C2 아키텍처를 변형한 메시지 중앙처리 기반의 Component 활용 기법

정 화 영

An Approach to Application of Component Based on Message Central Processing change the C2 Architecture

Hwa-Young Jeong

요 약

CBD를 지원하는 소프트웨어 개발기법이 최근 많은 관심과 함께 도입되면서 이를 효과적으로 운용할 수 있는 아키텍처기반의 조립 및 활용분야가 활발히 연구되고 있다. 특히, C2 아키텍처는 GUI를 지원하기 위한 메시지방식의 컴포넌트 조합기법이라는 점에서 많은 관심을 받고 있다. 그러나, 컴포넌트들의 계층적인 순차성과 메소드 호출방식의 서버컴포넌트의 경우 이를 적용하려면 컴포넌트의 수정이 불가피하다.

따라서, 본 논문에서는 C2아키텍처에서 일부 변형하여 메시지핸들링부분을 컴포넌트와 커넥터가 아닌 메시지 자체에 둠으로써 메소드 호출방식이라도 컴포넌트의 수정이 필요 없이 조합 및 운용할 수 있는 Plug-and-Play를 지원하도록 하였다. 또한, 계층적인 순차성을 갖지 않고 메시지를 사이에 둔 컴포넌트들의 병렬성을 갖도록 하여 보다 자유로운 메시지 핸들링이 가능하도록 하였다.

ABSTRACT

Recently, Software development method supported CBD is applied with many concern and is researched with part of application and composition based-on architecture effectively use it. Effectively, C2 architecture has been concern with the point of component composition method based-on message driven for supported GUI. But, In case of classified sequence in component and method call method in server component, component must be modified to apply it.

Thus, In this paper, Message handling part with a part of C2 architecture change is locate in the message neither component not connector. So, Although method call method it can be composit and operate component for support Plug-and-Play without modification. Also, it's possible the more flexible message handling with parallel composition of component between message without classified sequence.

키워드

EJB, C2, 컴포넌트 기반 개발(CBD), 컴포넌트 기반 소프트웨어 공학(CBSE), 컴포넌트 활용, Multi-Server Application

Ⅰ. 서 론

컴포넌트 기술은 소프트웨어 프로그래밍에서 하드웨어 개발 환경처럼 소프트웨어 Plug-and-Play 방식으로 시스템을 구축하는 '합성을 통한 시스템 구축'으로의 전환을 목적으로 한다[1, 2]. 따라서 CBD를 효과적으로 지원하기 위해서는 응용 컴포넌트들이 서로 정확하게 결합하여 작동할 수 있는 아키텍처를 기반으로 컴포넌트의 생성과 합성작업이 이루어질 수 있어야한다[3]. 이를 위

예원예술대학교

접수일자 2003. 3. 19

하여 , 컴포넌트 합성방법에서는 컴포넌트간의 인터페이스 불일치를 해결할 수 있어야하며, 독립적인 컴포넌트의 메소드 수정 없이 합성할 수 있어야 한다[12]. 아키텍처 기반기술은 UNIX의 Pipe-and-filter아키텍처와 일반적인 어플리케이션에서 오랜 기간에 사용된 Blackboard 아키텍처 기술[4]에서부터 Style 기반의 Unicon[5], Aesop[6], C2[7], 시멘틱 모델 기반의 Wright[8], Rapide[9], Domain Spec 기반의 실시간 객체지향 구조의 ROOM[10] 등을 들 수 있다. 이들 중, C2 스타일 아키텍처[11]는 컴포넌트간의 직접적인 메소드 호출방식이 아닌 메시지 전달방식의 비 동기적인 상호작용을 지원하는 대표적인 구조라 할 수 있다. 그러나 C2 스타일은 각 컴포넌트들 간의 Top, Bottom을 이용한 커넥터상의 계층적 구조를 가짐으로써 중간노드의 컴포넌트에 관한 결과가 요구될 때 바로 결과를 확인할 수 가없고, 여러 단계의 커넥터 포트를 통하거나 최하위계층의 컴포넌트에 관한 Notification을 통하여 확인된다. 또한, 컴포넌트간 메시지 전달을 위하여 In/Out벡터 2개와 각 컴포넌트마다 top, bottom 메시지 공간 등 최소 4개의 메모리공간을 필요로 한다. 또한, 메소드 호출방식을 기반으로 한 서버측 컴포넌트 모델인 EJB의 적용시 C2 스타일 구조의 변형이 불가피하다. 즉, 메소드 호출방식을 메시지호출방식으로 전환하여야 한다[12].

