컴포넌트 처리시간을 고려한 우선순위기반의 커넥터 상호작용

A Priority Process Based Connector's Interaction considering Component Processing Time

정 화 영*
Hwa-Young Jeong

요 약

컴포넌트 기반 개발방법에서 컴포넌트간의 상호작용을 수행하는 커넥터의 역할은 매우 중요하다. ADL을 기반으로 한 대부분의 커넥터 운용방식은 컴포넌트의 요구순서에 따른 FIFO 방식을 택하고 있다. 그러나 이러한 방식은 다양한 특성을 가지는 많은 컴포넌트의 요구발생시 효율적인 운용이 어렵다. 본 연구에서는 컴포넌트의 처리시간을 고려한 우선순위 커넥터를 설계 및 구현하였다. 또한, 정형화된 명세를 위하여 Wright 아키텍처를 이용하였다. 제안된 커넥터의 적용결과는 기존의 FIFO 방식과 비교하여 전체 처리시간에서 388ms 더 소요되었다. 그러나 처리시간이 짧은 컴포넌트들부터 우선적으로 처리할 수 있었다. 또한, 커넥터에서 컴포넌트 처리대기시간은 기존의 FIFO방식이 23323.1ms이며, 제안기법은 12731.27ms으로 나타남으로서 컴포넌트 처리 대기시간을 감소시킬 수 있었다.

Abstract

Connector's role between components is very important in the CBD(Component Based Development). The most connector has process ADL based method was choosing FIFO method by component request. But, in case many component's with various characteristics request, it is difficult that this method operate efficiently. In this research, I did design and implement priority connector considering component's processing time. Also, I used Wright architecture for formal specification. Application result of proposed connector was spend more 388ms compares with existent FIFO method in total processing time. But, this method could handle preferentially from components that have short processing time. Also, in case of component's waiting time in connector, existent FIFO method is 23323.1ms and proposal method is 12731.27ms. So, proposal method could reduce waiting time for component process

☞ Keyword : Component composition, Connector interaction, Wright, ADL

1. 서 론

최근에는 소프트웨어의 독립적 기능모듈단위인 컴포넌트들을 이용하여 합성되는 소프트웨어 시스템 개발 방법에 의한 컴포넌트 기반 패러다임으로 구현되고 있다[1]. 컴포넌트 기반 패러다임은 기능성 향상과 재사용성에 의해 소프트웨어 개발에 있어 보다 좋은 생산성과 품질을 제공한다[2]. 대부분의 소프트웨어 컴포넌트 연구들은 컴포넌트의 기능적인 형태에 초점을 맞추고 있다[3]. 즉, 정보는 컴포넌트들 사이의 상호작용을 통하여 분류되며, 이러한 상호작용은 컴포넌트들의 공통된 인터페이스를 통한 통신규약을 가진다[4]. 이러한 인터페이스를 이용하여 컴포넌트는 상호작용 포트들과 특별한 기능들에 의해 나타내어지며, 컴포넌트 기능은 요구와 응답의 집합에 의하여 표현된다[5]. ADL(Architecture Description Language)은 서로 연결된 컴포넌트들의 상호운용에서 효율적으로 시스템 구조를 명세하기

* 정 회 원 : 경희대학교 교양학부 전임강사
hyjeong@khu.ac.kr(제 1저자)
[2004/03/26 투고 - 2004/04/09 심사 - 2005/03/09 심사완료]

위한 전문적인 언어이다[6]. ADL에서 아키텍처 기반기술은 UNIX의 Pipe-and-filter아키텍처와 일반적인 어플리케이션에서 오랜 기간에 사용된 Blackboard 아키텍처기술에서부터 Style 기반의 Unicon, Aesop, C2, 시멘틱 모델 기반의 Wright, Rapide, Domain Spec 기반의 실시간 객체지향 구조의 ROOM등을 들 수 있다. 그러나 ADL을 기반으로 한 전체적인 커넥터 운용방식은 FIFO 방식을 따른다. Wright는 컴포넌트와 커넥터의 연결에서 요구된 컴포넌트의 순서에 따라 이를 처리하고 있다. Weaves는 연결된 호출 컴포넌트들의 요구 정보를 커넥터에서 큐를 이용하여 저장하고 응답 컴포넌트에게 큐에 저장된 순서에 따라 먼저 요구된 컴포넌트를 먼저 처리하게 된다. C2 또한 커넥터에서 쓰레드 방식으로 컴포넌트의 요구를 처리하고 있으나 FIFO 포트를 통하여 먼저 요구된 컴포넌트의 요구 메시지를 먼저 처리하게 된다. 이러한 선 요구 선 처리방식의 커넥터 운용은 다중 컴포넌트들의 요구에 효율적으로 대처하기 어려우며, 처리 응답시간이 긴 컴포넌트가 먼저 요청된 경우에는 처리를 기다리는 요구 컴포넌트들의 대기열이 길어지게 된다. 따라서 연결 컴포넌트들의 특성을 고려하여 이를 효율적으로 핸들링 할 수 있는 커넥터 운용기법이 필요하다.

