프레임워크 기반의 서비스 개발환경

정회원 조 세 형*

Service Creation Environment: A Framework-based Approach

Sehyeong Cho* *Regular Member*

요 약

지능망 서비스 개발환경은 컴포넌트 기반의 재사용 가능한 통신망 서비스 개발을 지향하지만 서비스 개발환경에서의 새로운 서비스 독립 빌딩블록 (SIB) 의 도입은 여전히 전문가에 의한 작업에 의존해야만 한다. 비록 네트워크 플랫폼 자체가 어떤 SIB의 기능을 근본적으로 제공할 수 있도록 변경이 되었더라도 새로운 SIB를 이용하여 이 기능을 포함하는 새로운 서비스를 제공하는 것은 전혀 새로운 차원의 문제이다. 본 논문에서는 객체 프레임워크의 방식으로 서비스 개발환경을 설계함으로써 새로운 SIB의 기능을 서비스 개발환경에 손쉽게 추가할 수 있음을 보였으며 이 개념의 실현을 위한 구현 결과와 자동화 도구들을 설명하고 있다.

ABSTRACT

While service creation environment was intended for component-based, re-usable development environment, the development of a new SIB is still considered a task only for specialized engineers. Even if we had the capability of executing the new SIB in the network platform, it is not easy to modify the SCE platform so that we can design and create a new service logic program using the new SIB. This paper describes a framework-based approach to service creation environment. This approach enables the evolution of a service creation environment, by being able to add support for new SIBs into the service creation platform, without any modifications to the SCE platform. We also discuss automated tools used for creation of new SIBs.

I. Introduction

A service-independent building block, or a SIB, is an abstraction of certain network capability for use in an intelligent network[1]. SIBs were originally created as a tool for defining the capability of a new network, but also can be used as reusable service components when designing new services[2]. In the latter case, a SIB is the basic unit of service logic specification and implementation. In the past, the service- independent building blocks were hard-wired in a service creation environment. Therefore it was not possible for ordinary SCE users to modify the service creation environment to incorporate new SIBs. The SCE developers, in order to add a new SIB to the SCE, should modify the code for graphic-editing capability of a new SIB, simulation capability of the SIB, code-generation capability of the SIB, and so on and so forth. The code for these are normally grouped by functions, not by SIBs (Figure1). This forces the SCE developer to modify the whole source code for SCE, resulting in a completely new implementation. From a software engineering point of view, it is a bad practice because it fails to localize the potential

* 명지대학교 전자정보통신공학부(shcho@mju.ac.kr)
논문번호 : K01025-0118, 접수일자 : 2001년1월18일
※ 본 연구는 명지대학교 신임교수 연구지원과제로 수행되었습니다.

instability, but introduces it throughout the whole system.

In this paper, we propose an architecture for SIBs in a framework-based SCE, and introduce the tools for creating SIB components to be used in the evolving SCE. Using SCE framework, it is easier to add or delete SIB's to the SCE.

Section 2 introduces the concept of framework, and surveys related work. Section 3 discusses a framework-based SCE architecture. It describes in detail how the framework is structured and how SIB applications are developed. Section 4 discusses the implementation issues. Finally, section 5 discusses the implication and future work.

Ⅱ. SCE and Framework Technology

Object-oriented paradigm provides a sound basis for reusing software. This is so because objects are greater in cohesion, and gives out simpler ways of interaction with the rest of the system (i.e., lower coupling). While objects are reused "individually," frameworks[3] provide an organized environment for running a collection of objects. It also provides tools that let you construct components that are willing to play by the framework's rules. When we use simply objects, we need to modify the main thread of control for using the new object. On the other hand, when we use framework, the framework - which is a sort of 'main program' - doesn't need to change when we add new objects or components, just like we add a new plug-in for a Web browser. Therefore frameworks provide a simpler and safer way of incorporating new components into exiting software.

It would be desirable to have an architecture for service creation environment based on the concept of framework, and also automated tools to help developing SIB application programs. Frameworks have a structure for easier installation and removal of new components. In a framework-base service creation environment, the set of functions that supports a SIB can become a component. All information about the SIB(i.e., the