따라서 본 논문에서는 C2 스타일을 변형하여 컴포넌트와 컴포넌트간의 직접적인 메시지 핸들링을 컴포넌트 조립구조가 아닌 메시지 자체에 두는 메시지 중앙처리방식을 제안한다. 이는, 컴포넌트간의 계층적 구조가 아닌 메시지공간을 매개체로 한 병렬적 구조를 갖는다. 또한, 1개의 메시지 명 벡터와 Request/Notification 메시지 공간 등 3개의 메시지공간만을 사용하였다.

II. 관련연구

C2 스타일 아키텍처의 기본원칙은 메시지기반의 컴포넌트간 통신, 멀티쓰레드, 각 계층의 독립성, 메시지 라우팅 커넥터를 통한 컴포넌트들의 연결구성, GUI 소프트웨어 요구사항 지원 등이다. 또한, 분산 및 이기종 환경, 분할된 주소공간을 갖지 않는 환경에서의 컴포넌트실행, 다중사용자 및 다중사용자 툴킷, 실행시간에서 변화될 수 있는 동적 구조 등에 적합하다[4]. C2 스타일의 기본구조는 <그림 1>과 같이 각 계층을 기반으로 메시지의 상호작용을 하도록 되어있다.

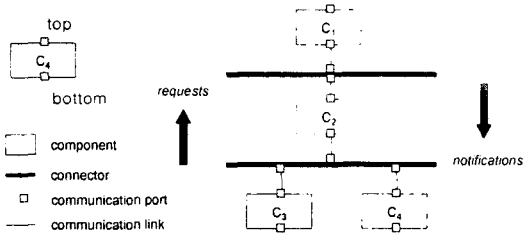

그림 1. C2 스타일 아키텍처의 기본구조

각 계층은 컴포넌트와 커넥터 2개 형식의 블록으로 이루어져 있으며, 각 컴포넌트와 커넥터에서 top 포트 와 bottom 포트를 가지고 있다.

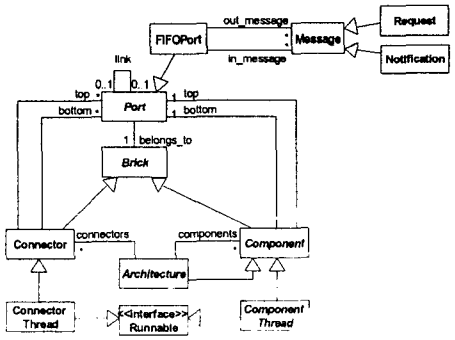

그림 2. C2 스타일 아키텍처 프레임워크

이를 통한 메시지교환은 컴포넌트의 top 포트에서 상위계층의 컴포넌트로 보내는 Request 메시지와 상위계층 컴포넌트의 bottom 포트로부터 전달되는 Notification 메시지를 받을 수 있다. 따라서 각 계층의 컴포넌트들은 독립적이며 최하위계층의 컴포넌트의 Notification 메시지를 통하여 최종결과를 확인할 수 있다.

또한, 이를 활용하기 위한 프래임워크의 구조는 <그림 2>와 같이 포트를 통하여 전달받은 메시지는 해당 커넥터와 컴포넌트로 전달된다. 이를 위

하여, 각 커넥터와 컴포넌트에서 쓰레드를 동작시켜 메시지를 확인하고 있다.

