논•문 17-1-1

소프트웨어 아키텍처에서 쓰레드를 이용한 커넥터 확장
(An Extension of Connector using Thread on the Software Architecture)

정 화 영
예원대학교 정보,경영학부
jmichael@hanmir.com

요 약

소프트웨어 개발은 객체지향 기법에서 최적의 효율성을 찾고 있으며 이에 따라 디자인 패턴, 컴포넌트 기반 개발기법 등이 연구 및 도입되고 있다. 즉, 소프트웨어는 라이브러리나 모듈기반의 단위에서 하나의 독립기능단위 부품인 컴포넌트를 조립 및 합성하여 새로운 시스템을 구축하는 기법으로 변화되고 있다. 소프트웨어 아키텍처는 이에 관한 각 기능역할 및 구조를 체계적으로 표현하고 있다. 그러나, 기존의 아키텍처 기반 조립 및 합성은 연결구조를 갖는 커넥터에서 단순히 메시지 큐를 이용한 직접적인 연결처리 방식으로 처리되고 있어서 컴포넌트의 다중연결구조를 갖는 조립형태에서는 각 컴포넌트의 요청에 효율적으로 대처할 수 없었다.

따라서, 본 연구에서는 커넥터부분의 메시지 큐를 쓰레드의 형태로 구현하였으며 이를 통하여 다중의 컴포넌트에서 처리 메시지 요구시 자동으로 처리를 할당하도록 하였다. 또한, 여러 컴포넌트에서 비동기적으로 요청되는 메시지에 대하여 메시지 큐와 커넥터를 분리하여 처리함으로서 커넥터에서 메시지의 처리 할당을 효율적으로 할 수 있도록 하였다.

1. 서론

소프트웨어 개발에 있어서 객체지향기법은 가장 효율적인 개발방법론으로서 대변되어왔으며, 연구, 발전되어 왔다. 이러한 노력은 최근 디자인 패턴, CBSD(Component Based Software Development)등의 기법들을 제시하였다. 컴포넌트 기반 개발기법(CBD)은 소프트웨어 프로그래밍에서 하드웨어 개발 환경처럼 소프트웨어를 Plug- and-Play 방식으로 시스템을 구축하는 '합성을 통한 시스템 구축'으로의 전환을 목적으로 한 방법이다[1]. 즉, 독립적인 단위 부품모듈인 컴포넌트를 설계 및 구현하거나, 기존에 개발된 컴포넌트를 이용하여 커넥터를 통한 조립 및

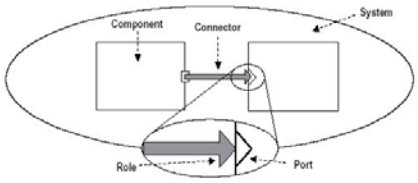
합성을 함으로서 전체 시스템을 완성하게된다[2]. 따라서, 시스템은 독립모듈들, 이름, 소프트웨어 컴포넌트들이 잘 정의된 소프트웨어 아키텍처의 형식을 갖는다[3]. 소프트웨어 아키텍처 기반기술은 UNIX의 Pipe-and-filter아키텍처에서부터 스타일 기반의 ACME[4], Aesop[5], C2[6], 시멘틱 모델 기반의 Wright[7], 도메인 명세기반의 실시간 객체지향구조의 ROOM[8] 등을 들 수 있다. 이러한 기술에서, 커넥터는 프로시저 호출, 이벤트 통지, 간단한 상호작용 형식을 포함하며, 상호작용에서의 호출하는 부분(Caller)과 받는 부분(Callee)에 의하여 역할(Roles)로 정의되는 인터페이스를 갖는다[9]. 그러나, Caller와 Callee로 연결되는 Roles구조는 단일 커넥터에 다중의 컴포넌트가 연결된 조립형태에서 비 동기적으로 처리요구가 발생시 커넥터가 요구처리 할당을 효과적으로 대처하기 어렵다.

