New Age of Software Development:
How Component-Based Software Engineering Changes the Way of Software Development ?

Mikio Aoyama
Department of Information and Electronics Engineering
Niigata Institute of Technology
1719 Fujihashi, Kashiwazaki 945-11, Japan
Tel: +81-257-22-8129, E-mail: mikio@iee.niit.ac.jp

ABSTRACT

The dawn of a new age of software development is coming.

Wide spread of the Internet technology and PCs opened up new market of software as well as new architecture of software. However, conventional software development technology could not catch up with the speed.

Component-Based Software Engineering (CBSE) is an emerging paradigm of software development. Its goal is composing applications with plug & play software components on the frameworks. CBSE is aiming at realizing long-waited software reuse by changing both software architecture and software process. It may vastly change the way we develop software. However, we still see many problems to solve.

Let's open up discussions, and take a step toward the new age of software development.

Keywords
Componentware, Object-Orientation, Reuse, Framework, Java and Distributed Object

1 INTRODUCTION: CHANGE

Change is a badge of modern corporate. No corporate can do business without software. Now, change is a badge of software tribe.

As illustrated in Fig 1, we are witnessing a historical encounter of change of environment and change of software and technology.

Widespread use of the personal computers and the Internet make computers commodity goods. And created new market and users. This requires substantial change to software industry. New users, most of them are consumers, require to drastically reduce the price and/or cost of software in order to match the ever-decreasing hardware price. Demands to new applications such as electronic commerce and groupware are high. However, conventional methodologies have not achieved such drastic gain of the productivity and quality yet. We are requested to fundamentally re-think the way of software development.

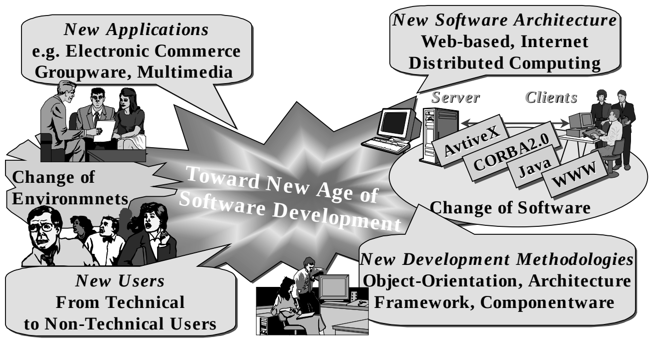

Fig.1 Change of Environment and Software

The use of components is the primary source of the productivity and quality. It is the law of nature in any matured engineering discipline [Szyp98].

Making applications from software components has been a dream in software engineering community since its very early time. As quoted in the literatures, McIlroy wrote in the NATO conference in 1968 [McIl68]; "*My thesis is that the software industry is weakly founded, in part because of the absence of a software components subindustry. … A components industry could be immensely successful*". However, wide spread reuse of software components over the industry has not come true.

Why? A number of obstacles have been identified. However, it seems clear that the most fundamental problems are lack of mechanisms to make components interoperable and lack of "really reusable" components.

Since early 1990's, so-called **Componentware** [Udel94] or **Component-Based Software Engineering (CBSE)** have emerged [Aoya96][Brow96][Kiel98] [Same97][Szyp98]. At the early time, CBSE emphasized on the *EUC (End-User Computing)* such as composing applications on the PCs. However, the use of *COTS (Commercial Off-The-Shelf)* software promoted the CBSE in the development business applications [Wall98]. Furthermore, quick evolution of the Internet technology such as Web and Java-based technologies even open up new possibilities of CBSE such as network distribution of components, and the reuse and interoperation of components over the Internet. Now, a few set of technologies have been widely deployed and are still evolving. They include ActiveX/DCOM from Microsoft [Micr98], CORBA from OMG [OMG98] and JavaBeans from SUN Microsystems [Thom97].

2 COMPINENT-BASED SOFTWARE ENGINEERING

2.1 What is Software Components

There are a number of definitions on software components [Same97]. However, in the context of CBSE, we emphasizes **plug & play** software components so that software can be composed with components like hardware. Thus, s*oftware components are binary units of independent production, acquisition, and deployment that interact to form a functioning system* [Szyp98].

