

Dynamic Virtual Enterprise Integration via Business Rule Enhanced Semantic Service Composition Framework

G. Chen¹, W. Ren¹, J. B. Zhang², Z. Yang¹, C. P. Low¹, C. Sun³, D. Chen⁴

¹School of Electrical and Electronics Engineering, ³School of Computer Engineering, Nanyang Technological University, Singapore 639798

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

⁴School of Computing & Information Technology, Griffith University, Australia 4111

Abstract—Effective collaboration is crucial to the success of Collaborative Virtual Enterprise (CVE), an emerging business paradigm driven by the increasing trend of globalization. In this paper, we adopt a service-oriented architecture for enterprise integration and collaboration based on Web Service standards. In order to tackle the technical challenge associated with the dynamic formation of business workflows, a service composition framework is presented and analyzed in this paper. Comparing with existing composition systems, our framework enjoys two major improvements: (1) the description of each Web Service is enhanced with rule-based modeling of the essential business logic behind the service interface; and (2) the divide-and-conquer strategy is explored in our framework to handle complex service composition tasks through a hierarchical composition architecture. A PC manufacturing prototyping system further presents a concrete demonstration of our framework in practical applications.

I. INTRODUCTION

The ongoing trend of globalization gives rise 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 [1]. Aimed at fully supporting collaboration in CVE, this paper explores the service-oriented paradigm for enterprise integration based on Web Service standards. In general, the ultimate business goal of a CVE will be divided into a group of closely related sub-goals. Each of these sub-goals will be further realized through a set of business units. Web service, which is a domain-independent, flexible, and highly robust infrastructure technology, will be utilized to implement the interfaces of these business units.

Even with the support of web service technology, CVE cannot be applied to its full potential due to two important issues. (1) Depending on the internal business logic, web services will impose a strong influence on the overall *performance* of a business workflow it participates. Using standard service description languages such as WSDL [2], the suitability of any web service for fulfilling a particular business functionality can only be judged based on the corresponding service inputs and outputs. Obviously decisions

limited to such judgments are highly unlikely to produce satisfactory workflows. (2) Business workflows need to be dynamically formulated, its business processes need to be dynamically configured and executed in order to respond to the dynamic market demand. In an ever-changing business environment, efficient discovery and exploitation of suitable web services to fulfill a business goal is a challenging research issue of CVE.

In an attempt to bring the emerging CVE paradigm into practical use, this paper presents a new service composition framework based on the booming semantic technology. Comparing with relevant service composition systems recently published in the literature, our framework enjoys two major enhancements aimed at solving the two essential research issues mentioned above:

(1) This paper promotes a standard approach for service description. Particularly, the semantic markup language for web services, OWL-S, which is a promising service description standard, will be seamlessly integrated with our service composition framework. In addition to modeling service inputs and outputs using semantics, which has already been suggested by previous research works, service descriptions will be 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), in this paper business rules will be utilized to model the preconditions as well as the effects of invoking any business service in domain-dependent and semantic-rich manner. Through processing these business rules in the context of any given business workflow, our composition framework is able to evaluate (or predict) the detailed impact of a component service on the whole workflow. The process to identify satisfactory workflows will be greatly facilitated by this much improved service description.

(2) The formation of a business workflow is in general a complex and computation-intensive task. To make this task achievable, the divide-and-conquer strategy will be explored in this paper through a hierarchical service composition architecture. In a nutshell, business services in our framework will be grouped into different service categories according to

their internal functionalities. In regard to every request from end customers, a business workflow will be formed at three different levels based on the hierarchical structure of service categories. First an abstract workflow will be constructed at the highest category level. This abstract workflow will be further refined with each business sub-goal in the workflow being realized through a group of business units, giving rise to a concrete workflow at the second composition level. Finally, the workflow grounding will be obtained by selecting specific web services to fulfill each of the enclosed business unit.

We have designed and developed a prototyping system to examine the effectiveness of our composition framework. This prototyping system will be briefly described in this paper. However, as we are still in the process of testing our prototype, detailed experiment results of our composition system will be reported later.

The remaining part of this paper is organized as follows. Section II provides a literature review of closely-related research works. Section III introduces business rules as well as their applications in describing web services. The hierarchical service composition architecture and service composition algorithm will be described in Section IV. Section V presents a prototyping system that exploits our composition framework. Finally Section VI concludes this paper.

