

# Research on an MDA Based COP Approach <sup>1</sup>

Seung-Yun Lee, Oh-Cheon Kwon, Min-Jung Kim and Gyu-Sang Shin

Computer & Software Technology Laboratory  
ETRI(Electronics and Telecommunications Research Institute)  
161 Gajung-Dong, Yusung-Gu, Daejun, Korea

{ coral, ockwon, minjkim, gsshin }@etri.re.kr

## Abstract

*Object-Oriented Programming (OOP) has some weaknesses in that it does not always produce reusable software and is not suitable for a large project and does not support the complete encapsulation of classes due to the inheritance of subclasses. As an evolutionary method of OOP, Component-Oriented Programming (COP) or Component-Based Development (CBD) has recently been hot issues for the Software Engineering community, and the component technology market is also growing rapidly. However, components built from various component platforms cannot be operated in a specific component platform. In addition, binary components produced from the third party cannot be reused without modifications due to different business processes among organizations. For these reasons, MDA (Model Driven Architecture) has been introduced as an evolutionary approach to COP. In this paper, the authors present the previous work on COP and the current work on MDA, and describe strengths of approaches related to COP and MDA.*

**Keywords:** Component, COP, CBD, COBALT Constructor, COBALT Assembler, MDA

## 1. Background

Component-Oriented Programming (COP) or Component-Based Development (CBD) methodology has been one of the most promising methods for software development since the middle of 1990's, because it enables software developers to develop highly reusable components and to assemble the components into a specific application, leading to elevate software development productivity and quality. An application development in CBD is carried out through the steps of selecting, adapting and composing components like the conventional reuse method, rather than developing software from scratch.

CBD has similar points to Object-Oriented Programming (OOP). For example, both code and data are defined in a component, as in an object. In particular, the component has interfaces separated from its implementation which is in a "black box" style. Therefore, component reusers only need to understand the interfaces of components and assemble the components into an application system [12].

In order to make CBD successful, there are several problems to be solved. Firstly, components should be reusable in order to be reused repeatedly for similar systems. Secondly, components can be adapted according to different requirements before reusers or integrators reuse the components. Thirdly, the components should be assembled into a specific application system by visual plug-&-play composition. The vision of component technology is to build a software system by drag-&-drop assembly like a lego game or puzzle game.

The OMG's mission is to help software developers to solve integration problems by providing open, language-vendor-neutral interoperability specifications. Therefore, the OMG has announced several standards such as Object Management Architecture (OMA), Common Object Request Broker Architecture (CORBA), and Unified Modeling Language (UML). Although CORBA has been a huge success in the market, different component architectures have been announced and used by a number of organizations. In fact, it is difficult for a large enterprise to standardize on a single middleware platform. Some

---

<sup>1</sup> This work was supported by the National Research Laboratory (NRL) Program of the Ministry of Science and Technology of Korea.

enterprises could use more than one because their different departments have different requirements, others because mergers or acquisitions created a mix. Even the lucky enterprise with a single middleware choice still has to use multiple middleware to interoperate with other enterprises and B2B markets [9].

In order to solve problems with integrating various platforms, the Model Driven Architecture (MDA) that is language-, vendor- and middleware-neutral, has been adopted as a standard by the OMG in September 2001. The MDA defines the platform independent specification that excludes the platform specific information from system specifications. Then, the Platform Specific Model (PSM) can be acquired by mapping the Platform Independent Model (PIM) to a particular platform. The major objectives of the MDA are to transform from PIMs to PSMs using UML metamodels, and to automatically generate code from PIM models written in Action Semantics Language (ASL) [11].

To make MDA successful, it is clear that the MDA tool for the above two major objectives should be provided. In addition, reuse of domain architectures should be supported for effective reuse of PIM models. Finally, since the MDA standard focuses on building components rather than composing pre-built components, a method of assembling components into a large-grained component or a specific software system should be integrated in the MDA process. In this paper, the authors present the previous work on COP and the current work on MDA, and describe strengths of tools related to COP and MDA.

In Section 2, the authors present the results from the previous project on CBD, the current work on the MDA project, and strengths of our approaches related to CBD and MDA. In Section 3, the authors compare our work with previous work on MDA. Finally, Section 4 concludes and gives future directions for our research work.