2.2 What Differentiates CBSE from the Conventional Reuse

(1) Conventional Software Reuse and CBSE

Although object-oriented technologies have promoted software reuse, there is a big gap between the whole systems and classes. To fill the gap, many interesting ideas have emerged in object-oriented software reuse for last several years. They include *software architecture* [Shaw96], *design patterns* [Gamm95], and *frameworks* [Faya97]. Fig. 2 illustrates layers of such reusable elements.

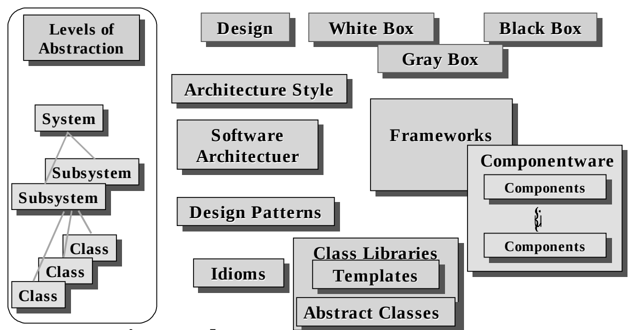

Fig. 2 Elements of Software Reuse

(2) CBSE Approach

CBSE takes different approaches from the conventional software reuse in the following manner.

(1) *Plug & Play:* Component should be able to plug and play with other components and/or frameworks so that component can be composed at run-time without compilation.

(2) *Interface-centric:* Component should separate the interface from the implementation and hide the implementation details so that they can be composed without knowing their implementation.

(3) *Architecture-centric:* Components are designed on a pre-defined architecture so that they can interoperate with other components and/or frameworks.

(4) *Standardization:* Component interface should be standardized so that they can be manufactured by multiple vendors and widely reused across the corporations.

(5) *Distribution through Market:* Components can be acquired and improved though competition market, and provide incentives to the vendors.

3 COMPONENT-BASD SOFTWARE DEVELOPMENT

The nature of CBSE suggest that the model of component-based software development should be different from the conventional development model. Table 1 summarizes major characteristics of conventional software development and component-based software development, which are briefly discussed in the following sections.

Table 1 Comparison of Development Models

Characteristics	Conventional	CBSE
Architecture	Monolithic	Modular
Components	Implementation & White-Box	Interface & Black-Box
Process	Big-bang & Waterfall	Evolutional & Concurrent
Methodology	Build from Scratch	Composition
Organization	Monolithic	Specialized: Component Vendor, Broker, & Integrator

3.1 Architectrue

CBSE emphasizes modular architecture so that we can partially develop a system and incrementally enhance the functions by adding and/or replacing components. To make such design possible, we need a sound foundation of software systems, that is, software architecture. Most component-based systems assume underlying software architecture such as MFC (Microsoft Foundation Class) and CORBA. They are provided in the form of frameworks. Frameworks are workable reference to the underlying software architecture.

To be effective, framework can be hierarchical up from domain independent to domain specific. Examples include Andersen Consulting's Eagle project [Ande98] and IBM's San Francisco project [IBM98][Laza98]. With standardized and modular software architecture, the CBSE can avoid ad hoc and monolithic design as illustrated in Fig. 3 [Mowb97].

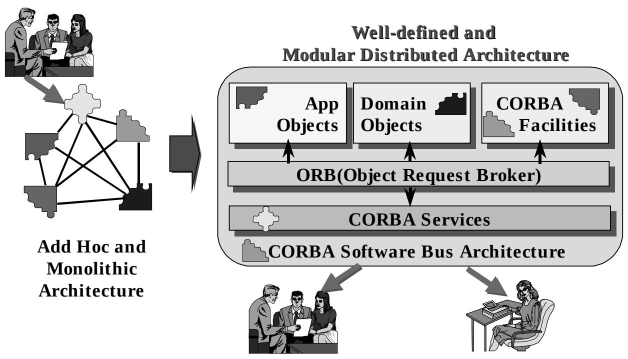
Well-defined and
Modular Distributed Architecture
App Objects
Domain Objects
CORBA Facilities
ORB(Object Request Broker)
CORBA Services
CORBA Software Bus Architecture
Add Hoc and Monolithic Architecture

Fig. 3 Architecture-Based Design

3.2 Components

Components can be product-specific, domain-specific or domain-independent as illustrated in Fig. 4.

