

Semantic Enhanced Rule Driven Workflow Execution in Collaborative Virtual Enterprise

Wei Ren¹, Gang Chen¹, Zhonghua Yang¹,
Junhong Zhou²,

¹School of Electrical and Electronics Engineering,

⁴School of Computer Engineering,

Nanyang Technological University, Singapore 639798

weiren@ntu.edu.sg

Jing Bing Zhang², Chor Ping Low¹, David Chen³ and
Chengzheng Sun⁴

²Singapore Institute of Manufacturing Technology
(SIMTech), Singapore 638075

³School of Computing & Information Technology,
Griffith University, Australia 4111

Abstract—Globalization leads to an efficient new business paradigm generally known as Collaborative Virtual Enterprise (CVE) which demands flexible service orchestration and robust workflow execution. As two major service orchestration strategies, OWL-S and BPEL have their own strength and deficiencies. To meet the challenge in CVE, in this paper, we propose a Workflow Execution System (WES) that takes advantage of the complementary strengths of these two technologies. On the one hand, having semantic support, OWL-S is used in dynamic service discovery and composition at high level. On the other hand, at the concrete level, industry-based BPEL is exploited in service execution. The description of each Semantic Web service is enhanced with rule-based modeling of the essential business logic behind the service interface. In order to realize interoperability in OWL-S and BPEL without loss of semantic information, we further proposed an OWLS2BPEL Mapper to facilitate the workflow robustness and support rule evaluation to increase responsiveness to customers. A concrete scenario in PC manufacturing Collaborative Virtual Enterprise is analyzed to demonstrate the effectiveness of our workflow execution system.

Keywords—workflow, Semantic Web service, BPEL, OWL-S, rule

I. INTRODUCTION

Globalization leads to an efficient new business paradigm generally known as Collaborative Virtual Enterprise (CVE), where companies increasingly concentrate on their core competencies and outsource all other functions to their partners. In a broad sense, CVE refers to a timely alliance of enterprises that share their skills and resources in order to better respond to business opportunities in collaborative environments. Aimed at fully supporting workflow execution in CVE, this paper explores the semantic-rich service-oriented paradigm for enterprise integration based on Web Service standards. To achieve the complex business workflow execution across business boundaries, multiple services should be orchestrated first through service composition which is widely considered as a promising solution towards enterprise application integration.

There are two active communities for service orchestration: one is driven by industrial applications, the other is inspired by the booming research development in Semantic Web. Among

several emerged industry standards, we believe Web Services Business Process Execution Language (also known as WS-BPEL, or BPEL for short) [3], is the most matured technology. However, BPEL mainly focus on the syntactic aspects of Web service composition and is lack of semantic support. In a parallel effort, the Semantic Web community seeks to address these problems through introducing semantic-rich languages for high-level services description and computer programs for processing semantic information. One representative technology is Web Ontology Language for Services (OWL-S) [2]. As two major service orchestration strategies, OWL-S and BPEL have their own strength and deficiencies. Section III will give further details.

Target at fully supporting flexible service orchestration and robust workflow execution in CVE, in this paper, we propose a Workflow Execution System (WES) which has following features: (1) in order to offset each other's weakness, we make an attempt to combine the complementary strengths of OWL-S and BPEL. On the one hand, having semantic support, OWL-S is used in dynamic service discovery and composition at high level. On the other hand, industry-based BPEL is exploited in service execution at concrete level; (2) in addition to modeling service *inputs* and *outputs* using semantics, which has already been suggested by previous research works, service descriptions are further enhanced with business rules that capture the underlying business logic behind the service interfaces. To be more specific, using Semantic Web Rule Language (SWRL), one of the recommended rule languages in OWL-S specification, business rules is utilized to model the *preconditions* as well as the *effects* of any business service in a domain-dependent and semantic-rich manner. The WES will take different actions to fulfill the workflow according to correspondent *preconditions/effects* which subsequently influence the workflow execution. Through processing these business rules in the context of any given business workflow, our execution system makes decisions at a functional level to increase responsiveness to customers. The robustness of critical business processes will be greatly facilitated by this much improved service description (Ref. Section V); (3) OWL2BPEL Mapper is further proposed to translate OWL-S process to BPEL process by converting data, data flow and control flow into BPEL counterparts. It bridges the gap between OWL-S and BPEL and maintains the semantic

information during their interoperation. This paper is a summarization of our initial effort towards the above goals.