따라서, 본 연구에서는 컴포넌트의 연결구조를 갖는 커넥터 부분을 메시지 큐와 커넥터를 분리하고, 이를 쓰레드로 처리함으로서 다중의 비 동기적인 요청 메시지 발생시에도 효과적으로 운용될 수 있도록 하였다. 이를 위하여, 컴포넌트는 EJB를 사용하였으며, 커넥터 부분은 JAVA를 이용하여 구현하였다. 본 연구는 다음과 같이 구성되어 있다. 2장에서는 소프트웨어 아키텍처에 의한 조립기법 중 본 제안기법을 적용한 ACME와 기존의 커넥터 구조 및 운용에 대한 연구들을 살펴보고자 한다. 3장에서는 본 연구에서 제안된 커넥터의 구조 및 설계를 나타내며, 이를 통하여 4장에서 예제 시스템에 적용함으로서 제안된 기법이 실제 직접 적용될 수 있음을 보인다. 마지막으로 5장에서는 결론 및 향후 연구 방향으로 끝을 맺고자 한다.

2. 관련연구

2.1 ACME 아키텍처

소프트웨어 아키텍처는 컴포넌트들간의 상호작용에 대한 그래프의 형태로 나타내며, 커넥터를 이용하여 컴포넌트들간의 상호작용을 표현한다[10]. 이러한 기법에서 ACME는 아키텍처 명세교환을 지원하며, 각 구성요소들 사이의 아키텍처를 잘 나타내는 것으로 알려져 있다[4]. ACME는 컴포넌트, 커넥터, 시스템, 포트, roles, 표현, repmap 등 7가지 요소를 갖으며, 이들 중 가장 기본적인 요소는 컴포넌트, 커넥터, 시스템을 들 수 있다. <그림 1>은 ACME의 기본 구성을 나타내며, 커넥터 부분인 role에 따라 컴포넌트의 요구 처리를 수행한다.



<그림 1> ACME 기본구성

2.2 기존의 커넥터 구조 및 운용

커넥터 연결구조는 등록된 컴포넌트들을 기반으로 role에 의하여 연결되며, 컴포넌트의 요구에 대한 처리는 연결구조와 속성에 따라 선요구 선처리 방식으로 이루어진다. 다음은 ACME명세 기반의 컴포넌트 등록과 커넥터의 연결과정을 나타낸다[11].

```
exists client, server, rpc |
    component(client) ^
    component(server) ^
    connector(rpc) ^
    attached(client.send-request, rpc.caller) ^
    attached(server.receive-request, rpc.callee) ^
    client != server ^
```

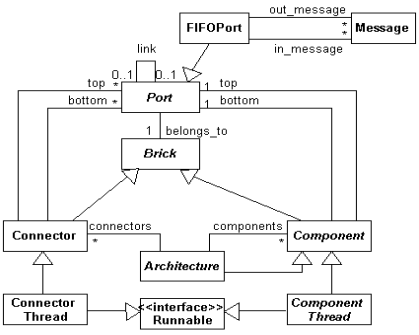
```
server != rpc ^
client != rpc ^
(for ∀y : component(y)⇒y=client | y=server) ^
(for ∀y : component(y)⇒y=rpc) ^
(for ∀p, q : attached(p,q)⇒
    (p=client.send-request ^ q=rpc.caller)
    | (p=server.receive-request ^ q=rpc.callee))
```

이에 따라, 호출부분(Caller)와 호출받는 부분(Callee)에서 각 컴포넌트들의 요소들을 연결함을 볼 수 있다. 이는, 아래와 같이 커넥터에 연결하는데 roles에서 caller와 callee를 연결하며, 프로토콜은 Wright, Aesop, ACME등의 형식을 지정할 수 있다.

```
connector rpc = {
    roles {caller, callee}
    property {protocol : (Wright|Aesop-style|...) }
```

이들의 인터페이스는 컴포넌트의 처리요청 생성시 각 연결구조(roles)에 따라 순차적으로 할당 처리되고있다[12].

