An Approach to Abstracting and Transforming Web Services for End-user-doable Construction of Service-Oriented Applications*

Jian Yu[1], Jun Fang[1,2], Yanbo Han[1], Jianwu Wang[1,2], and Cheng Zhang[1,2]

[1] Software Division, Institute of Computing Technology,
CAS, Beijing 100080
{yujian, yhan}@software.ict.ac.cn
http://sigsit.ict.ac.cn/
[2] Graduate School of CAS, Beijing 100039
{fangjun, wjw, zhch}@software.ict.ac.cn

Abstract: End-user-programmable business-level services composition is an effective way to build virtual organizations of individual applications in a just-in-time manner. Challenging issues include how to model business-level services so that the end users can understand and compose them; how to associate business-level services to underlying Web services. This paper presents a service virtualization approach called $VINCA_{Virtualization}$ to supporting the abstraction, transformation, binding and execution of Web services by end users. Four key mechanisms of $VINCA_{Virtualization}$, namely semantics annotation, services aggregation, virtualization operation and services convergence are discussed in details. $VINCA_{Virtualization}$ has been implemented and its application in a real-world project is illustrated. The paper concludes with a comparative study with other related works.

1 Introduction

End-user-programmable services composition is of importance in realizing virtual organizations of individual applications on spontaneous business requirements in a just-in-time manner. Typical scenarios having such requirements include dynamic supply chain management, handling of city emergency, and management of massive public events. Since current Web services composition languages such as BPEL4WS [ACD03] and BPML [Bp02] are developed for IT professionals, end-users involvement is not well promoted. As a matter of fact, to enable end-user-programmable services composition, we have to stride a number of hurdles, e.g. how to derive user-understandable, business-level services from business requirements, how to relate business-level services to software-level Web services, how to enable end users to assemble business-level services in an appropriate way, and how to ensure the interoperability and QoS constraints of participating business-level services.

* The research work is supported by the National Natural Science Foundation of China under Grant No. 90412005 and Grant No.90412010, National 863 High Technology Development Plan Project of China under Grant No.2003AA414330

In meeting the above challenges, we have proposed an approach to end-user-programmable, business-level services composition, and defined a corresponding composition language VINCA [HGL03], [YWH04]. The key concepts of VINCA include business services, service spaces, and Web service virtualization. A business service is a user-understandable, large-granularity service abstraction with business-level semantics. Service spaces organize and manage business services according to the user's actual needs by grouping business services that either are interrelated or share some common features. Web service virtualization is the process of abstracting away the Web service's technical details, describing it with business-related semantics, aggregating semantically substitutive Web services into service proxies, combining service proxies into a composite one with virtualization operators if necessary, and then converging service proxies and business services. After this virtualization process, Web services cannot be seen and used directly, rather they delegate their capabilities to business services. We introduce in this paper the service virtualization part of VINCA, called $\text{VINCA}_{\text{Virtualization}}$.

The rest of this paper is organized as follows. Section 2 describes a real-world example highlights the requirements of $\text{VINCA}_{\text{Virtualization}}$. Section 3 presents the basic principle of $\text{VINCA}_{\text{Virtualization}}$. Section 4 illustrates the process of virtualization. Section 5 discusses the implementation of $\text{VINCA}_{\text{Virtualization}}$. An application of $\text{VINCA}_{\text{Virtualization}}$ is given in section 6. In section 7, we discuss related works namely Service Domain, Service Cluster and XMS Service Views. And finally we conclude in section 8.

2 A Real-World Example

Let's take a real-world case as an example. This example is excerpted form the AMGrid (Advanced Manufacturing Grid) project [Ca02] that is to build a virtual enterprise for a large Chinese electronics manufacturing enterprise.

