Ontology-Based Business Process Customization for Composite Web Services

Qianhui Liang, Xindong Wu, E. K. Park, Taghi M. Khoshgoftaar, *Member, IEEE*, and Chi-Hung Chi

***Abstract*—A key goal of the Semantic Web is to shift social interaction patterns from a producer-centric paradigm to a consumer-centric one. Treating customers as the most valuable assets and making the business models work better for them are at the core of building successful consumer-centric business models. It follows that customizing business processes constitutes a major concern in the realm of a knowledge-pull-based human semantic Web. This paper conceptualizes the customization of service-based business processes leveraging the existing knowledge of Web services and business processes. We represent this conceptualization as a new Extensible Markup Language (XML) markup language Web Ontology Language-Business Process Customization (OWL-BPC), based on the de facto semantic markup language for Web-based information [Web Ontology Language (OWL)]. Furthermore, we report a framework, built on OWL-BPC, for customizing service-based business processes, which supports customization detection and enactment. Customization detection is enabled by a business-goal analysis, and customization enactment is enabled via event–condition–action rule inference. Our solution and framework have the following capabilities in dealing with inconsistencies and misalignments in business process interactions: 1) resolve semantic mismatch of process parameters; 2) handle behavioral mismatches which may or may not be compatible; and 3) process misaligned rendezvous requirements. Such capabilities are applicable to business processes with heterogeneous domain ontology. We present an architectural description of the implementation and a walk-through of an example of solving a customization problem as a validation of the proposed approach.**

***Index Terms*—Business process customization, composite Web services, consumer centric, goal analysis, ontology-based, semantics.**

Manuscript received November 15, 2008; revised April 23, 2009; accepted May 30, 2009. Date of publication April 29, 2011; date of current version June 21, 2011. This work was supported in part by the 973 Program of China under Award 2009CB326203 and in part by the National Natural Science Foundation of China under Grant 60828005. This paper was recommended by Editor W. Pedrycz.

Q. Liang was with the School of Information Systems, Singapore Management University, Singapore 178902. She is now with HP Labs Singapore, Fusionopolis, Singapore 138632.

X. Wu is with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China, and also with the Department of Computer Science, University of Vermont, Burlington, VT 05405 USA (e-mail: xwu@cs.uvm.edu).

E. K. Park is with the Department of Computer Science Electrical Engineering, University of Missouri at Kansas City, Kansas City, MO 64110 USA.

T. M. Khoshgoftaar is with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431 USA.

C.-H. Chi is with the School of Software, Tsinghua University, Beijing 100084, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCA.2011.2132710

I. Introduction

THE VISION of the Semantic Web has been evolving in recent years as a knowledge-based framework aimed at crossing the chasm from the current Web of unstructured information resources to a Web equipped with metadata and oriented to the delegation of tasks to software agents [1]. Meanwhile, such a knowledge framework offered by the Semantic Web should support a shift of social interaction patterns from a producer-centric paradigm to a consumer-centric one [1]. The topic of human semantic Web has thus emerged to meet this need. In the domain of business management, the knowledge-pull-based human semantic Web by nature provides a solution that customer-centric businesses are looking for in general to maximize their possible business profit and success.

In consumer-centric business modeling, an important task is to develop semantic-based frameworks that make a business process easier for consumers to do business with. This will demand a measure of business process customization. Automating this task has been made easier by service-oriented architecture. In a service-based business process, each activity in the process is treated as a message exchange with an operation supported by some Web service. The process itself can then be described as a composition of Web services using a standardized language such as the Business Process Execution Language (BPEL) [2] or Web Ontology Language for Web Services (OWL-S) [3]. A service-based business process by nature allows more agility in the process due to loose coupling, service reuse, and dynamic binding.

In a service-based business process, customization may be enabled by automatically adapting the process to match the business partner's practice indicated by their business processes. Such practice includes service interface specifications, Web Ontology Language (OWL)-service profiles, process models, and grounding. We would like to point out that, in this paper, we focus on the business scenarios where the business processes can be supported by dynamic and automatic service composition. In such scenarios, the instantiation of business processes allows a certain degree of flexibility in selecting business partners and adjusting the process parameters for the partners. In other words, here, we will only discuss service-based business processes where the idea of automation of adaptation is applicable. We do not address many other circumstances where the choreography between processes must be well defined before execution in order to avoid any unacceptable conflict and loss.

We refer to the *customization of business process* as a machine-enabled capability of adapting a business process of

a company according to the process of the customer or business partner that it is collaborating with. A generic solution to this issue has not been proposed so far due to reasons, such as lacking a proper definition of a body of knowledge for the customization of business processes and lacking its standardized representation and rationales of inference.

Our research efforts reported in this paper seek to establish a generic solution to the problem of customization of service-based processes from the following three aspects. First, we present a conceptualization definition for business process customization that leverages existing knowledge of business processes and Web services. For such a definition, we have developed a vocabulary of business process customization for modeling the meanings of concepts and the relationships between these concepts. Second, we present a representation of this conceptualization in a new Extensible Markup Language (XML) markup language, based on the de facto semantic markup language for Web-based information, i.e., OWL [4]. We name the conceptualization OWL-BPC for OWL on Business Process Customization. Third, we present a framework for customizing service-based business processes based on OWL-BPC by first identifying the possible causes of discrepancies/inconsistencies between collaborating business processes (*customization detection*) and then taking suitable remedial actions (*customization enactment*). Our solution and framework can do the following: 1) deal with semantic inconsistencies like semantic mismatching of process parameters; 2) resolve behavioral mismatches between services which may or may not be compatible; and 3) address misaligned rendezvous requirements. Such capacities are applicable to business processes with heterogeneous domain ontology.

The remainder of this paper is organized as follows. We review related work in Section II. The design of the conceptualization of business process customization and its representation are provided in Section III. In Section IV, we discuss the rational for detecting and enacting business process customization. Customization detection is based on event expectation by business goal analysis and process parameter comparisons. The customization enactment mechanism uses an event-trigger-rule engine, which is to be described in the same section. In Section V, we give an architectural description of the implementation of the proposed approach. We also introduce an example business process customization problem and walk through how the framework solves the problem as a validation of our approach. Section VI concludes this paper.

II. Related Work

In the past ten years, there have been a number of efforts to apply formal semantics to the Web, including special interest groups such as the Special Interest Group on Semantic Web and Information Systems of the Association for Information Systems (AIS, see http://www.aisnet.org [1]) and several World Wide Web recommendations such as the OWL [4]. The Semantic Web has been a vision that helps shape the future directions of many research topics in computer science and information systems. For example, there is research work dedicated to a study of formal semantics [5] and analysis for Web Services Business Process Execution Language (WS-BPEL) [6]. The Semantic Web also brings benefits to a range of applications, including manufacturing, e-learning and digital libraries [7], electronic government, and biomedical information search, e.g., [8]. It has been proven to facilitate interdisciplinary efforts.

To follow a user-centered philosophy throughout the software, product life cycle has been a need in many application domains. The main idea is that what the end users need, want, and are constrained from doing, given existing interfaces, documents, or products, need extensive attention. This philosophy entails many related issues and efforts regarding a piece of software. One issue is that the design process must be user centered. The testing must be user centered, and the interface of the final software product must be user centered [9]. Another issue is that software in different domains may demonstrate different user-centered requirements. For example, systems designed to serve the electronic business needs of users have to cater to users with a wide range of capacities and background. Therefore, in order to serve a certain group of users better, a system needs a certain "helper" [10] agent, or negotiation mechanism [11] with a customized design to be added to the original system. As another example, environment software and medical visualization systems may need advanced interaction techniques or 2-D and 3-D virtual reality in order to be user centered [11], [13].

A branch of research efforts on semantic Web seeks to integrate a machine-understandable knowledge framework with the user-centric human factors. It is attracting more and more attention as the vision of the Semantic Web has received a wide acceptance in many application domains. This so-called "human semantic Web," in addition to the focus of semantic Web technology in its more original sense, pays extra attention to human factors in the process of compiling, presenting, processing, and applying the knowledge framework. Its aim is to address the challenge of taking care of the individual needs of each consumer and partner. Most recent advances on human semantic Web, from the knowledge presentation and processing level, have laid foundations of a knowledge-pull paradigm. Such research work mainly focuses on the tools and languages for metadata and ontology extraction, reasoning-based knowledge discovery, semantic Web applications including search and information retrieval, grid computing, and enterprise information integration [14], [15].

