IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITJ2911

Ontology and Service Oriented Programming

Bing Li, Arizona State University, USA
Wei-Tek Tsai, Arizona State University, USA

ABSTRACT

This paper presents a novel methodology to develop and integrate distributed applications. It starts from analyzing requirement specifications based on a service's point of view. Thereafter, it is required to describe each service using ontology. The modeling and describing procedures are regarded as a new way to program Ontology and Service Oriented (OSO) programming, and descriptions obtained in the procedure are called OSO code. Moreover, OSO code has the features of interpretability, transformability, comparability, composability, and portability. Those features support code interpretation and generation. In addition, OSO code is also human-readable. This feature minimizes development efforts. Finally, since business logic in OSO code is represented in a machine-understandable format, the procedure of business process integration is completed automatically, based on business logic understanding.

Keywords: *business process integration; ontology theory; semantic matching; service integration; service oriented architecture*

INTRODUCTION

This paper presents a novel methodology, Ontology and Service Oriented (OSO), to develop and integrate distributed applications. The OSO is based on two fundamental techniques: Service-Oriented Modeling (SOM) and Ontology-Oriented Programming (ONOP), which provide developers with a new point of view to analyze an application domain and a new approach to program. OSO not only saves efforts on implementing distributed applications but also facilitates the goal of automatic system integration.

In OSO, the concept of service is the primary key to develop and integrate distributed applications. First, a service is a building block of distributed applications. SOM requires analyzing requirement specifications based on the concept of service. One of the major tasks in the procedure is to figure out what services exist in a distributed application to be developed and how those services are organized together. Second, a service is a function, a duty, or an action. Each service has the capability to receive a request, complete a task, and generate a corresponding response. Interactions occur between the service and a human being or another service. Therefore, when decomposing an application domain into multiple services, it is essential to make sure that each of those services provides certain functionalities. Third,

a service is an autonomous computing component, which means that a service works as an independent entity to provide functions, duties, or actions. In the procedure to service, no support or intervention is needed. The relations among services are loose and constructed by messages transmitted among them. In short, a service is a self-contained computing unit that is loosely coupled with each other to form a system.

Although SOM obtains all the services and their corresponding associations, only a skeleton of the system is built up. To make it work, ONOP is used to describe each service in detail. In OSO, a service consists of four elements: provider, messages, contracts, and business logic. The provider is an actor or a supplier of functions, duties, or actions. The messages are requests/responses transmitted among services. The contracts define the relations among services. Business logic represents the procedure to generate responses according to requests. Moreover, each element represents certain knowledge (i.e., business logic) in a service. In OSO, business logic is described in an ontology-oriented approach. In fact, ONOP is an approach to describe business logic inside a service by defining all the vocabularies of the service. After all the vocabularies inside a service are defined, the service is capable of receiving requests, updating the provider's status, and generating corresponding responses (i.e., business logic in the service is specified using ONOP). The skeleton of services and business logic inside services is the result of SOM and ONOP, and the results are represented in an ontology-oriented format. To simplify, the description is called OSO code.

OSO code is not only machine executable, but is also machine understandable. There are two ways to execute OSO code: interpretation and compilation. An interpreter is designed to execute OSO code, based on interpretation. Since OSO code is independent of any specific platforms, it is required that the interpreter be capable of associating the platform-independent descriptions with specific platforms or applications. Another way to run OSO code consists of two steps: transforming OSO code to a specific programming code and then compiling the code into a binary one to execute. Similar to the OSO interpreter, an OSO transformer is designed to achieve the previously mentioned goal. In short, OSO code has the features of interpretability and transformability.

One of the major advantages in developing distributed applications using OSO is that developers' efforts are focused on describing business logic through specifying ontology. Furthermore, since OSO code is represented in meaningful terminologies, it is human-readable specification. Thus, it is convenient for developers to program. In addition, in the procedure, no implementation details are required to be cared for by developers. Efforts are minimized when developing distributed applications using OSO. Another major advantage is that it is feasible to carry out system integration automatically on the basis of OSO code. There is a precondition to achieve this goal (i.e., a standard ontology library exists in a specific domain). In OSO, system integration is a procedure to build a new business process through discovering and selecting reusable services and composing final service instances according to a service integration requirement specification (SIRS). According to measure metrics, business logic similarity, business logic satisfaction, and instance similarity, reused service instances are discovered, selected, and integrated. In summary, OSO code's features, understandability, portability, and composability are important to the integration procedure.

Compared with the current popular programming modeling approach, Object-Oriented Programming (OSO; Coad & Yourdon, 1991) provides developers with some benefits. Although both an object in OO and a service in OSO are basic components to model an application domain, an object is not an independent application, but a service is a unique application that provides particular functionalities in a distributed environment. In addition, another important difference is that business logic in OO is specified in a machine-executable format. However, business logic in OSO is not only machine-executable, but is also machine-under-