Searching for Service-Oriented Strategies of Dynamic Composition of Web Services: A Comparative Perspective

Wei Ren¹, Gang Chen¹, David Chen², Chor Ping Low¹, Chengzheng Sun³, Jing Bing Zhang⁴ and Zhonghua Yang¹

¹School of Electrical and Electronics Engineering, ²School of Computing & Information Technology, Griffith University, Australia 4111
³School of Computer Engineering, Nanyang Technological University, Singapore 639798
⁴Singapore Institute of Manufacturing Technology (SIMTech), Singapore 638075

Contact email: WeiRen@ntu.edu.sg

Abstract – Determining the service composition strategy suitable for service-oriented applications is an essential design issue. As a first step towards understanding service composition, in this paper, we present a comparison study with two different service composition strategies. One is driven by industrial application—BPEL, the other is inspired by the booming research development in semantic web—OWL-S. We compare the two major strategies with each other through the Collaborative Virtual Enterprise (CVE) scenario. Five key dimensions for service composition requirement have been identified. These dimensions belong to two types. One type of dimensions is essential requirement including component model, orchestration model and data flow model. The other type is desirable requirement including error handler and transaction, and quality of service. We provide the comparison between BPEL and OWL-S against the five dimensions. Our comparison shows that OWL-S is more suitable for dynamic environments and BPEL is an ideal choice for controlled workflow.

I. INTRODUCTION

Service-Oriented Architecture (SOA) has emerged as a promising paradigm for building loosely coupled, standardized and Web-enabled distributed applications and systems. SOA is an architectural style for building software applications that use services available in the network [1]. The concept of services refers to as a web-based software application which exposes its well-defined and self-described interface over the Internet using XML-based technologies.

Nowadays, many enterprises expose their business applications as Web Services which enables the enterprise collaboration, both within their organization and across enterprise boundaries. However, in many cases, a standalone service has a limited functionality and is not sufficient to meet user’s request. To achieve complex business goals in real-life applications, very often execution of multiple services should be orchestrated through service composition. Service composition is widely considered as a promising solution towards rapid application development, service reuse, and enterprise application integration [2].

There are two active research communities for service composition: one is driven by industrial applications, the other is inspired by the booming research development in semantic web. Over the years, many technologies have been introduced to support service composition, such as BPEL, Web Service Choreography Description Language (WSDL), Business Process Modeling Language (BPML) [3]. Among them, we believe BPEL is most matured technology with a complete bundle of software tools and running environments catered for business applications. In this paper, we will compare BPEL with Web Ontology Language for Services (OWL-S), which is a technology based on semantic web and has attracted enormous research interests in the academic community. Our intention is that: by comparing these two representative technologies, we are able to have a general view of the state-of-the-art research on service composition, which will guide us for our future research endeavors.

Our major research interest is to support collaborative virtual enterprise through SOA. Based on the specific requirement for dynamic service composition in CVE, we have identified five main comparison dimensions, which we believe are essential and desirable for service composition. They are component model, orchestration model, data and data flow model, error handling and transactions, and quality of service. In this paper we will provide a detailed comparison between BPEL and OWL-S with regard to the five dimensions.

This paper is organized as follows. In section II the Web Service infrastructure is introduced. Section III gives an overview of service composition requirement. Section IV describes a scenario—Collaborative Virtual Enterprise (CVE) that has been developed. In section V, we compare the two different service composition strategies BPEL and OWL-S in details. We give a discussion in section VI. Section VII concludes the article.
II. STRATEGIES OF WEB SERVICE COMPOSITION

As two leading emerging service composition strategies, in this section, we will introduce the overview of BPEL and OWL-S.

A. BPEL

In a service-oriented environment, many different business partners expose their functionalities as web services. There is a requirement to have the ability to define logic over a set of service interactions. Thus, Web Services Business Process Execution Language (also known as WS-BPEL, or BPEL for short) [4] was developed to define business process behavior by orchestrating web services. BPEL is an XML language for process-oriented service composition which combines the IBM Web Service Flow Language (WSFL) and Microsoft XLANG. It is developed by IBM, Microsoft, BEA, SAP and Siebel. The current version is BPEL4WS 1.1, the next version, renamed as WS-BPEL 2.0, was submitted to the Organization for the Advancement of Structured Information Standards (OASIS) and is still in the standardization process.

Comparing with Web Services which use stateless interaction model, business processes which are defined in BPEL and built on WSDL represent stateful long-running interactions through Web Service operations. Whenever a client starts a new business process, a new instance of that process is created and lasts until the instance ends. During its lifecycle, a business process instance might communicate with other business partners. It is important to have a message correlation mechanism to guarantee that messages are delivered to the correct instance of the business process [4].