II. RELATED SERVICE COMPOSITION SYSTEM

As one of the key technologies of the service-oriented architecture (SOA) and CVE, dynamic service composition has attracted tremendous research attention over the past few years and a wealth of service composition systems have been published recently in the literature. Due to the space limitation, this Section will only briefly summarize a few widely-cited composition systems.

In [3], Sirin *et. al.* proposed a semi-automatic user-driven Web Service composition system. Different from our composition framework, composite services in their system will be established through a chaining procedure driven primarily by human decisions. Essentially, services selected to fuel a business workflow are direct outcomes of non-functional constraints specified by users. At each composition step, an interactive user interface will list a group of Input/Output compatible services for users to select and filter. After making certain selection decisions, another group of compatible services will be further presented to the user until the whole workflow is formed. In comparison to their approach, our framework represents an automatic composition solution with very limited human intervention. High-level automation is achievable in our framework due to the ability to model and process back-ground business logic in the form of service preconditions and effects.

In an attempt to reduce human involvement, AI planning techniques such as the SHOP2 system [4] have been extensively explored to achieve automatic service composition. For example, in [4], an algorithm is proposed to transform service composition tasks to planning problems which are directly solvable through SHOP2. In order to build up a rigorous transformation procedure, assumptions and compromises

have been imposed in [4], leading to undesirable loss of information in service descriptions. Unfortunately, such loss of information will significantly affect the quality of composable business workflows. Without excluding the potential of using AI planning tools as suggested in [4], our framework emphasizes the expressiveness of service descriptions, especially the modeling and processing of business logic behind each business service, as well as its impact on the composed workflows.

It is argued that the notion of composeability significantly facilitates the reuse of services and dynamic composition [1, 5]. Inspired by the concept of composeability, another service composition framework was proposed, which is based on WSDL service descriptions with extended semantic capability [5]. To guide the service composition process, a composability model is proposed in [5] with two sets of composability rules to compare both syntactic and semantic features of Web Services. Syntactic rules include rules for operation modes and the binding protocols of interacting services. Semantic rules comprise (1) message composability, (2) operation semantics composability, (3) qualitative composability, and (4) composition soundness. Our composition framework can be considered as a complement (or extension) to this framework since the composeability of a service will be further evaluated within our framework based on its underlying business logic [1]. As we believe, business logic will play a decisive role in determining the ultimate composeability of a service and their existence may significantly alter the landscape of composable services for any business workflow.

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. Different from our composition framework, SWORD uses Entity-Relation (ER) model to describe Web Services. Given inputs and outputs of a service, a Horn rule is defined with service inputs as *antecedent* and services outputs as *consequent*. When the *antecedent* is satisfied, the *consequent* will take effect. A detailed workflow will be generated using a rule-based expert system upon providing the initial inputs and expected final outputs of the workflow to SWORD. As the composition process considers only the inputs and outputs of existing services, it is highly likely that the generated workflow will fail to meet the real business objectives of collaborating companies in a CVE.

III. USING BUSINESS RULES TO ENHANCE SERVICE DESCRIPTION

This Section presents the technical details of our business rule enhanced service descriptions. 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. In this paper, we adopt Web Ontology Language for Services (OWL-S) as our service description language because it is well-developed with a broad user base.

OWL-S [7] provides a collection of OWL upper ontology that underlines the important functional and non-functional

properties of Web Services. An OWL-S description of a Web Service is composed of three main components: *Service Profile*, *Process Model*, and *Service Grounding*. From the perspective of service composition, *Service Profile* is of particular importance as it presents high-level service abstractions in terms of service inputs, service outputs, the preconditions for accessing a service, as well as the effects of a service call (namely IOPE). In comparison to *Service Profile*, *Process Model* and *Service Grounding* focus more on the low-level implementation details of a Web Service. The *Process Model* describes how the service works. And *Grounding* contains the necessary information for invoking the service.

Service composition in general is a collaborative activity performed at the enterprise level. When a complete business workflow across the local administrative boundaries is to be formulated, the technical detail of each component Web Service usually is of minor significance as compared to the high-level service abstractions. In line with this view, our service composition framework will work primarily with *Service Profiles*. To be more specific, the exact inputs and outputs of a service will be marked up with domain ontologies. And the preconditions and effects of a service, which capture essentially the underlying business logic, will be modeled using business rules.

