Web service-based business process automation: the location of services based the capabilities of published services, and the composition of business processes based on the business requirements of submitted process requests. This paper discusses solutions to these problems, and, especially, focuses on the following aspects: the specification of the capabilities of services and the requirements of requests, and algorithms for matching published services and submitted process requests in terms of service capabilities and requested business requirements.

The rest of this paper is structured as follows: Section 2 summarizes the previous work on the problems of interest, discusses their limitations, and explains how the work presented in this paper addresses them. Section 3 addresses issues involved with the specification of business requirements in process request documents. Section 4 presents a matching algorithm for locating services based on service capabilities and properties. Section 5 presents matching algorithms that are deigned to satisfy the business requirements and provide optimal solutions in terms of meeting certain business objectives. In Section 6, conclusions are drawn and future work is outlined.

2 Related Work

Recently, there have been active studies related to the Web service-based process automation in both academia and industry. Industrial effort for the business process automation is centered around the Business Process Execution Language for Web Services (BPEL4WS), which is an XML-based workflow definition language that allows companies to describe business processes that can both consume and provide Web services [14]. Along with complementary specifications, WS-Coordination and WS-Transaction, BPEL4WS provides a basis for a business process automation framework, and is viewed to become the basis of a Web service standard for composition. With the BPEL4WS specification, vendors such as IBM provide workflow engines (e.g., BPWS4J [13]) on which business processes written in BPEL4WS can be executed. Running on Web application servers such as Apache Tomcat, the workflow engines support the coordinated invocation, from within the

process, of Web services. There are some studies, mostly from academia, done for the specification of service capabilities and process requests by using semantic knowledge-based markup languages, notably, OWL-S(formerly known as DAML-S) [2].

For matching published services and submitted process requests in terms of service capabilities and requested business requirements, we propose a system multiple matching algorithms, a *micro-level matching algorithm*, which matches the capabilities and attributes of published services with activities in a process request, and *macro-level matching algorithms*, which are used to compose a business process by selecting one service for each activity among the candidate services selected by the micro-level algorithm. Some previous work envisioned the task of business process composition as an AI-inspired *planning* problem [3, 11]. They represent a Web service by a rule that expresses the service capable of producing a particular output, given a certain input. Then, a rule-based expert system is used to automatically determine whether a desired composite process can be realized using existing services, and construct a plan that instantiates the process.

3 Requirement Specification

In this section, we address issues involved with the specification of business requirements and objectives in process request documents. We discuss what information on business requirements and preferences need to be specified in process request documents and how the information may be used in the discovery of services and the composition of processes. We extend the BPEL4WS specification to accommodate this information in business process documents. Business process documents written in BPEL4WS mostly consist of the following parts, which are primary components of BPEL4WS [14]:

- Process definition,
- Partner definition,
- Container definition,
- Flow model, and
- Fault handling.

Fig. 1 shows an example of specifying several requirements for a business process, i.e., cost, time and quality of services. In Section 5, we will explain how this requirement information is used in selecting services for composing a business process with an algorithm for optimizing certain business objectives, or a multi-attribute decision analysis algorithm that maximizes the total utility value of selected service combinations.

```
<businessRequirements>
  <requirement name="processBudget"
        type="cost"
        value="30000.00"
        unit="USD"
        limit="maximum"
        weight="10" />
```

```
    <requirement name="processTime"
            type="time"
            value="365"
            unit="days"
            limit="maximum"
            weight="7" />
    <requirement name="processAvailability"
            type="quality"
            value="98.0"
            unit="%"
            limit="minimum"
            weight="5" />
</businessRequirements>
```

Fig. 1. Specification of business requirement

In addition to business requirements, users of business processes sometimes need to express their preferences in selecting Web services for implementing processes. An example is the preference regarding whom a company prefers (or does not prefer) partnering with in a business process depending on its existing business relationship with service providers.