B. OWL-S

The service-oriented environment can be enhanced with rich semantics using Semantic Web technologies. A Semantic Web service is a Web Service whose description is in a language that has well-defined semantics [5]. To realize Semantic Web Services, OWL-S was proposed to facilitate the automation of Web Service tasks including service discovery, execution, interoperation, composition and execution monitoring [6].

OWL-S is a collection of loose upper ontologies and is description oriented. It focuses on capabilities, functional properties, and other non-functional properties (such as QoS). The behavior of a web service is modeled with a process/action paradigm rooted in situation calculus. Inputs and outputs of a service are named and typed using either OWL-S ontologies or data types that XML Schema provides. They together constitute the main interface for the purpose of interacting with the service [1].

The current version of OWL-S builds on top of Web Ontology Language (OWL) which is the W3C semantic markup language for publishing and sharing ontologies on the Web. OWL is the most expressive of the ontology language based on Resource Description Framework (RDF), RDF Schema (RDF-S), DARPA Agent Markup Language and Ontology Inference Layer (DAML + OIL) which was the unifying effort of the American proposal DAML and the European language OIL.

III. SERVICE COMPOSITION REQUIREMENT

In this section, we will present and explain the five dimensions of a service composition model which are utilized to compare BPEL and OWL-S. As we mentioned, this paper serves as the first step towards deeply understanding the service composition problem. These five dimensions are chosen because they are most relevant to CVE, which is our main research area. Similar dimensions have also been identified recently in the literature [7].

A. Component Model

Services can be seen as a set of network endpoints. In a service composition, each component service should be able to be defined in a structured way to identify its nature [7]. As a general requirement, component model should be able to describe the type of component services and service interface.

B. Orchestration Model

In a service composition, orchestration describes the way in which different services can be brought together into a coherent composite service to provide a value-added service. It is realized by control flow constructs that specify the order in which individual operations of services are executed, and the conditions under which a certain service may or may not be invoked [7]. One composition language should provide the control constructs to define logic over a set of service interactions. To support a variety of practical composition requirements, control construct in many aspects should resemble a programming-language. Specifically, the control construct should be able to break up the flow of execution by employing decision making (if-then-else, switch), looping (while, for), and branching (break, return, continue), enabling the composed service to conditionally execute particular operation. Therefore, control flow is one of the essential requirements for service composition.

C. Data and Data Flow Model

Service components interact with each other by exchanging data which should be defined and accessed in an explicit ways [7]. From service composition point of view, data flow refers to the flow of information in a composed service. One composition language should provide means to define data, to describe how data can be exchanged between component services being composed, and to describe inputs and outputs of the composed service.

D. Error Handling and Transactions

It is desirable for a composition system to offer an efficient and effective error handling and compensation mechanism. Such mechanism will help to identify faults (i.e. error handling mechanism) and undo work that is partially or successfully completed (i.e. compensation mechanism). When error occurs
during execution, the system can deal with exceptional behavior to guarantee system stability.

E. Quality of Service (QoS)

With the increase of Web Services as a business solution to enterprise application integration, the quality of service (QoS) offered by Web Services will become more and more important for service providers and their partners. A better QoS for a Web Service will make it more competitive than others. Therefore, it is desirable to carry the quality of service information which is a key non-functional property for service composition.

IV. A Scenario—Collaborative Virtual Enterprise (CVE)

We have built a virtual enterprise manufacturing system using Web Service technologies. In order to further support collaborative operation in this system, our next step is to realize dynamic service composition. Dynamism is a salient feature of CVE. A CVE needs to be dynamically formulated, its business processes need to be dynamically configured and executed to respond to the dynamic market. A CVE needs to quickly integrate its systems, applications, and services to fulfill its business goal. Within a service-oriented environment, the enterprise integration problem can be further viewed as a problem of service composition [1].

![Fig. 1. High level service process in an enterprise manufacturing system](image)

Each component in the high level service process as shown in Fig. 1 is itself a composite service. For example, Plan Schedule component is virtualized through several web services including Plan Master Product Schedule service, Plan Material service, Plan Capacity service and so on as shown in activity diagram Fig. 2. The Plan Master Product Schedule service creates the weekly production plan and calculates the Required Capacity and Required Stock Material. The Plan Capacity and Plan Material services evaluate the initial production quantities to determine whether sufficient capacity and material exist. As we can see in Fig. 2, Plan Master Product Schedule service and the other two services (Plan Material and Plan Capacity) are in the sequence order. Plan Material service and Plan Capacity service can be executed in parallel.