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. For example, with respect to a service that handles product orders, a plausible precondition would be that the client's bank account must be valid. On the other hand, when modeling service effects, business rules can be used to represent related but different results after calling the service. For example, we can use a business rule to model the simple fact that when a product is purchased, \$100 will be deducted from the client's bank account. Obviously, preconditions and effects together form the logic core of a business service.

In our service composition framework, the Semantic Web Rule Language (SWRL) [8] 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) [9]. 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. A

complete specification for SWRL can be found in [8]. In this paper, we will illustrate the use of SWRL through a simple example.

Suppose we want to model one potential effect of a service that sells PC monitors. Using natural English, this effect 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 less than 500; **THEN** the total service cost incurred is (?qX256) dollars.

The antecedent of rule R1 contains two conditions C1 and C2. Assume that a client issues a request to this service for purchasing twenty 17-inch monitors. This request satisfies both conditions C1 and C2. According to R1, when the service call is finished successfully, (20X256) dollars will be deducted from the client's bank account. This effect is reflected through the change of total service cost contained in the OWL knowledge base.

One prerequisite for specifying R1 using SWRL is to design a proper group of OWL ontologies. In the context of R1, two ontology concepts can be identified, namely, *Monitor* and *Service cost*. These two concepts are implemented via OWL classes. For the *Monitor* class, it contains two properties: *monitorSize* and *quantity*. Meanwhile, we can create another property, namely *cost*, for the *Service cost* class. Based on the identified OWL classes and properties, the SWRL description for rule R1 is presented in Fig. 1.

```

Monitor(?m)  $\wedge$  quantity(?m, ?q)  $\wedge$  swrlb:lessThan(?q, 500)  $\wedge$ 
monitorSize(?m, ?s)  $\wedge$  swrlb:equal(?s, 17)  $\wedge$  swrlb:multiply(?c, ?q, 256)
 $\wedge$  ServiceCost(?sc)  $\rightarrow$  cost(?sc, ?c)

```

Fig. 1. SWRL description for rule R1

As shown in Fig. 1, seven atomic conditions are involved in defining the antecedent of R1. They are linked together using the conjunction relation \wedge . It is easy to verify that the first three atomic conditions realize condition C2. The fourth and the fifth atomic conditions together realize condition C1. The build-in command swrlb:multiply is utilized in the sixth atomic condition to calculate the total service cost. Finally, the calculated cost is used to update the *cost* property of *Service cost* in the consequent of R1.

A business rules processing subsystem (BRPS) has been developed in this paper to support our service composition framework. Our BRPS relies on two open-source projects, Protégé [10] and Jess Rules Engine [11]. 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.

The impact of component services on the whole business workflow is reflected through the modified knowledge base.

Our BRPS is designed to handle workflows with realistic complexity. Upon evaluating a workflow, 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 evaluated in sequence based on their preconditions and effects. The following two rules are imposed to control the evaluation procedure:

- R2. If the preconditions of a service cannot be satisfied with the given OWL knowledge base, then a failure will be returned by the BRPS, indicating that the workflow is *unfeasible*.
- R3. If a service is *applicable*, then the effects of the service will be further processed to infer new changes to the OWL knowledge base. This updated OWL knowledge base will be utilized to assess succeeding services in the workflow.

When business rules associated with all component services in a workflow have been processed, the performance of the workflow can be derived from the obtained OWL knowledge base according to domain-dependent criteria, such as the total service cost and total service processing time. Through changing both the structure of workflows and their component services, our service composition framework therefore is able to identify the most suitable workflow based on the evaluated performance of these workflows.

IV. HIERARCHICAL SERVICE COMPOSITION ARCHITECTURE

The formation of a business workflow is in general a complex and computation-intensive task. To make this task achievable, the divide-and-conquer strategy will be explored in this paper via a hierarchical service composition architecture. It is common knowledge that services involved in a business workflow can be divided roughly into several categories. For example, in a manufacturing-centric application, a typical high-level workflow can be identified as in Fig. 2.

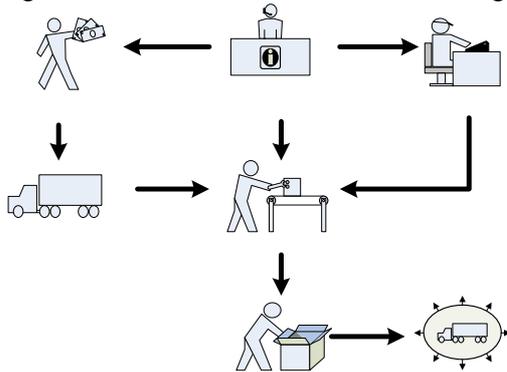


Fig. 2. High-level manufacturing-centric business workflow.