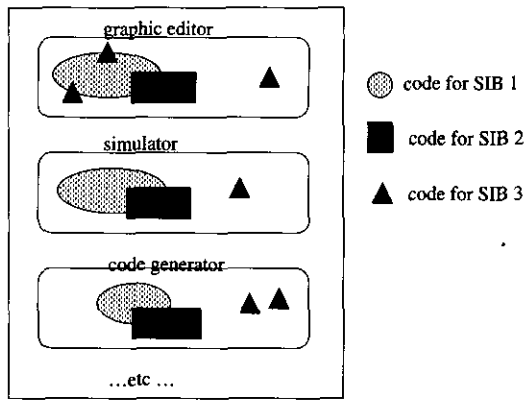

Fig. 1 Conventional SCE: the code for a single SIB is scattered throughout the system

data part), and the service specification operation, code generation operation, etc. (i.e., the operations part) constitute an application program (component) for a SIB support. When it is necessary to support new SIB capability, we do not re-code the SCE, but we simply register a new component, which is developed by using automated tools independent of the SCE framework. Once the SIB support component is registered, all SIB-dependent operations are performed by the SIB support component, while SIB-independent function are performed in the framework, which includes relations among SIB instances, event processing, and coordination. When for any reason a SIB is no longer used in the SCE, we simply de-register the SIB, and the service creation environment no longer supports the deleted SIB.

Recent work in TOSCA (TINA Open Service Creation Architecture) adopted framework approach [4,11]. In TOSCA, service components are created in three distinct phases: specification, design, and coding. In TOSCA, component API and behavior are distinguished, and API is defined by IDL, or ODL, and behavior is defined by using C++ language. TOSCA is aimed at TINA architecture[5], and the service which are developed in TOSCA is framework-based. The usage of framework in TOSCA differs from our case: in TOSCA the service software is framework based; in our case, the tool itself is framework-based.

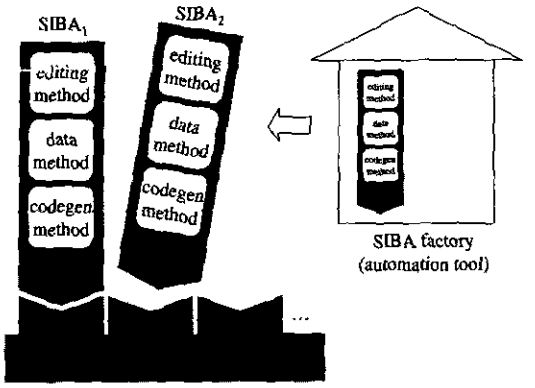

Fig. 2 OpenSCI framework and SIBA component

Outside the telecommunications world, there is OpenDoc[6], a pioneering work in object framework, although not much used currently. OpenDoc is based on SOM, or System Object Model, which is OpenDoc's own object bus. Microsoft's OLE(Object Linking and Embedding) is a framework based on Common Object Model, or COM[7]. In these frameworks, development of a component is carried out in two parts, the API and the behavior. The API is defined by IDL, and the behavior is defined by using the templates, or skeleton, generated by the IDL compiler.

The development of SIB presented in this paper also consists of specification phase and coding phase. In the specification phase, we use form-based GUI to define he SIB API. In the coding phase, C++ code is automatically generated based on the SIB specification. The generated code becomes the basis for further enhancement by the programmer. By using SIB automation tools, SIBs can easily be developed to be included in the service creation environment, saving a great deal of time.

Ⅲ. OpenSCI Architecture

OpenSCI(OPen ENvironment for Service Component Integration) is based on a framework architecture, where adding new components or deleting an old, unused components is easy. The functions provided in OpenSCI include SIB-based editing of global service logic, storage, SLP code generation, SIB registration and deregistration. In order to provide these features, OpenSCI framework comprises the OpenSCI container and OpenSCI components. SIB-specific features and information are contained in a SIB component called a SIB agent, or a SIBA for short. GSL-related storage, editing and code generation features are included in the container. Figure 3 shows the class hierarchy.

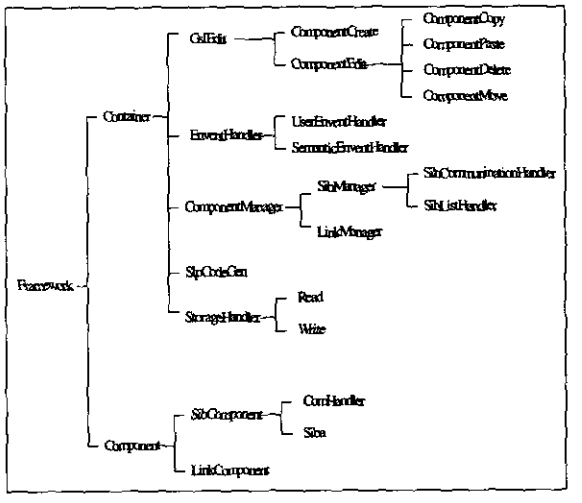

Fig. 3 Constituent Classes of OpenSCI

OpenSCI components comprise SIB components and link components. Link components are objects that are used to chain the SIBs together to form a service logic. The components, technically speaking, are independent application programs and do not alter OpenSCI framework in any ways. Components can be added or deleted at will. One who wishes to create a new component will do so by subclassing the 'component' in the class hierarchy shown in Figure 3. Currently OpenSCI is targeted to CS-2 SIB's, and tested on a CORBA-based IN CS-2[1] simulated environment. Using OpenSCI, one can create a SIB-based service specification and can automatically produce object-oriented Service Logic Programs.