V. Comparison of BPEL and OWL-S

In the sequel, we will compare BPEL and OWL-S based on the five requirements identified in Section 3. In each subsection, we will first discuss BPEL followed by OWL-S.

A. Component Model

In BPEL, the components are WSDL services. The result of composite service is called process. The external participating web service that interacts with the business process are called partner. BPEL defines <partnerLinks> for describing the behavior of a business process based on interactions between the process and its partners. WSDL is used to describe the service interfaces for participating partner [4].

A BPEL process is composed of a set of basic or structured activities. A basic activity is an actual "component" which describes elemental step of the business process behavior. A structured activity provides higher-level business logic and combines multiple basic or structured activities. They can be nested and define the order in which a set of activities is executed.

In contrast with BPEL, OWL-S uses an upper ontology to describe Web Services. It consists of three types of knowledge about a service [6]: ServiceProfile (what the service does), ServiceModel (how the service works), ServiceGrounding (how the service is accessed).

For every distinct published Web service, there exists an instance of SERVICE class. Each instance of a SERVICE will present (line 2) a ServiceProfile description, be describedBy (line 3) a ServiceModel description, and support (line 4) a ServiceGrounding description. See snippet in Fig. 3.

The ServiceProfile provides a concise description of the service to a registry in order to facilitate the search process, but once the service has been selected the Profile is useless. The client will use the ProcessModel to control the interaction with the service [6]. Both the ServiceProfile and the ServiceModel are abstract representations in OWL-S. Only the Service Grounding is the concrete level of specification which defined...
the details of how to access the service involving communication protocol, message format, serialization, transport and addressing. OWL-S makes use of WSDL as grounding mechanism. An OWL-S atomic process corresponds to a WSDL operation, the input and output sets of an OWL-S atomic process corresponds to WSDL’s message, the types (OWL classes) of inputs and outputs of an OWL-S atomic process corresponds to WSDL’s abstract types. The grounding connects the implementation of a Web Service with its semantic description.

It is noteworthy that each component service which is also described as OWL-S file has the same structure as the composed one.

```
1 <service:Service rdf:ID="PlanScheduleService">
2   <service:present rdf:ID="PlanScheduleProfile"/>
3   <service:describedBy rdf:ID="PlanScheduleProcess"/>
4   <service:supports rdf:ID="PlanScheduleGrounding"/>
5 </service:Service>
```

Fig. 3. Service in OWL-S

**B. Orchestration Model**

BPEL allows the definitions of structured activities, which combines multiple basic or structured activities to define order in which a set of activities is executed [4]. Fig. 4 shows the simplified snippets of control flow in BPEL for the composition of three services in Fig 3. The `sequence` structured activity (line 1) indicates the following activities `<receive>`, `<invoke>` activity and `flow` structured activity are executed sequentially. The `<receive>` basic activity (line 2) waits for receipt of a message from a partner "clientPL" via the predefined operation "process". The `<invoke>` basic activity (line 6) is used to invoke PlanMasterProductSchedule operation offered by a service. The `flow` structured activity (line 9) defines the two `<invoke>` activities (PlanMaterial and PlanCapacity operations at line 14 and 20) which should execute concurrently. Other structured activities in BPEL include `<if>`, `<while>` and `<pick>` activities.

The OWL-S process upper ontology provides similar control constructs as BPEL. The Service Model describes the execution of a service in details [6]. A detailed perspective of a service can be viewed as a Process which is defined as a subclass of the ServiceModel. This part of OWL-S describes the service interaction protocol according to the flow of data and control between services. OWL-S defines three subclasses of Process. Atomic processes have no subprocesses and can directly execute in a single step which corresponds to the invocation of a web service operation. Each atomic process is associated with a grounding that defines the details of how to access the service. Simple processes are not associated with a grounding and are single-step executions. They are used as abstraction elements for either atomic or composite processes. Composite processes can be decomposed into other processes by using control constructs such as Sequence, If-then-else, Split, Split + Join and so on. The decomposition shows how the various inputs and outputs of the subprocesses are mapped [6].

```
1 <sequence>
2   <receive partnerLink="clientPL"
3     portType="PlanSchedulePT" operation="process" .../>
4   ...>
5   <invoke partnerLink="PlanMasterProductSchedulePL"
6     portType="PlanMasterProductSchedulePT"
7     operation="PlanMasterProductSchedule" ... />
8   <flow>
9     <sequence>
10    ...>
11   <invoke partnerLink="PlanMaterialPL"
12     portType="PlanMaterialPT"
13     operation="PlanMaterial" ... />
14   <sequence>
15    ...>
16   <invoke partnerLink="PlanCapacityPL"
17     portType="PlanCapacityPT"
18     operation="PlanCapacity" ... />
```

