CIMO – Component Integration Model

Yan Xia, Anthony Tung Shuen Ho
School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore 639798
Tel. (65)7905465, Fax: (65)7904161, E-mail address: eyxia@ntu.edu.sg

YuCheng Zhang
Beijing Conuco Electronics Co. Ltd

Abstract

Component Object Model (COM) represents a binary interface standard that allows developers to build specialized software component that interface in a common way with other software components. After compiled, these components are integrated into an application and can interoperate with each other in a reliable, controlled manner. Can the components be integrated and operated into an application without re-compiled? This article describes the Component Integration Model (CIMO), a software platform that allows the components written by different software programmer to be integrated and operated into an application without re-compiling. Firstly, the paper concentrates on a general overview of the CIMO and describes the constitutions and functions of the CIMO architecture. Secondly, the paper presents the definition of CIMO component concept and addresses how CIMO facilitates users to establish scalable component-based applications, and how CIMO supports the synchronous and asynchronous communication between components. Thirdly, the paper explains how CIMO sets up the deployment of components and processes after Users develop components based on CIMO specification.

Keywords: *CIMO, Component Technology, Configuration, Integration*

1. Introduction

Component technology is very popular in the software world because COM represents a binary interface standard that allows developers to build specialized software component that interface in a common way with other software components. The components can be developed by separate project team members using a wide variety of language - independent tools, and can be used and reused in many applications if these components have the COM interface [1]. This technology provides tremendous design flexibility and opens a new world—and challenge—to software programmers, and demands new ways of thinking and working. However, the component technology does not pay more attention to the component integration problem. These components are not late-bound freely; they cannot be integrated or combined into a variety of applications without re-compiling. OLE-related technologies can integrate between functional components of all sorts, allowing the features of those components to evolve over time, wherever they may be – in the system or inside applications, inside in-process DLLs or out-of-process .EXE files [2]. But re-compiling for components is needed at most situations in OLE.

To meet the customer's requirements of adding or modifying functionality, or inserting new components to the existed application, we researched and developed the **Component Integration Model**. CIMO is a software platform that allows the components written by different software programmer to be integrated and operated into an application without re-compiling. Also, the CIMO can support to extend the functionality to component and to insert new components to the application. In this article, the CIMO technique and framework will be presented. Firstly, the paper gives a general overview of the CIMO and describes the constitutions and functions of the CIMO architecture. Secondly, the paper focuses on the definition of CIMO component concept and addresses how CIMO facilitates users to establish scalable component-based applications, and how CIMO supports the synchronous and asynchronous communication between components. Thirdly, the paper explains how CIMO sets up the deployment of components and processes, and how CIMO manages the components running in the application after Users develop components based on CIMO specification.

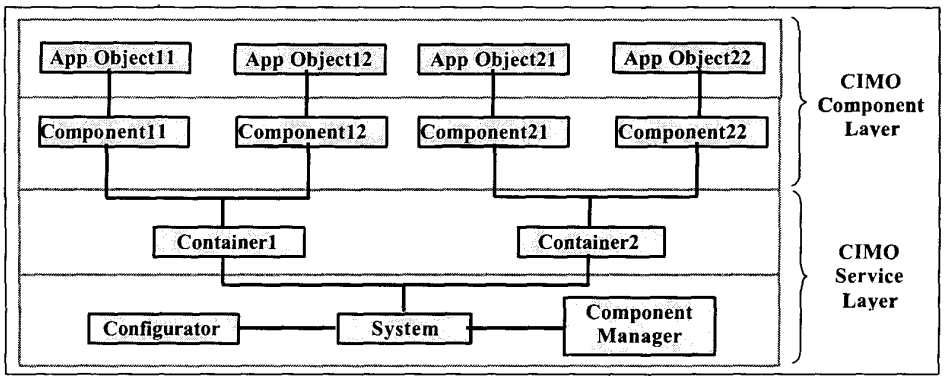

Figure 1 CIMO Architecture

2. Architecture of CIMO

How does CIMO facilitate customers to establish scalable component-based on application without re-compiling? What makes it such a useful and unifying model? To examine these problems that CIMO is meant to solve and how CIMO provides solutions for these problems, it will be helpful to first see the architecture of CIMO. CIMO has a layered structure: **CIMO Component Layer** and **CIMO Service Layer**. Figure 1 shows the static structure and feature of CIMO having objects at different levels and relationship between the various objects.