1. The Structure of a SIB Agent

A SIBA(SIB Agent) is the application program for a specific SIB class. A SIBA consists of

attributes, operations, and events. Attributes and operations which are publicized to the OpenSCI framework are used by the framework. Those that are not publicized are used inside the SIBA. SIBA attributes include visual attributes that are used by the graphical user interface, logical input and output, and the IPC[1] parameters. Figure 4 illustrates the operations of a SIBA.

SIB Data management operation initializes the SIB instance attributes, assigns new values, or manages the storage for some attributes. Visual processing operation takes the responsibility of visualizing SIB instances on the screen of the service creation environment. SIB parameter editing operation implements the GUI that interacts with the user to assign appropriate values for the SIB instance to behave the way it is intended. SLP creation operation produces part of the service logic that will be combined with the rest of the code that is pertinent to the global service logic and other SIB's. The generated code uses IDL(Interface Definition Language). OpenSCI framework itself is the container and has the responsibility of coordinating the code generation for the global service logic. It traverses the service logic graph, producing the overall control structure, and indivisual SIBA's (the componenets) provide SIB-specific code generation when requested by the framework. Figure 5 shows the relation between the framework and the components while doing the code generation.

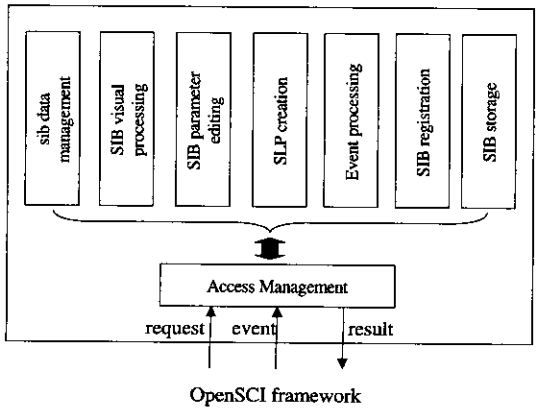

Fig. 4 SIBA Operations

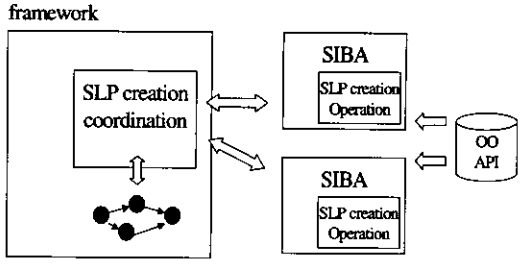

Fig. 5 Relation among SIBAs and framework in SLP code generation

Event processing operation is a set of actions that are related to external events. SIB registration operation is responsible for registering the SIBA into the OpenSCI platform when adding new SIBA, as well as for deregistration, when no longer necessary. SIB storage operation serializes and stores the SIB instance to or reads from external storage. This is done when there is a request from the storage coordinator inside the framework. Like the code generation coordination, the framework is responsible for storing the global information, while SIBA storage operation is responsible for the storage of the individual SIB instance. Currently, SIBAs communicate with the framework by using MicroSoft's COM(Common Object Model)[7].

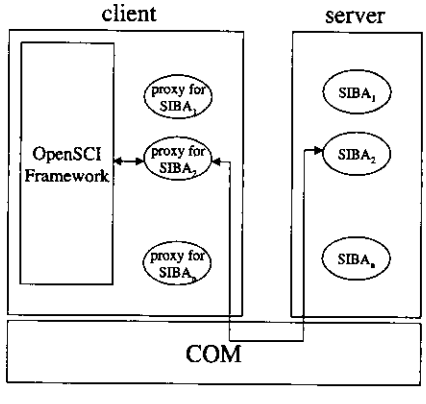

Fig. 6 How OpenSCI framework communicates with SIBAs

Inside the OpenSCI framework, operations are performed using a client-server model. The framework container is the client, and the SIBAs are servers. The container forms proxy objects in the client area and communicates directly with the

proxy, not the SIBA. The proxy, in turn, uses COM to communicate with the SIBAs, giving the illusion that the client(framework) is directly communicating with the server (SIBA).