Fig. 4. Control Flow in BPEL

Fig. 5 shows the snippets of control flow in OWL-S for the composition of three services in Fig. 2. The Sequence control construct (line 1) indicates the following component processes (PlanMasterProductSchedule at line 5 and Split-Join component starting at line 10 ) can be executed sequentially. The Split-Join control construct (line10) defines concurrent execution of two process components (PlanMaterial at line 14 and PlanCapacity at line 19).

```
1 <process:Sequence>
2 <process:components>
3 <process:ControlConstructList>
4 <list:first>
5   <process:Perform rdf:ID="PlanMasterProductSchedule"/>
6 </list:first>
7 <list:rest>
8 <process:ControlConstructList>
9 <list:first>
10   <process:Split-Join>
11     <process:components>
12       <process:ControlConstructBag>
13 <list:first>
14       <process:Perform rdf:ID="PlanMaterial"/>
15 </list:first>
16       <list:rest>
17       <process:ControlConstructBag>
18 <list:first>
19       <process:Perform rdf:ID="PlanCapacity"/>
```

Fig. 5. Control Flow in OWL-S

With regarding to the control flow, the BPEL approach is easier to understand and use than OWL-S. The previous approach is more close to conventional programming language. The same conclusion can be found in the following section with respect to data flow.
C. Data and Data Flow Model

Variables in BPEL provide means to maintain the state of a business process which includes the messages that are exchanged as well as intermediate data used in business logic and in composing messages sent to partners [4]. BPEL variables which are declared by a name and type are similar to variables in programming languages. BPEL uses three kinds of variable declarations: WSDL message type, XML Schema type, and XML Schema element. The <assign> activity (line 6) can be used to copy data from one variable to another as show in Fig. 6. Two variables PlanMasterProductSchedule_Output (line 1) and RequiredStockMaterial_Input (line 3) are declared as WSDL message type. The first variable is the output of Plan Master Product Schedule service. And the second one is the input of the following component service--Plan Material. BPEL realizes the input/output mapping through <assign> activity by copying data from PlanMasterProductSchedule_Output variable to RequiredStockMaterial_Input variable.

In OWL-S, a composite process is one that maintains some state; each message the client sends advances it through the process [6]. A process can produce some new data based on data it is given and the world state. The data transformation is described by the inputs and outputs of the process. Moreover, a process can make a state change in the world. The state transition is described by the preconditions and results of the process. This together is known as IOPR. When defining a composite process, we need to describe how the output of a process component can be mapped into input of the following step. The input and output parameters can be declared as XML Schema data types or OWL classes (concepts). Fig. 7 shows the snippet of specifying data flow and variable bindings in OWL-S.

The composite process PlanScheduleProcess (line 2) has an input ProduceSpecificationSchedule (line 4) which is declared as a string type (line 7) in XML Schema. One of its outputs is DailySchedule (line 12) which is declared as a concept in predefined ontology Manufacture.owl (line 15). Line 21 to 32 specifies the data flow from process PlanMasterProductSchedule (line 28) to process PlanMaterial (line 19). That is the output variable PlanMasterProductSchedule_Output (line 30) from process PlanMasterProductSchedule is mapped into the input variable (line 24) RequiredStockMaterial_Input.

```
1 <variable name="PlanMasterProductSchedule_Output"
2  messageType="PlanMasterProductScheduleResponse"/>
3 <variable name="RequiredStockMaterial_Input"
4  messageType="PlanMaterialRequest"/>
5 ... ...
6 <assign>
7 <copy>
8 <from variable="PlanMasterProductSchedule_Output" ... />
9 <to variable="RequiredStockMaterial_Input" ... />
10 </copy>
11 </assign>

Fig. 6. Data Flow in BPEL
```

```
1 <process:CompositeProcess
2  rdf:about="#PlanScheduleProcess">
3 <process:hasInput>
4  <process:Input rdf:ID="ProducSpecificationSchedule">
5   <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
6     "http://www.w3.org/2001/XMLSchema#anyURI">
7   </process:parameterType>
8   <process:Inputrdf:ID="ProducSpecificationSchedule">
9 </process:Input>
10 ... ...
11 <process:hasOutput>
12  <process:Output rdf:ID="DailySchedule">
13   <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
14     "http://www.icis.ntu.edu.sg/Manufacture.owl#DailySchedule"
15   </process:parameterType>
16 </process:Output>
17 ... ...
18 <process:Perform rdf:about="#PlanMaterial">
19 ... ...
20 <process:hasDataFrom>
21  <process:InputBinding>
22   <process:toParam rdf:resource="#RequiredStockMaterial_Input"/>
23 </process:InputBinding>
24 <process:fromProcess rdf:resource="#PlanMasterProductSchedule">
25  <process:ValueOf/>
26   <process:fromProcess rdf:resource="#PlanMasterProductSchedule">
27  </process:ValueOf>
28  <process:ValueOf>
29   <process:theVar rdf:resource="#PlanMasterProductSchedule_Output"/>
30 </process:ValueOf>
31 </process:ValueOf>
32 </process:ValueOf>
```