- CIMO Component Layer contains CIMO components, which work together and make up a CIMO application.
- CIMO Service Layer consists of CIMO service components -- **CIMO Configurator**, **CIMO Manager**, **CIMO Container**, **and CIMO System**. CIMO Service Layer is mainly intended for supporting the management, communication and configuration of CIMO components. A CIMO System and a group of CIMO Containers make up a platform for CIMO components - **CIMO Platform**.

CIMO will provide all the classes in CIMO Service Layer and instructions for application composers to establish the application components. Application composers will write application components and scripts for configuring the application system. Application users will use the final application.

3. CIMO Component Layer

CIMO components are located in the CIMO Component Layer. These components will be developed by application composers based on the component specification, and be integrated, managed and supported by CIMO Service Layer Components. CIMO components may be distributed on network and assigned to different processes.

3.1. CIMO Component Definition

Generally, a CIMO component is a Microsoft COM object. Each CIMO component is a discrete unit of code built on Component technologies that delivers a well-specified set of services through well-specified interfaces [2][3]. Compared with other component (standard Microsoft COM component), CIMO component has not only common COM interface, but also special **CIMO Component Interface**, as shown in figure 2.

- Each CIMO component may have a set of **Source Interfaces** that implemented by the component and have a set of **Sink Interfaces** that used by the component. It consumes messages incoming through its Source Interfaces and sends out messages through its Sink Interfaces.
- Each component has a group of services. These services handle the incoming messages through Source Interfaces, and the services may produce messages and send out through its Sink Interface.

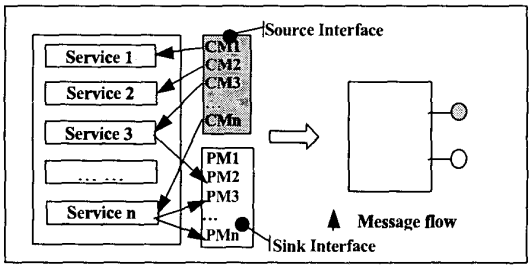

Figure 2 Structure of CIMO Component

A component is not a process. It may be a thread or an object staying in the process of a container. It can send and receive messages to other components via the CIMO Container or even via the CIMO System. It knows nothing about other components even if they are in the same CIMO Container. Component creator can extend functionality to components by adding more services in addition to the basic ones. Only in this way, concrete application can be realized.

3.2. CIMO Interface

The **CIMO Interface** is a group of semantically related functions, or "methods". Taken together, the methods in an interface define a logical group of services that a CIMO component object can provide the expected behavior and responsibilities. **CIMO Source Interface:** the CIMO Interface implemented by a CIMO component is a CIMO Source Interface, through which the component receives incoming messages. **CIMO Sink Interface:** the CIMO Interface used by a CIMO component is a CIMO Sink Interface, through which the component sends out-coming messages.

When a Sink Interface is coupled with a Source Interface, which type is the same as the Sink Interface, a client-server communication connection of two components is established. Figure 3 depicts how the three components are coupled together. Services existing in the CIMO Platform support the communication between components.

4. CIMO Service Layer

The CIMO software platform includes a **CIMO System** and a number of **CIMO Containers**, a **CIMO Configurator**, and a **CIMO Manager.** These components are located in the CIMO Service layer, and would guarantee safe operation in terms of process management, component management, memory management, and component communication.

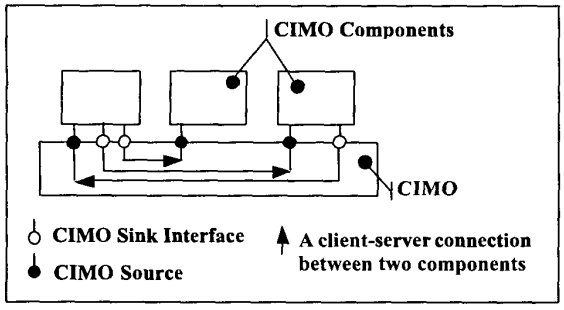

Figure 3 CIMO Component Integration

4.1. CIMO Platform

The CIMO platform provides a platform for CIMO components running on it. It supports the setup, integration, communication and management of the components in the CIMO application. A CIMO platform has one CIMO system to coordinate the whole system and has a number of CIMO containers, which directly contains CIMO components.