30 different kinds of Web services are prepared beforehand. Some are encapsulated from legacy systems such as stock-management system, order-processing system, etc. Some are provided by its suppliers and dealers according to the service interface agreements. And others are newly created by the enterprise itself. One major goal AMGrid is to let normal business users express their basic requirements by assembling and manipulating underlying services. The above-mentioned business services are thus needed. For the dynamism of business, business services are changed and created frequently. A semantic matchmaking algorithm based on inputs and outputs match [PKP02] is used to automatically bind business services to underlying web services. But frequently, the algorithm either can't establish the binding for inputs and/or outputs mismatch or the binding created is not correct. The following typical scenarios appeared in the project manifest this problem:

– Web services with similar functionalities often need to be aggregated. Only part of the Web service outputs is needed. For example, if there is a Web service operation named "*queryProductStock*" constructed from legacy stock management system whose function is to get the whole stock information. But when a business service

only needs part of its outputs according to some selection criteria, additional treatment is need on the outputs of this Web service operation.

– Some business service needs to bind to more than one Web services. For example, if there is a business service wants to query the products that are not in stock. But only "*queryProduct*" and "*queryProductStock*" Web service operations are available. In fact, we do not need to program a new Web service operation, the outputs of "*queryProduct*" exclude the outputs of "*queryProductStock*" will satisfy this business service. Another example is that there is a business service wants to get the product parts information from all the suppliers. For each supplier only gives its own parts information through a Web service operation, only by combining the outputs of these operations can the business service be satisfied.

3 Basic Principle of the Virtualization Approach

Figure 1 illustrates a basic reference model of $\text{VINCA}_{\text{Virtualization}}$. $\text{VINCA}_{\text{Virtualization}}$ follows the convergent engineering methodology, which tries to bridge the gap between business domain and software domain and enable software to adapt to ever-changing business [Ta95]. In $\text{VINCA}_{\text{Virtualization}}$, business services are modeled and created from business requirements in a top-down fashion; Web services are virtualized into service proxies in a bottom-up way; and then they are converged in the middle.

Business services are user-understandable, large-granularity service abstractions defined by domain experts independent of the underlying implementation details. Their definition is based on domain-specific norms of concepts and functionalities. A business service is formulated based on meaningful combinations of business activities (verbs) and business concepts (nouns) as defined in domain standards, e.g. the OTA travel information standard [Ot05]. The structure of a business service includes five parts: identification, functionality, inputs, outputs, and nonfunctional properties (NFP in short). The formal definition of business service can be found in [YWH04]. With the support of VINCA service space and VINCA studio, the visual programming environment, end users can explore and then compose business services into Web-based applications easily.

Service proxies are the aggregation of Web services. When providing Web services, the service provider also needs to provide its semantics. Such semantics can become the descriptions of service proxies, so functional substitutive Web services can be aggregated into one service proxy. The structure of a service proxy includes six parts: functionality, inputs, outputs, nonfunctional properties (NFP), Web services links, and virtualization operation (VO). The VO field is used in the convergence between business services and service proxies, which will also be discussed in detail in section 4.3. We implement the convergence algorithm based on both semantic matching and virtualization operation. As to the semantic matching, we adopt the matching algorithm proposed by [PKP02] where the inputs and outputs of business services and service proxies are matched and their matching degrees are calculated semantically. But according to the matching algorithm of [PKP02], whenever one of the business service's outputs is not matched by any of the service proxy's outputs, the matching fails. At this time, the virtualization operation as our novel approach takes effect. The virtualization operation transforms/combines service proxies into a new service proxy with one of the virtualization operators. For example, if both service proxy VS1 and service proxy VS2 can not match business service BS alone for the mismatch of outputs, but the combined outputs of VS1 and VS2 can match BS. Then we can use the virtualization operator "Union" on these two service proxies to create a new service proxy whose outputs are the combination of its component service proxies to achieve the matching. At a certain situation, the convergence between a business service and a service proxy can't be established after the effort of both semantic matching and virtualization operation, and this business service will be marked as DISSOCIATIVE and will not be seen by the end user until a convergence relationship is established successfully.