III. C2 아키텍처를 응용한 메시지 중앙처리 방식의 Component 활용 기법

3.1. C2 아키텍처를 변형한 메시지 중앙 처리 방식의 구조

컴포넌트의 조립 및 활용은 Plug-and- Play 조립방식을 지원함으로서 기존의 컴포넌트에 관한 수정작업이 없이 조립 가능하여야한다. C2 아키텍처에서는 서버측 컴포넌트 모델인 EJB의 경우 컴포넌트간의 직접적인 메소드 호출방식을 택하고 있으며 EJB에서 쓰레드를 시작, 중지, 재시작 할 수 없으므로 이에 적용하기 위해서는 많은 수정이 필요하다. 이를 위하여 <그림 3>과 같은 방법이 제시된바있다[12].

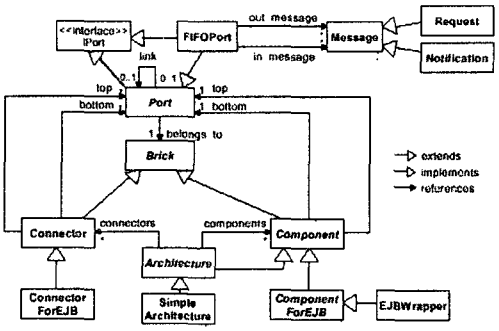

그림 3. EJB적용에 관한 제시된 방법

그러나 이러한 방식은 EJB 컴포넌트와 C2 컴포넌트를 맵핑하기 위한 별도의 EJBWrapper가 필요하며, C2의 계층적 구조를 따름으로써 중간단계의 조립 컴포넌트에 대한 결과를 확인하기 위하여 각 조립계층에 의한 역순의 메시지 전달과정이 필요하다. 따라서 본 논문에서는 위 문제를 해결하기 위한 방안으로 C2스타일 프레임워크를 <그림 4>와 같이 변형하였다.

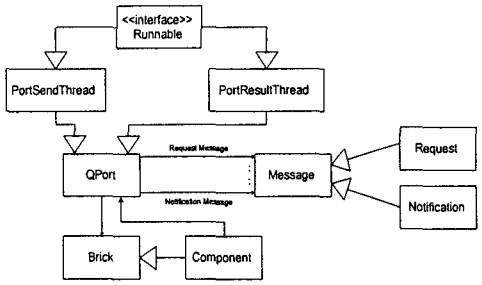

그림 4. C2 스타일 아키텍처를 변형한 제안된 프레임워크

즉, 메시지의 처리를 컴포넌트 조립구조에서 구성하지 않고 메시지 자체에서 쓰레드를 통하여 자체 핸들링 하도록 하였다. 이에 따라, 컴포넌트의 메시지는 QPort를 통하여 메시지영역에 저장되며 PortSendThread와 PortResultThread부분에서 핸들링 함으로써 기존의 컴포넌트에 관한 수정 없이 조합 및 운용할 수 있다. 조립형태는 <그림 5>에서와 같이 핸들링 되는 메시지의 이름공간을 두어 이를 식별하고 그 결과를 확인할 수 있도록 하였으며, 다음의 메시지를 진행할 수 있도록 하였다.

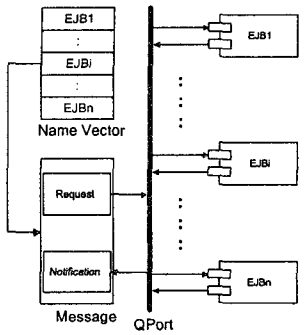

그림 5. C2 스타일 아키텍처를 변형한 제안된 메시지 중앙저리방식

<그림 6>에서는 본 논문에서 제시된 기법을 활용하여 각 컴포넌트를 조립하고 이를 운용하는 프레임워크를 나타낸다.