## 2. Our Position

### 2.1 Results from the CBD Project

The authors have carried out the CBD project in order to solve the three problems with promoting the CBD technology such as identification of reusable components, adaptation of binary components and visual server side component assembly by plug-&-play, as described in Section 1. The first task of the project is to develop a CBD methodology named **MaRMI III** [16] that specifies the whole CBD process and procedure for supporting component building, legacy wrapping and assembly in the EJB component platform. The second task of the project is to develop the CBD tool, including **COBALT (COmponent-Based Application deVeLopment Tool) Constructor** [10] which supports the whole component development process such as component identification, component modeling, component implementation, component extraction, component deployment, component testing and report generation, and **COBALT Assembler** [12] which supports architecture modeling, visual component assembly, component deployment, component testing and component performance measurement. The developed CBD methodology and the COBALT tools have several strengths in terms of the world first's EJB-based CBD methodology, reusable component identification, the world first's EJB component composition by visual plug-&-play assembly, and easy to use binary component adaptation.

The component assembly tool called COBALT Assembler has been a finalist for the Best of Comdex Awards in Nov. 2002. In addition, **MaRMI III** has been open to the public through the KCSC (Korea Consortium for Software Component promotion) homepage, and downloaded about 6000 times as of April 2003, leading to building the base for promoting the component industry.

Among CBD processes, the identification of reusable components is one of key and difficult processes. Currently, component identification depends mainly on domain experts' intuition and experience. In order to automate the identification of reusable components, the authors have proposed a systematic method for identifying software components using object usages and dependencies between object-oriented domain models such as use case diagrams, class diagrams and sequence diagrams, and implemented it in the COBALT Constructor [10]. Figure 1 shows the component identification wizard and domain models that are included as subsystems of the COBALT Constructor.

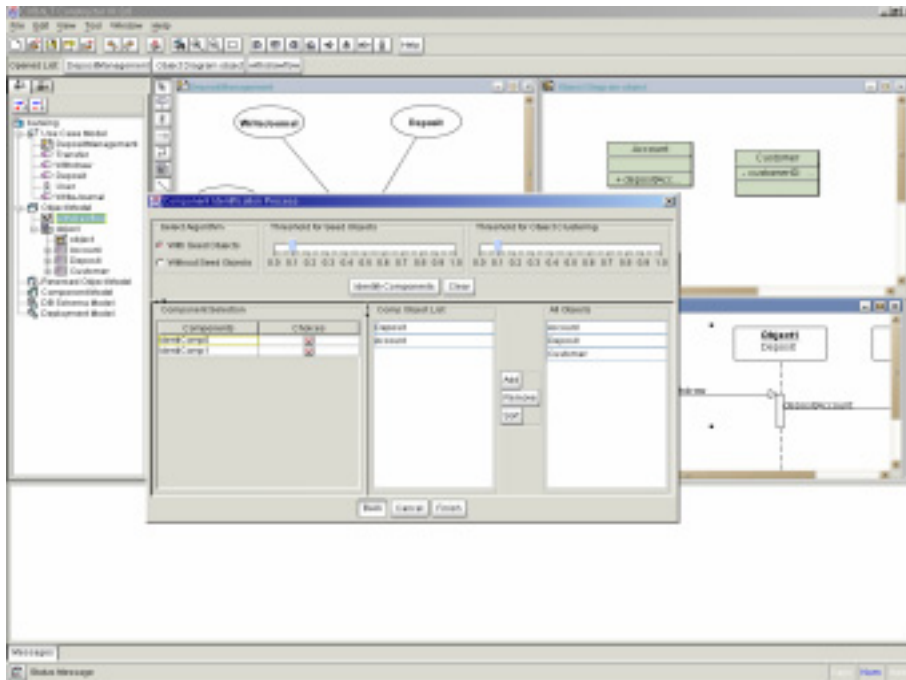


Figure 1. Identification of Reusable Components Using Domain Models

EJB is a server side component model emerged to reduce the complexity of software development and to facilitate reuse of components. However, EJB does not support component assembly by plug-and-play due to the hard-wired composition at the code level. To solve this problem described in Section 1, the architecture for EJB component assembly is defined at the abstract level and the inconsistency between system architecture and its implementation should be eliminated at the implementation level. In addition, as described in Section 1, component reusers require component adaptation since the interfaces of a component might not be exactly matched. Component assembly is carried out only when the contents of the interfaces such as an interface and attribute name, an operation name, and a parameter name and type match up completely. In other words, in order to get the component matched with changing requirements of a specific application, it is generally needed to adapt the binary component through an adaptation technique [12]. The authors have proposed a tool, COBALT Assembler, to support the design and implementation of EJB component assembly by plug-and-play based on the architecture, and of binary component adaptation.

