Component-Oriented Development Method and Environment for Advanced Enterprise Information Systems

Katsuhiko Yuura
Koji Kagei
Hirofumi Danno

OVERVIEW: The development of enterprise systems by object-oriented methods is now being realized. Object technology is a way to grasp targets as objects, and its characteristics are ease of understanding and extensibility. To push business reengineering forward by the development of advanced information systems, it becomes important that the software structures precisely reflect the enterprise’s business needs. The component-oriented development method is a system development method to support the interchange of design information among end-users, systems analysts, and systems engineers. In this method, we adopt a concept of components that encapsulate independent business functions in addition to a concept of classes that show each process and data. With this method, the designer describes concrete business flows on scenarios with components. It is possible to represent components as large objects in the design of the software structures, because the components are defined by the data, the operations, and the actions on each business. The component oriented development method connects modeling tools at each component design step, and realizes integrated management of the design information.

INTRODUCTION

THE development of enterprise systems by object-oriented methods is realized with the generalization of distributed objects, the basis of distributed systems. In object-oriented development, we design the development targets as objects to deal with large-scale complex businesses and advanced software technologies. An object is a design unit that has data, methods, and actions — and it approaches the general form of understanding “things.”

With object-oriented technology, we design the flows of businesses by the actions among the objects. We design each business process by the data attributes and operations of the objects. To execute application systems, we finally develop programs, screen definitions, and database definitions; however, we define those relationships as models based on objects before that.

Modeling serves to make things that we cannot see directly with our eyes — such as business flows — visual, to change them into electrical data, and to specify consistent relationships among them. Though many man-hours are needed to define the models, we can expect to effectively reuse the components and to easily expand the system continuously by taking an extensive view of the whole system.

When we think of the development of systems in terms of two aspects such as business flows and individual business processes, we design each process at the same time and tend to design all flows for each process in object-oriented technology, rather than design the flow first as in the usual procedural-type development. This seemed to be a good approach from the point of view of modeling the targets, but there was the criticism that the object-oriented method is only for some specialists, because of the difficulty of the development.

Therefore, Unified Modeling Language (UML)\(^1\), a language for specifying, visualizing, constructing, and documenting the artifacts of software systems (Fig. 1) that has been proposed to the Object Management Group (OMG), adopts the use case diagrams (Fig. 2) that show relations and flows among the outside substances and the businesses visually. In this paper, we describe the component-oriented development method that was developed to support design information interchanges among systems engineers.

IDEA OF COMPONENT-ORIENTED DEVELOPMENT METHOD

Granularity of Objects

In object-oriented development, the granularity of
the definitions of the objects makes a difference in the approach (Fig. 3).

1) Precise target model (Fine grain): This model realizes advanced reuse of programs by using such properties as inheritance until the programming phase. The program unit in this model is a class. It is possible to use spiral development with a small number of developers.

2) Business design model (Medium grain): The program units in this model are the extracted objects from the design of business reengineering. We call the units business components.

3) Distributed systems integration model (Coarse grain): The program units in this model are the units of encapsulation of the distributed systems such as outside systems, positions of businesses, and organizations. We call the units application components.

**Introduction of Components**

The definition of components is coarser than usual class libraries: it is a large object that is easy to grasp from the viewpoint of the business. In OMG,
standardization is being discussed by using the figure of speech business object.

There are two viewpoints regarding components. One viewpoint is that design by the object-oriented development proceeds in a stepwise manner. It is difficult to generalize object-oriented development as a broadly-based development method for designing a precise target model directly from the two sides of the flows of the business and individual business processes, because advanced design skill is indispensable.

If the target is divided into components, development can be partitioned by limiting the influence range. Thus the degree of progress and location of problems can easily be comprehended during the development. This is especially effective during large-scale development.

The other viewpoint is that readily available software components become the unit of reuse. This can be expected to raise development efficiency, because the usage in the application design is clear from the point of having a function independent of each business and the unit of the reuse is large. The two viewpoints of the component-oriented development method treat two granularities of components.

An application system is composed of application components. It becomes the unit of systems integration in the case of distributed systems. It is an application framework to manage control among the application components. For example, the framework for a WWW-based and distributed objects system of Hitachi, Ltd. provides a framework that includes a client’s WWW dialog environment and the client’s secure information management. Also, application components are used as the unit to share development. Therefore, it is desired to design a simple integration method among components.

A business component is the unit of the business specifications that constitute an application component; also, it is the unit of software reuse. It is sometimes implemented as distributed objects. For example, in the above-mentioned framework for a WWW-based and distributed objects system, we realize distributed objects related to the user authorization, the management of the user needs (mass-customization), and the like.

A business component is composed of fine grained classes. These are designed to raise the efficiency of the reuse of the business component. Design of the target as a fine grained precise model becomes effective when complex business logic is designed.

**CHARACTERISTICS AND DEVELOPMENT PROCESS OF THE COMPONENT-ORIENTED DEVELOPMENT METHOD**

Hitachi developed a component-oriented development method, which includes the development of the systems integration model and the precise target model, to support the implementation of the model for the continuous push of business reengineering (refer to the preceding chapter “Granularity of Objects”) as
our first goal. The component-oriented development method is arranged as a menu to drive the system development planning phase with an object-oriented model on the upper stream of the information systems development method in Hitachi, Ltd. (Fig. 4). It has the following characteristics.

(1) Business components extraction based on scenarios

The component-oriented development method attaches importance to the ease of understanding from the business side and the smooth transmission of design information from the software side. A scenario contains the flows of the business; it also contains start conditions, inputs and outputs, divergence, and the like.