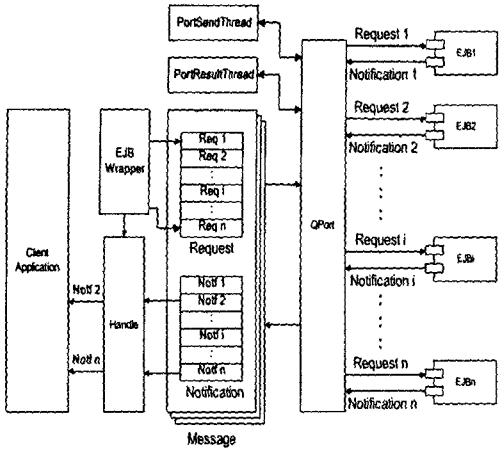

그림 6. 제안된 메시지 중앙저리방식의 컴포넌트 조합 프레임워크

본 기법에서 EJB Wrapper는 조합될 컴포넌트들의 인터페이스 정보를 갖고 있으며, 이를 통하여 각 컴포넌트들의 요구 메소드들을 초기화한다. Handle에서는 Wrapper에서의 컴포넌트 정보들을 기반으로 컴포넌트들의 호출 메소드들을 메시지공간의 Request영역과 이름 벡터에 적재한다. 적재된 Request메시지들은 PortSendThread를 통하여 확인되고 각 컴포넌트의 조립순서에 따라 메소드 메시지들을 호출하며, 확인된 결과 메시지들은 Notification에 적재한다. PortResultThread는 Notification에 적재된 메시지들을 확인하고 Handle을 통하여 요구된 컴포넌트의 결과를 사용자에게 나타낸다. 본 논문에서 제안된 메시지 중앙처리방식을 적용할 경우 각 컴포넌트들의 인터페이스정보 및 조립순서들을 Wrapper에 등록시키고 이를 통하여 메시지단계에서만 각 컴포넌트들의 통신을 핸들링 함으로써 컴포넌트의 수정 없이 조합 및 운용이 가능하다. 조립구성에 따른 결과는 이름 벡터와 Notification 메시지를 통하여 확인이 가능하다.

3.2. 메시지 중앙처리방식의 조립 및 활용에 관한 모델링

본 기법의 활용을 위한 모델링은 UML을 이용하였다. 이에 따라, Deployment Diagram은 <그림 7>과 같이 나타내며 각 EJB 서버를 별도로 두는 다중서버환경에 이용할 수 있다.

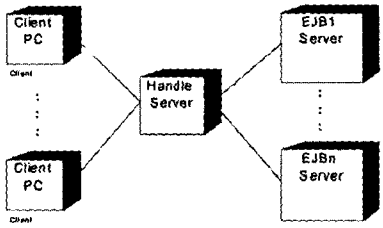

그림 7. Deployment Diagram

또한, 컴포넌트의 조합 및 운용을 위한 메시지 핸들링 기법은 다음 <그림 8>과 같이 EJBWrapper에서 조립 컴포넌트를 초기화 한 후, Handle에서 각 컴포넌트의 호출 메소드를 Request 메시지로 변환하여 저장하고 실행된 Notification 메시지를 클라이언트에 나타낸다.

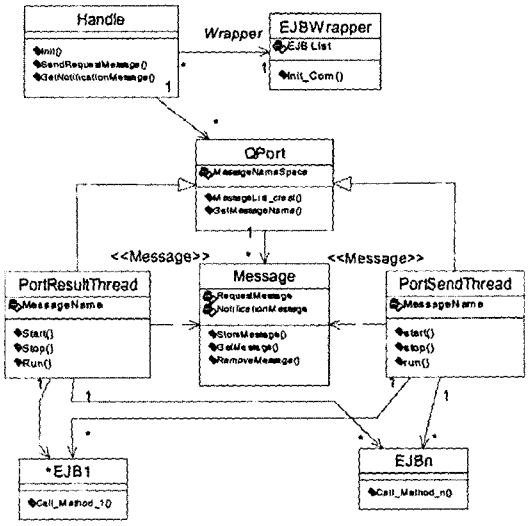

그림 8. Class Diagram

이를 구체적인 각 단계별로 처리흐름을 나타내면 다음 <그림 9>의 Sequence Diagram과 같다.