Figure 2 shows the architecture editor and component specification editor which are subsystems of the COBALT Assembler. The COBALT Assembler provides the graphical and textual architecture editors to build the system architecture. The tool user can drag and drop EJB components (EJB Jars) to assemble from the Composite Palette which browses EJB Jars in the directory and shows the files in it. A component structure is built using the component specification editor that automatically converts a component's methods to interfaces of message type.

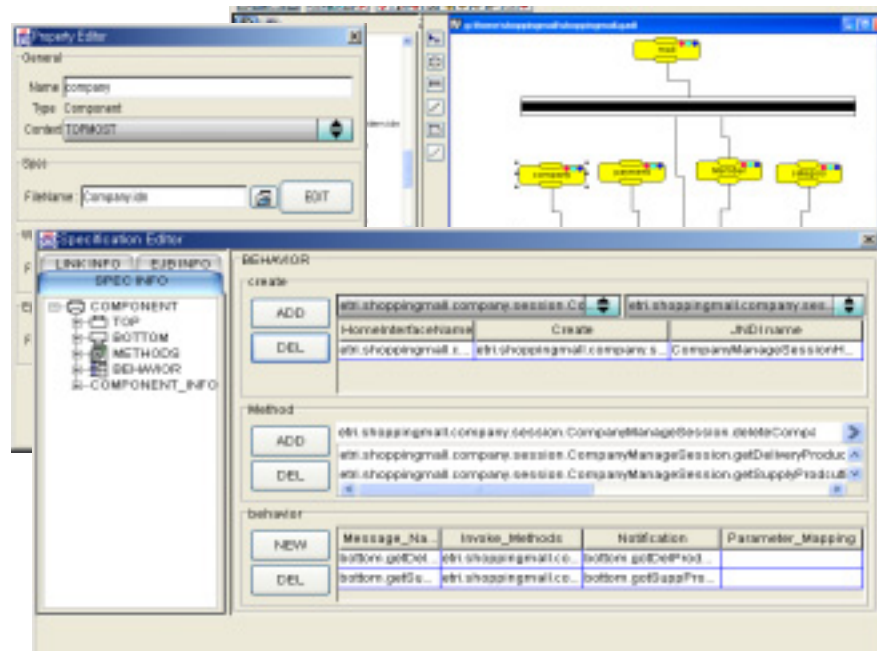


Figure 2. Architecture Editor and Component Specification Editor

## 2.2 MDA Based CBD Approach

As software technology changes rapidly, software developers are facing with problems such as system integration and interoperability between different technologies or platforms. To cope with these problems, the OMG proposed MDA that is a method of specifying and developing systems based on models, using OMG's existing standards (i.e., OMA, CORBA, UML) [1]. Thus, the MDA enables software engineers to integrate, change and maintain a technology specific system. The MDA approach produces PIM that is a technology independent model for system design and specification, then builds PSM that is associated with an actual implementation through an automatic mapping process, leading to efficient integration and maintenance of systems [2]. The MDA is supported by four key OMG modeling standards such as UML [5], MOF [6], CWM [7], and XML [8].

The authors propose MDA based CBD approach in order to support MDA core features and other additional key features such as reuse of domain architectures and visual component assembly by plug-&-play style, including CBD features. The MDA core features that we support are as follows.

- Model repository that stores and manages all models related to the MDA approach; various UML Profiles, transformation model, Meta editor model, PIM and PSM instance model, project model, etc.
- Model transformation that transforms automatically PIMs to PSMs according to transformation meta model. A model mapping rule is defined based on transformation metamodel and PIM is transformed to PSM following the mapping rule.
- Meta Editor that generates various modeling editors. Visual modeling of various models such as PIMs, PSMs. UML modeling is provided and a new modeling for the system architecture can be supported by managing the meta model of view models.