본 논문에서는 연결 컴포넌트들의 요구에 대하여 컴포넌트들의 처리시간을 고려한 우선순위에 따라 핸들링 할 수 있는 커넥터를 설계 및 구현하였다. 즉, 다중의 비동기적인 컴포넌트 요구처리에 대하여 컴포넌트의 처리시간에 따라 처리순서를 결정하도록 하였다. 제안된 커넥터의 적용결과, 컴포넌트들의 요구 처리에 대하여 커넥터에서의 우선순위 처리는 단위 시간당 컴포넌트 처리 수의 증가와 각 컴포넌트들의 처리할당을 위한 평균 대기시간 감소의 효과를 가져왔다. 즉, 처리시간이 긴 컴포넌트가 먼저 요청이 된 경우에도 각 컴포넌트들이 커넥터의 처리 대기열의 정체를 감소시켰다.

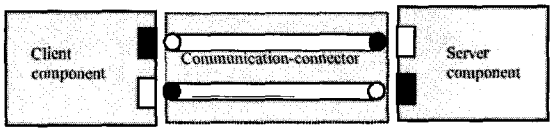
2. 관련연구

2.1 CBD

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[7]. 따라서, 시스템은 독립모듈들, 이름, 소프트웨어 컴포넌트들이 잘 정의된 소프트웨어 아키텍처의 형식을 갖으며, 컴포넌트 합성은 인터페이스 또는 커넥터가 컴포넌트사이의 교환을 나타냄으로서[8] 이루어진다. 컴포넌트 기반 개발은 컴포넌트, 컴포넌트 인스턴스, 인터페이스 형식, 인터페이스, 요구 인터페이스, 응답 인터페이스, 인터페이스 인스턴스, 커넥터 형식, 커넥터, 커넥터 인스턴스로 구성된다.

2.2 컴포넌트 기반개발에서의 커넥터

컴포넌트 기반 개발에서 중요한 문제는 합성이며, 합성의 중요한 단계는 활동에 기반한 컴포넌트들의 실제적인 상호작용이 된다[9]. 커넥터는 컴포넌트들 사이의 상호작용을 나타내는 아키텍처 요소이며 그들 상호작용에 주어지는 규칙이다. 커넥터는 프로시저 호출과 같은 단순한 상호작용 메커니즘에서부터 client-server 프로토콜, 데이터베이스 엑세스 프로토콜등과 같은 복잡한 상호작용으로 나타낼 수 있다[10]. 대부분의 커넥터 운용방식인 RPC와 같은 동기식 커넥터를 통한 client-server스타일에서 컴포넌트 통신은 요구와 응답 이벤트를 이용한다. client 컴포넌트는 요구를 보내고 server 컴포넌트로부터 응답을 받는 하나의 마스터 포트를 가지며, server는 각 client를 위한 다중의 하부 포트를 가진다[11]. 그림 1은 client-server의 간단한 연결 구조를 나타낸다. 이는 client 컴포넌트와 server 컴포넌트 사이에 상호작용을 담당하는 커넥터가 위치한다. 컴포넌트



〈그림 1〉 Client-Server style

들 사이의 상호작용은 client 컴포넌트에서 서비스 처리의 요구를 커넥터에게 보내고, 커넥터는 연결된 server 컴포넌트에게 중개하게 된다. 또한, server 컴포넌트에서 처리가 완료된 후 client 컴포넌트의 요구에 대한 응답을 커넥터에게 보내면 커넥터는 연결된 client 컴포넌트에게 응답 메시지를 전달한다[12].

2.3 Wright 아키텍처

Wright[13]는 Carnegie Mellon대학에서 개발되었으며 Robert Allen의 연구에서 정형화의 기반을 다졌다.