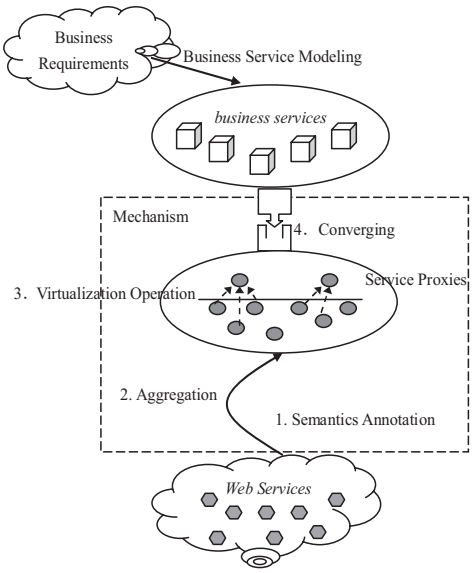

Figure 1: Basic Principle of $VINCA_{Virtualization}$

4 Process of Virtualization

In this section, we present the core mechanisms of $\text{VINCA}_{\text{Virtualization}}$ that are also illustrated in Figure 1. These mechanisms include *semantics annotation*, *Web services aggregation*, *virtualization operation*, and *convergence of business services and service proxies*.

4.1 Semantics Annotation

Semantics annotation is critical to reach unanimous understanding among syntax-diverse but functionality-similar Web services. As stated in the next subsection, our aggregating mechanism bases on the semantics of services. According to the structure of service proxy, a Web service operation is annotated with functional classification, inputs, outputs, and nonfunctional properties. Such annotation is saved in OWL [Ow04] format. To annotate, service providers use the ontology repository that is an extension to the one in business domain. Currently we only support the "equivalent class" extension, which facilitate service providers describing services with their own jargon.

4.2 Services Aggregation

$\text{VINCA}_{\text{Virtualization}}$ aggregates Web service operations with the same functional classification, inputs and outputs into one service proxy. Web services often operate in a dynamic environment as providers remove, modify, or relocate their services frequently. Aggregation of services is an effective mechanism to cope with the dynamism of Web services [BSD03]. When services are aggregated into one service proxy, they can be selected dynamically and application-level fault tolerance is yielded [MM05]. Every service proxy has its corresponding nonfunctional-property-based service selection policy. When executing a service proxy, it can avoid selecting an unavailable Web service and select the right Web service to run according to its nonfunctional properties.

4.3 Virtualization Operation

It's not always possible to bind a business service to the right service proxy for the incompatibility of semantics. At this time, virtualization operators can be used to transform/combine service proxies into a new service proxy to meet the semantics of this business service. We design our virtualization operators from two perspectives, one is on the selection of services and the other is on the filtering/uniting of the outputs of services. Fundamental virtualization operators include select, project, union, and difference etc. Next we will introduce these operators together with some examples.

4.3.1 Selection

The selection operator, denoted as σ, is used both in filtering the execution outputs and filtering the involved services based on their NFP.

For example as for filtering execution outputs, we need the Web service operation "*queryProductStock*" only present to the sale agent the corresponding product stock information they can sale while the default result can produce the whole product stock results. Following virtualization operation (VOP in short) creates a new service proxy "*queryProductStock_fr*" by filtering the execution outputs of service proxy "*queryProductStock*" with condition "*prodcutType="AA'"*", so only part of the result satisfying the selection condition is given.

$$queryProductStock\_fr = \sigma_{out.productType="AA"}(queryProductStock) \quad (1)$$

As for selecting involved services, following VOP creates a new service proxy "*supplyparts_fs*" by selecting the component Web services of "*supplyparts*" service proxy with response time less than 5 days. It means that the supplier that can provide its service in five days will be selected as the part supplier.

$$supplyParts\_fs = \sigma_{responseTime<5day}(supplyParts) \quad (2)$$

4.3.2 Projection

The projection operator, denoted as π, is used to select one or more attributes of the execution outputs. Following VOP creates a new service proxy "*queryProduct*_p" by only rendering the price information of the execution outputs of service proxy "*queryProduct*".