4 Service Discovery with Micro-Level Matching

This algorithm returns a (pre-specified) number of services that sufficiently match with an activity in the request. It is based on the previous work in [3, 9] which allows service providers to advertise their services in OWL-S service profile markup, and match submitted requests again in OWL-S profile markup with appropriate services. Unlike this previous work, our work does not depend on OWL-S profile, but utilizes the specification of service capabilities and request requirements directly stored in UDDI records and BPEL4WS documents, respectively. This algorithm is referred to as a *micro-level matching algorithm*, because it mostly deals with a single atomic process of a request.

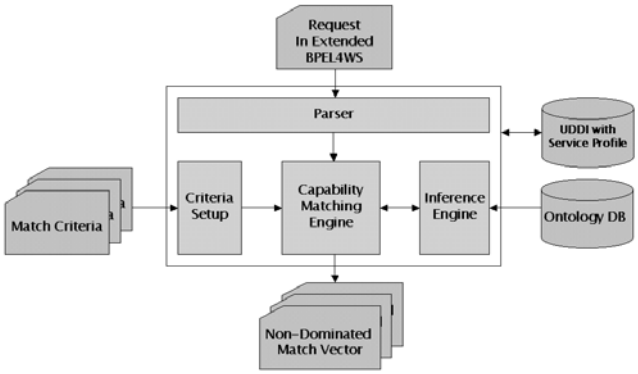

Fig. 2. The micro-level matching algorithm

Fig.2 depicts the architecture of the micro-level matching algorithm. The Parser module is capable of parsing an input BPEL4WS document and creates objects storing business requirements specified in the documents. The Inference Engine module parses and reasons with ontologies that provide the working model of entities and interactions in knowledge domains of interest, specified in the OWL language [5, 6]. The Capability Matching Engine is based on the semantic matching algorithms outlined in [3, 9]. While the matching algorithm presented in [9] is constrained to match only input and output messages of Web services, the algorithm proposed in [3] generalized the previous algorithm to match for any attribute of services and requests by parameterizing the match criteria such as quality, service categories as well as input and output messages. Figure 4 outlines the main control loop of the matching algorithm, which is based on the work in [3]. The *degree of match* is a measure of the semantic distance between the conceptual meanings of the service attributes [3, 9]. Each attribute has a lexical concept attached to it that is defined in the Ontology Database available to the Inference Engine. We use three different degrees of matches based on specialization relationship as defined in [9]. As given in the degreeOfMatch module of Fig.3 , the degrees of match are preferentially ordered based on the semantic distance that the degree represents: an EXACT match between concepts is preferred to a PLUG_IN match, and a PLUG_IN match is preferred over a SUBSUMES match [9].

```
matchAttribute(request, service, matchCriteria) {
   for each criteria in matchCriteria do {
      requestAttributes = request(attributeCriteria);
      serviceAttributes = service(attributeCriteria);
      for each requestAttribute in requestAttributes do {
         for each serviceAttribute in serviceAttributes do {
            degreeMatch = degreeOfMatch(requestAttribute,
                                                serviceAttribute);
            if (degreeMatch < matchLimit)
               return fail;
            if (degreeMatch < globalDegreeMatch)
               globalDegreeMatch = degreeMatch;
         }
      }
    }
    return success;
}
degreeOfMatch(requestAttribute, serviceAttribute) {
  if requestAttribute is SameClassAs serviceAttribute return EXACT;
  if serviceAttribute is SubClassOf requestAttribute return PLUG_IN;
  if requestAttribute is SubClassOf requestAttribute return SUBSUMES;
  else return FAIL;
}
```