2. SIB Automation Tool

The SIB Automation tool produces the application program of which the structure is shown in section 3.1. It also produces the resource file that characterizes the graphical user interface. The procedure for using the tool to create a new SIBA is shown in Figure 7.

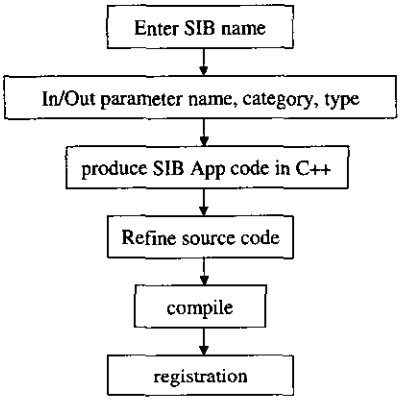

Fig. 7 SIB development using the SIB Automation Tool

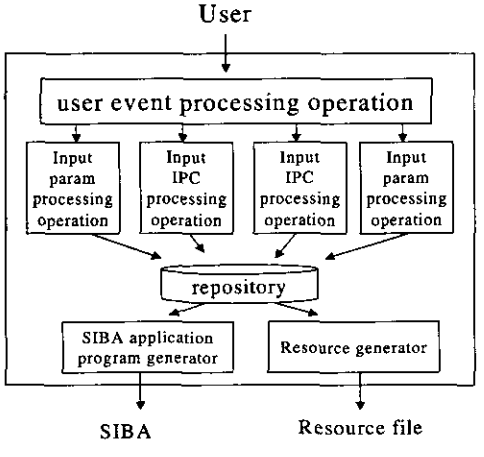

Fig. 8 Structure of the SIB Automation Tool

First, the user specifies the name of the new SIB, and then defines the name of the SIB parameters. When defining the I/O parameters and IPC parameters, the user specifies whether it is a service support data (SSD), call instance data (CID), or service instance data(SID). Also the types of the data (string, integer, etc) are specified, so that appropriate dialog is created and memory is allocated. When all data are specified by the SIB developer, SIB automation tool creates the SIBA application program and the resource that characterizes the GUI. The SIB developer also can modify the generated source code for additional features. After compiling the program together with the resource file, it is registered to the OpenSCI framework. Figure 8 shows the structure of the SIB automation tool.

The SIB application program created using the tool inherits SIBA class and ComHandler class that are in the hierarchy of OpenSCI shown in Figure 1. SIBA class is the class that implements the basic SIB structure template and all common operations. ComHander is the class for communication between SIBA and the framework(i.e., the container). In terms of implementation, Com Handler is implemented by inheriting Microsoft COleControl class. All SIBs are defined by subclassing SIBA and ComHandler.

Ⅳ. Implementation

Current implementation of OpenSCI is MS Windows-based. Instead of implementing the whole framework from scratch, we used Microsoft's ActiveX control and the container for rapid implementation. Only the automation tools are built from scratch.

A SIBA is created by using Microsoft's Visual C++ and the SIBA definition tool introduced in the previous section. The MS Visual C++ ActiveX Control wizard is used to create ActiveX control programs and ActiveX control resource files. Using ClassWizard, various properties are added, including SIBA name, SIB editing method signature, and user Events. The SIB application program is inserted into the ActiveX control program. The control program is then refined as necessary. Also the SIB application can be refined if desired. They are then compiled and registered to the OpenSCI framework. Technically, this is accomplished by

using the Windows system registry.

Figure 9 shows the snapshot of defining an input parameter 'Screen list name' for Screen SIB.

On the left side of Figure 9, Screen list name will act as either an SSD or a CID, as this was chosen, and the data value will be of type string. The right side of the Figure shows the dialog for defining an output IPC 'POI-Initiate', for Initiate Service Process SIB.