At the beginning of this century, researchers in the Web service community provoked the discussion on applying a semantic Web vision to the Web service technology and on how it helps make Web services better understood and used by computer software. The de facto semantic markup language for Web services is OWL-S [3], which models the semantics of a service-based business process. Researchers have also taken into consideration the following as required semantics of service-based business processes: run-time status and event processing, intraorganizational and interorganizational security and access control [16], and configurability and availability [17], [18]. McIlraith *et al.* [19] developed markups of Web services that benefit Web service discovery, execution, and composition. The semantics of service requests are also among the research issues attracting much attention. In [20], the authors

pointed out that BPEL lacks the flexibility in responding to the unforeseen situation. They reported a request language called XML Service Request Language (XSRL) that integrates artificial intelligence planning and constraint satisfaction techniques and a planning architecture that accepts requests in XSRL. In [21], the authors presented a semantic model for modeling requests of composite services and a mechanism of incorporating the presentation of nonfunctional requirements into service requests for better facilitating service composition.

A few papers have discussed business process customization and optimization, e.g., [22] and [23]. However, most of these discussions study the customization of a specific system from the application programming interface and component level. No efforts have focused on the study of customizing a business process in general and from a knowledge presentation and process level. Business process compatibility issues have also been studied. Such work mainly focuses on the behavioral aspect of the business processes and does not cover other aspects of possible inconsistencies [22]. We conceptualize the problem of business process customization, present an ontology (OWL-BPC) for it, and report a framework. The semantic approach that we have taken relies on a knowledge framework that is supported by a rule-based inference. It has the advantage of a comprehensive representation as well as a rigorous and automatic processing mechanism. Moreover, our study also includes enactment upon any possible inconsistencies to adapt the process to the need of the process of the partner or customer.

III. Conceptualization of Service-Based Process Customization

A generic solution to business process customization is among the major forces that will hopefully lead to the realization of the vision of human semantic Web in a business setting. A general solution to the problem can be described as follows: A business process needs to communicate with another by calling its Web services and exchanging XML messages with it. In order to meet this need, it has to observe its partner within the scope of the relevant collaboration. Then, it customizes its own process accordingly in order to ensure a smooth collaboration. To allow the customization of processes, among the first things required are the following: 1) the establishment of a conceptualization of business process customization together with the associated markup language; 2) the creation of an ontology for this specific problem; 3) the instantiation of this ontology in describing a specific instance of customization; and 4) the application of efficient reasoning to automatically determine the meaning of certain customization instructions and actions. With the proposed idea of "process customization," processes do not have to merely follow the fixed business rules and/or templates in performing required tasks in order to achieve a business goal but are allowed to effect changes in themselves, such as a change in the process flow or an upgrade of the delivery means. Process customization should enable users to build, fit, or alter a process for making the life of its business partners easier in doing business automatically. A service-based business process provides a standard foundation on which to build our customization. In the rest of this paper, we base our work on OWL. In general, both the conceptualization and methodology of this customization framework are applicable to any service-based business process.

A. Motivation and Advantages

Here, we present the motivating tasks of conceptualizing the customization of service-based processes. For each take, we also analyze the potential advantage of using such a customization ontology, like the proposed OWL-BPC.

1) Automatic Customization Detection: Automatic customization detection is an automated process of detecting possible elements or variables of a business process that need to be especially treated in order to suit the requirement of the other process(es). We refer to the business process to be customized as the *primary business process* or PBP and those that it collaborates with as *secondary business process(es)* or SBP(s). For example, the original design of the workflow corresponding to the PBP introduces a flow activity for the concurrent execution of multiple activities. These activities may be actually designed to invoke the Web services in the SBP, which are orchestrated to execute in a sequential order. Currently, the business process modeler needs to investigate the PBP line by line in order to find the location of the corresponding "flow activities." In this aspect, the advantage of having a markup language is that the information necessary for understanding where and how to perform customization can be described in a machine-understandable format and analyzed by a machine. Using an analytic tool of the ontology, software agents can be designed to automatically detect the customization needs according to the analysis results.

2) Automatic Customization Enactment: Automatic customization enactment is an automated process of taking actions to perform the customization on the PBP according to the detected customization spots and the automatic reasoning on the customization conceptualization knowledge framework. Let us use the same example of PBP with a flow construct and SBP with sequential activities. Currently, the modeler of the business process, upon understanding the necessary customization requirement, needs to manually locate the flow construct, move the concurrent activities originally under the flow construct out and add them in a sequential order to the process, and then remove the flow construct. In this aspect, the advantage of a language like OWL-BPC is to allow software agents to take actions of customization on the specific constructs of the process automatically.

In the next section (Section III-B), we will describe the metadata for semantic Web that the ontology is designed to support in terms of data and knowledge retrieval. The detailed function of the ontology is discussed in Section III-C.

B. Metadata and Reasoning

As we have explained, the kind of metadata that we capture are the discrepancies between two interacting business processes. We choose the Resource Description Framework (RDF) [24] as the presentation format of the metadata in support of describing and interchanging knowledge of customizing service-based processes. In particular, we apply the concepts

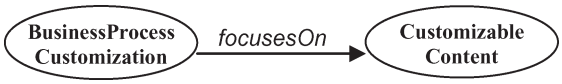

Fig. 1. Concepts and relations of metadata.

```
- <rdf:RDF xml:base="http://www.mysmu.edu/faculty/althealiang/ontologies/bpc/#">
  + <!---->
  - <owl:Ontology rdf:about="">
    - <rdfs:comment>
        Top Level OWL Ontology for Business Process Customization Author: Qianhui Liang
      </rdfs:comment>
    </owl:Ontology>
  + <owl:Class rdf:ID="BusinessProcessCustomization"></owl:Class>
    <!-- Owner of Business Customization -->
  + <owl:DatatypeProperty rdf:ID="owner"></owl:DatatypeProperty>
  + <!---->
  + <owl:DatatypeProperty rdf:ID="primaryBusinessProcess"></owl:DatatypeProperty>
  + <!---->
  + <owl:DatatypeProperty rdf:ID="secondaryBusinessProcess"></owl:DatatypeProperty>
    <!-- Customizable Content -->
  - <owl:DatatypeProperty rdf:ID="focusesOn">
      <rdfs:domain rdf:resource="#BusinessProcessCustomization"/>
      <rdfs:range rdf:resource="#CustomizableContent"/>
    </owl:DatatypeProperty>
    <!-- Customizable Function -->
  - <owl:DatatypeProperty rdf:ID="conducts">
      <rdfs:domain rdf:resource="#BusinessProcessCustomization"/>
      <rdfs:range rdf:resource="#CustomizationFunction"/>
    </owl:DatatypeProperty>
    <!-- For Semantic Misalignment -->
  - <owl:Class rdf:ID="Range">
```

Fig. 2. Metadata in OWL.

described in OWL-S [3] regarding service profiles and process models as the basis to build the metadata of the objects in this particular problem domain. We use resource descriptions to define concepts in customization and relations between them using OWL-S statements. We connect these statements to form a semantic network.

Using the RDF, each concept in the metadata is modeled as a resource with a Uniform Resource Identifier (URI). For example, the root concept of customization can be written as a class with the URI "http://www.serviceprocess.org/BusinesProcessCustomization.owl#BusinessProcessCustomization" in the RDF. A basic statement in the RDF is a <subject, property, object> triple, which models a relation in the metadata. A relation is between two concepts, where one corresponds to the subject of the relation and the other the object. They are connected by a property or a "predicate." For example, in Fig. 1, we show a graphical representation of the relation between an object of class "#BusinessProcessCustomization" and an object of "#CustomizableContent," which depicts a "focusesOn" property. In other words, it prescribes that customization must focus on some certain content. Its definition in the RDF is shown in Fig. 2.

A more detailed description of the metadata modeled by the ontology is given in Section III-C.