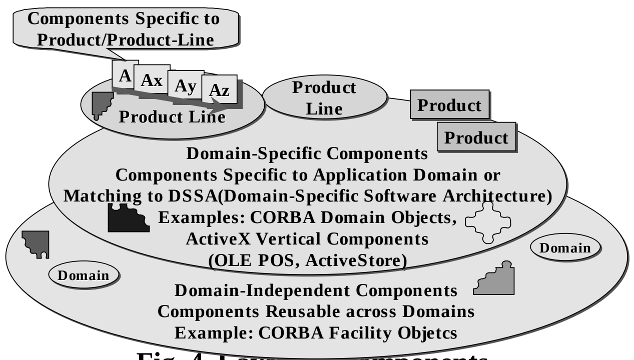
Components Specific to Product/Product-Line
A
Ax
Ay
Az
Product Line
Product Line
Product
Product
Domain-Specific Components
Components Specific to Application Domain or
Matching to DSSA(Domain-Specific Software Architecture)
Examples: CORBA Domain Objects,
ActiveX Vertical Components
(OLE POS, ActiveStore)
Domain
Domain
Domain-Independent Components
Components Reusable across Domains
Example: CORBA Facility Objetcs

Fig. 4 Layers of Components

3.3 Process

CBSE makes software development and delivery be evolutional. Since some parts of a system can be acquired from the component vendors and/or be outsourced to other organizations, some parts of software process can be done concurrently.

(1) Architecture of Software Process

To make software reuse happen, software process should be reuse-oriented so that designers can reuse artifacts at different levels of abstraction along with software process. Fig. 5 illustrates conventional water-fall process and an example of CBSE process.

CBSE process consists of two processes; component development and component integration. Since these two processes can be done by different organizations, these two process can be concurrent.

Unlike conventional process, CBSE process need a new process for component acquisition.

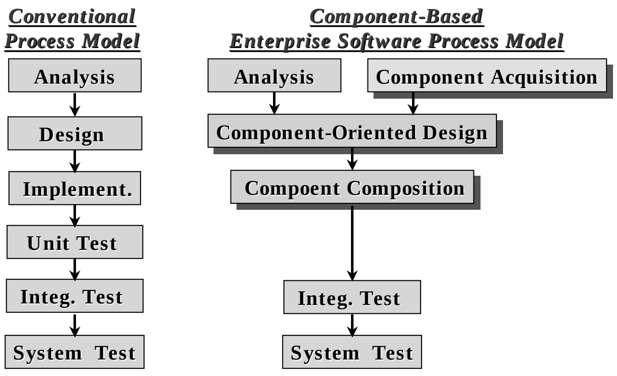

Fig. 5 Conventional Process and CBSE Process

3.4 Methodology

Fig. 6 illustrates an overview of development methodologies of CBSE. As illustrated, methodologies need to deal with both component development and component composition.

Most of conventional methodologies such as object-oriented methodology assume development from scratch and have not provided much help for reuse-oriented development. Furthermore, plug & play software components separated interface from the implementation and provide interface

CBSE focuses on composition of components through their interface. Composition also requires to design collaborative behavior of multiple components. So, CBSE methodologies need to help *interface-centric* and *behavior-oriented* design such as Catalysis [DSou98] and connection-oriented programming [Szyp98].

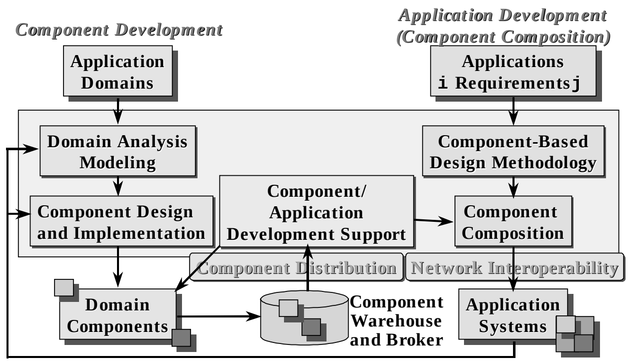

Fig. 6 CBSE Development Methodologies

3.5 Organization

The separation of component development and component integration created a new role of component broker. Component broker can sell and distribute software components.