이는 시스템에서 사용되는 상호작용의 프로토콜을 기술하도록 설계된 ADL(Architecture Description Language)이며, 아키텍처 명세의 정적인 분석을 지원한다. 또한 소프트웨어 아키텍처와 관련하여 행위로부터 구조를 분리한다. 이러한 정형언어는 컴포넌트의 포트와 roles을 포함하는 커넥터, roles와 포트를 연결하는 glue를 포함한다. 컴포넌트는 외부 통신을 위한 포트와 내부 처리 명세인 Computation을 가지며, 커넥터는 연결을 나타내는 role과 합성 및 처리 명령순서를 나타내는 glue를 가진다. 또한 Constraints는 합성 시스템의 정형적인 명세를 포함한다. 시스템 아키텍처에서 Wright 구조는 컴포넌트와 커넥터 집합의 배합처럼 표현된다. Wright명세에 의한 Client-server 시스템을 기술하면 그림 2와 같다[14].

```
System SimpleExample
component Server =
        port provide [provide protocol]
        spec [Server specification]
component Client =
        port request [request protocol]
        spec [Client specification]
connector C-S-connector =
        role client [client protocol]
        role server [server protocol]
        glue [glue protocol]
Instances
        s: Server
        c: Client
        cs: C-S-Connector
Attachments
        s.provide as cs.server;
        c.request as cs.client;
end SimpleExample.
```

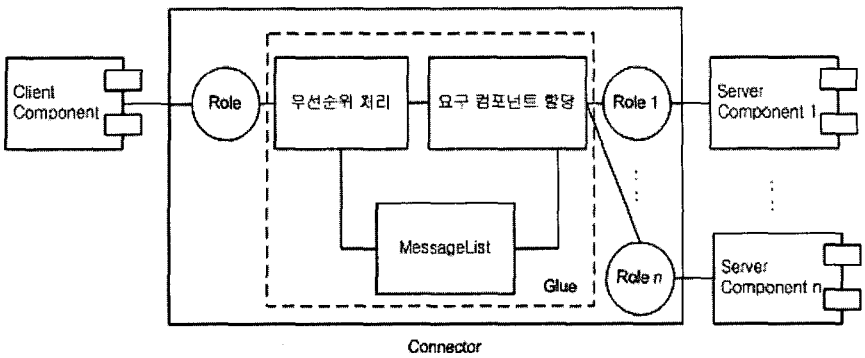
〈그림 2〉 간단한 client-server 시스템

3. 우선순위 처리를 고려한 커넥터

3.1 우선순위 처리 커넥터의 구성

제안된 커넥터에서 사용된 우선순위처리를 위한 컴포넌트 처리시간 값은 컴포넌트의 기능테스트를 통하여 얻으며, 컴포넌트와 커넥터에 연결할 때 커넥터의 메시지리스트에 저장되어 이용된다. 그림 3은 제안된 우선순위 커넥터를 중심으로 한 컴포넌트들의 연결 구조를 나타낸다. 즉, 서비스를 요청하는 Client 컴포넌트와 서비스 처리를 하는 다중 Server 컴포넌트 환경을 연결하는 커넥터는 요구된 서버 컴포넌트를 우선순위에 따라 할당하고 처리결과를 Client 컴포넌트에 보내게 된다.

그러나 적용된 컴포넌트 처리시간 값에서 특정 변수에 따라 컴포넌트의 처리시간 수치가 달라질 수 있는 경우는 고려하지 않았으며, 임의의 샘플변수에 따라 일반적인 컴포넌트 기능테스트에 의한 수치만을 반영하였다. 컴포넌트의 우선순위 처리는 다음과 같이 수행되었다. 즉, 컴포넌트들의 처리시간을 $R_1 \sim R_n$이라 할 때, 최소값을 가지는 컴포넌트가 우선순위가 가장 높으며 최대값을 가지는 컴포넌트는 우선순위가 가장 낮으므로 $R_1 \sim R_n$의 값을 정렬한 순서가 우선순위 처리 순서가 된다.