직접적인 연결구조인 ACME에 반하여 메시지 기반의 C2는 비 동기적인 연결구조를 갖으며 메시지의 조립 및 처리를 위한 컴포넌트와 커넥터의 쓰레드를 수행한다. 또한, 메시지에 대한 Name Vector를 Port와 메시지에 각각 두었다. <그림 2>는 C2 프레임워크를 나타낸다.

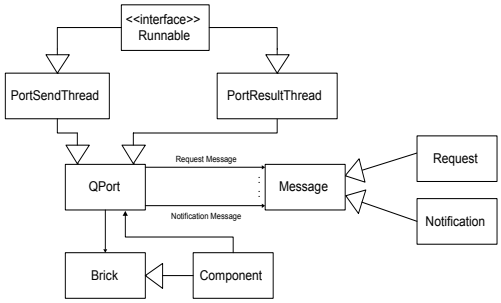


<그림 2> C2 프레임워크

3. 쓰레드를 이용한 커넥터 확장

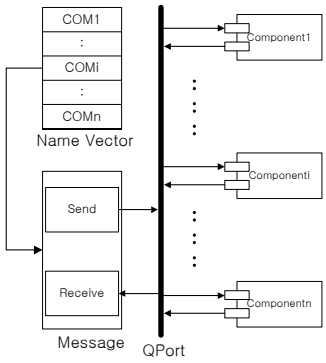
기존의 커네터구조는 다음과 같은 단점을 갖고 있다. 첫째, ACME의 경우 컴포넌트 사이에 단일 커넥터가 직접적으로 연결되어 있음으로 다중의 컴포넌트 조립 및 요구처리가 어렵다. 또한, 먼저 할당한 처리가 종료되기 전에는 다음에 요청된 메시지는 할당 대기 상태가 된다. 둘째, C2의 경우 단일 커넥터에 다중의 컴포넌트가 연결되어 운용되나 요구메시지 처리 및 할당을 위하여 컴포넌트와 커넥터 쓰레드가 각각 수행되면서 체크되어야 하며, 메시지와 Port 부분에서도 메시지 저장 및 핸들링을 위한 송,수신 Name Vector가 필요하다 또한, 복잡한 아키텍처 프래임워크의 적용이 요구된다.

따라서, 본 연구에서는 커넥터 부분에서 요구 메시지 처리 및 할당을 위한 프레임위크를 제안하였다. 또한, 컴포넌트의 요구처리 및 할당을 위한 쓰레드는 메시지를 핸들링하는 Port에만 한정하였다. 즉, 메시지의 처리를 커넥터 조립구조에서 구성하지 않고 메시지 자체에서 쓰레드를 통하여 자체 핸들링하도록 하였다. 컴포넌트의 조립구조에서는 효율적이고 쉬운 조립운용을 위하여 간단한 구조를 갖는 ACME 명세기반으로 설계 및 구현되었다. 이에 따라, <그림 3>과 같이 컴포넌트의 메시지는 커넥터 부분의 QPort를 통하여 메시지영역에 저장되며 PortSendThread와 PortResultThread부분에서 컴포넌트의 요구처리를 핸들링 하였다.



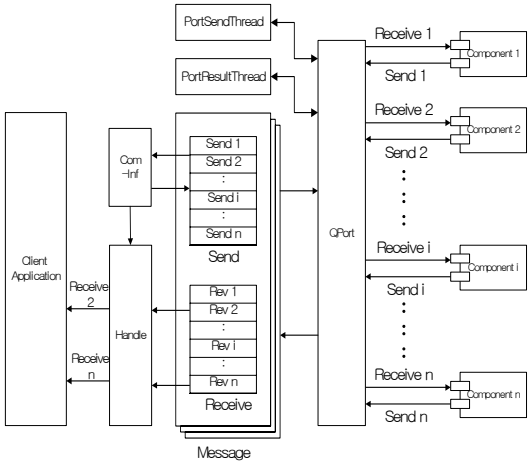
<그림 3> 제안된 커넥터 프레임워크

또한, 중간단계의 컴포넌트 결과확인이 필요할 경우 <그림 4>에서와 같이 핸들링되는 메시지의 Name Vector을 두어 이를 식별하고 그 결과를 확인할 수 있도록 하였으며, 다음의 메시지를 진행할 수 있도록 하였다.