Fig. 7. Data Flow in OWL-S

D. Error Handling and Transactions

BPEL introduces systematic mechanisms for dealing with business exceptions and processing faults. While a BPEL process is running, faults can originate from partner services that are invoked or from within the BPEL process itself. For faults that are due to partner service invocations, the WSDL defines a fault message to be returned. Within the process, abnormal behavior can be either thrown explicitly through <throw> activity or identified by the BPEL Execution engine. BPEL fault handlers specifies activity which should be executed when a fault caught by catch or catchAll construct. In addition, BPEL compensation handlers provides a mechanism to define activity to reverse intermediate results in cases where exceptions occur or a partner requests reversal [4]. Snippet in Fig. 8 shows the fault handlers in BPEL process.

When a fault cannotPlanMaterial occurs, regular processing is terminated and control is transferred to corresponding faultHandlers which uses a <reply> activity to return a fault to the client.

Inside the OWL-S specification 1.2, we can not find any mechanisms specifically designed for error handling and compensation.
Fig. 8. Fault handlers in BPEL

E. Quality of Service (QoS)

In addition to describe what the service does at a high level, the OWL-S Service Profile allows the description of properties that are used to describe features of the service, such as quality of the service. Moreover, Service Profile provides an unbounded list of service parameters that can contain any type of information which might include QoS parameters such as responsiveness and efficiency of the service [6].

In contrast, BPEL's constructs do not support the capability to describe the QoS. However, BPEL's extensibility makes up the deficiency by using WS-Policy to attach QoS policies to different parts of BPEL process [8].

VI. DISCUSSION

BPEL is high-level workflow language that can be used to create business processes that constitutes a number of related business functions [11]. However, the business process is pre-defined. In contrast to the manual composition techniques provided by BPEL, OWL-S has more ambitious goal to enable automatic service discovery, invocation, composition and interoperation. By describing domain concepts (IOPR) using ontology, OWL-S provides a semantic description for discovering and composing web services automatically [10]. The semantic description is unambiguously machine-interpretable. It supports rigorous reasoning about service description and capability. It facilitates maximal dynamism in Web service discovery, selection, composition, negotiation, and invocation. This feature provides the sound technical foundation for achieving dynamic integration via service-composition.

Comparing with BPEL which has been developed for many years and almost reaches maturity, OWL-S is still in its early stage. The adoption of OWL-S hasn't been prevalent for lack of tools to support the development. Furthermore, the learning curve for the ontology languages can be steep for common developers [9]. Unlike OWL-S, BPEL gets more supports from industry. Currently, several BPEL orchestration servers which provide a runtime environment for executing BPEL business processes are available for both J2EE and Net platform. They are Oracle BPEL Process Manager, Active BPEL Engine, IBM WebSphere Business Integration Server, BEA WebLogic Integration, Microsoft BizTalk and OpenStorm Service Orchestrator. In addition, BPEL has correspondent process design environment with a graphical user interface such as Oracle JDevelop.

VII. CONCLUSION

Determining the service composition strategy suitable for service-oriented applications is an essential design issue. As a first step towards understanding service composition, in this paper, we present a comparison study with two different service composition strategies. One is driven by industrial application—BPEL, the other is inspired by the booming research development in semantic web—OWL-S. We compare the two major strategies with each other through the Collaborative Virtual Enterprise (CVE) scenario. Five key dimensions for service composition requirement have been identified. These dimensions belong to two types. One type of dimensions is essential requirement including component model, orchestration model and data flow model. The other type is desirable requirement including error handler and transaction, and quality of service. We provide the comparison between BPEL and OWL-S against the five dimensions. Our comparison shows that OWL-S is more suitable for dynamic environments and BPEL is an ideal choice for controlled workflow.

ACKNOWLEDGMENT

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

REFERENCES