Fig. 3 Capability matching algorithm

5 Macro-Level Matching

The micro-matching algorithm works with other matching algorithms, *macro-level matching algorithms*, which are used to compose a business process by selecting one service for each activity in the request. The output from the macro-level matching algorithms satisfies the business requirements of the submitted request, and provides optimal solutions in terms of meeting a certain objective, e.g., minimizing the cost or execution time, or maximizing a certain quality measure. In this paper, we model the macro-level matching problem as a variation of the *multiple-choice knapsack problem* [8], and design a configurable, generic optimization engine, which can be repeatedly run with variations of configuration criteria in search for a business process solution best fit the need. In addition, we alternatively model the macro-level matching problem as a *multi-attribute decision making problem*. This model is particularly useful when it is not sufficient to provide an optimal solution for a single measure, but requires maximizing the total utility value of multiple business measures of interest. Our algorithm is based on *multi-attribute decision analysis*, which computes the scores of the candidate service combinations by considering their attributes values and capabilities, ranks the candidates by score, and selects services among the top-rankers.

5.1 Multiple-Choice Knapsack Algorithm

Fig.4 displays the architecture of the macro-level matching algorithm. The input to the matching algorithm is a set of Non-Dominated Match Vectors, one vector for each atomic activity in the request, which were generated by the micro-level matching algorithm. The output of the optimization engine is a set of services selected from the input, one service from each Non-Dominated Match Vector. The match engine can be customized for different business objectives and constraints as specified in another input to the engine, the Configuration.

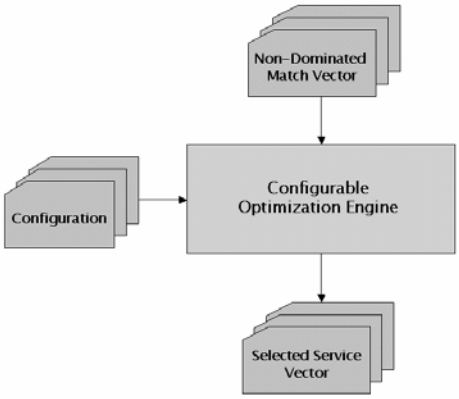

Fig.4. The macro-level matching algorithm

We model the macro-level matching problem as a variation of the *multiple-choice knapsack problem* [8]. The "multiple-choice" term in this problem designation refers

to the requirement of selecting exactly one service from each candidate list, i.e., each Non-Dominated Match Vector. For a specific example, consider the following problem: We are given a set of m business activities in our business process request, $a_1, ...,a_m$ such that activity, a_i, contains n_i candidates of Web services from the micro-level matching step. The j-th candidate for activity a_i has cost c_{ij}, and execution time t_{ij}. Given the total execution time limit T for this business process, the goal of this macro-level matching algorithm is to compose an implementation plan for this business process by selecting one and only one Web service candidate from each candidate list such that the overall cost is minimized without exceeding our total execution time limit. If we use indicator variable x_{ij} to indicate whether the j-th service from the candidate list for activity a_i was selected, we can formalize the problem with the following equations:

$$\text{minimize} \qquad C = \sum_{i=1}^{m}\sum_{j=1}^{n_i} c_{ij}x_{ij}$$

$$\text{subject to} \qquad \sum_{i=1}^{m}\sum_{j=1}^{n_i} t_{ij}x_{ij} \le T$$

$$\sum_{j=1}^{n_i} x_{ij} = 1, i = 1,..., m$$

$$x_{ij} \in \{0,1\}, \forall i, j.$$

The multiple-choice knapsack problem is known to be NP-hard [8]. It is possible to exactly solve the above problems using branch-and-bound algorithms, but because the worst-case running time of these algorithms is exponential in both the number of activities and the number of candidates on each list, branch-and-bound algorithms are often too slow to be useful. An alternative approach is to use dynamic programming techniques, and there are a number of algorithms known in this direction [8]. By using off-the-shelf software packages of optimization algorithms such as IBM's OSL [12], the given problem can be implemented in a straightforward manner. With this model in place, we can vary the problem with different objective functions and constraints. The variation of the problem can be implemented by using the Configuration component in Fig.4. For example, some processes may need to be optimized for execution time, while other measures such as cost will be treated as a constraint. In this case, the problem can be re-formulated as follows: We are given a set of m business activities, $a_1, ...,a_m$ such that activity, a_i, contains n_i candidates of Web services. The j-th candidate for activity a_i has cost c_{ij}, and execution time t_{ij}. Given the total cost budget C for this business process, the goal of this algorithm is to compose an implementation plan for this business process by selecting one and only one Web service candidate from each candidate list such that the overall execution time is minimized without exceeding our total execution time limit.