<그림 4> 제안된 커넥터의 연결구조

<그림 5>에서는 본 연구에서 제안된 기법을 활용하여 각 컴포넌트를 조립하고 이를 운용하는 프레임워크를 나타낸다. 본 기법에서 Com-Inf는 조합될 컴포넌트들의 인터페이스 정보를 갖고있으며, 이를 통하여 각 컴포넌트들의 요구 메소드 들을 초기화한다.



<그림 5> 제안된 커넥터의 컴포넌트 연결 프레임워크

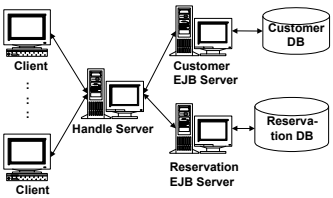
Handle에서는 Com-Inf에서의 컴포넌트 정보들을 기반으로 컴포넌트들의 호출 메소드 들을 메시지공간의 Send영역과 Name Vector에 적재한다. 적재된 전송 메시지 들은 PortSendThread를 통하여 확인되고 각 컴포넌트의 조립순서에 따라 메소드 메시지들을 호출하며, 확인된 결과 메시지들은 Receive에 적재한다. PortResultThread는 Receive에 적재된 메시지들을 확인하고 Handle을 통하여 요구된 컴포넌트의 결과를 사용자에게 나타낸다. 이에 따라, 각 컴포넌트들의 인터페이스정보 및 조립순서들을 Com-Inf에 등록시키고 이를 통하여 메시지 단계에서만 각 컴포넌트들의 통신을 핸들링 함으로써 요구 메시지의 관리 및 처리가 용이하였다. 또한, 조립순서에서 중간단계의 컴포넌트에 대한 결과를 확인하고자할 경우 쓰레드를 기반으로 커넥터를 운용함으로 이를 Name Vector와 Receive메시지를 통하여 확인이 가능하며, 계속 진행하여 최종단계의 컴포넌트에 대한 결과를 도출할 수 있다.

위 제안구조에 따라 다음과 같이 커넥터를 운용할 수 있다.

```
exists client, server, rpc |
    component(client) ^
    component(server) ^
        :
        :
connector rpc = {
    roles {caller, callee}
    property {thread.start()}
```

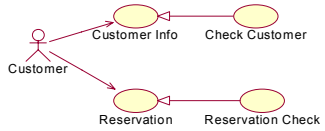
4. 예제 시스템 적용

본 연구에서 제안된 기법을 이용하여 간단한 예제를 적용하여 보았다. 적용된 시스템 환경은 각 서버에 대하여 윈도우 XP에서 JDK1.3.1과 J2SDKEE1.2.1을 이용하여 구현 및 테스트되었으며, 데이터베이스를 위하여 Informix사의 Cloudscape를 사용하였다. <그림 6>은 이에 관한 배치도를 나타낸다. 예제 구현 시스템은 고객정보를 검색하는 CustomerEJB와 해당고객의 좌석예약상황을 확인하는 ReservationEJB로 각각 구현되었으며 두 EJB서버 사이에 Handle서버를 두어 컴포넌트들을 조합 및 운용하였다.

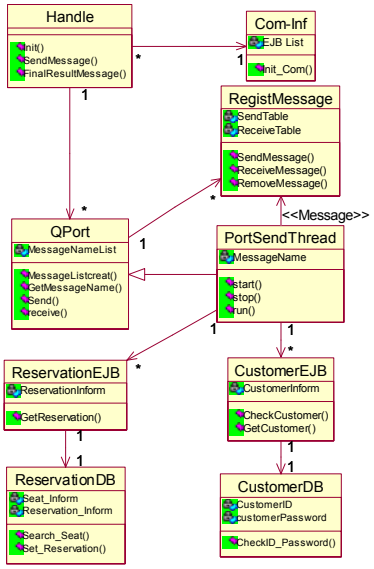


<그림 6> 좌석 예약 시스템 배치도

즉, 조합된 EJB정보는 Handle서버에 두었으며 이를 통하여 원격지의 Customer 서버 시스템과 Reservation 서버 시스템을 각각 두어 활용함으로써 3대의 다중서버 시스템환경으로 구현하였다.