Figure 10 shows the SCE platform before and after registration of a new SIB agent. The new SIB inserted is Compare SIB, which compares two strings and makes branches according to the comparison result. The graphical icon in the toolbar (rightmost one) is for the selection of Compare SIB. The Global service logic shown in Figure 10 is part of a Video-On-Demand service,

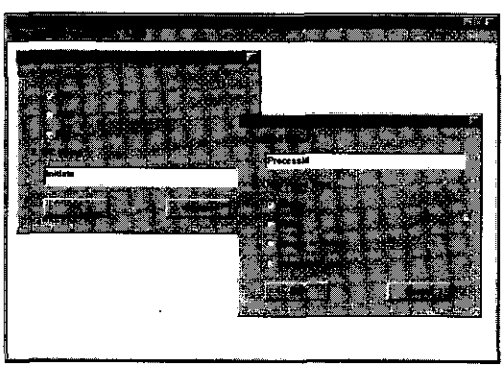

Fig. 9 Input and Output Parameter definition using the automation tool

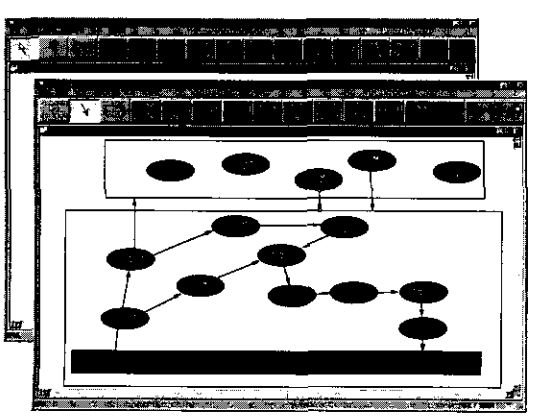

Fig. 10 SCE platform before (left) and after(right) registration of "Compare" SIBA. Note that a new button has been created in the toolbar.

which consists of two service processes. The VOD service created has been tested to run on a CORBA-based IN CS-2 platform.

V. Conclusion

Object framework technology can be used to build a flexible service creation environment. We presented a framework structure for a service creation environment, and demonstrated that an SCE can be extended to support new SIBs without complete overhaul of the SCE implementation. We also introduced an implementation of such SCE and the automation tools for developing new SIB capabilities that are to be included in the existing SCE.

In OpenSCI, a SIB becomes a component, and the component is an application that comprises the SIB-specific data, service specification operation, code generation operation, and so on. By using SIB automation tools, a SIB in this framework can easily be developed. This enables fast evolution of service creation environment.

The implication is this. First, as we alluded to above, it is easier to create a SCE that is more robust, in terms of modifications and addition of new SIBs. Second, SIBAs can even be executed remotely, with slight modification (using Distributed COM). Finally, using this architecture, SIBAs can be developed independent of the platform and then integrated. This means a change in the value chain of service creation: not only services are developed by third parties, but service components and service creation components can be developed by third parties. Although the methodology has been tested only in the IN CS-2 domain, it is equally applicable to CS-3[8], TINA, or other network API's including Parlay [9] and JAIN [10].

However, we only dealt with the framework-based implementation of 'high-level SCE', that is, the GUI-based composition tool, but not the entire service creation process, such as requirement analysis, deployment and testing. In order to be more effective, it is also desirable to incorporate

tools for such processes in a tightly integrated manner. This, at the moment, is left as a future work.

References

[1] ITU-T Recommendation Q.1223: Global Functional Plane for IN Capability Set 2, ITU-T, Geneva, Mar. 1997.

[2] Sehyeong Cho, "Service Creation Environment - principles and practice," Tutorial Notes, IEEE Intelligent Network Workshop, Melbourne, Australia, May 1996

[3] Robert Orfali, Dan Harkey, Jeri Edwards, "The Essential Distributed Objects Survival Guide", John Wiley & Sons, Inc., 1996.

[4] TOSCA Deleverable 6. Service Creation: The TOSCA Paradigm and Framework Approach, August 1997.

[5] TINA-C, Service Architecture, June 1997.

[6] IBM, "OpenDoc programming Guide", http://doofus.mit.org/OpenDOC/html/guide/

[7] Microsoft, "COM specification." On-line developer library, http://msdn.microsoft.com/library/default.asp?URL=/library/books/inole/s10d8.htm

[8] ITU-T Recommendation Q.1233: Global Functional Plane for IN Capability Set 3, ITU-T, Geneva, Mar. 1999

[9] Parlay API Business Benifits White Paper, version 1.0, June 1999, the Parlay Group

[10] D. Tait, J. de Keijzer, and R. Goedman , "JAIN: A New Approach to Services in Communication Networks," IEEE Communications Magazine, January 2000

[11] R.O.Sinnott, "Frameworks: the future of formal software development?," Computer Standards and Interfaces 19 (1998) pp.375-385

조 세 형 (Sehyeong Cho) 정회원

1981년 2월 : 서울대학교 섬유공학과 졸업

1983년 2월 : 서울대학교 계산통계학과 석사

1992년 5월 : 펜실바니아 주립대학 박사 (전산과학)

<주관심 분야> 통신서비스, 분산처리, 인공지능