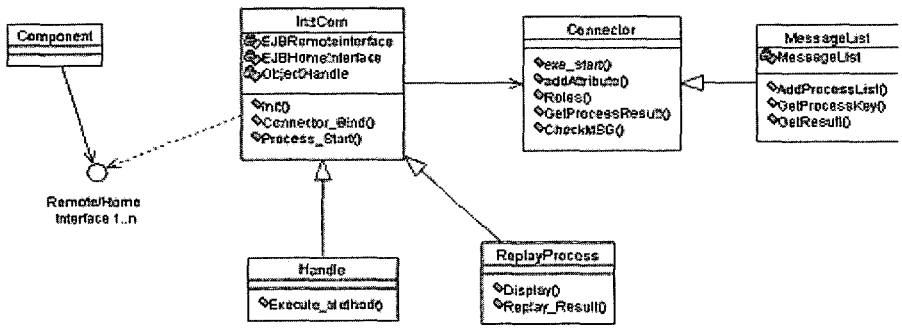
〈그림 3〉 우선순위 커넥터에 의한 컴포넌트 연결

3.2 우선순위를 고려한 커넥터 설계 및 구현

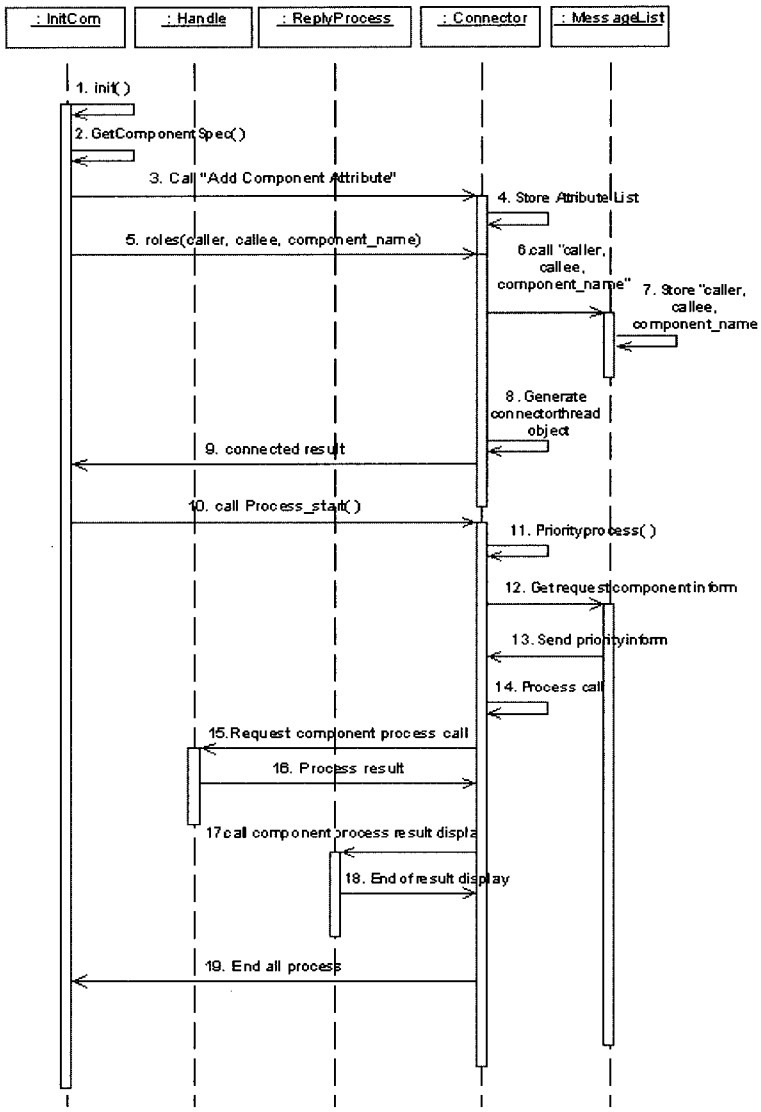
본 제안 커넥터의 설계를 위하여 UML을 이용하였으며, Window XP에서 JDK1.3.1을 이용한 Java 언어로 구현되었다. 그림 4는 클래스 다이어그램 중 Client와 Server 컴포넌트들 사이의 관계를 나타낸다.

즉, 컴포넌트에서 제공되는 Remote와 Home 인터페이스 정보를 기반으로, Client 컴포넌트 부분에 해당되는 InitCom의 init()에서는 컴포넌트 호출 및 운용을 위한 컴포넌트 등록을 수행한다. Connector_Bind()는 컴포넌트들의 처리시간 값을 Connector의 addAttribute()로 보냄으로써 커넥터에서 우선순위를 산출할 수 있도록 그 정보를 메시지리스트에 저장한다. 또한, 컴포넌트들과 커넥터의 연결 관계에 따라 연결정보를 Connecotr의 Roles()로 보냄으로써 커넥터에서 컴포넌트 연결에 따른 처리 할당을 위한 연결정보를 저장한다. Process_Start()는 커넥터에 존재하는 우선순위 산출과 컴포넌트 처리할당에 따른 컴포넌트 처리 서비스를 시작한다. 또한, ReplyProcess와 Handle은 각각 실행결과에 따른 화면표시정보와 컴포넌트 리모트 인터페이스에 의한 실제 호출 메소드를 가진다. 각 컴포넌트들과 커넥터 사이의 메시지 흐름은 그림 5와 같다.