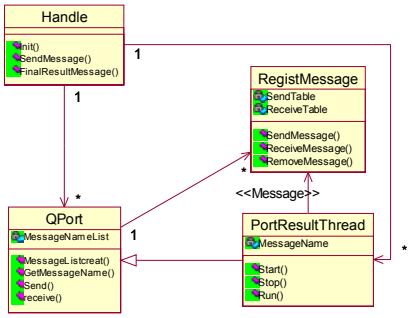


<그림 7> Usecase Diagram

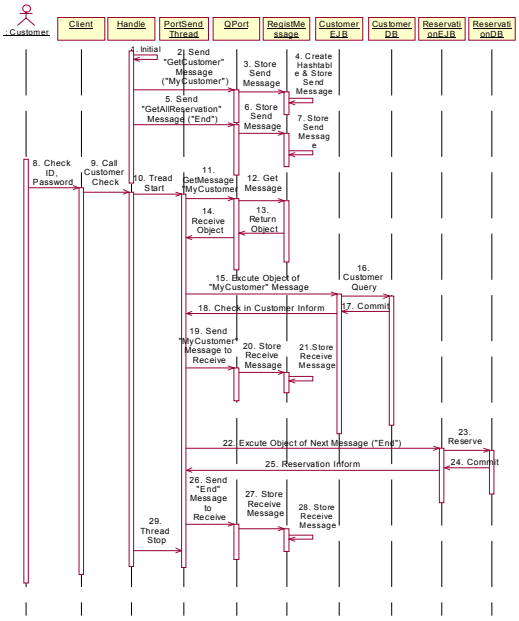


<그림 8> Send 메시지 처리 클래스 다이어그램

이에 따른 사용자 요구사항을 <그림 7>와 같이 나타낸다. 또한, 데이터베이스를 포함한 각 클래스들의 연관관계로서 <그림 8>에서는 조립된 컴포넌트들의 Send 메시지처리 및 Receive 메시지적재까지를 나타낸다. <그림 9>는 적재된 Receive 메시지 의 처리를 나타낸다.

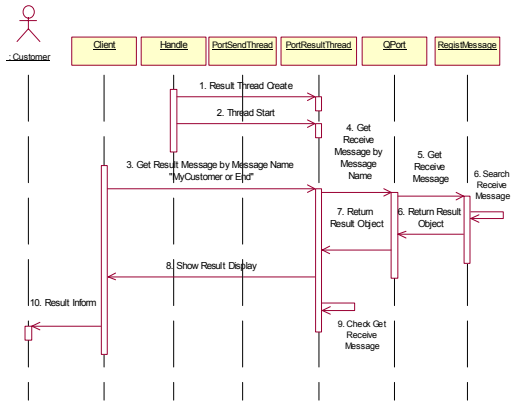


<그림 9> Receive 메시지 처리 클래스 다이어그램



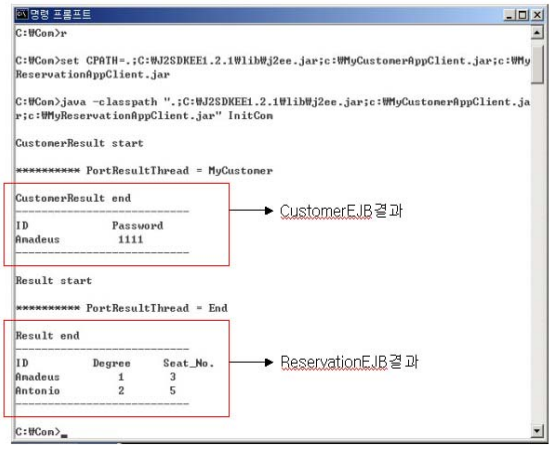
<그림 10> Request 메시지 처리 시퀀스 다이어그램

이에 따라, 각 단계의 처리순서는 Send 메시지처리 부분은 <그림 10>에서 Receive 메시지 처리부분은 <그림 11>과 같이 각각 나타낸다.