If we use indicator variable x_{ij} to indicate whether the j-th service from the candidate list for activity a_i was selected, we can formalize the problem with the following equations:

$$\text{minimize} \qquad T = \sum_{i=1}^{m}\sum_{j=1}^{n_i} t_{ij}x_{ij}$$

subject to

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i} c_{ij}x_{ij} \le C$$

$$\sum_{j=1}^{n_i} x_{ij} = 1, i = 1,..., m$$

$$x_{ij} \in \{0,1\}, \forall i, j.$$

Yet another variation of this problem is an optimization on an interesting metric such as the degree of match described in the previous section. For example, the problem can be formulated as follows: We are given a set of m business activities, $a_1, ...,a_m$ such that activity, a_i, contains n_i candidates of Web services. The j-th candidate for activity a_i has combined degree of match d_{ij}, cost c_{ij}, and execution time t_{ij}. Given the total cost budget C and the total execution time limit T for this business process, the goal of this algorithm is to compose an implementation plan for this business process by selecting one and only one Web service candidate from each candidate list such that our overall degree of match is maximized without exceeding our total cost budget and the total execution time limit.

If we use indicator variable x_{ij} to indicate whether the j-th service from the candidate list for activity a_i was selected, we can formalize the problem with the following equations:

maximize

$$D = \sum_{i=1}^{m}\sum_{j=1}^{n_i} d_{ij}x_{ij}$$

subject to

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i} c_{ij}x_{ij} \le C$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i} t_{ij}x_{ij} \le T$$

$$\sum_{j=1}^{n_i} x_{ij} = 1, i = 1,..., m$$

$$x_{ij} \in \{0,1\}, \forall i, j.$$

The degree of match of an activity can be more important than those of other activities. In such a case, the variant importance of degree of match of different activities can be reflected in the model by the assignment of weight w_i for each a_i. Then the objective model is slightly modified as follows:

maximize

$$D = \sum_{i=1}^{m} w_i \sum_{j=1}^{n_i} d_{ij}x_{ij}$$

5.2 Multi-Attribute Decision Analysis

Another approach to solving the macro-level matching problem is a *multi-attribute decision analysis*. This method is particularly useful when it is not sufficient to

provide an optimal solution for a single measure, but requires maximizing the total utility value computed by considering multiple business measures such as cost, execution time, degree of match, quality, category, and business entity relationship. The input to this algorithm is a set of n service combinations, s_1, ..., s_n such that service combination, s_i, contains m Web services, one service for each activity in the given business process. Also, each service combination has k business attributes, x_1, ..., x_k such that business attribute, x_j, is assigned a relative weight w_j (Remember the weight attribute of the <requirement> tag in Fig. 1.). Then this algorithm uses additive value function in order to compute the scores of the alternative service combinations. The system then ranks the alternative combinations by score, and selects the winning combinations among the top-rankers. The basic hypothesis of this multi-attribute decision analysis algorithm is that in any decision problem, there exists a real valued function U defined along the set of feasible alternatives, which the decision maker wishes to maximize. This function aggregates the criteria x_1, ..., x_k. Besides, individual (single-measure) utility functions $U_1(x_1)$, ..., $U_n(x_n)$ are assumed for the k different attributes. The utility function translates the value of an attribute into "utility units". The overall utility for an alternative is given by the sum of all weighted utilities of the attributes. For an outcome that has levels x_1, ..., x_k on the k attributes, the overall utility for an alternative i is given by

$$U(\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{k}) = \sum_{i=1}^{k} w_i U(x_i)$$

The alternative with the largest overall utility is the most desirable under this rule. Each utility function $U(x_i)$ assigns values of 0 and 1 to the worst and best levels on that particular objective and

$$\sum_{i=1}^{k} w_i = 1, \; w_i > 0$$

Consequently, the additive utility function also assigns values of 0 and 1 to the worst and best conceivable outcomes, respectively. A basic precondition for the additive utility function is preferential independence of all attributes, which has been the topic of many debates on multi-attribute utility theory [1, 4].