The remaining part of this paper is organized as follows. Section II provides a literature review of related research works. Section III introduces the system architecture. The implementation will be described in Section IV. Section V presents an analysis through the CVE scenario. Finally Section VI concludes this paper.

II. RELATED WORK

As one of the key technologies of the service-oriented architecture (SOA) and CVE, dynamic service orchestration has attracted tremendous research attention over the past few years. Most of them focus on the two composition languages -- OWL-S and BPEL. In addition to composition languages, a wealth of service composition systems has been published recently in the literature. In [4], Sirin *et. al.* proposed a semi-automatic user-driven Web Service composition system. Composite services in their system will be established through a chaining procedure driven primarily by human decisions. In an attempt to reduce human involvement, AI planning techniques such as the SHOP2 system [5] have been extensively explored to achieve automatic service composition. In addition to AI planning, rule-based composition systems such as SWORD [6] can automatically verify whether a desirable composite service can be built from existing services.

Only few research works directly concentrate on the relation between OWL-S and BPEL. [7] presents a comparison study. [8] and [9] provides a concrete mapping between the two languages. One of our system components -- OWLS2BPEL Mapper is similar to work [8] in which the *precondition* and *effect* are left out of mapping. As a matter of fact, *precondition* and *effect* can also control the workflow execution at concrete level (Ref. Section V). We extend the mapping to *preconditions* and *effects* by creating a partner for Rule Evaluator Web Service to check *preconditions* and *effects*. In work [9], they present a mapping strategy to map BPEL process to OWL-S ontology to overcome the semantic limitation of BPEL which is different from our work in the reversed order. Unlike our top-down approach, [1] takes a bottom-up approach to extend BPEL with automatic service discovery and semantic translation. However, their system relies on a DAML-S based Semantic Discovery Service (SDS) to find and invoke potential service partners during the workflow execution. It does not support automated service composition. In contrast, we address the issue at two different levels. At high level, we rely on OWL-S to automate service discovery and composition; at the concrete level, we rely on BPEL to automate workflow execution. OWLS2BPEL Mapper bridges the gap between them. In addition, to our knowledge, no mapper addresses *preconditions* and *effects* mapping as studied in this paper.

III. SYSTEM ARCHITECTURE

Figure 1 shows the architecture of our WES. We will briefly introduce the models in the architecture. More technical details can be referred to the next section.

BPEL mainly focus on the syntactic aspects of Web service composition and is lack of semantic support which

subsequently results in manual service composition. Although it provides the mechanisms to bind *partnerLinks* dynamically, very often, the way BPEL partners bind to actual service instances is static and inflexible. That means at design time, partners are already bound to the actual service instances. When the BPEL process is deployed, the actual service instances participating in the orchestration are known right away and can not be changed dynamically. This shortcoming makes BPEL lack of flexibility and dynamicity in service discovery and composition.

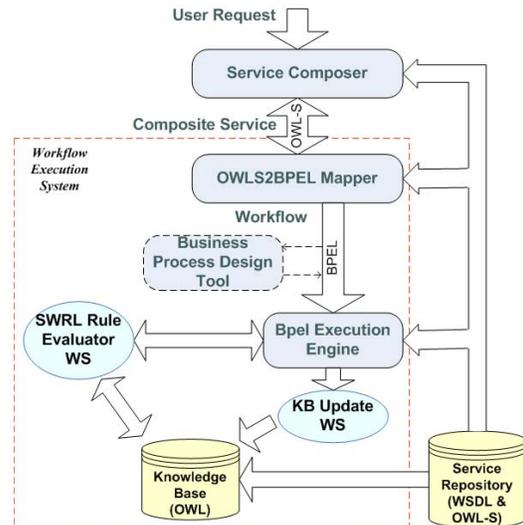


Figure 1. System Architecture

In contrast with BPEL, OWL-S provides semantic-rich descriptions of service to automate service discovery and composition by defining data (concept) in ontologies which support reasoning about service capability. This advantage, further, facilitates the dynamicity in service composition. Especially, when new Web Services emerge and old Web services are not available temporarily, the enterprises involved in a service composition might change their business partners to fulfill the business goal.