As shown in Fig. 2, customers' requests are to be realized through a sequence of six major steps. A request will be handled first by order processing service to determine the feasibility of the request and to generate detailed requests for raw material procurement and product manufacturing. These

detailed requests will drive the succeeding steps until requested products have been manufactured and delivered to customers.

Each step in Fig. 2 is a high-level abstraction of a series of business activities. For example, *Product Manufacturing* in Fig. 2 is usually comprised of a chain of elementary manufacturing units, ranging from raw material processing to product assembly and quality assurance. In view of this, a hierarchical structure of service categories can be established. The six steps in Fig. 2 stand for top-level service categories. Underlying each step, detailed service categories can be identified and realized through Web Services. Service categories are domain dependent. In this paper, the available categories as well as their relationships will be modeled through OWL classes and the *subclassof* property.

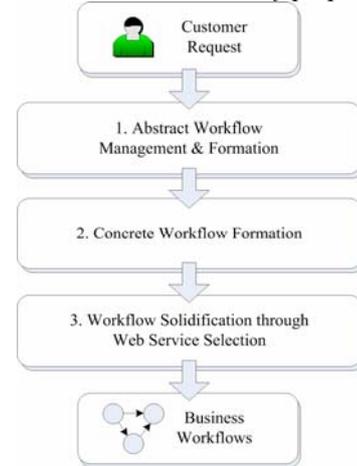


Fig. 3. Hierarchical service composition architecture.

We introduce a new concept termed *business unit* in our service composition framework. A business unit refers to an atomic business operation that belongs to the lowest-level service categories. For instance, the PC casing manufacturing is normally performed by a single factory within a business workflow and therefore should not be divided further into more manufacturing activities. As a result, PC casing manufacturing will be treated as an individual business unit.

In accordance with a service category, a *business unit* also contains a group of service inputs and service outputs. For the case of PC casing manufacturing, the input would be a manufacturing request, and the output would be the manufactured casings. However, a business unit is different in concept from a web service. In essence, it is convenient to view a business unit as representing a group of web services that belong to the same service category and exhibit the same set of service inputs and service outputs.

Based on service categories and available business units, a hierarchical service composition architecture has been proposed in Fig. 3. According to Fig. 3, in order to satisfy any business request from end customers, business workflows will be constructed at three different abstraction levels. These three levels will be discussed separately as follows.

Composition level 1: In line with our manufacturing scenario in Fig. 2, whenever a customer orders certain amount of

products, an abstract workflow will be formulated initially at the highest service category level. This workflow serves as the basis for the subsequent formation of concrete workflows at composition level 2. From the problem-solving perspective, abstract workflows stand for general divisions of complex service composition problems into multiple sub-problems. Each sub-problem is associated with a distinct step as in Fig. 2. For convenience, these sub-problems are also termed sub-goals.

It is eligible to view abstract workflows as service composition templates. They can effectively reduce the number of alternative workflows to be considered during the service composition process. Instead of challenging the validity of any given abstract workflow, the composition process will concentrate on realizing every sub-goal introduced in the workflow. For this reason, abstract workflows are usually pre-defined and centrally managed by system administrators. Overtime new composition solutions may also be discovered and added into our composition framework to expedite the service composition process.

Composition level 2: Based on the abstract workflows identified at composition level 1, concrete business workflows will be further formulated at this level. Different from level 1, in order to address the inherent dynamicity of business applications, service composition algorithms will be explored to dynamically construct a chain of business units with respect to every sub-goal in the abstract workflow.

To facilitate the construction of concrete workflows, a service composer based on the forward chaining algorithm has been designed and developed by us. The details of our composer can be found in [12]. By leveraging the semantic markup of service inputs and service outputs associated with every business unit, our composer will aggressively chain a group of business units together in order to satisfy the specific requirements of any sub-goals. We will demonstrate our composer through a prototyping system in the next Section.

Composition level 3: In order to construct an *executable* business workflow, all the business units of a concrete workflow obtained at composition level 2 must have proper *groundings*. This is achieved through a selection process that associates each business unit with a Web Service that belongs to that unit. Obviously, a single concrete workflow can lead to diverse groundings. To determine the eligibility of any grounding, the BRPS introduced in Section III will be used to process the preconditions and effects of each chosen Web Service along the workflow. Groundings that produce the highest overall performance will be employed to generate the final business workflow.