Inference on the metadata of service-based process customization is important, mainly for answering the queries where implied knowledge of customization is needed. Such knowledge has to be derived from explicitly known facts or existing knowledge. For this purpose, we may use existing forward-chaining or backward-chaining rule engines for inference. Examples of such engines include Java Expert System Shell (Jess) [25] and XSB [26] (a logic programming and deductive database system). A final thing that we need is a "bridge" that translates the formal semantics in processing information encoded in OWL-S. There have been a few such open-source systems that we can choose from, including

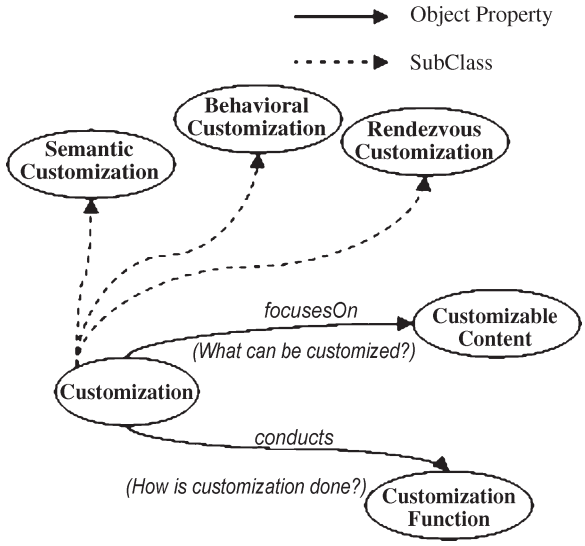

Fig. 3. Top level customization ontology.

SWRLJessTab [27], OWL Inference Engine in Flora-2 (F-OWL) [28], and Pellet [29]. With the rule engine and the "bridge," we can process customization scenarios that need inference. A good example is the following: The sequence of these two tasks in our process is different from that of our partner. The reasoner will detect that "CustomizableSequence" is a subclass of "CustomizableControlConstruct" through inference and instructs to apply all the customization actions defined for the class of "CustomizableControlConstruct."

C. OWL-BPC Ontology

The top level of the customization ontology is centered on the essential types of knowledge of the customization of service-based business processes. Each type of knowledge answers one question in the following set of two questions that characterize all that are needed to know about business process customization.

1) *What may be customized?* The answer to this question is given in content, which is used to identify the elements in the business process that possibly need to be taken special care of. To capture this knowledge, each instance of class Customization needs to present a CustomizableContent.
2) *How is customization done?* The answer to this question is given in a function, which is used to describe the required actions on particular parts of the business process. To capture this knowledge, each instance of class Customization needs to present a CustomizationFunction.

The OWL class Customization provides an organizational blueprint of the customization ontology in Fig. 3. The class of CustomizableContent is shown in Fig. 4.

IV. Customization Detection and Enactment

Based on the OWL-BPC ontology presented earlier, a machine is able to reason against it for detecting the necessary customizations and producing the instructions of doing the customizations. In this section, we present OWL-BPC-based

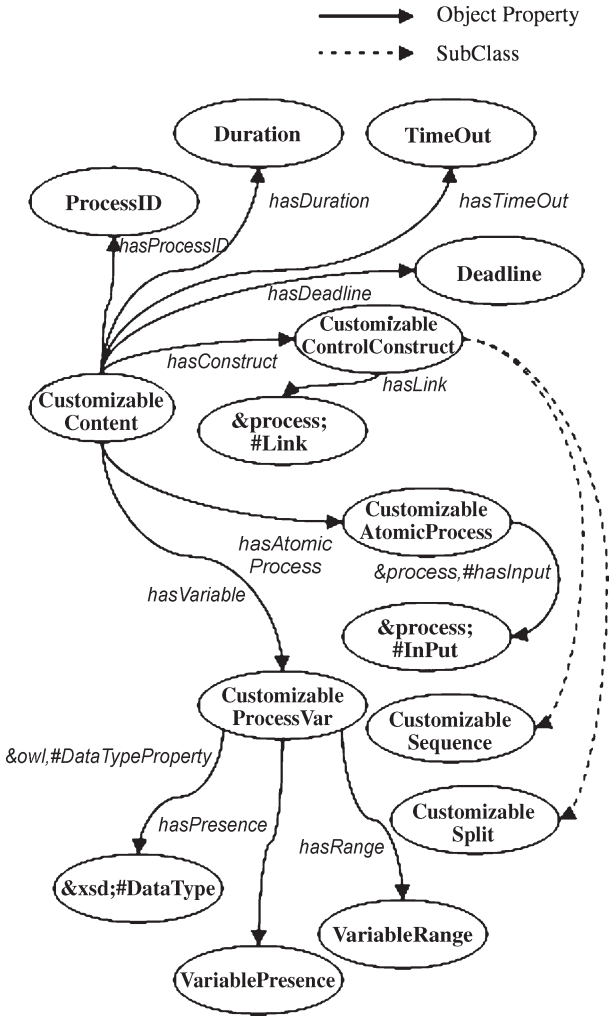

Fig. 4. Selected classes and properties of customizable content.

business process customization detection, instrumentation, and enactment.

A. Heterogeneous Ontology Matching—OnExCat

Two important elements that enable an automatic way of processing business process customization are upper ontology of a business process and upper ontology of the process customization. Both upper ontologies have to interface with domain ontology during the inference by an inference engine.

The upper ontology of services, such as OWL-S, refers to the types of knowledge about a service such as what the service provides for prospective clients and how it is used [2]. For a service-based business process, the classes and properties defined within the upper ontology of services directly convey all important information of a business process. Together with the structural constructs of the process, they facilitate the automation of executing the business process. Furthermore, with an upper ontology on business process customization like OWL-BPC, automatic customization of business processes is made possible due to the presence of a machine-understandable knowledge framework on what, when, and how customization should be performed.

The issue of using homogeneous domain ontology in process descriptions needs to be handled if business process customization is to be automated. The choice of domain ontology is left open to the business process and service owners. As a result, they may choose a different domain ontology, and this makes understanding and conveying knowledge difficult.

Our framework of business process customization handles the heterogeneity of domain ontologies in semantic business process descriptions by categorizing the terms extracted from their corresponding process descriptions. Only those terms that correspond to the values of the key classes or key properties in a given upper process ontology are categorized. The categorization function is provided by an ontology extraction and categorization tool called OnExCat that we have developed earlier [30]. OnExCat processes new domain terms in the service-based process description documents statistically and determines the semantic similarity of ontology instances by a probabilistic categorization measurement that incorporates relationship among the terms in the upper ontology for services. Now, we briefly review the tool.

In OnExCat, the categorization of ontology instances is treated as a term category search problem. The techniques in the document classification of information retrieval are used. Just as document classification classifies a large collection of documents, the categorization of domain ontology instances classifies the terms in the textual or semistructured service-based business process descriptions to the concepts defined in given ontologies or thesaurus.

An improved version of two complementary categorization methodologies: 1) Single Random Variable with Multiple Values (SVMV) [31] and 2) co-occurrence are applied. Both technologies have been improved for processing Web service descriptions by incorporating service semantic information obtained during instance extraction.

1) Probabilistic Text Categorization: SVMV, as a model of probabilistic text categorization, defines a better way to derive the probability that a document d is categorized into a category c, $p(c|d)$. This model, when applied to term categorization, provides a calculation of the similarities of the terms' linguistic patterns, assuming that words used in similar grammatical patterns are similar. Since the grammar of languages that mark up semistructured documents bears semantics just like a grammar in natural languages, relationships in the service description language can be exploited to discover term semantic similarities. An improvement to SVMV is to introduce patterns that pertain to service-based process description documents. In particular, the following relationships for OWL-S documents are considered:

- serviceName-hasParameter, serviceName-hasInput, serviceName-hasOutput, serviceName-hasPrecondition, serviceName-hasResult, serviceName-serviceParameterName, serviceName-serviceProduct, serviceName-wsdlInput, serviceName-wsdlOutput for OWL-S.

Each ontology instance of a part or an element is considered similar to a noun in a natural language. These terms are referred to as nounlike terms. Each ontology instance of an operation or a service is considered similar to a verb in a natural language.

These terms are referred to as verblike terms. A nounlike term will have a set of verblike terms with which it appears to be related. In OWL-S, each serviceName has its related hasInput, hasOutput, hasResult, hasParameter, etc. Therefore, each term can be represented by a set of the terms that co-occur with it.