6. CONCLUDING REMARKS

In this paper, we addressed two primary problems of the Web service-based business process automation: the location of services on the basis of the capabilities of published services, and the composition of business processes on the basis of the business requirements of submitted process requests. We proposed a solution, which comprises multiple matching algorithms, a micro-level matching algorithm and a macro-level matching algorithm. The first algorithm reasons with semantic information of services and returns services that sufficiently match with an activity in the request. The second algorithm solves a variation of the multiple-choice knapsack problem that models the macro-level matching problem for optimizing a business objective and fulfilling other business constraints. In addition, we proposed a multi-attribute decision analysis algorithm, which can be used with the

optimization algorithm in a complementary fashion for a better process composition result. This algorithm is particularly useful when it requires maximizing the total utility value computed by taking multiple business measures into account. For securing information required for the execution of the matching algorithms, we explained how existing standards, UDDI and BPEL4WS, could be used and extended to specify service capabilities of services and business requirements, respectively.

7. REFERENCES

1. R. T. Clemen, Making Hard Decisions: an Introduction to Decision Analysis, Wadsworth Publishing Company, Belmont, CA, 1996.
2. DAML-S Coalition, "DAML-S: Web Service Description for the Semantic Web," Proceedings of the 1st International Semantic Web Conference, June 2002.
3. P. Doshi, R. Goodwin, R. Akkiraju, S. Roeder, "A Flexible Parameterized Semantic Matching Engine," IBM Research Report, 2002.
4. W. Edwards, "How to Use Multi-Attribute Utility Measurement for Social Decision Making," IEEE Transactions on Systems, Man, and Cybernetics SMC, vol. 7:326-340, 1977.
5. D. Fensel, I. Horrocks, F. van Harmelen, D. L. McGuinness, and P. F. Pate, "OIL: An Ontology Infrastructure for the Semantic Web," IEEE Intelligent Systems, Vol. 16, No. 2, 2001.
6. J. Hendler, and D. L. McGuinness, "DARPA Agent Markup Language," IEEE Intelligent Systems, Vol. 15, No. 6, 2001.
7. F. Leymann, D. Roller, and M. T. Schmidt, "Web Services and Business Process Management," IBM Systems Journal, Vol. 41, No. 2, 2002.
8. S. Martello, and P. Toth, Knapsack Problems, Chichester, New York, John Wiley & Sons, 1990.
9. M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic Matching of Web Services Capabilities," Proceedings of the 1st International Semantic Web Conference, June 2002.
10. M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Importing the Semantic Web in UDDI," Workshop on Web Services, e-Business, and the Semantic Web: Foundations, Models, Architecture, Engineering and Applications, Toronto, Ontario, Canada, May 2002.
11. S. R. Ponnekanti, and A. Fox, "SWORD: A Developer Toolkit for Web Service Composition," Proceedings of the 11th World Wide Web Conference, Honolulu, Hawaii, May 7-11, 2002
12. IBM Optimization Solutions and Library, http://www-3.ibm.com/software/data/bi/osl/index.html.
13. "BPWS4J," IBM Corporation, http://alphaworks.ibm.com/tech/bpws4j, August 2002.
14. "Business Process Execution Language for Web Services, Version 1.0," BEA Systems, IBM Corporation, and Microsoft Corporation, Inc., http://www.ibm.com/developerworks/library/ws-bpel/, July 2002.