Since component development and component integration requires different expertise, it is natural to specialize the organizations into *component vendors* and *component integrators*. This specialization will requires the mediators between two organizations, that is, *component brokers*. This organization structure, illustrated in Fig. 7, can be called *vendor-broker-integrator model* [Ning97].

As the software component vendors have been growing, a software component market is emerging. Since software can be distributed over the Internet, web-based software component brokers have emerged [Aoya98a].

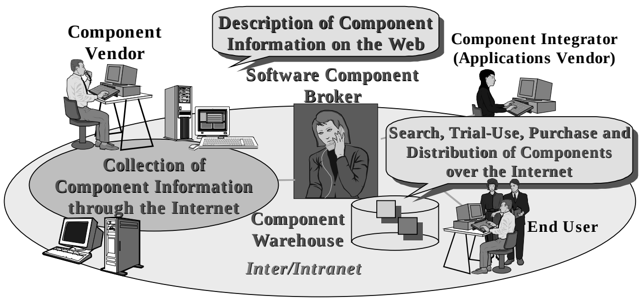
Component Vendor
Description of Component Information on the Web
Component Integrator (Applications Vendor)
Software Component Broker
Search, Trial-Use, Purchase and Distribution of Components over the Internet
Collection of Component Information through the Internet
Component Warehouse
End User
Inter/Intranet

Fig. 7 Vendor-Broker-Integrator Model

4 EARLY EXPERIENCE

We have observed a number of component-based software development. As an example, Fig. 8 shows the size of code written and effort of two pilot projects of component-based software development conducted in a Japanese software company [Aoya97][Aoya98b]. As the size of code drastically is reduced, so the required work load is.

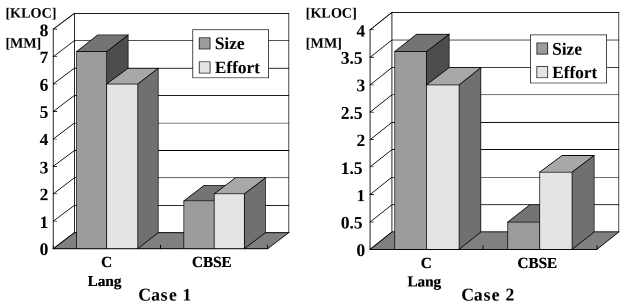

Fig. 8 Case Study: Size and Effort

Fig 9 shows the workload distribution along with software process. Data of case 1 and 2 are collected from the pilot projects above mentioned. For the reference, an estimation based on COCOMO is also illustrated. Although the number of data is small, the following characteristics of component-based software development are revealed:

1) CBSE requires a new process that is component acquisition, and
2) The workload for testing is drastically reduced.

Besides these cases, a number of component-based software development have been conducted.

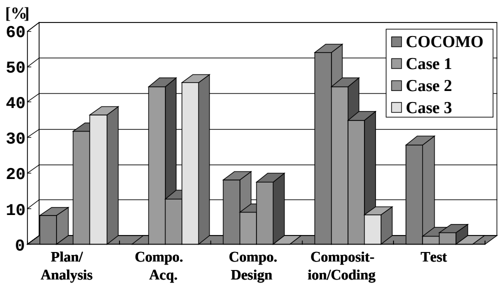

Fig. 9 Case Study: Workload Distribution

5 VISION TO A NEW AGE OF SOFTWARE DEVELOPMENT WITH CBSE

With CBSE, we can change the way of software development as illustrated in Fig. 10. As suggested, software development should be with modular process, modular architecture and specialized organization so that we can accumulate our technology and expertise.