The adapted probability calculations and estimations of the SVMV model can be described as follows. A probability value that is calculated a term t (presented as a part, an element of a part, a parameter, a product, a result, etc. in a Web service description document) is categorized into an ontology concept c_i. In (1), r_i refers to a verblike term with which t co-occurs

$$P(c|t) = \sum_{r_i} P(c|t, R = r_i)P(R = r_i|t). \quad (1)$$

Assuming conditional independence between c and $R = r_i$ given t, by applying Bayes' theorem to $P(c|R = r_i)$, (2) is also true

$$P(c|t) = P(c) \sum_{r_i} \frac{P(R = r_i|c)P(R = r_i|t)}{P(R = r_i)}. \quad (2)$$

As shown in (2), the probability that a term belongs to category c can be calculated using the following three probabilities: 1) the probability that a randomly selected verblike term co-occurring with a nounlike term is r_i, given that the nounlike term is an instance of ontology concept c, i.e., $P(R = r_i|c)$; 2) the probability that a randomly selected verblike term co-occurring with a term t is r_i, i.e., $P(R = r_i|t)$; and 3) the prior probability that a randomly selected term co-occurring with a randomly selected nounlike term is r_i, i.e., $P(R = r_i)$. Terms that are assigned to the same category form a single cluster. New terms will be compared to each of these clusters to get similarity scores. The instance is then categorized with the cluster with the highest score.

With this probabilistic model, nounlike instances are assigned to the most probable category, assuming the conditional independence between the category and the term that the nounlike term with which it co-occurs. Domain ontology instances that are assigned to the same category form a single cluster. Any new ontology instance will be compared to each of the clusters to get similarity scores. The instance is then categorized with the cluster with the highest score.

2) Term Co-Occurrence Analysis: The clustering analysis used is asymmetric, which is different from other term co-occurrence algorithms such as the cosine analysis. Therefore, the fact that there exists a link from one term to the other does not necessarily mean that the opposite is true. Even if there are links from terms A to B and also from B to A, their weights are usually different. The cluster function defines the term similarity weights from t_j to t_k and from t_k to t_j by the combined weight of both terms t_j and t_k in process description document i and the inverse document frequency. Both the combined weights and the inverse document frequency are then determined by the co-occurrences of the two terms in each process description document and the number of description documents that the two terms co-occur. Due to the limited number of collected description documents, a different definition of the similarity form is used. In particular, instead of indexing description documents by i, the definition is shown in

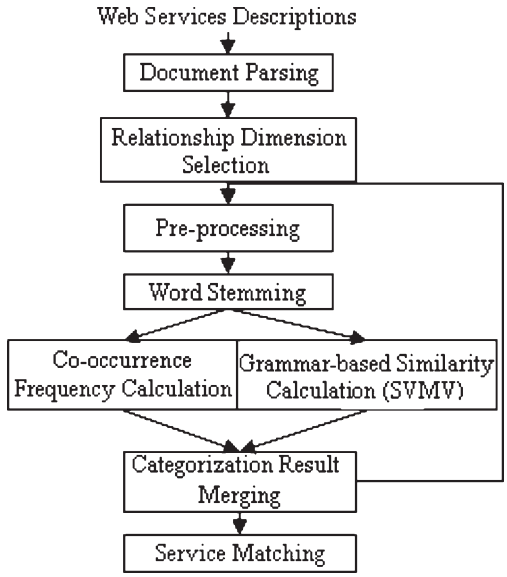

Fig. 5. Process of ontology matching.

$$Similarity(t_j, t_k) = \frac{weighting\ factor(t_k) \times \sum_{i=1}^{m} c_{ijk}}{\sum_{i=1}^{m} c_{ij}}. \quad (3)$$

The similarity from t_j to t_k, or *Similarity* (t_j, t_k), is defined over the following three factors: 1) the weighting factor of t_k; 2) the summation of the combined term weights of t_j and t_k, c_{ijk}, over all M categories in the description document collection; and 3) the summation of the combined weights of t_j over all M categories, as in (3). All the three factors can be calculated as the number of categories that the two terms co-occur and the number of their co-occurrences in a particular category. Fig. 5 shows the process of domain ontology matching.

Both SVMV and the co-occurrence technique are used in OnExCat, and the results are integrated by a simple voting scheme. The experiments on OnExCat have shown a correct categorization rate of over 85% [30].

B. Customization Detection

1) Scoping Customizable Contents: The first step in detecting necessary customizations is to separate customizable contents from the rest of the contents of a service-based business process. In short, we rely on *a goal analysis* to scope the customizable contents. Given two interacting and collaborating business processes and their goal taxonomies, goal analysis is to perform reasoning against the goal taxonomies in order to identify all matched functions (atomic processes in OWL-S) that may be customized for business process customization for the interactions of them. The details are explained in the following.

Consider an interactive business collaboration, e.g., supply chain, between two or more partners each of which follows their individual business process. Any business process is to achieve some goal, which is set beforehand, by a partial order of activities. Therefore, the goal of a process is usually identified and described in a simple and not procedural way without having to refer to the detailed process. We reasonably assume that the goals of the business processes of collaborating partners are provided. If the goals of the interacting parties match perfectly, the interactive collaboration will be successfully brought to an end. In other words, there will only be a sequence of normal business events informing the achievement of certain goals, e.g., shipmentConfirmation. On the other hand, if there is any inconsistency between the goals, due to internal or external reasons, the process will emit business events of exceptions, e.g., shipmentDelay. This will happen at the moment when the goals of the interacting partners start to drift apart due to semantic discrepancy, behavioral misalignment, and rendezvous misalignment. Such an event indicates discrepancies and misalignments and, thus, the scope of customizable contents. It can be used to trigger adaptation to possibly adjust the process and then continue the execution.

Based on the aforementioned assumption, we rely on a static analysis on the goals of interacting partners in an interactive collaboration in order to suggest possibly appearing events pertinent to the business process. The approach taken is to compare goals of the partners semantically by traversing the ontology graph. The goal analysis for scoping customizable contents can be briefly described as an algorithm listed in Algorithm 1.

Algorithm 1

1: ScopingCustomizableContents

$$((G_0, O_0, bp_0), (G_1, O_1, bp_1), \ldots, (G_m, O_m, bp_m), \ldots (G_M, O_M, bp_M))$$

bp_0	The PBP, which is to be customized.
G_0	The goal taxonomy of bp_0.
O_0	The domain ontology of bp_0.
$bp_m (m! = 0)$	The mth SBP that bp_0 collaborates with.
$G_m (m! = 0)$	The goal taxonomy of bp_m.
$O_m (m! = 0)$	The domain taxonomy of bp_m.
$M + 1$	is the total number of processes in consideration.

2: $p = 0$;
3: **L0**: For bp_p and each business process b_p in the set of business processes $\{bp_m\}$, which bp_p collaborates with, {
4: $i = j = k = 0$;
5: Identify the goal of the business process in the goal taxonomy G_m;
6: Assign the goal to g;
7: **L1**: Reduce g to a set of subgoals $\{sg_i\}$ by reductions defined in G_m;
8: for each subgoal sg in $\{sg_i\}${
9: if sg is not implementable{
10: $g = sg$;
11: Go to **L1**;}
12: else
13: for each objective o returned by $obj(sg)$, which implements {
14: /*$obj(sg)$, defined in G_m, returns a list of operational objectives that implement sg*/
15: Append o to $\{o_j\}$;}}
16: for each objective o in $\{o_j\}${
17: for each action a returned by $esb(o)$, which establishes o {
18: /*$esb(o)$, defined in G_m, returns a list of actions that establish o* /
19: Append a to $\{a_k\}$;}}
20: Run *OnExCat* on the actions in $\{a_k\}$;
21: Infer on $\{a_k\}$ to identify actions dependent on each other;
22: Record a set of tuples made of actions $\Gamma^p = \{t | t = (a^0_{k0}, a^1_{k0}, \ldots, a^q_{ki}, \ldots)\}$, where a^q_{ki} is an action in the collaborating processes that is relevant to action a_{ki} of bp_p (q enumerates through the actions that are relevant to a particular action a_{ki}).
23: if $p < M$ {
24: $p = p + 1$
25: Go to **L0**;}
26: Return $\{\Gamma^m\}$

The input to the algorithm is a number of collaborating business processes together with their domain ontology and their goal taxonomies, which may be heterogeneous. The output of the algorithm is the related or dependent actions in their goal taxonomies being identified and clustered together.