The service selection problem as described above can be very complex since the services in the workflow may interact with each other in highly sophisticated manner, resulting in capricious outcomes. To tackle this problem, stochastic search strategies based on evolutionary algorithms have been exploited in our service composition system. The algorithm is still being implemented currently and its details will be omitted in this paper. It is to be noticed that our service composition framework does not exclude other algorithms

from being applied to solve the selection problem. Depending on the application domains, efficient algorithms may be designed in the future to further improve practicality of our framework.

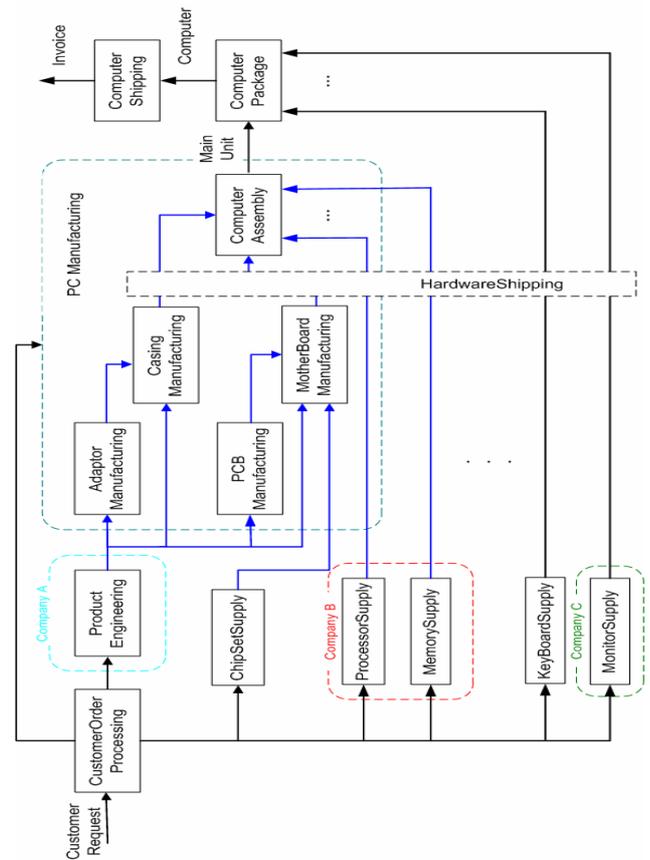


Fig. 4. PC Manufacturing CVE.

V. PC MANUFACTURING PROTOTYPING SYSTEM

As a practical demonstration of our composition framework, a prototyping system is being built based on a PC manufacturing scenario. With new material and technology emerging, computer products upgrade frequently. In this ever-changing environment, business workflows for PC manufacturing need to be dynamically formulated, its business processes need to be dynamically configured and executed in order to respond to the dynamic market demand. Thus, PC manufacturing business possesses dynamism feature. In addition, most of computer manufacturers increasingly concentrate on their core competencies and outsource other functions to third parties, such as product design function and casing manufacturing, etc. In order to fulfill market demand for computers, a timely alliance of enterprises will be formed to share their functionalities and resources in a collaborative manner. As shown in Fig. 4, company A, B, C and many others join together to formulate a Collaborative PC manufacturing Virtual Enterprise.

From the functional perspective, our CVE system aims at automating the manufacturing process after receiving customers' requests. Its proper operation relies essentially on the collaboration of various business functionalities (e.g. product engineering, material management, manufacturing and ship-

ping, etc) as shown in Fig. 2. The collaboration of these functionalities forms an abstract workflow at the composition level 1.

Each of the functionality in Fig. 2 is a high-level abstraction of a series of business units. We use a service composer based on the forward chaining algorithm to identify a group of business units that together form the concrete workflow at composition level 2 (Ref. Section IV). The composer relies on the semantic markup of service inputs and service outputs associated with every business unit.

For example, to realize the *Product Manufacturing* function at level 1 in Fig. 2, a group of business units, including *Adaptor Manufacturing*, *Casing Manufacturing*, *PCB Manufacturing*, *MotherBoard Manufacturing* and *Computer Assembly*, etc, will be chained together to establish the path from initial customer requests to manufactured main units. The details can found in Fig. 4 (business units connected by arrows in blue). In addition to *Product Manufacturing*, the forward-chaining algorithm will also be applied to other abstract functions such as *Material Purchasing* at level 1. When this process finishes, the complete PC manufacturing workflow that links suppliers, manufactures, and distributors as a whole will be formed (depicted in Fig. 4).