<그림 11> Receive 메시지 처리 시퀀스 다이어그램

본 예제 시스템의 실행을 위하여 J2EE 서버를 서버측에서 가동한 후, 클라이언트측의 최종결과인 좌석 예약 상황의 화면은 <그림 12>와 같다.



<그림 12> 좌석예약 시스템 결과

5. 결론

소프트웨어 아키텍처에서 컴포넌트 조립 및 합성은 매우 중요한 요소임에도 불구하고

많은 연구에서 커넥터 부분은 단순히 컴포넌트를 조립 연결하고, 이를 순차적으로 핸들링하는 단계에 머물렀다. 따라서, 본 연구에서는 컴포넌트 조립 및 운용에서 커넥터부분에 대한 기존의 단순한 단일구조에서 컴포넌트 요구 메시지 처리부분을 확장함으로서 비동기적이고 다중적인 조립형태에서도 효율적으로 운용될 수 있는 기법을 제시하였다. 기존의 방법에서는 요구 메시지 처리를 위하여 쓰레드가 컴포넌트와 커넥터 모두 필요하였으며, 조립운용에 적용하기 위하여 복잡한 프레임워크에 따른 컴포넌트 수정 및 커넥터에서의 많은 부가정보가 필요하였다. 이에 관하여, 본 연구에서는 요구 메시지 처리를 위한 쓰레드는 메시지부분에서만 한정하였다. 그 결과, 본 제안기법의 적용을 위하여 컴포넌트 부분은 조립구조에 따른 수정이 필요 없었으며 메시지 처리를 위한 로직을 단순화 할 수 있었다. 또한, 간단한 조립구조를 갖는 ACME 명세기반으로 설계함으로서 쉽고 효율적으로 운용될 수 있었다.

그러나, 본 연구는 대량의 컴포넌트 요구가 발생시 상대적으로 쓰레드에서 나타날 수 있는 처리속도 저하에 대하여 고려되지 않았으며, 메시지 큐에서 쓰레드를 통한 처리할당에 대한 부하를 충분히 고려하지 않았다. 따라서, 이러한 처리가 이루어진다면 보다 완전한 커넥터의 활용이 가능할 것으로 사료된다.

참고문헌

[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] 이은서, 이경환, "컴포넌트의 재사용과 확장성을 위한 개발방법", 한국정보처리학회논문지 D 제9-D권 제5호, 2002. 10.

[3] Ioannis Georgiadis, "Self-Organising Distributed Component Software Architecture", University of London, Ph.D Thesis, 2002. 1.

[4] David Garlan, Robert T. Monroe, David Wile, "Acme: Architectural Description of Component-Based Systems", Cambridge University Press, 2000.

[5] 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

[6] 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

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

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

[9] David Garlan, "Software Architecture", Encyclopedia of Software Engineering, 2001.

[10] Bradley Schmerl, David Garlan, "Exploiting Architectural Design Knowledge to Support Self-repairing Systems", Fourteenth International Conference on Software Engineering

and Knowledge Engineering, July 15-19, 2002.
[11] David Garlan., et, al., "Acme: An Architecture Description Interchange Language", Proceedings of CASCON '97, November 1997.
[12] David Garlan, Bradley Schmerl, "Model-based Adaptation for Self-Healing Systems", ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02), November 18-19, 2002.

정화영

1991년~1994년 경희대학교 전자계산공학과(공학석사)
1998년~2000년 한남대학교 컴퓨터공학과(박사과정)
2000년~2001년 경희대학교 전자계산공학과(박사수료)
2000년~현재 예원대학교 정보경영학부 전임강사
관심분야 : 소프트웨어공학, OOP/S, S/W 재사용, 컴포넌트기반 개발 방법론.