The algorithm repeats the following steps based on each partner's business process. First, it identifies the goal of the business process in the goal taxonomy, which is part of some ontology. The goal is reduced to subgoals iteratively by reductions defined in the same ontology. Leaf subgoals are implemented by some operational objectives. These operational objectives can be established by actions. Once the above is done for each business process, the algorithm applies term categorization provided by OnExCat to the actions of all business processes. Based on the categorization result, the algorithm uses an inference engine to short-list all actions that are dependent on each other. The purpose of the last two steps is the following: Given one particular action of a business process, it identifies all the actions in the other business processes that are related to (either dependent or depending on) it. After the execution of this algorithm, for each action in any of the given set of business processes, its related actions of the remaining processes can be easily retrieved.

C. Customization Instrumentation

The next step for detecting customization is to compare interacting actions of the collaborating business processes and identify which parts of the actions need customization and what kind of customization is needed. The process can be described as the following.

TABLE I
CUSTOMIZATION EVENTS

Event Name	Event Parameters	Event Datatype	Event Description
None-Corresponding-Content	ContentID ContentID_BE ContentID_AF	XPath String XPath String XPath String	A customizable content in PBP does not have a corresponding content in SBP.
Different-Order	ContentID1 ContentID2	XPath String XPath String	A customizable content in PBP appears in a different order (in relevance to another) than that in SBP.
Concurrent-Serialized-Conflict	ContentID ConstructID	XPath String XPath String	A customizable content in a concurrent construct in PBP appears to be in a sequential construct in SBP.
Null-Content	SBPContentID ContentID_BE ContentID_AF	XPath String XPath String	A content in SBP does not have a required corresponding content in PBP.
Different-Variable-Type	VariableID SBPType PBPType	XPath String XSD DataType XSD DataType	A variable has a different type in PBP from that in SBP.
Incompatible-Variable-Type	VariableID SBPType PBPType	XPath String XSD DataType XSD DataType	A variable in PBP has an incompatible (primary) type with that in SBP.
Inconsistent-Variable-Range	VariableID SBPRange PBPRange	XPath String Data Interval Data Interval	The range of a variable in PBP is inconsistent with that in SBP.
Inconsistent-Naming	ContentID SBPName PBPName	XPath String XML QName XML QName	The naming of the corresponding parameters in PBP and SBP depends on different ontologies.

The input is a number of collaborating business processes together with their domain ontology, their goal taxonomies, and the set of tuples, each associating an action with all relevant actions of the other business processes. The output of the algorithm is a collection of records of the customizable contents in the PBP that have the following characteristics: 1) have missing contacting contents in the SBPs; 2) have a different sequential order (in relation to another customizable content of the PBP) than the corresponding contents in the SBPs; 3) are in a concurrent construct which are sequential in an SBP; and 4) do not exist and are needed in interactions from SBPs.

It goes through a very similar process for each action of the PBP and those of the SBPs. For each action of the PBP, it repeats the following steps. It first runs OnExCat to retrieve all component processes of the PBP whose service name is categorized together with the action. For each of such component process, based on OWL-BPC, it finds a customizable component of the component process. Again using the onExCat, it retrieves its relevant (either depending or dependent) components in SBPs, which are categorized together with both the component in the PBP and the associated action of its owning SBP. One such relevant component is retrieved for each SBP. According to different symptoms observed from the SBP, the algorithm places a record, including that a required contact in SBP is missing, that the concurrent structure is actually a sequential one in the SBP, that a different order in the SBP is discovered, and that there are semantic or rendezvous discrepancies. It then goes to find the next customizable content in the component process and repeats the process until all customizable contents are exhausted. It continues on to do the same for the next component process and their customizable contents.

The process of finding dependent or depending component processes that may require the customization of the PBP is similar to the above. Due to space limit, we did not list the pseudo code here.

D. ECA-Based Customization Enactment

Record output is used to directly arrange various customization activities as defined in the CustomizationFunction ontology. The enactment of customization is based on a model of event–condition–action (ECA) rules [32]. ECA rules, also referred to as active rules, are widely used in active database systems where the systems wait until a predefined situation (a composite event pattern) is matched to trigger an action in the databases. For business process customization, we expect the discrepancies and misalignments of collaborating business processes to cause occurrences of exceptional business events. We also expect rules to be executed to enact appropriate customizations in response to the discrepancies and misalignments. In this case, ECA is a suitable model for enacting customizations of the BPB.

1) Events: Events in ECA are of our interest if they have happened or will happen. In the context of the problem of business process customization, events are always related to the customizable contents of the business process. For example, the collection of records of the customizable content output can be directly converted to the events that occur. Events will be passed to the event server for processing. If the event matches the event part of a rule, the rule will be fired.

Events may be designed to carry additional information, referred to as parameters, which the rule may use in its execution. This is especially important in our case. For example, customizing a content is required to be taken at the same time that another content is customized. FunctionTime defined in the CustomizationFunction ontology needs to be passed to the rule so that the timing of customization is guaranteed.

Following the conceptualization of Customization-Function defined in OWL-BPC, we list all the possible events together with their parameters that we will process for the purpose of business process customization in Table I.

TABLE II
RULES AND THEIR ACTIONS

Rule Name	Description of RHS of the rule
AddContent	Since the SBP requires an additional content to be present, a customization that adds a corresponding content in the PBP is conducted.
ReorderContent	Since the SBP requires a different sequential order of the activities, a customization that re-orders the contents in the PBP is conducted.
SerializationContent	Since the SBP requires a sequential execution of the activities, a customization that serializes the content is performed.

2) Rules: The processing of events mainly involves inferences by an inference engine in a production system. The inference engine is designed to infer on rules in a rule base, given the occurrence of an event. Each rule has a left-hand side (LHS) and a right-hand side (RHS). The LHS of the rule contains patterns of events, and the RHS represents the actions that will be taken once the event that occurred matches any pattern in the LHS of the same rule. A rule is assigned a name and is designed to take in parameter values. An ECA also provides an option to conditionally specify an action. When the rule is invoked upon the occurrence of some event, the condition will be evaluated. True evaluations lead to the execution of the RHS. Otherwise, no action will be taken.

We show three examples of the rules that have been implemented by us in Table II.

E. Consistency and Equivalence

After the enactment of customization is completed, both the potential structural consistency of the adapted process and the functional equivalence between the original process and the adapted one need to be checked. This is to keep the process customization from being arbitrary.

Structural consistency is used to refer to a modified workflow schemata description as describing a legal workflow [33]. Similarly, here, by structural consistency, we refer to the structural validity of the adapted business process. Structural consistency only deals the static part of the workflow control patterns [34] and workflow activities. Functional equivalence means that the process after customization remains the same function as before. We have adopted the theorem–proof-based approach discussed in [35] to the problem of checking the consistency and equivalence of the adapted process. Below, we briefly review the process.

The characterization that we use is logic-based formulas. Each task in the business process is characterized as a predicate or a term, and each individual step in the process is characterized as a logic formula. The processes that we consider here always have two common tasks s_1 and s_2 which it starts with and ends with, respectively. This assumption has a strong support in the workflow community and is canonized into the WS-BPEL specification [2]. In our approach, the entire process is represented by a set of logic formulas. In particular, the characterization of process P is a set of logic formulas F with the corresponding notations summarized as follows. We have used $\hat{}$, O, and $\oplus$ to denote logic control structures of parallel,

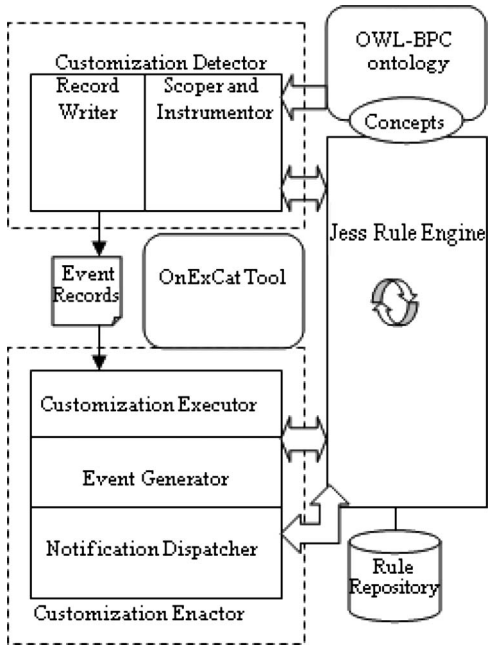

Fig. 6. Architecture of the framework.

branch (one or more branches get executed), and XOR branch (exactly one branch gets executed). (Other alternative notations are also available to use, e.g., $|\ |$, $\vee$, and $|$, respectively [5].) Each logic formula has a symbol of "$\rightarrow$" which connects the LHS of the formula to the RHS of the formula. The formula represents the truth of the left side of "$\rightarrow$" and derives the truth of the right side. In other words, we use "$\rightarrow$" to denote the complete and necessary condition of invocation of a task in terms of invocation of predecessor tasks and it corresponds to the notion of logical implication.