$$queryProduct\_p = \pi_{out.productname,out.price}(queryProduct) \quad (3)$$

4.3.3 Union

The union operator, denoted as $\cup$, is used for combining execution outputs of two service proxies. Suppose that the information of suppliers and their current supply parts is needed. But there are only two Web service operations, one can provide the information of suppliers and the other can provide the information of these suppliers' supply parts. Then we can define a new service proxy to present the union of these two operations. Following VOP creates a new service proxy "*query_u*" by uniting the execution outputs of both "*querySupplier*" and "*queryParts*".

$$query\_u = querySupplier \cup queryParts \quad (4)$$

4.3.4 Difference

The difference operator is denoted as $\ominus$. The following equation creates a service proxy "*query_d*" which gives the attributes provided by "*queryParts*" excluding the attributes also provided by "*queryDisks*".

$$query\_d = queryParts \ominus queryDisks \quad (5)$$

4.3.5 Power

The power operator is denoted as ψ. This operator is used when a business service needs to combine all the outputs of the Web service operations that aggregate to one service proxy. The following equation creates a service proxy "*queryParts_p*" whose execution outputs will combine all the outputs of the Web service operations that aggregate to service proxy "*queryPart*".

$$queryParts\_p = \psi(queryParts) \quad (6)$$

4.3.6 Intersection

The intersection operator is denoted as $\cap$. The following equation creates a service proxy "*query_i*" which provides the attributes provided by both "*querySupplierA*" and "*querySupplierB*".

$$query\_i = querySupplierA \cap querySupplierB \quad (7)$$

4.4 Convergence of Business Services and Service proxies

Convergence is the process of creating the binding relations between business services and service proxies. This process includes two phases:

- The first phase is automatic: The system will check the semantics of both business services and service proxies, the binding between a business service and a service proxy is established if they are semantically equivalent in functionality, inputs, and outputs.
- The second phase is manual: The Domain expert is responsible for converging a business service and a service proxy by constructing a new service proxy with virtualization operators if automatic binding can not be established for this business service.

5 Implementation

We have implemented a prototype of $\text{VINCA}_{\text{Virtualization}}$ that works as the core module of VINCA integrated business-end development and execution environment [YWH04]. For $\text{VINCA}_{\text{Virtualization}}$ is an indispensable part of VINCA, it is necessary for us to introduce the overall architecture of VINCA.

As illustrated in Figure 2, the architecture of VINCA is constituted of three parts: Basic Service Layer, Service proxy Layer, and End-User Programming Environment.

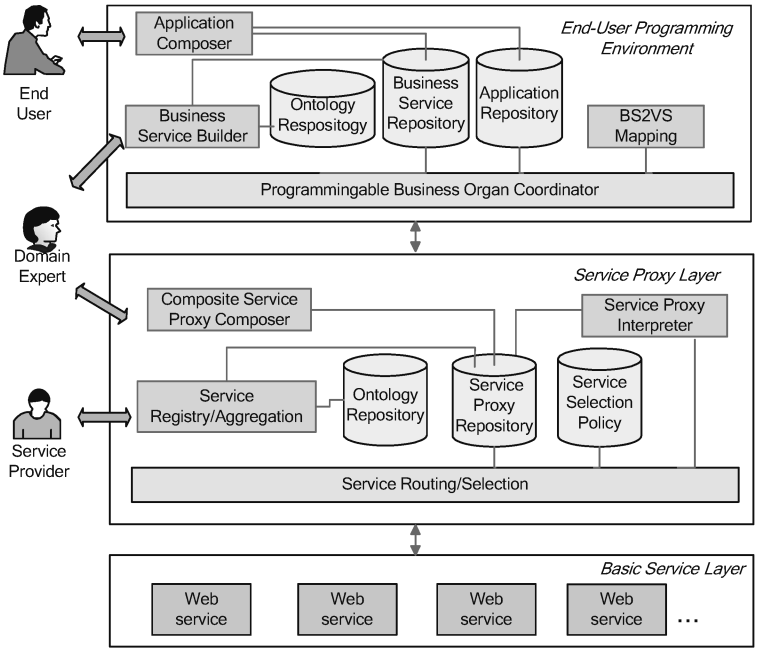

Figure 2: The Architecture of VINCA Environment

At the bottom layer, various available resources should be wrapped into web services, and we can access and invoke specific services according to their WSDL descriptions.