In line with this view, we adopt OWL-S as the composition strategy at high level by taking advantage of its flexibility and dynamicity in service composition. The *inputs* and *outputs* of a service will be marked up with domain ontologies. In addition, service descriptions will be further enhanced with business rules that capture the underlying business logic behind the service interfaces. Specifically, using SWRL, one of the recommended rule languages in OWL-S specification, business rules will be utilized to model the *preconditions* as well as the *effects* of any business service in domain-dependent and semantic-rich manner. The WES will take different actions to fulfill the workflow according to correspondent *preconditions/effects* which subsequently influence the workflow execution. Through processing these business rules in the context of any given business workflow, our execution system makes decisions at functional level to increase responsiveness to customers. The robustness of critical business processes will be greatly facilitated by this much improved service description (Ref. Section V).

Although OWL-S has its own strength in composition, it is still in its early stage. The adoption of OWL-S hasn't been prevalent for lack of tools to support the development. Moreover, there is no mechanism designed for OWL-S to identify faults during execution. Nevertheless, BPEL has strong points which can offset OWL-S' weakness. Currently, major vendors successively provide their runtime environment for executing BPEL business processes for both J2EE and .Net platform. In addition, BPEL has correspondent process design environment with a graphical user interface such as Oracle JDevelop. Furthermore, BPEL provides systematic mechanisms for dealing with business exceptions. In view of these facts, BPEL is exploited in service execution at concrete level.

The user inputs to our system are the requested service capability which refers primarily to the required outputs of a service given the available inputs. The service composer based on the forward chaining algorithm has been designed and developed by us. The details of our composer can be found in [10]. By leveraging the semantic markup of service inputs and service outputs, our composer will aggressively chain a group of Semantic Web services together in order to satisfy the specific requirements. The output from Service Composer is a list of composite services which consists of one or more atomic services described in OWL-S with *Input/Output/Precondition/Effect* (IOPE).

OWLS2BPEL Mapper takes composite service as input and outputs a standard BPEL workflow without losing semantic information. It involves with IOPE mapping, process model mapping and semantic service mapping. The first two mappings focus on syntactic aspect such as data (IOPE), data flow and control constructs transformation. Since BPEL interacts with WSDL service interface, in order to get correspondent BPEL part, OWL-S process model and its grounding to WSDL are utilized in this mapping.

Due to lack of semantic support in BPEL, the semantic service mapping is about how to convert from each Semantic Web service to a BPEL sub-workflow while maintaining the semantic information. For the *precondition/effect* of each Semantic Web service, the Mapper will create a partner for rule Evaluator Web service to process the correspondent business rule. When BPEL engine calls the Evaluator Web Service with business rules as input, the rule Evaluator will process the *preconditions/effects* of each chosen Web Service along the workflow and return a Boolean value. Here, the Evaluator plays a role of decision maker that can influence the workflow execution at functional level in domain-dependent and semantic-rich manner.

For each Semantic Web service itself, the Mapper will generate a partner for the correspondent Web service to perform actual business function. Very often, running a Web service will create new knowledge which should be kept as dynamic execution environment. In order to maintain the semantic information, (1) outputs from BPEL execution engine should be transformed into knowledge in OWL class format; (2) a Knowledge Base is necessary for storage and update of semantic information during workflow execution. Therefore, the Mapper will generate another partner for KB Update Web

service to transform outputs of the Web service to knowledge format first and then store it in the Knowledge Base. The updated OWL knowledge base will be utilized to assess succeeding services in the workflow. This guarantees the semantic information will not be lost during the execution of BPEL workflow.

Before the BPEL workflow is deployed into BPEL Execution Engine, business process design tools such as Oracle JDevelop can be used to refine the converted BPEL workflow. Error handling and compensation mechanism can be added to increase reliability and robustness in business workflow at technical level.

All the services written in WSDL and OWL-S advertised by *Service Providers* are stored in the Service Repository. In addition to keep the dynamic execution environment, the Knowledge Base also stores the domain ontologies written in OWL which are utilized to mark up IOPE of semantic service. To access Service Repository and Knowledge Base, the Protégé OWL API library is used which is developed for Semantic Web services by Stanford University.

IV. IMPLEMENTATION

In this section, we will give implementation details about three important components in our workflow execution system. They are OWLS2BPEL Mapper, SWRL Business Rule Evaluator, and Knowledge Base Update.