Now, with such a characterization, the verification of the structural consistency of the customized business process is converted to a proof that the following predicate-logic formula is true: $\forall s_1 \forall s_2, s_1 \rightarrow s_2$; in other words, $F \Rightarrow \forall s_1 \forall s_2, s_1 \rightarrow s_2$, where F is the set of logic formulas representing the process.

V. FRAMEWORK IMPLEMENTATION AND EXAMPLE PROBLEM

The implementation is done on a genetic desktop PC. It is deployed as a Web application with a user-friendly interface for an easy Web access. The Web application is deployed with Apache Tomcat 6.0.18. We have used the java-based Jess 7 as the inference engine for ontology and rule inference. The architecture of the implementation is shown in Fig. 6.

The implementation of the framework consists of two parts: the Customization Detector and the Customization Enactor. In the Customization Detector, the Scoper and Instrumentor identifies all the customizable contents of the PBP and identifies the ones that do need a customization because of their discrepancies with the SBP(s). The result is recorded by the Record Writer in Event Records. The Customization Detector relies on the Jess Rule Engine to inference on the OWL-BPC ontology for the knowledge of business process customization.

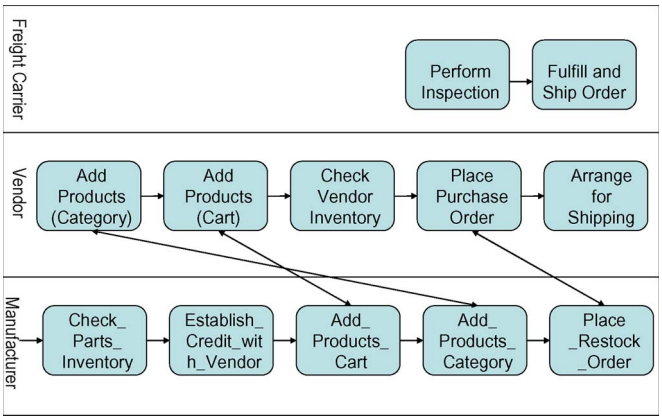

Fig. 7. Process of placing restock.

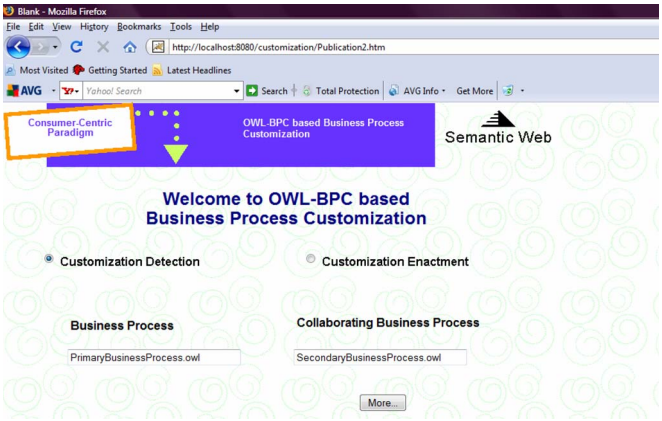

Fig. 8. Screenshot.

It also relies on the categorization tool—OnExCat—to overcome the semantic heterogeneity of various business process descriptions.

The Customization Entactor consists of the Event Generator, the Customization Executor, and the Notification Dispatcher. The Event Generator takes in the event records and generates events. The events will be passed to the Jess Rule Engine for processing. The rules of customization are pre-edited through a rule editor, which is not shown in the figure. The rules are stored in the Rule Repository connected to the Jess Rule Engine. The Jess Rule Engine will pass the action command, if any rules are triggered, to the Customization Executor to execute the customization. It will also pass the events to the Notification Dispatcher, which notifies the owners of the PBP and the SBPs.

In order to explain how the framework works and prove the capability of our solution, we present a simplified version of the business processes that are used to check the inventory and place restock orders for supply chain management as our example problem for business process customization. The business processes for the participants in the example problem can be depicted as shown in Fig. 7. Although the processes are simplified, the customization demand in the example is nontrivial. As we can see from the figure, the facts that there is a mixture of two different types of customizations and there are interfaces of two types of semantics make the scenario complex. The screenshot of using the framework is shown in Fig. 8.

The example customization problem is described as follows. A procurement officer of a manufacturer logs into the online supply chain management system and checks the parts inventory. After inspecting the parts database, the officer discovers that a few items are low in stock. This person then goes to click the restock button and enters the ordering system to restock those items. At this time, the business process of the vendor for order processing is enacted. Since the manufacturer has just signed the agreement with the supplier, it is new to the vendor. Therefore, the vendor faces the problem of customizing its own business processes to suit the manufacturer's business style.

In this particular example, the framework will be used to investigate the processes of both the vendor and the manufacturer and detect and possibly enact two customizations: 1) Two activities of the vendor need to be reordered to make life easier for its customer, and 2) the delivery time of the vendor's business process is not consistent with the one required by its partner manufacturer's inventory lead time and therefore needs to be adjusted. Here, the vendor's process is the PBP, and the manufacturer's process is the SBP. The process flow in the framework is as follows.

First, goal analysis is performed on both processes to find relevant actions. The goal of "OrderSatisfied" is identified in the goal taxonomy of the manufacturer. OrderSatisfied can be reduced by either a) "FinishedProductAvailable," b) "FinishedProduct-Shipped," and c) "BuyerNotified" or by keeping the product stock as large as possible; the subgoal FinishedProductAvailable can be reduced to subgoals "PartsAvailable" and "MachineAvailable." The subgoal "PartsAvailable" can be implemented to the operational objective of "SufficientInventoryLevel."

"SufficientInventoryLevel" can be established by the action of a) "Check_Parts_Inventory" and b) "Place_Restock_Order."

On the side of the vendor, the goal of "FinishedProduct-Shipped" is identified in the goal taxonomy of the vendor. FinishedProductShipped can be reduced to three subgoals, i.e., "OrderInPlace," "ReasonableShippingTime," and "AcceptedShipping-Insurance." OrderInPlace can be implemented by "PlacePurchaseOrder."

By running OnExCat and Jess' inference on the domain ontology, it is then known that "Place_Restock_Order" is relevant to "PlacePurchaseOrder," that "AddProducts(Category)" is relevant to "Add_Products_Category," and that "AddProducts(Cart)" is relevant to "Add_Products_Cart."

Then, the Scoper and Instrumentor runs the algorithm to identify all the relevant customizable contents of the two business processes based on their relevant actions on the goal taxonomies. The Scoper and Instrumentor also investigates whether there exist any discrepancies and misalignments of the relevant customizable contents. In this case, the "DeliverTime" in the vendor's business process is found to be relevant to the "InventoryTime" in the manufacturer's process. Furthermore, it is discovered that the range of InventoryTime specified by the manufacturer's process is to be [5–8 days] and the range of DeliveryTime specified by the vendor's process is to be [9–10 days]. Since DeliveryTime does not fall in the range of InventoryTime, there is an inconsistency which needs to be especially taken care of. It is also detected that there exists a

discrepancy on the sequence of adding products from the cart and adding products from the category between the processes of the vendor and of the manufacturer.

There are two records to be written by the Record Writer, which are passed to the Event Generator. The Event Generator generates two events of the types "Inconsistent-Variable-Range" and "Different-Order." The engine executes the rule to advise revising the DeliveryTime to at most 8 days for the sake of its customer and switching the order of "Add Products (Category)" and "Add Products (Cart)."