CIMO System

The CIMO system may contain a number of CIMO containers and coordinate them to provide a system-level management for the CIMO application. CIMO System is a process that is responsible for communication between Containers. And it also plays a partial role in communication between Components in different Containers. It is started by the end user; it gets the application topology information from application configuration and does work accordingly. For example, it constructs and starts all the containers, establishes communication channel to these containers, and requests them to construct components by sending messages to them according the configuration information. Additionally, the CIMO System knows only its children, which containers are in it, oblivious of its grand children, which components it has. The CIMO System talks to containers but not to components directly. It can send and receive messages, and communicates with Configurator and Component Manager directly, as shown in figure 4.

CIMO Container

High performance is an important requirement for component software architecture. While cross-process and cross-network transparency is a laudable goal, high performance is critical for software structure. So, a CIMO Container is used to organize the components that will interact within the same address space to utilize each other's services without any undue "system" overhead. A CIMO container contains a number of CIMO components and has its own process. It can also setup, integrate, manage components and provide communication management for these components. The CIMO Container communicates only with the System directly instead of other containers. If it wants to communicate with other containers, it has to perform via the CIMO System. It can send and receive messages, and play a role in communication between components in different containers.

CIMO Platform's Communication Mechanism

The CIMO components can only process and send the messages, know nothing about other components, and

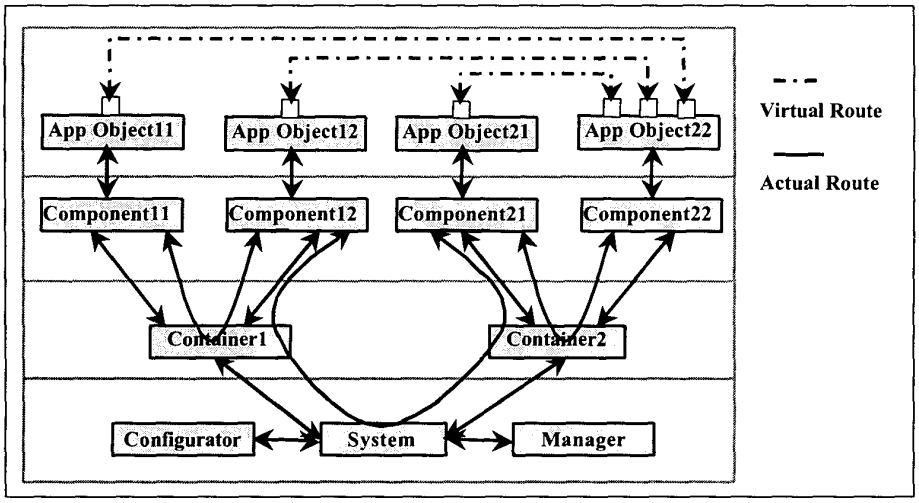

Figure 4 Communication Routes

additionally, may be distributed on network and assigned to different processes. So, communication plays an very important role in CIMO. The messages need to be exchanged smoothly and accurately between layer and layer, and among layer, and need also to run in one process, in different processes, and even at different computers. Fortunately, Microsoft COM technology has provided the communication mechanism for components; MS COM object can transfer any messages even over LANs. Additionally, in order to realize asynchronous transportation, asynchronous OLE automation and multithread technology are employed.

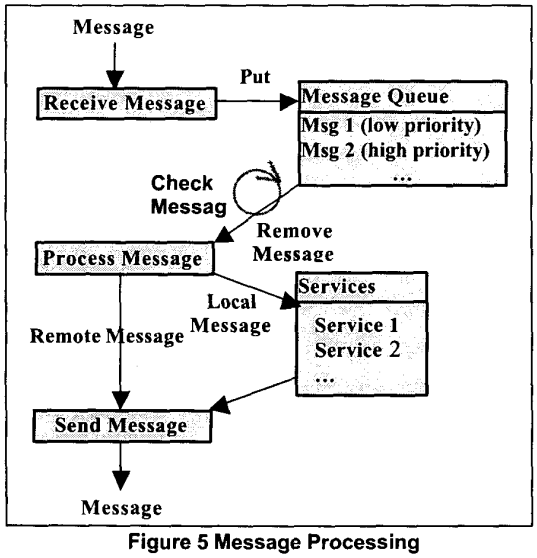

Figure 5 Message Processing

In the CIMO Platform, whenever a node, maybe a system object, a container object, or a component object, receives a message, it puts the message into its messages queue and this message is then waiting for processing. This node will check its message queue regularly and process the messages in it according to their priority, as shown as figure 5. When a node processes a message, it checks if the destination is itself. If so, the corresponding service will be invoked. If not, it checks whether the destination is one of its children, if so, the messages will be sent out again to this child. Otherwise the messages will be sent to its parent for further dispatching.