In addition to the MDA core features, we provide several features to support CBD. The distinctive and competitive features of our approach are as follows:

- An architecture centric approach that integrates a component development process with a component assembly process in a system architecture which consists of components and can be produced as an instance of a domain architecture.
- Architecture reuse that brings greater benefits to component reusers since reuse of architectures at a higher abstract level can guarantee more benefits than reuse of components in terms of software development productivity, cost and quality.
- Component identification that is automatically carried out by business modeling results that includes relationships between object models and use case models, and relationships between object models and activity models.
- Composition of components built from various component platforms such as J2EE, CORBA and .NET whereas the COBALT Assembler enables reusers to compose only J2EE/EJB components.
- Providing major distinguishing features of previous CBD project results such as visual component assembly by plug-&-

play, and binary component adaptation.

The authors are currently implementing the MDA based CBD tool that supports not only the core features of MDA but also the distinctive and comparative features described above. Figure 3 shows the structure of a tool system that consists of 13 subsystems. The authors describe some of them which are core functionalities as follows:

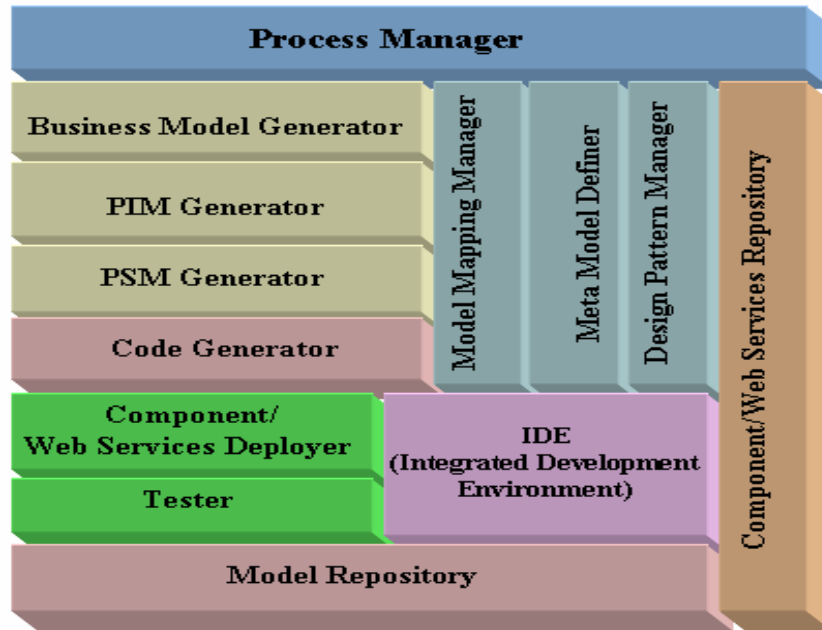


Figure 3. Structure of MDA Based CBD Tool

- Business Model Generator analyzes functional and nonfunctional requirements for a system to be built, and then identifies reusable components using relationships between object models, use case models and activity models.
- PIM Generator defines the architecture of a system to be developed, models the elements of the system according to UML 1.5 [11], and verifies generated models.
- PSM Generator receives PIM models, produces PSM models that relate to a specific platform using Model Mapping Manager, and refines the models.
- Model Mapping Manager enables users to describe model mapping rules according to the metamodel for model transformation, and automatically transforms PIM models to PSM models.
- Code Generator generates business logic code corresponding to a component platform from component information designed using ASL(Action Semantics Language) [11] in PIM stage.
- Process Manager generates a new process using profiles and additional process stages, traces and manages models that are built, changed and deleted during the MDA process.
- Meta Model Definer defines metamodels to produce models adequate for domain characteristics, and generates diagram editors for editing models of metamodels.
- Design Pattern Manager defines and manages design patterns that can be reused repeatedly for Business Model Generator, PIM Generator and PSM Generator, and enables users to apply related patterns to the design process.
- Model Repository stores and manages information on meta models and models produced from the tool. They manage not only UML Profiles but also transformation metamodel and meta editor model.