A. OWLS2BPEL Mapper

As there exists similarity in the conceptual model of OWL-S and BPEL [9], our OWLS2BPEL Mapper is to convert OWL-S composite service to a BPEL workflow. It includes syntactic mapping and semantic mapping.

1) *Syntactic Mapping*: The syntactic mapping deals with the transformation from OWL-S process model to BPEL process. It is similar to the work in [8, 9]. We will briefly introduce it here, for more details please refer [8, 9].

a) *Atomic Service*: In general, "Perform" construct in OWL-S is used to perform an atomic service, BPEL has primitive activity `<invoke>` to perform some operations. If the atomic service has inputs and/or outputs, `<receive>` and `<reply>` primitive activity in BPEL are used to receive input messages and/or send output messages from/to some other Web service. The mapping from OWL-S atomic service to BPEL process could be:

```
<sequence> <receive.../>
<invoke inputVariable = "..." outputVariable = "..." .../>
<reply../> </sequence>
```

b) *Composite Service*: Control constructs in OWL-S composite process has similar logic behavior to BPEL structured activities. Figure 2 gives a summary of control constructs mapping between OWL-S and BPEL.

c) *Data and Data Flow*: Data mapping is primarily involved with the mapping between IOPE of OWL-S and BPEL variables. In OWL-S, grounding builds a mapping between each input/output of the OWL-S atomic service and a WSDL message type. On the other hand, BPEL uses WSDL

message type as variable declarations. Therefore, we can map the input as well as output of the OWL-S atomic service to BPEL variables with the WSDL message type specified in the OWL-S grounding. Since *precondition* and *effect* are modeled by business rules in this paper, we refer to these rule files by using their URL in *ServiceParameter* of OWL-S *profile*. The Business Rule Evaluator Web Service has one input message (URL of rule file) and one output message (a Boolean value). Thus, the URL of rule file (representing precondition/effect of an OWL-S atomic service) will be mapped into BPEL variable first and then the rules will be processed by the Rule Evaluator Web service. We will introduce the rule processing in the next sub-section. The data flow between atomic services can be described using BPEL `<assign>` activity combined with `<copy>`, `<from>` and `<to>` elements.

| OWL-S | BPEL |
|---|-------------------------------|
| Sequence | <code><sequence></code> |
| IfThenElse | <code><switch></code> |
| RepeatWhile, RepeatUntil, Iterate | <code><while></code> |
| Split + Join | <code><flow></code> |

Figure 2. Control Constructs Mapping

2) Semantic Mapping:

Semantic mapping aims at keeping semantic information in the BPEL workflow during transformation. As illustrated in Figure 3, each Semantic Web service can be mapped into a BPEL sub-workflow which is composed of four Web Services calls. They are two Evaluator Web Services for processing *precondition* and *effect* respectively, one Web Service (ws1) performing the actual business logic and one KB Update Web Service.

The IOPE of a Semantic Web service (sws1) are mapped into BPEL variables respectively. The first Evaluator Web Service takes the *Precondition* as input, evaluates the condition required by the sws1 and returns a Boolean value. If the condition to make the service accessible is satisfied, the workflow will go ahead to execute the actual Web service. Otherwise, it will be terminated or move to other services. In such a case, the Mapper will find an alternative service from a list of available services which already got from the Service Composer. Subsequently, this alternative semantic service will be mapped into a sub-workflow in the same way by utilizing BPEL control constructs such as `<IfThenElse>`.

Outputs of the service could create new information which must be recorded. The KB Update Web Service transforms the *Outputs* into knowledge in OWL format and stores it in the OWL knowledge base which served as a dynamic execution environment.

The second Evaluator Web Service evaluates the *Effect* and return a Boolean value which can influence the workflow execution. If the value is *True*, the workflow will move to the next semantic service. Otherwise, it will report to client. During

the whole mapping procedure, all the high-level semantics contains in OWL-S composite service are embedded in low-level BPEL workflow.