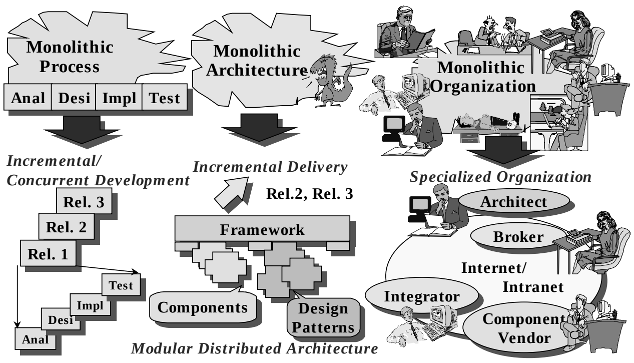

Fig. 10 Vision to New Age Software Development

How we can make CBSE happen ?

As above mentioned, CBSE can be a fundamental technology for software development so that it requires to re-think various aspects of software development. Besides technical issues, non technical issues such as commerce of components and management issues are also important.

To make our dreams come true,. new technology and professional will be needed.

REFERENCES

[Ande98] http://www.ac.com/aboutus/tech/eagle.

[Aoya96] M. Aoyama, Componentware: Building Applications with Software Components, *J. of IPSJ*, Vol. 37, No. 1, Jan. 1996, pp. 71-79 (In Japanese).

[Aoya97] M. Aoyama, Process and Economic Model of Component-Based Software Development, *Proc. 5th IEEE SAST (Symp. on Assessment of Software Tools)*, Jun. 1997, pp. 100-103.

[Aoya98a] M. Aoyama, et al., An Architecture of Software Commerce Broker over the Internet, *Proc. WWCA (Worldwide Computing and Its Applications) '98, LNCS Vol. 1368*, Springer-Verlag, Mar. 1998, pp. 97-107.

[Aoya98b] M. Aoyama, et al. (ed.), *Componentware*, Kyoritsu Shuppan, 1998 (In Japanese).

[Brow96] A. W. Brown, *Component-Based Software Engineering*, IEEE CS Press, 1996.

[DSou98] D. F. D'Souza and A. C. Wills, *Objects, Components and Frameworks with UML: The Catalysis Approach*, Addison Wesley, 1998 (http://www.iconcomp.com).

[Faya97] M. E. Fayad and D. C. Schmidt (ed.), Object-Oriented Application Frameworks, *CACM*, Vol. 40, No. 10, Oct. 1997.

[Gamm95] E. Gamma, et., *Design Patterns*, Addison-Wesley, 1995.

[IBM98] http://www.ibm.com/Java/Sanfrancisco/.

[Kiel98] D. Kiely, Are Components the Future of Software?, *IEEE Computer*, Vol. 31, No. 2, Feb. 1998, pp. 10-11.

[Laza98] B. Lazar, IBM's San Francisco Project, *Software Development*, Vol. 6, No. 2, Feb. 1998, pp. 40-46.

[McIl68] M. D. McIlroy, Mass-Produced Software Components, *Software Engineering Concepts and Techniques (1968 NATO Conference on Software Engineering)*, Van Nostrand Reinhold, 1976, pp. 88-98.

[Micr98] http://www.microsoft.com/activex/.

[Mowb97] T. J. Mowbray and W. A. Ruh, *Inside CORBA*, Addison-Wesley, 1997.

[Ning97] J. Ning, A Component-Based Software Development Model, *Proc. COMPSAC '96*, Aug. 1996, pp. 389-394.

[OMG98] http://www.omg.org.

[Same97] J. Sametinger, *Software Engineering with Reusable Components*, Springer-Verlag, 1997.

[Shaw96]M. Shaw and D. Garlan, *Software Architecture*, Prentice Hall, 1996.

[Szyp98] C. Szyperski, *Component Software*, Addison-Wesley, 1998.

[Thom97] A. Thomas, *Enterprise JavaBeans: Server Component Model for Java*, White Paper, Dec. 1997, http://www.javasoft.com/products/ejb/.

[Udel94] J. Udell, ComponentWare, *BYTE*, Vol. 19, No. 5, May 1994, pp. 46-56.

[Wall98] K. Wallnau and D. Carney, *COTS Products and Technology Evaluation: Concepts and Pragmatics*, ICSE '98 Tutorial, Apr. 1998.