In scenarios, the business flows are designed as the chains of the actions of business components, and the business processes are designed as the operations of the attributes of business components. Actions correspond to messages of the object, and operations of the attributes correspond to the attributes of the object and the messages of the operations of the attributes. Thus, business components get forms as objects.

(2) Components design with 4 steps

Objects that cope with business are designed by four graduated steps to develop large-scale systems stably and efficiently.

(a) Application components design: Describe use case diagrams in accordance with the application requirements, and extract application components. At this time, consider the utilization of existent subsystems, or introduction of existent packages. Refer to the specifications of the existent application’s frameworks.

(b) Business components design: Describe the concrete scenarios of the business reengineering, and extract business components. Consider the reuse of existent business components at this time.

(c) Structural model design: Design the class group that realizes business components in accordance with the configuration definition of the hardware and software and adopted application frameworks.

(d) Fine grained model design: Design class groups in detail. Incorporate design patterns and class libraries.

When only a distributed systems integration model [refer to the preceeding chapter “Granularity of Objects
(3) is designed, you can proceed with the design after
the application components design of (a) above by
another method; for example a procedural-type
development method. To reflect a business
reengineering model [refer to the preceding chapter
“Granularity of Objects (2)’] on the design result and
to enhance the effect of the introduction of the
distributed objects, the structural-model design of (c)
above is indispensable. To adopt an applicable precise
model [refer to the preceding chapter “Granularity
of Objects (1)’] in the main parts and to design a highly
reusable business reengineering model, it is necessary
to carry out the detailed model design of (d) above.
(3) The parallel design of four matters

For the proper personnel arrangement and to
shorten the delivery time, it is necessary to proceed
with the investigation and design of the data base, and
the examination of the system requirements and the
like at the same time as the design of the components
class (Fig. 5). In the design of objects, the work of
defining business structures — in other words, the
work of defining components or classes — is the main
part of the task.

To decide business structure, though, design
information of business movements for such cases as
message sequence, graphical user interface (GUI)
movement, and data (entities and data items) become
important information. Also necessary are the
extraction of the system requirements such as the
performance and security, and the confirmation of the
design result.

In the component-oriented development method,
personnel who have suitable skills are allocated to the
different aspects of the business structure; including
business movement, data, and system considerations;
and they work cooperatively. At present, in object-
oriented development, which is now in a growth phase,
there tends to be a lack of talented people with object-
oriented development skills to take charge of business
structure design. With this method, though, it is
possible to develop large-scale systems with a rather
small number of specialists.

(4) UML conformity

Standardization is based on UML international
standardization methodology. This method implements
international division of system development, and
provides integration with tools of other companies.

COMPONENT-ORIENTED DEVELOPMENT
METHOD DEVELOPMENT SUPPORT TOOLS

Design information for the component-oriented development method is described with notation based on international standards such as class figures and message sequence figures — and individual work sheets and specifications. The environment of the input, editing, and display is realized with Hitachi’s development support tools and integration (Component CASE) with the well-known tools available for each field.

Design information for each of the tools can be obtained from the Web environment, because the information is internally managed as an applicable design model inside the tools (Fig. 6). In the software development process after the business design, the source program of the object and the customization of frameworks and components are managed in a unified manner by the main tool.

The main tool of component-oriented development support defines a business scenario visually (Fig. 7). The tool describes a business component as a character that has an icon, and indicates message movement step by step (scene). The tool indicates the start condition of a message to define the scenario, the reference of the data, and renewal and divergence with respect to other scenarios at each scene.

The user can avoid misunderstanding even if the chain of the message and the relations of the data operation are complicated because it is indicated step by step. This method supports acquisition of agreement with end users by taking an extensive view of the business world, and showing each business movement and the flow clearly3). Also, the tool can give the other development support tools the design information of business-components or messages.

CONCLUSIONS

In this article, we described the component-oriented development method. The component-oriented development method is applicable to the development of systems by object-oriented technology that lead the
markets such as electronic commerce systems or financial market systems. In the future, Hitachi will apply the method to development by object-oriented technology for a wide range of related tasks such as the financial big-bang and work-flow applications.

Based on these applications, Hitachi will exert increased effort toward the evaluation and the expansion of the method, and provide additional manuals and the development support. The support tools have been applied partly to the advanced development of the preceding, and increase the effectiveness of getting agreement about the business design, and Hitachi will continue to develop the tools at the same time as it develops the method.

REFERENCES
(1) Rational Inc. et.al., UML Semantics, OMG ad/97-08-03 (1997-9).

ABOUT THE AUTHORS

Katsuhiko Yuura
Joined Hitachi, Ltd., in 1980 and now works at the Dept. 2, Business & Information Systems Development Div. He is currently engaged in the development of information systems design technology based on object-oriented method and advanced application systems. Mr. Yuura is a member of Information Processing Society of Japan, Japan Society for Software Science and Technology, and can be reached by e-mail at yuura@bisd.hitachi.co.jp

Koji Kagei
Joined Hitachi, Ltd. in 1993 and now works at the Dept. 2, Business & Information Systems Development Div. He is currently engaged in the development of the information systems development methods. Mr. Kagei is a member of the Information Processing Society of Japan, and can be reached by e-mail at kagei@bisd.hitachi.co.jp

Hirofumi Danno
Joined Hitachi, Ltd. in 1970 and now works at the 2nd Research Dept. of Systems Development Laboratory. He is currently engaged in research and development of business componentware technology with object-oriented method. Ms. Danno is a member of the Information Processing Society of Japan, and can be reached by e-mail at danno@sdl.hitachi.co.jp