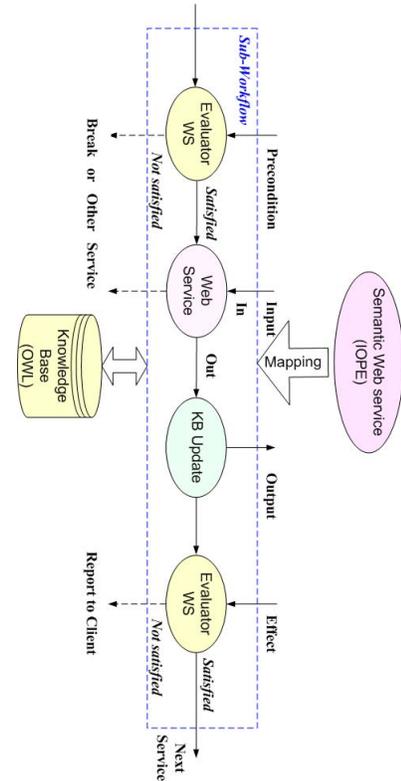


Figure 3. Semantic Web service Mapping

B. SWRL Business Rule Evaluator

This Section presents the technical details of our SWRL business rule evaluator. The basic idea is to utilize business rules as a general vehicle to drive the modeling and processing of the essential business logic behind every Web Service interface.

1) Business Rule depicted in SWRL:

Business rules serve as a powerful and important tool in describing the operations, definitions and constraints that apply to an organization in achieving its business goals. A business rule is comprised of two parts, namely, the *antecedent* and the *consequent*. The *antecedent* governs the conditions for the corresponding rule to become satisfactory. It usually assumes the form of a *conjunction* of *atomic conditions*. When all these *atomic conditions* are satisfied, the *consequent* as defined in the rule will take effect.

Business rules can be applied to model both the *preconditions* and the *effects* of a Web Service. In case a rule is used to model the *preconditions*, the *antecedent* of the rule specifies the situations for the service to become accessible. On the other hand, when modeling service effects, business rules can be used to represent related but different results after calling the service. Obviously, *preconditions* and *effects* together form the logic core of a business service.

In our WES, the SWRL [11] is utilized for writing business rules. SWRL is an emerging standard rooted in the Web

Ontology Language (OWL), a popular ontology language based on the Resource Description Framework (RDF). Through extending the set of OWL axioms, Horn-like SWRL rules can be designed to manipulate OWL classes and properties, and to infer new knowledge from existing OWL knowledge bases. SWRL also provides a range of build-ins to support various mathematical and logical calculations. In this paper, we will illustrate the use of SWRL through a simple example.

Suppose we want to model one typical *Precondition* of a service that sells PC monitors. Using natural English, this *Precondition* can be described with rule R1 below.

R1. IF (C1) the monitor size is 17 inch; and (C2) the quantity of monitors to be purchased (?q) is greater than 100; THEN return True ; ELSE return False.

The antecedent of rule R1 contains two conditions C1 and C2. Assume that a client issues a request to this service for purchasing one hundred and twenty 17-inch monitors. This request satisfies both conditions C1 and C2. According to R1, the Boolean value *True* will be returned to indicate that the workflow can go ahead to invoke the actual Web service. Otherwise, the workflow will be terminated or move to other service which can satisfy the precondition.

One prerequisite for specifying R1 using SWRL is to design a proper group of OWL ontologies. In the context of R1, one ontology concept can be identified, namely, Monitor. This concept is implemented via OWL class which contains two properties: *monitorSize* and *quantity*.

2) Rule Evaluator:

A business rules Evaluator Web Service has been developed in this paper to support our WES. It plays a role of decision maker that can influence the workflow execution at functional level. It receives an URL of a rule file as an input and returns a Boolean value as an output. Our Evaluator relies on two open-source projects, Protégé and Jess Rules Engine. Upon given an OWL-S service description, APIs offered by Protégé will be first applied to parse the service description in order to identify the corresponding service inputs, service outputs, preconditions, and effects (IOPE). Using SWRL Rule Engine Bridge bundled with the Protégé distribution, SWRL rules contained in both the preconditions and effects will be further converted to Lisp-like rules suitable for processing by the Jess Rules Engine. Evaluating these rules may result in modifications of an OWL knowledge base that serves as the local execution context of the service.

C. Knowledge Base Update

Very often, running a Web service will create new knowledge. The OWL knowledge base should be updated to reflect such change. The impact of component services execution on the whole business workflow is reflected through the modified knowledge base. Upon executing a workflow by the BPEL Engine, an OWL knowledge base will be initialized with the client's request (modeled as OWL classes), which is expected to be satisfied through the workflow. Driven by this knowledge base, component services along the workflow will be executed in sequence. For each component service, its outputs of execution in BPEL engine are typed with XML

Schema which can not be used to update the OWL Knowledge Base directly. An XSLT style sheet should be applied to converts the service *Outputs* characterized in XML Schema into OWL classes typed outputs. The style sheet script can be obtained in each atomic OWL-S service grounding. This updated OWL knowledge base will be utilized to assess succeeding component services in the workflow.