### 3. Comparisons

Several MDA based CBD tools have recently been announced to the software industry. ArcStyler [3] is an MDA tool developed by iO(Interactive Objects) Software, and develops model based software using the Convergent Architecture concept that is the most effective architecture style for integrating the Internet based enterprise. OptimalJ is a pattern based MDA tool developed by Compuware [4]. OptimalJ generates PIM models that are domain models such as classes and services, and automatically produces PSM models for the J2EE platform using a pattern based approach.

These tools follow the MDA standard and support the process for building software based on MDA. However, the tools loosely support UML Profiles and their transformation. They do not support visual component composition that assembles pre-

built components into a large size component or a specific software application. It is clear that a system should be modeled considering available components at the earlier stage in order to assemble reusable components, and that it is required to model the system, focusing on the interactions among components rather than internals of components. In addition, they do not support automatic component identification, reuse of domain architecture, binary component adaptation, and tightly integration of the component development process and assembly process, which can be supported by our MDA based CBD approach and tool.

## 4. Conclusions and Further Work

COP or CBD has recently been adopted as the better method for software development. However, CBD has problems with system integration and interoperability between different technologies or platforms. In addition, binary components produced from the third party cannot be reused without modifications due to different business processes among organizations. For these reasons, MDA has been introduced as an evolutionary approach to CBD. The development paradigm is currently shifting from CBD to Model Driven Development (MDD) based on the MDA standard. In order to solve the problems with CBD and to catch up the MDD paradigm, the authors have carried out MDA based CBD project, employing the results from our previous CBD project [10, 12] as possible as we can.

Comparing our approach with existing MDA solutions, our approach has some strength in terms of model management based on the meta model, architecture reuse, automatic component identification, binary component adaptation, visual component composition by plug-&-play, tight integration of the component development process and assembly process, and composition of components built from various component platforms such as J2EE, CORBA and .NET.

The tool related to our MDA based CBD approach has currently been implementing after building detail design specifications. The model mapping rule of our approach will be verified through some case studies. In addition, the automatic code generation process will be tested and refined towards generating nearly executable code. Consistency and completion checking between models will be supported by a formal model checking process.

## 5. References

- [1] Object Management Group, "Model Driven Architecture", OMG document ormsc/01-07-01.
- [2] Jon Siegel and the OMG Staff Strategy Group, "Developing in OMG's Model Driven Architecture", <ftp://ftp.omg.org/pub/docs/omg/01-12-01.pdf>, November 2001.
- [3] Interactive Objects Software, ArcStyler, [www.io-software.com](http://www.io-software.com).
- [4] Compuware, "Using OptimalJ", <http://javacentral.compuware.com>.
- [5] Unified Modeling Language Specification, Version 1.4, OMG document <http://cgi.omg.org/cgi-bin/doc?ad/01-02-13>.
- [6] Object Management Group, "Meta Object Facility (MOF) Specification, Version 1.4", OMG document formal/2002-04-03.
- [7] Object Management Group, "Common Warehouse Metamodel (CWM) Specification, Version 1.0", OMG document ad/01-02-01.
- [8] XMI 1.1 Specification, OMG document <http://cgi.omg.org/cgi-bin/doc?ad/99-10-02>.
- [9] Object Management Group, UML for Enterprise Richard Soley and the OMG Staff Strategy Group, "Model Driven Architecture", <ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf>, November 2001.
- [10] Woo-Jin Lee, Oh-Cheon Kwon, et al., "A method and tool support for identifying domain components using object usage information", ETRI Journal, April, 2003.
- [11] Action Semantics Models, Unified Modeling Language Specification, Version 1.5, OMG document, formal/03-03-01, March 2003.
- [12] Oh-Cheon Kwon, Gyu-Sang Shin, et al., "Technique for Adapting EJB Components according to Adaptation Patterns", ISE2001, Las Vegas, USA, June 2001.
- [13] Enterprise Distributed Object Computing Specification, OMG Adopted Specification ptc/02-02-05.
- [14] Distributed Object Computing Joint Final Submission Part II Supporting Annexes v0.1, July 2001.
- [15] Object Management Group, UML Profile for Sun Java Community Process JSR-26 currently under public review, <http://jcp.org/jsr/detail/26.jsp>.
- [16] MaRMI III, EJB Supporting CBD Methodology, ETRI(Electronics and Telecommunications Research Institute), Korea, 2001.