The Service proxy Layer is in the middle, its main responsibility is to create, manage and execute the aggregated service proxies. It provides service registering and annotating tool for service providers submitting their services. A newly registered service will be aggregated into an already-existing service proxy or it will trigger the creation of a new service proxy according to its annotated semantic description. During registering a service, its semantics is given referring to the Ontology Repository. Note that the Ontology Repository in this part is an extension to the one in the upper End-User-Programming Environment. For the Ontology description language, we use OWL and the "*equivalentClass*" relationship is supported for Ontology mapping. Virtualization tools could assist the domain expert to combine service proxies into a composite service proxy according to business needs. Then he could bind the new service proxy to business service. To execute a service proxy, first it is processed by the Service Proxy Interpreter component that will decompose composite service proxies into fundamental ones, and then the Service Selection/Routing component will select the right web services to execute according to the QoS specification of services and the Service Selection Policy.

6 Application

In this section we demonstrate the application of $\text{VINCA}_{\text{Virtualization}}$ by a real-world example. Suppose that one of the end users creates a business service named "*queryOutOfStock*", whose function is to query the products that are out of stock. The end user can construct this business service with Business Service Builder. There are also numbers of Web services registered in our system. Presently, this business service can't be automatically bound to any service proxy with BS2VS Mapping tool. At this time, the domain expert can use Service Proxy Composer to produce a new service proxy from two existing ones namely "*queryProductStock*" and "*queryProduct*". The tool can achieve this goal by the following virtualization operation:

$$queryOutOfStock = \pi_{producttype}(queryProduct) \ominus \pi_{producttype}(queryProductStock) \quad (8)$$

Then this new service proxy could be bound to the business service mentioned above. This process is illustrated in Figure 3.

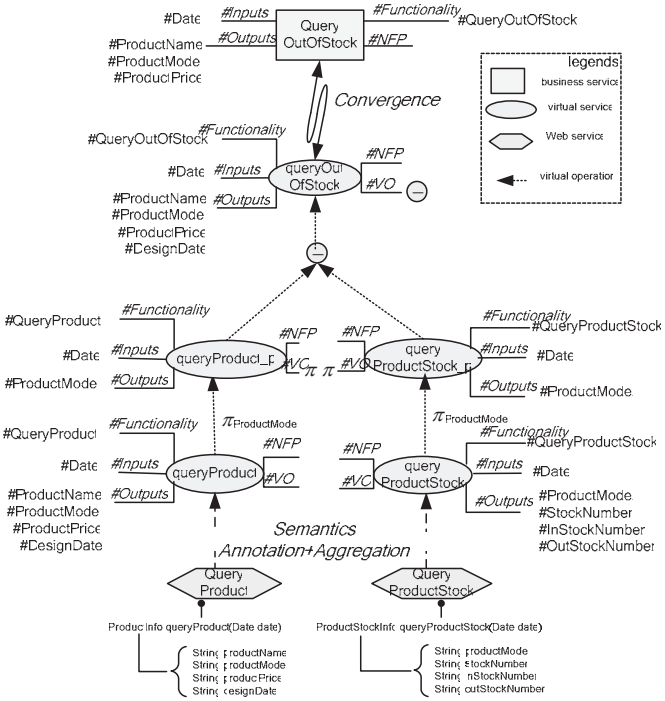

Figure 3: An Example of Service Virtualization

7 Discussion

A common characteristic of all service virtualization approaches is that it aggregates functional substitutive services into a virtualized service. Such aggregated virtualized service is given different name in different approaches.

IBM proposed the Service Domain technology [TTB03]. A Service Domain is an abstraction to a cluster of functional interchangeable Web services. It applies autonomic computing principles for aggregating Web services and Grid services. The Service Domain technology provides a service proxy layer – so called service Grid that can create, filter, discover, cluster, organize, select, route, recover, and switch Web services and Grid services autonomically. An impressive feature of Service Domain Technology is its service selection mechanism: the selection of a service instance is not just based on availability, but can also be based on QoS characteristics dynamically. The performance of Web service instances is monitored and their service level may be modified dynamically. It can also perform failover processing if required.