V. SCENARIO ANALYSIS

In order to respond to the dynamic market demand for computers, business workflows for PC manufacturing need to be dynamically formulated and executed to share their functionalities and resources in a collaborative manner. Figure 4 shows a part of workflow formulated in a Collaborative PC manufacturing Virtual Enterprise.



Figure 4. Part of PC Manufacturing CVE

To illustrate the mapping from a semantic service to a sub-workflow in BPEL, we use a typical product selling service MonitorSupplier that sells PC monitors as an example, followed by Hardware Shipping service. As shown in Figure 5.

One typical precondition of the service is that the quantity of monitors to be purchased should not be less than 100. This business rule will be processed by the Evaluator Web Service first. If client's request satisfies the condition, the MonitorSupplier service can be accessible and the sub-workflow will continue. Otherwise, it will be terminated or move to other services. For example, if the requested minimal quantity is not satisfied, the Evaluator will return a Boolean value *False*. Subsequently, the Mapper will contact with the Service Composer to get an alternative semantic MonitorSupplier service first. Then the Mapper will construct BPEL activities for the alternative service by utilizing BPEL control constructs such as <IfThenElse>. Finally, the alternative semantic service is transformed to a sub-workflow. In case there is no alternative service is returned by the Service Composer, the Mapper will terminate the workflow or report to the client. It is worth to note that this is not achievable with OWL-S only. Through processing these semantic enhanced rules and utilizing the BPEL control constructs, our WES greatly increases responsiveness to customers and improves system robustness.

The actual MonitorSupplier Web Service receives a PurchaseOrder as an input and produces a DeliveryOrder as an output. However, invoking Web services often encounter problems which can not be handled by OWL-S due to lack of exception handling mechanism. In case this situation happens, the whole system won't have any response and clients will be in the dark without any knowledge about what's happening and which service is running at the moment.

In order to facilitate the reliability and robustness, the Mapper takes advantage of compensation handling mechanism in BPEL during the transformation procedure. It will select an alternative semantic MonitorSupplier service from a list of available services which already get from the Service Composer to handle the execution failure. Then the Mapper

will construct BPEL activities for the alternative service by utilizing BPEL compensation handling mechanism <compensationHandler>. The alternative semantic service is transformed to a sub-workflow. It is worth to note that the <compensationHandler> can be nested until no more alternative service is available in the list. In such a case, the execution engine will terminate the workflow or report to the client.

If the invocation succeeds, the output which is typed with XML Schema in the BPEL execution level will be fed into KB Update Web Service and transformed into OWL classes typed output. Then the output can be stored into OWL Knowledge Base for later processing by other services.

When the MonitorSupplier service call is finished successfully, the Evaluator Web Service will process the effect of this semantic service if it has. In this scenario, the ownership of monitors will be changed. The succeeding Hardware Shipping semantic service can be mapped in the same way.

During workflow execution, the knowledge base serves as a semantics container to store all the dynamic semantic information. On the other hand, the OWLS2BPEL Mapper embeds all the operations into the BPEL workflow to facilitate interaction with semantic container using ontology processing. In such a way, all the semantics are kept.

VI. CONCLUSION

CVE demands flexible service orchestration and robust workflow execution. To meet the challenge in CVE, in this paper, we propose a Workflow Execution System (WES) by taking marriage strategy between OWL-S and BPEL. On the one hand, having semantic support, OWL-S is used in dynamic service discovery and composition at high level. On the other hand, at the concrete level, industry-based BPEL is exploited in service execution. The Semantic Web Service is equipped with rich business rules modeling of the essential business logic behind the service interface. In order to realize interoperability in OWL-S and BPEL without loss of semantic information, we further proposed an OWLS2BPEL Mapper to facilitate the workflow robustness and support rule evaluation to increase responsiveness to customers. A concrete scenario in PC manufacturing Collaborative Virtual Enterprise is analyzed to demonstrate the effectiveness of our workflow execution system.

REFERENCES

[1] D. J. Mandell and S.A. McIlraith. "A Bottom-Up Approach to Automating Web Service Discovery Customization, and Semantic Translation." in The Proceedings of the Twelfth International World Wide Web Conference Workshop on EServices and the Semantic Web. 2003. Budapest.