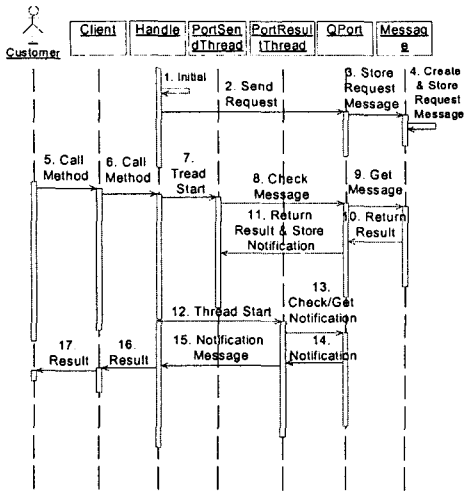

그림 9. Sequence Diagram

IV. 결과 및 평가

본 제안 기법은 C2아키텍처를 변형하여 메시지의 핸들링 부분을 컴포넌트와 커넥터가 아닌 메시지부분에서 처리하도록 하였다. 즉, 컴포넌트와 커넥터부분은 각각의 쓰레드를 포함하여 2개씩이나 본 제안기법에서는 메시지 핸들링부분을 메시지에 둠으로써 컴포넌트부분을 수정하지 않는다. 또한, 포트부분에서 C2아키텍처의 경우 메시지의 창구 역할만 함으로써 포트와 FIFOPort를 두고 있으나, 본 제안기법은 메시지부분으로부터의 처리까지 담당함으로 QPort, PortSendThread, PortResultThread 등 3개의 포트가 필요하다. 그러나 메시지처리를 위한 Thread의 수는 본 기법과 C2아키텍처 모두 같다. 이에 따라, 실제 본 제안기법과의 구조 및 운용상의 차이점은 다음 <표 1>와 같이 나타난다.

컴포넌트간 병렬적 처리부분이나 중간단계의 컴포넌트 결과 확인시 C2아키텍처의 경우 컴포넌트의 Notification메시지를 직접 확인하지 못하고 커넥터의 포트를 이용하여 몇 개의 단계를 거쳐야만 확인이 가능하다. 그러나 본 기법에서는 메시지 단계에서 바로 핸들링이 가능하다.

표 1. 구조 및 운용상의 차이점

특징	C2아키텍처	본 제안 기법
구조	컴포넌트간 계층적 순차구조	메시지 List를 통한 컴포넌트간 · 병렬구조
컴포넌트간 순차적인 메시지처리	가능	가능
컴포넌트간 병렬적인 메시지처리	여러 단계의 처리필요	쉬움
조립된 중간단계의 컴포넌트 결과확인	여러 단계의 처리필요	쉬움
메시지 저장소	4개	3개

또한, 메시지 저장소 부분에서는 C2아키텍처의 경우 in/out벡터 2개와 Request/Notification 메시지 2개가 필요하지만, 본 기법의 경우 메시지 명 벡터, Request/Notification 메시지 등 3개만이 사용된다. 따라서 많은 메시지공간의 활용은 시스템의 처리 속도를 저하시키며 프로세스의 복잡성을 가져온다.

V. 결 론

본 논문에서는 C2아키텍처의 구조를 변형하여 메소드 호출방식의 컴포넌트에서도 쉽게 활용할 수 있으며, 컴포넌트간 병렬 조립구조를 갖는 환경에 맞도록 고안하였다. 즉, C2아키텍처 구조에서의 메시지 핸들링부분을 컴포넌트와 커넥터가 아닌 메시지 자체에서 처리하도록 하였으며, In/Out벡터를 메시지 명 벡터 하나만으로 운용함으로써 컴포넌트의 수정이 없이 조립될 수 있는 Plug-and- Play를 지원하였다. 따라서 메소드 호출방식의 서버컴포넌트의 경우 C2아키텍처를 적용하려면 컴포넌트와 커넥터부분을 수정하거나 조립하는 컴포넌트와 C2구조 사이의 접속코드 역할을 하는 Wrapper를 포함하는 일부 구조를 변형하여야 했으나, 본 제안기법에서는 메시지의 핸들링을 QPort부분에서 담당함으로써 컴포넌트의 수정이 필요 없이 조립 및 활용이 가능하였다. 또한, C2아키텍처와 같은 순차적인 계층 조립구조를 갖지 않고 메시지를 사이에 둔 병렬 적인 컴포넌트

의 조립구조를 가짐으로써 컴포넌트간 유연한 조립 및 활용이 이루어질 수 있었다.