4.2. CIMO Component Manager

To guarantee the uniqueness of the components within the application's memory, each instantiated components must be registered into a centralized r egistry. The CIMO Component Manager is a centralized registry and manages the components running in the application. It belongs to the CIMO Service Component, intended for the registry and management of the CIMO Components and CIMO Interface, even the CIMO Messages. It records a reference to the running system. After a system starts up, only an object of the CIMO Component Manager can be created and then starts to store the reference of whole application's system. The CIMO components in the application's system are usually identified in the CIMO Component Manager using their unique identifiers. The CIMO Components Manager connects to an existing application system, acts just like a database manager to manipulate components, and exists until the system object terminates.

4.3. CIMO Configurator

The CIMO Configurator is a very important CIMO Service Component because it implements from basic software components to a running application system. It coordinates with the CIMO platform to configure applications, that is, reads configuration information to construct and start the application system, tells the application to create all the nodes (including a CIMO System, many CIMO Containers and Components) in the application, sets up message-service mapping within one node, and sets up links between components. It controls which components and services are loaded during the initialization process. After the Configurator has finished its task, it can exit without any effect to the platform. The tasks are as follow:

- Configure CIMO applications, such as process topology, component topology, message route within a component, link between components
- Terminate after an application system has been established. Also it can connect to and delete an existing application system
- Send messages to the application system in the debug mode.
- Show the application system structure and detailed information within each node, such as message route within a component and links of components.

Now, we support scripts as messages to construct the application system. Script file is a plain text file in a database. Each line is a message in the file. The configurator reads and runs the script files in a sequential order, and performs tasks taccordingly, either to create a system or to send messages to this system, Fig.6.

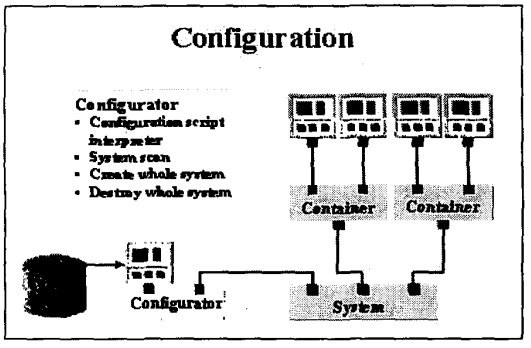

Figure 6 Concept of Configuration

5. Conclusion

We have successfully used the CIMO for control software for a new generation machine of Multi-Factor for MIKRON AG, a world-leading company of industrial automated machine in Switzerland. Although it is a simple software platform, it integrates and operates the CIMO components into an application without re-compiling.

- CIMO has a layered structure. CIMO Component Layer contains CIMO components, which work together and make up a CIMO application. The CIMO Service Layer objects intend for supporting the management, communication and configuration of CIMO components. Business logic deployed on centralized services rather than scattered user service makes CIMO to be changed easier so that it consistently gears to the customer's requirements.
- CIMO supports COM technology. The CIMO component has both COM Interface and CIMO Interface. They can be written by different software engineers based on the CIMO specification and can be integrated and operated into an application without re-compiling.
- CIMO is a fully configurable application system; the CIMO Configurator configures both the application and application messages, such as process topology, component topology, message route within a component, links between components, and so on.
- The CIMO components only process and send the messages. The CIMO provides the communication mechanism for messages exchanged, and supports synchronous and Asynchronous communication between components, distribution of components.
- Manageability. While cross-process and cross-network transparency is a goal, CIMO container is used to organize group components that will interact within the same address space, which helps in addressing high performance requirements.

Reference

[1] Sara Willliams and Charlie Kindel, "The Component Object Model: A Technical Overview", Dr. Dobb's Journal, the newsstand special edition, December 1994.

[2] Kraig Brockschmidt, "OLE Integration Technologies: A Technical Overview", Dr. Dobb's Journal, the newsstand special edition, December 1994.

[3] Paul Stafford and Joel Powell, "COM: A Model Problem Solver", March 10, 1995