VI. Conclusion and Discussion

In this paper, we have presented a conceptualization of customizing service-based business processes according to the discrepancies and misalignments discovered in their business process description documents. We have also presented an ontology, i.e., the OWL-BPC, which we have developed for this purpose. Our main contribution in this paper is to tackle the problem of possible inconsistencies of collaborating business processes, including the following: 1) possible semantic inconsistency such as semantic mismatching on process parameters; 2) behavioral mismatches which may or may not be compatible; and 3) misaligned rendezvous requirements. This ontology is used to build our framework for detecting and performing service-based business process customization. We have presented the detailed algorithm of the framework and its architecture.

Our research tackles the topic of human semantic Web from the aspect of customizing the procedures of automated businesses transactions by accommodating the various characteristics of individual business partners. The way that businesses operate nowadays is more reliant on electronic devices and documents and in a networked way. Effectively using such devices and documents is largely supported by the semantic Web technology. However, the automation of the interaction patterns among the business partners will only be meaningful if the uniqueness of each partner is also part of the knowledge base equipped to the automated process by the semantic Web technology. The implications of our research in real-world applications range from cultivating automatic service composition for a more streamlined service-based business process integration scheme to increasing the reuse of existing business process artifacts and from improving the scalability of a business process to increasing the agility of the operations of the entire business.

We are aware of the fact that our ontology matching tool, which the customization framework uses, does not have a correct matching rate of 100%. We are in the process of improving its matching rate. At this time, to avoid any problem that may be caused when the automatic matching results are erroneous, we involve an extra step of human screening of the machine-produced matching results. On the one hand, the automatic matching here provides the first-cut matching results with a relatively high correct rate. The benefit of such automatic matching includes the off-load of the complicated task of matching from humans by replacing it with a simple "yes-or-no" checking for most cases. For only a small portion of the cases, humans will have to perform rematching. On the other hand, we feel that such human involvement is necessary although this may cause extra efforts in 85% of the cases when the matching results returned by the machine are actually correct. Such involvement provides a guarantee that incorrect matching results will not be propagated to customization detection and enactment. One way to reduce such extra efforts is to only check on the cases with a confidence level lower than some threshold. In other words, when the matching scores do not seem as discriminate, they may imply a need of further human investigation.

The behavioral customization in nontrivial cases may depend on the run-time context of the business process execution. We are developing a more comprehensive theoretic framework to address this issue. Currently, the framework is only able to perform customization statically before the business process is instantiated. Since the OWL-BPC is designed to handle both static and dynamic customization, we are in the process of implementing a dynamic customization capacity during or after the instantiation time.

Appendix A
Ontology (Content Properties)

Customization of semantic content

hasVariable

Ranges over CustomizableProcessVar instances of this ontology. Class CustomizableProcessVar is a subclass of ProcessVar in the Process ontology. Customization of variables in a process can be on its type, its presence, and its ranges. These three aspects of customization on class CustomizableProcessVar correspond to properties of *&owl;#DataTypeProperty*, *hasPresence*, and *hasRange*.

hasAtomicProcess

Ranges over CustomizableAtomicProcess instances of this ontology. Class CustomizableAtomicProcess is a subclass of AtomicProcess in the Process ontology. Customization of atomic processes in a process can be performed on any items defined in an atomic process, including its service name and all its parameters and local variables. Only property *&process;#hasInput* and its range InPut are shown for the sake of saving space.

Customization of behavioral content

hasConstruct

Ranges over Customizable ControlConstruct instances of the Process ontology. Concurrent constructs and sequential constructs are subclasses of class CustomizableControlConstruct. Customization of constructs is applied on the links that connect the activities within such control constructs. Therefore, ControlConstruct has a hasLink property, which ranges over the Link instances of the Process ontology.

Customization of rendezvous content

hasDuration

Ranges over Duration instances in this ontology. Duration of a process may need to be adjusted according to the preference of the partner process.

hasDeadline

Ranges over Deadline instances in this ontology. Deadline of a process may be adjusted according to the deadline constraint of the partner process.

hasTimeOut

Ranges over TimeOut instances in this ontology, which represent the time out of the process.

hasProcessID

Ranges over ProcessID instances in the ontology. There may be constraints on the Process instances to interact with the partner process.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their accurate and constructive comments, which have helped to improve this paper greatly.

REFERENCES

[1] A. Naeve, "The human semantic Web: Shifting from knowledge push to knowledge pull," *J. Semantic Web Inf. Syst. (IJSWIS)*, vol. 1, no. 3, pp. 1–30, 2005.
[2] BPEL. [Online]. Available: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev =wsbpel
[3] OWL-S. [Online]. Available: http://www.ai.sri.com/daml/services/owl-s/1.2/
[4] OWL. [Online]. Available: http://www.w3.org/TR/owl-features/
[5] R. Milner, "Operational and algebraic semantics of concurrent processes," in *Handbook of Theoretical Computer Science*, vol. B, *Formal Models and Sematics*, 1990, pp. 1201–1242.
[6] C. Ouyang, E. Verbeek, W. M. P. van der Aalst, S. Breutel, M. Dumas, and A. H. M. ter Hofstede, "Formal semantics and analysis of control flow in WS-BPEL," *Sci. Comput. Program.*, vol. 67, no. 2/3, pp. 162–198, Jul. 2007.
[7] A. Naeve, M. Lytras, W. Nejdl, J. Harding, and N. Balacheff, "Advances of semantic Web for e-learning: Expanding learning frontiers," *Brit. J. Educ. Technol.*, vol. 37, no. 3, pp. 321–330, 2006.
[8] D. He and X. Wu, "Ontology-based feature weighting for biomedical literature classification," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, 2006, pp. 280–285.
[9] R. Katz-Haas, "Ten guidelines for user-centered Web design," *Usability Interface*, vol. 5, no. 1, Jul. 1998. [Online]. Available: http://www.stcsig.org/usability/newsletter/9807-webguide.html
[10] X. Wu, "User-centered agents for structured information location on the Web," in *Proc. 10th IEEE Int. Enterprise Distrib. Object Comput. Conf.*, 2006, p. 19.
[11] Y. Zhuang, S. Fong, and M. Shi, "Knowledge-empowered automated negotiation system for e-Commerce," *Knowl. Inf. Syst.*, vol. 17, no. 2, pp. 167–191, Nov. 2008.
[12] O. H. Juarez-Espinosa, "Development of user centered environmental software systems," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, 1999.
[13] J. Fechter, T. Grunert, L. M. Encarnação, and W. Straßer, "User-centered development of medical visualization applications: Flexible interaction through communicating application objects," *Comput. Graph.*, vol. 20, no. 6, pp. 763–774, Nov./Dec. 1996.
[14] K. Marsolo and S. Parthasarathy, "On the use of structure and sequence-based features for protein classification and retrieval," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 59–80, Jan. 2008.
[15] B. Rosenfeld and R. Feldman, "Self-supervised relation extraction from the Web," *Knowl. Inf. Syst.*, vol. 17, no. 1, pp. 17–33, Oct. 2008.
[16] R. Singh and A. F. Salam, "Semantic information assurance for secure distributed knowledge management: A business process perspective," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 3, pp. 472–486, May 2006.
[17] Q. A. Liang, H. Lam, L. Narupiyakul, and P. C. K. Hung, "A rule-based approach for availability of Web service," in *Proc. IEEE Int. Conf. Web Serv.*, 2008, pp. 153–160.
[18] P. C. Xiong, Y. Fan, and M. Zhou, "QoS-aware Web service configuration," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 4, pp. 888–895, Jul. 2008.
[19] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic Web services," *IEEE Intell. Syst.*, vol. 16, no. 2, pp. 46–53, Mar./Apr. 2001.
[20] M. Papazoglou, M. Aiello, M. Pistore, and J. Yang, "Planning for requests against Web services," *IEEE Data Eng. Bull.*, vol. 25, no. 4, pp. 41–46, Dec. 2002.
[21] Q. A. Liang, J. Y. Chung, and D. Miller, "Modeling semantics in composite Web service requests by utility elicitation," *Knowl. Inf. Syst. (KAIS) J.*, vol. 13, no. 3, pp. 367–394, Nov. 2007.
[22] D. König, N. Lohmann, S. Moser, C. Stahl, and K. Wolf, "Extending the compatibility notion for abstract WS-BPEL processes," in *Proc. 17th Int. Conf. World Wide Web*, 2008, pp. 785–794.
[23] Q. Liang, X. Wu, and H. Lau, "Optimizing service systems based on application-level QoS," *IEEE Trans. Serv. Comput.*, vol. 2, no. 2, pp. 108–121, Apr.–Jun. 2009.
[24] [Online]. Available: www.w3.org/RDF/
[25] [Online]. Available: http://www.jessrules.com/jess/charlemagne.shtml
[26] [Online]. Available: http://xsb.sourceforge.net
[27] [Online]. Available: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessTab
[28] [Online]. Available: http://fowl.sourceforge.net
[29] [Online]. Available: http://clarkparsia.com/pellet/
[30] Q. A. Liang and H. Lam, "Web service matching by ontology instance categorization," in *Proc. IEEE Int. Conf. Serv. Comput.*, 2008, pp. 202–209.
[31] M. Iwayama and T. Tokunaga, "A probabilistic model for text categorization: Based on a single random variable with multiple values," in *Proc. Conf. Appl. Natural Lang. Process.*, 1994, pp. 162–167.
[32] N. W. Paton and O. Díaz, "Active databases survey," *ACM Comput. Surv.*, vol. 31, no. 1, pp. 63–103, Mar. 1999.
[33] F. Casati, S. Ceri, B. Pernici, and G. Pozzi, "Workflow evolution," *Data Knowl. Eng.*, vol. 24, no. 3, pp. 211–238, 1998.
[34] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *J. Distrib. Parallel Databases*, vol. 14, no. 1, pp. 5–51, Jul. 2003.
[35] Q. A. Liang and J. L. Zhao, "Verification of unstructured workflows via propositional logic," in *Proc. 7th Int. Conf. IEEE/ACIS*, 2008, pp. 247–252.