그러나 본 연구는 조립된 컴포넌트간의 메시지 처리에 관한 우선순위 알고리즘이 고려되지 않았고, 아키텍처에 대한 ADL, IDN, ADN, ACN등의 구조적인 명세가 언급되지 않았다. 또한, 조립에 관한 효과적인 검증방안을 기반으로 보다 효율적인 컴포넌트간 핸들링부분이 필요하다.

참고 문헌

[1] F. Brosard, D. Bryan, W. Kozaczynski, E. S. Liongorari, J. Q. Ning, A. Olafsson, and J. W. Wetterstrand, "Toward Software Plug-and-Play", in Proc. of the 1997 Symposium on Software Reusability, 1997.

[2] P. C. Clements, "From Subroutines to Subsystem : Component-Based Software Development", Component Based Software Engineering, IEEE CSPress, 1996.

[3] 신동익 외6인, "C2 스타일의 아키텍처 기술을 지원하는 ADL 지원도구의 개발", 한국정보처리학회 논문지 Vol. 8-D, No 6. 2001.

[4] Taylor, R. N., Medvidovic, N., Anderson, K. M., Whitehead, E. J., Jr., Robbins, J. E., Nies, K, A., Oreizy, P. and Dubrow, D. L., "A Component-and Message-Based Architectural Style for GUI Software", IEEE Transactions on Software Engineering, Vol.22. No.6., June, 1996.

[5] M. Shaw, R. DeLine, D. V. Klein, T. L. Ross, D. M. Young, and G. Zelesnik, "Abstractions for Software Architecture and Tools to Support Them", IEEE Transactions on Software Engineering, Vol. 21, No. 4, April 1995, pp. 314-335

[6] D. Garlan, R. Allen, and J. Ockerbloom, "Exploiting Style in Architectural Design Environments", Proceedings of SIGSOFT 94 Symposium on the Foundations of Software Engineering, Dec. 1994

[7] N. Medvidovic, P. Oreizy, and R. N. Taylor, "Using Object-Oriented Typing to Support Architectural Design in the C2 Style", Proceedings of the 4th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE4), San Francisco, CA, Oct. 1996

[8] R. Allen and D. Garlan, "Formalizing Architectural Connection", Proceedings of 16th Int l Conference on Software Engineering, Sorrento, Italy, May 1994

[9] D. C. Luckham and J. Vera, "An Event-Based Architecture Definition Language", IEEE Transactions on Software Engineering, Vol. 21, No. 9, Sept. 1995, pp. 717-734

[10] B. Selic, G. Gullekson, and P. T. Ward, "Real-Time Object-Oriented Modeling", John Wiley & Sons, Inc., 1994

[11] The C2 Style, "http://www.ics.uci.edu/pub/arch/ c2.html", Information and Computer Science, University of California, Irvine.

[12] 최유희, 권오천, 신규상, "C2 스타일을 이용한 EJB 컴포넌트 합성방법", 한국정보처리학회논문지, Vol. 8-D, No 6. 2001.

저자 소개

정화영(Hwa-Young Jeong)

1991년 2월 : 목원대학교 수학교육과 졸업(이학사)

1994년 2월 : 경희대학교 전자계산학과 졸업(공학석사)

2001년 8월 : 경희대학교 전자계산공학과 박사수료

2000년 3월 ~ 현재 예원예술대학교 멀티미디어디자인학과 조교수

※ 관심분야 : CBSD, 웹 엔지니어링, 전자상거래 시스템 구축, 객체지향 소프트웨어 개발 기법, EJB