In order to execute the concrete workflow obtained at composition level 2, each business unit in the workflow must be substituted by a Web Service with compatible service inputs and outputs. To determine the eligibility of choosing any Web Service to replace a specific business unit, the BRPS introduced in Section III will be utilized at composition level 3 to evaluate the preconditions and effects of this service within the context of the workflow. Only the Web Service that produces the highest overall performance (e.g. total service cost) will be selected to generate the executable business workflow.

TABLE I
MONITOR PRICE OF SERVICES W_1 AND W_2

Monitor (Cost S\$)		Quantity			
		< 500		>= 500	
		W_1	W_2	W_1	W_2
Monitor	17	256	260	243	240
Size (inch)	19	280	280	266	266

As an example, suppose that two Monitor Supply services W_1 and W_2 (Ref. Fig. 4) are available for selection in a given concrete workflow. The business rules that govern the respective cost (considered as service effects) of the two services are summarized through a decision table (Table I). The upper-left cell in table I stipulates that when the monitor's demand quantity is less than 500, the unit price of 17-inch monitors to be charged by service W_1 will be 256 dollars. Obviously, in this situation, service W_1 will stand for a better choice than service W_2 .

VI. CONCLUSION

Effective collaboration is crucial to the success of Collaborative Virtual Enterprise (CVE), an emerging business paradigm driven by the increasing trend of globalization. In

this paper, we adopted a service-oriented architecture for enterprise integration and collaboration based on Web Service standards. Aimed at tackling the technical challenge associated with the dynamic formation of business workflows, a service composition framework based on the booming semantic technology was presented and analyzed in this paper. Comparing with existing composition systems, our framework enjoys two major improvements: (1) the description of each Web Service is enhanced with rule-based modeling of the essential business logic behind the service interface; and (2) the divide-and-conquer strategy is explored in our framework to handle complex service composition tasks through a hierarchical composition architecture built on top of existing service categories and basic business units. A PC manufacturing prototyping system further presented a concrete demonstration of our framework in practical applications.

ACKNOWLEDGMENT

This work was supported in part by the Agency for Science, Technology, and Research (A*STAR) of Singapore under SERC TSRP IMSS Projects 0521160070 and 0521160078.

REFERENCES

- [1] Z. Yang, J. B. Zhang, and C. P. Low, "Towards Dynamic Integration of Collaborative Virtual Enterprise using Semantic Web Services," *The 4th International IEEE Conference on Industrial Informatics*, Singapore, 2006.
- [2] R.Chinnici, M.Gudgin, J.-J.Moreau, J.Schlimmer, and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0," 2.0 ed: W3C, 2004.
- [3] E. Sirin, J. Hendler, and B. Parsia, "Semi-automatic Composition of Web Services using Semantic Descriptions.," *Web services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS2003*, 2003.
- [4] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau, "HTN Planning of Web Service Composition using SHOP2," *Journal of Web Semantics*, vol. 1, pp. 377-396, 2004.
- [5] B. Medjahed, A. Bouguettaya, and A.K. Elmagarmid, "Composing Web Services on the Semantic Web," *The VLDB Journal*, vol. 12, 2003.
- [6] S. R. Ponnekanti, and A.Fox, "SWORD: A developer toolkit for Web service composition.," *Proceedings of the 11th World Wide Web Conference on Web Services*, Honolulu, HI, USA, 2002.
- [7] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Rayne, E. Sirin, N. Srinivasan, and K. Sycara, "OWL-S: Semantic Markup for Web Services," in W3C Member Submission, 2004.
- [8] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Groszof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," W3C, 2004.
- [9] G. Klyne, and J. J. Carroll, "Resource Description Framework (RDF): Concepts and Abstract Syntax," in W3C Recommendation 2004.
- [10] "Protégé Ontology editor and knowledge-base framework", <http://protege.stanford.edu/>
- [11] E. Friedman-Hill, "Jess the Rule Engine for the Java Platform ", <http://www.jessrules.com/>.
- [12] G. Chen, J. B. Zhang, C. P. Low, Z. Yang, W. Ren, and L. Zhuang, "Collaborative Virtual Enterprise Integration via Semantic Web Service Composition," *The 2nd IEEE Conference on Industrial Electronics and Applications*, Harbin, China, 2007.