Qianhui Liang received the Ph.D. degree in computer engineering from the University of Florida, Gainesville, Florida, USA.

She is a researcher at Cloud and Security Lab, HP Labs, Singapore. She has published in journals and conferences like the IEEE TRANSACTIONS ON SERVICES COMPUTING, the *International Journal of Web Services Research*, the *Knowledge and Information Systems*, the International Conference on Services Computing (SCC), the International Conference on Service-Oriented Computing, the International Conference on Web Services, the International Conference on Information Reuse and Integration (IRI), the International Conference on Enterprise Information Systems, etc. Her research interests are services computing (service composition, service modeling, and service ontology), semantic Web, cloud computing, and machine learning.

Dr. Liang is a member of the Association for Computing Machinery. She is currently the Work-In-Progress Track Chair of the IEEE International Conference on Services Computing (SCC 2011). She was the Program Vice Chair of the IEEE International Conference on Information Reuse and Integration (IRI 2009), the Publicity Chair of the IEEE International Conference on Services Computing (SCC 2009), the Publicity Chair of the 5th 2009 World Congress on Services (SERVICES-II 2009), the Publicity Chair of the IEEE 2009 International Conference on Cloud Computing (CLOUD-II 2009), and the Local Organization Cochair of the Fourth IEEE Asia-Pacific Services Computing Conference (APSCC 2009).

Xindong Wu received the Ph.D. degree in artificial intelligence from the University of Edinburgh, Edinburgh, U.K.

He is currently a Professor and the Chair of the Computer Science Department, University of Vermont, Burlington. He has published over 170 refereed papers in various journals and conferences, including the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, Association for Computing Machinery (ACM) Transactions on Information Systems, Data Mining and Knowledge Discovery, Knowledge and Information Systems (KAIS), International Joint Conference on Artificial Intelligence, Association for the Advancement of Artificial Intelligence, International Conference on Machine Learning, ACM Conference on Knowledge Discovery and Data Mining (KDD), International Conference on Data Mining (ICDM), and International World Wide Web Conference (WWW), as well as 22 books and conference proceedings. His research interests include data mining, knowledge-based systems, and Web information exploration. His research has been supported by the U.S. National Science Foundation (NSF), the U.S. Department of Defense, the National Natural Science Foundation of China, and the Chinese Academy of Sciences, as well as industrial companies including U.S. West Advanced Technologies and Empact Solutions.

Dr. Wu is the founder and current Steering Committee Chair of the IEEE ICDM, the founder and current Editor-in-Chief of KAIS (by Springer), the Founding Chair (2002–2006) of the IEEE Computer Society Technical Committee on Intelligent Informatics, and a Series Editor of the Springer Book Series on Advanced Information and Knowledge Processing. He was the Editor-in-Chief of the IEEE TKDE (by the IEEE Computer Society) between January 1, 2005, and December 31, 2008, and served as the Program Committee Chair for the ICDM 2003 (the 2003 IEEE International Conference on Data Mining) and as the Program Committee Cochair for KDD-07 (the 13th ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) International Conference on Knowledge Discovery and Data Mining). He is the 2004 ACM SIGKDD Service Award winner, the 2006 IEEE ICDM Outstanding Service Award winner, and the 2005 Chair Professor in the Changjiang (or Yangtze River) Scholars Program at the Hefei University of Technology appointed by the Ministry of Education of China. He has been an invited/keynote speaker at numerous international conferences, including the 8th Joint Conference on Knowledge-Based Software Engineering, NSF Next Generation Data Mining 2007, Pacific-Asia Conference on Knowledge Discovery and Data Mining 2007, IEEE International Enterprise Distributed Object Computing Conference 2006, IEEE International Conference on Tools with Artificial Intelligence 2004, IEEE/Web Intelligence Consortium/ACM International Conference on Web Intelligence 2004/International Conference on Intelligent Agent Technology 2004, International Conference on Software Engineering and Knowledge Engineering 2002, and the 1997 International Conference on the Practical Application of Knowledge Discovery and Data Mining (PADD-97).

E. K. Park received the Ph.D. degree in computer science from the Northwestern University, Evanston, IL.

He is currently a Professor of computer science with the University of Missouri at Kansas City, Kansas City. He is currently on an assignment serving as a Program Director of the Division of Computing and Communications Foundations, U.S. National Science Foundation. His research interests include data mining, bioinformatics, information and knowledge management, computer communications and networks, optical networks, distributed systems, and object-oriented methodology.

Taghi M. Khoshgoftaar (M'86) received the Ph.D. degree from Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

He is currently with Florida Atlantic University, Roca Baton, where he is a Professor of the Department of Computer Science and Engineering and the Director of the Empirical Software Engineering Laboratory and Data Mining and Machine Learning Laboratory. He has published more than 400 refereed papers. His research interests are in software engineering, data mining, and machine learning.

Prof. Khoshgoftaar is a member of the IEEE Computer Society and the IEEE Reliability Society. He was the Program Chair and General Chair of the IEEE International Conference on Tools with Artificial Intelligence in 2004 and 2005, respectively, and was the Program Chair of the 20th International Conference on Software Engineering and Knowledge Engineering (2008). He was the General Chair of the 21st International Conference on Software Engineering and Knowledge Engineering (2009). He has served on technical program committees of various international conferences, symposia, and workshops. Also, he has served as North American Editor of the Software Quality Journal, was on the editorial board of the Empirical Software Engineering journal, and is on the editorial boards of the journals Software Quality, Fuzzy Systems, and Knowledge and Information Systems.

Chi-Hung Chi obtained his Ph.D. degree from Purdue University.

He is currently a Professor with the School of Software, Tsinghua University, Beijing, China. After his Ph.D. from Purdue University, West Lafayette, IN, he was with Philips Research Laboratories and IBM Poughkeepsie before he returned to academia in 1993. Since then, he was with the Chinese University of Hong Kong, Shatin, Hong Kong, and the National University of Singapore, Singapore, Singapore, before he joined Tsinghua University in 2005. He has published about 200 papers and is the holder of the six U.S. patents. His current main research areas include distributed systems and computer network, content engineering, system security, and service engineering.

Dr. Chi was the Program Chairman of international conferences, including the Advanced Workshop on Content Computing 2004, Workshop on Web Content Distribution 2004, IEEE International Workshop on Service-Oriented System Engineering 2005, and International Conference on Service-Oriented Computing 2009, and is on the PC committee and panelist of many international conferences. The technologies that he developed on content engineering have also been transferred successfully into industry.