Service Container [BSD03] is a constitutive part of Self-Serv Web services composition environment. Its purpose is to cope with the dynamism of Web services. A Service Container is also a virtualized service that aggregates several substitutable services. An interesting feature of Service Container is its service functionalities related change management [ZBN01].

Service Views [Se05] are also abstractions and aggregation of back end services. It's an industry product and the main purpose is to ensure the QoS of Web services based applications and serve as an infrastructure for Service Oriented Architecture.

All the above-mentioned approaches aim at making the service oriented applications more reliable. So they put much effort on the mechanisms of service instances monitoring and QoS-based service selection. $\text{VINCA}_{\text{Virtualization}}$ focuses on how to virtualized Web services to semantically match the needs of business-level services. So $\text{VINCA}_{\text{Virtualization}}$ has its unique feature of service virtualization operators for coping with the incompatibility of semantics.

8 Conclusion and Future Work

In this paper, we have presented $\text{VINCA}_{\text{Virtualization}}$, an approach for Web service virtualization, which is a main component of VINCA business-end services composition approach. $\text{VINCA}_{\text{Virtualization}}$ has the core mechanisms of services annotation, services aggregation, virtualization operation, and services convergence that provide key support to the binding and execution of business services. With $\text{VINCA}_{\text{Virtualization}}$, end users can construct business-service based applications in an easy and effective way and business efficiency can be reached eventually.

This approach needs to be evaluated more exhaustively in the future based on diverse application scenarios. We also intend to incorporate service instances monitoring facility into $\text{VINCA}_{\text{Virtualization}}$ to enhance its capability in QoS-based service selection.

References

[ACD03] Andrews, T., Curbera, F., Dholakia, H. et al: Business Process Execution Language for Web Services. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/ (2003)

[Bp02] BPMI: Business Process Modeling Language. http://www.bpmi.org/ (2002)

[BSD03] Benatallah, B., Sheng, Q., Dumas, M.: The Self-Serv Environment for Web Services Composition. IEEE Internet Computing Jan/Feb 2003 (2003) 40–48

[Ca02] CAFISE group: AMGrid Project. Technical Report, Software Division, ICT, CAS (2002)

[HGL03] Han, Y., Geng H., Li H. et al: VINCA - A Visual and Personalized Business-level Composition Language for Chaining Web-based Services. Proc. of the 1st International Conference on Service-Oriented Computing, LNCS 2910, Springer-Verlag (2003) 165–177

[MM05] Michael, N., Muninda, P.: Service-Oriented Computing: Key Concepts and Principles. IEEE Internet Computing Jan/Feb 2005 (2005) 75–81

[Ot05] OTA: Open Travel Alliance. http://www.opentravel.org/ (2005)

[Ow04] OWL: OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/ (2004)

[PKP02] Paolucci, M., Kawamura, T., Payne, T., and Sycara, K.: Semantic matching of web services capabilities. International Semantic Web Conference, Sardinia, Italy (2002)

[Ta95] Taylor, D.: Business Engineering with Object Technology. John Wiley & Sons (1995)

[TTB03] Tan, Y., Topol, B., Vellanki, V. et al: Business service grid: Manage Web services and Grid services with Service Domain Technology. http://www-106.ibm.com/developerworks/grid/library/gr-servicegrid (2003)

[Se05] Service Views: XMS Service Views. http://www.westbridgetech.com/serviceview.html (2005)

[YWH04] Yu, J., Wang, J., Han, Y. et al: Developing End-User Programmable Service-Oriented Applications with VINCA. Proc. of the 2nd Ljungby Workshop on Information Logistics, Ljungby, Sweden (2004)

[ZBN01] Zeng, L., Benatallah, B., Ngu, A.: On-Demand Business to Business Integration. Proc. Int'l Conf. Cooperative Information Systems (CoopIS). Springer-Verlag New York (2001) 403–417