Component Architecture for Business Application Systems

Sanya Uehara
Fujitsu Laboratories Ltd.
Email: sanya@flab.fujitsu.co.jp

Abstract

In this paper, I analyze the technological trends of component architecture used for business applications. By architecture, I am referring to the software elements and the relationships among them. The two important technical goals are (1) separating business logic from implementation logic and (2) mapping business models for implementation. I also explain the difference between object orientation and component orientation in building business application systems.

Keywords: component architecture, business application systems, business logic, business modeling

1. Introduction

This paper discusses business application systems (information systems) such as sales management systems and manufacturing systems. Business logic in such systems is very important. Business logic is usually embedded in source code and is thus not clearly visible. Finding specific portions of business logic is a very time-consuming task for people who maintain the systems.

Object orientation has not contributed as much to the development of business application systems as it has to others such as operating and simulation systems. One reason is that most objects in business modeling are data rather than active objects. Object modeling does not give us good insights about software architecture, although it does give us good entity relationships for database schema. A 3-tiered architecture is widely employed for business application systems in which the data and process parts are assigned to a database server and an application server, respectively. Another claim against object orientation is that it does not enable good business process modeling, which is at the heart of business process reengineering [3].

Introducing object technology into business application systems is different from other types of software. In this paper, I analyze the technological trends that researchers and industry technology developers will follow.

2. Separating Business Logic from Implementation Logic

The separation of business logic from implementation logic is important for several reasons.

(1) Business logic changes frequently and thus the corresponding code often needs to be found and modified.

(2) Business logic is the heart of the system; it explains what and how the system performs.

(3) Implementation logic portions should be modified separately for many different reasons, one of which is the need to upgrade middleware. This may be a long-term goal, but it is aimed at the component-based software development of business application systems. This is the core idea of Enterprise JavaBeans[4]. The next, more difficult problem is how to separate business logic into its constituent components. Although Enterprise JavaBeans currently only distinguishes the process-centric component "session enterprise bean" and the data-centric component "entity enterprise bean", OMG BODTF recognizes more elements relating to business logic such as "events" and "business rules." OMG BODTF discusses these elements by specifying the interface of components rather than by implementing components.

Events and business rules are means to describe business logic at a more abstract level. For instance, procedure call is a mechanism by which a client specifies a specific server. Conversely, by issuing an event, a client may invoke multiple cooperative servers. An event such as "issuing a purchase order" is an element that models a real world business, and its associated business rules and processes are initiated to execute the appropriate process. Other elements can also be used to model business logic. These elements are not usually discussed in object-technology communities.

3. Construction of Components for Business Logic

Enterprise or business modeling is closely related to research aimed at developing component architecture for business application systems. By allowing us to describe the implementation as well as the interfaces of components via such business modeling vocabulary as events, business rules, roles and so on, we will be able to build, customize, and modify component-based systems efficiently. The refinement process or the generation of a business model to implementation is an important research topic. OMG discusses only the specifics of the interface of business object components by using a business-modeling vocabulary. The following are some definitions on enterprise modeling:

(1) Enterprise: A system of business objects that cooperate to accomplish a desired goal. The business objects of an enterprise include people, machines, buildings, processes, events, and information.

(2) Business function: A group of activities which, taken together, support one aspect of furthering the mission of the enterprise.

(3) Business process: A specified activity in an enterprise that is executed repeatedly.

(4) Business rules: The conditions, constraints, and policies that control the operation of an enterprise.

4. Conclusion

In this paper, I have discussed the "separation" of business logic and implementation logic and the "mapping" of a business model for implementation. All are closely related to future component architectures and the development technology for business application systems. Note that this claim is not applicable to other types of software and that object technology will not support these goals. The techniques we desire for business application systems are for the "separation of business logic" and the "mapping of business models" via a business modeling vocabulary, in which object technology only suggests real world modeling using classes with an inheritance mechanism.

References

[1]Data Access Technologies, Combined Business Object Facility, BODTF-RFP 1 Submission, 1998.

[2]P. Fingar, D. Read, J. Stikeleather, Next Generation Computing - Distributed Objects for Business, SIGS Books, 1996.

[3]T. Gale and J. Eldred, Getting Results with the Object-Oriented Enterprise Model, SIGS Books, 1996.

[4]Sun Microsystems, Enterprise JavaBeans, 1997.

[5]S. Uehara, Enterprise Modeling and Software Architecture for Enterprise-wide Collaboration, IEICE vol.J80-D-I, no.7, 1997 (in Japanese).