[2] OWL-SCoalition. "OWL-S: Semantic Markup for Web Services". W3C Member Submission 2004 Available from: <http://www.w3.org/Submission/OWL-S/>.

[3] A. Alves, et al. "Web Services Business Process Execution Language Version 2.0". 2006 Available from: <http://www.oasis-open.org/apps/org/workgroup/wsbpel/>.

[4] E. Sirin, J. Hendler, and B. Parsia. "Semi-automatic Composition of Web Services using Semantic Descriptions". in Web services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS2003. 2003.

[5] E. Sirin, et al., "HTN Planning of Web Service Composition using SHOP2". Journal of Web Semantics, 2004. 1(4): p. 377-396.

[6] S. R. Ponnekanti and A. Fox. "SWORD: A developer toolkit for Web service composition". in Proceedings of the 11th World Wide Web Conference on Web Services. 2002. Honolulu, HI, USA.

[7] W. Ren, et al. "Searching for Service-Oriented Strategies of Dynamic Composition of Web Services: A Comparative Perspective". in 33rd Annual Conference of the IEEE Industrial Electronics Society. . 2007. Taipei, Taiwan.

[8] S. Liu, R. Khalaf, and F. Curbera. "From DAML-S Processes to BPEL4WS". in Proceedings of the 14th International workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications (RIDE'04). 2004.

[9] MA. Aslam, S. Auer, and J. Shen. "From BPEL4WS Process Model to Full OWL-S Ontology". in Third European Semantic Web Conference (ESWC2006). 2006.

[10] G. Chen, et al. "Collaborative Virtual Enterprise Integration via Semantic Web Service Composition". in The 2nd IEEE Conference on Industrial Electronics and Applications. 2007. Harbin, China.

[11] I. Horrocks, et al. "SWRL: A Semantic Web Rule Language Combining OWL and RuleML". 2004 Available from: <http://www.w3.org/Submission/SWRL/>.

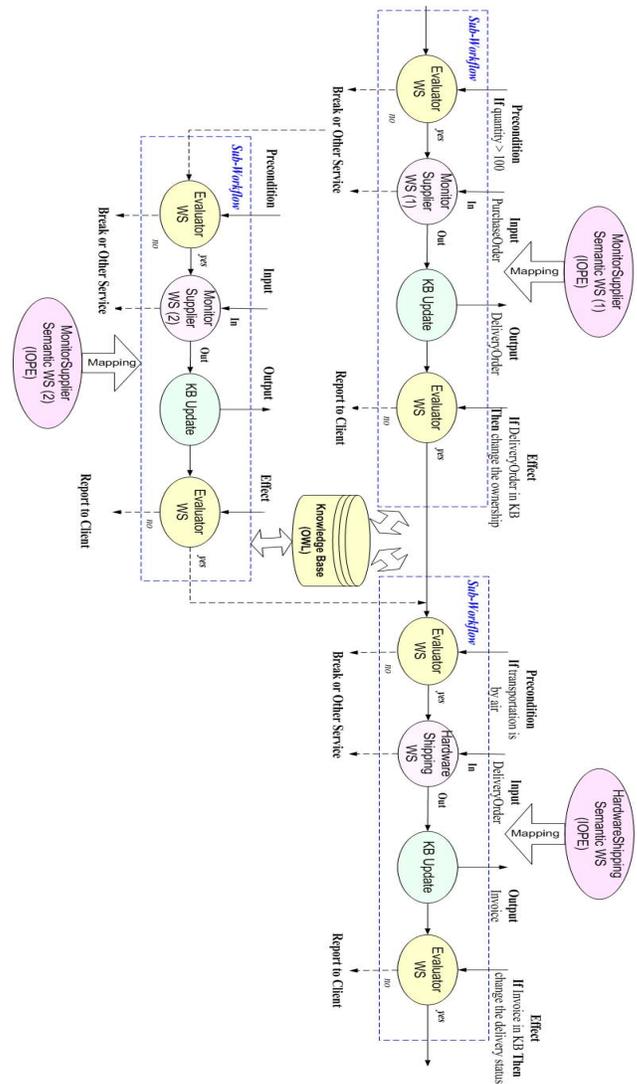


Figure 5. Scenario