즉, InitCom에서는 init()를 통하여 연결할 컴포넌트들의 Home 및 Remote 인터페이스 정보를 등록한다. 또한, GetComponentSpec()에서는 각 컴포넌트들의 처리시간 값을 얻으며, 커넥터에서 우선순위 산출에 이용할 수 있도록 컴포넌트 처리시간 값을 커넥터에 저장한다. 또한, 각 컴포넌트와 커넥터 사이의 연결구조에 따른 정보는 Roles()를 통하여 커넥터에 등록되며, 커넥터는 각 연결 및 호출정보를 MessageList에 저장한다.



〈그림 4〉 Client와 Server컴포넌트들 사이의 클래스 다이어그램



〈그림 5〉 Client와 Server 컴포넌트 사이의 커넥터 시퀀스 다이어그램

InitCom에서 컴포넌트 서비스의 시작인 Process _Start()를 호출하게 되면, 커넥터에서는 우선순위를 산출하는 PriorityProcess를 실행시킨다. 우선순위 산출을 위하여 커넥터에서는 각 연결정보를 가지는 MessageList에서 연결정보와 처리시간 값을 얻으며, 이를 통하여 우선순위를 산출한다. 산출된 우선순위에 따라 커넥터에서는 각 컴포넌트들의 실제 호출정보를 가지는 Handle에 대하여 컴포넌트를 호출한다. 처리가 완료됨에 따라 그 결과는 커넥터를 통하여 MessageList에 저장되며, Replyprocess에 의하여 처리결과를 나타낸다.

3.3 제안 커넥터의 Wright 명세

Wright는 컴포넌트 합성명세에서 합성구조와 행위를 가장 잘 나타내는 ADL로 알려져 있다.

```
Style Client-server
   Component ClientComponent
       Port p = request□reply□p □ §
       Computation = internalCompute□ p.request□p.reply□Computation □ §
   Component ServerComponent1
       Port p1 = request□reply□p ∏ §
       Computation = internalCompute□ p.request□p.reply□Computation ∏ §
           :                    :
           :                    :
   Component ServerComponentn
       Port pn = request□reply□p ∏ §
       Computation = internalCompute□ p.request□p.reply□Computation ∏ §
Connector Link
       Role c=p.request□p.reply□c □ §
       Role s1=p1.request□p1.reply□s1 ∏ §
                 :              :
                 :              :
       Role sn=pn.request□pn.reply□s ∏ §
       Glue=c.request□sn.request□Glue ∏{GetComponentEfficient □ CalcWeight
        □ Sorting}□sn.reply□c.reply□Glue ∏ §
 Constraints
       ∃!s∈Component, ∃!e∈EfficientValue,
       ∀c∈Component :
 TypeServer(s, e)∧TypeClient(c)⇒
       connected(c,s,e)
EndStyle
```

〈그림 6〉 우선순위를 고려한 Wright기반의 합성명세

따라서 본 연구에서는 제안된 커넥터의 정형화된 명세를 위하여 Wright 명세를 이용하였으며, 연결된 컴포넌트 처리시간을 고려한 우선순위 에 따라 Wright명세기반에 의해 그림 6과 같이 커넥터를 설계 및 구현하였다. 이때, ∏은 프로세스 내부의 처리 선택을 의미하며, □은 외부의 처리 선택을 의미한다. 따라서 커넥터 Link는 Client 컴포넌트의 요구 및 응답을 나타내는 Role c를 가지며 Server 컴포넌트로의 호출을 요구한다. 또한, Role s1...sn은 server 컴포넌트로서 Client 컴포넌트로부터 요구되는 정보를 받아 이를 처리한 후 응답을 보낸다. Glue는 컴포넌트들의 연결에서 행위를 묘사하며, {GetComponentEfficient □ CalcWeight □ Sorting}를 통하여 우선순위 처리를 반영한다.

4. 적용 및 결과비교

제안 커넥터의 적용을 위하여 각 처리시간을 다르게 가지는 테스트 컴포넌트 20개를 구현 하였으며 그림 3에 따라 합성하였다. EJB에서 무상태 세션빈으로 구현된 테스트 컴포넌트는 처리시간을 다르게 가지기 위하여 중첩 반복문을 사용함으로서 응답시간을 지연시켰다. 구현된 테스트 컴포넌트들은 웹로직의 성능 테스트를 통하여 측정된 결과 표 2와 같이 나타났다.

본 연구의 적용결과로는 Wright에 의하여 효율성 특성을 고려하지 않는 기존의 FIFO 방식과 우선순위를 고려한 제안된 커넥터 처리방식을 각각 적용하여 그 결과를 비교하였다. 이러한 결과로서 FIFO방식의 커넥터에서는 컴포넌트 초기화 및 등

〈표 1〉 테스트 EJB 컴포넌트의 처리시간

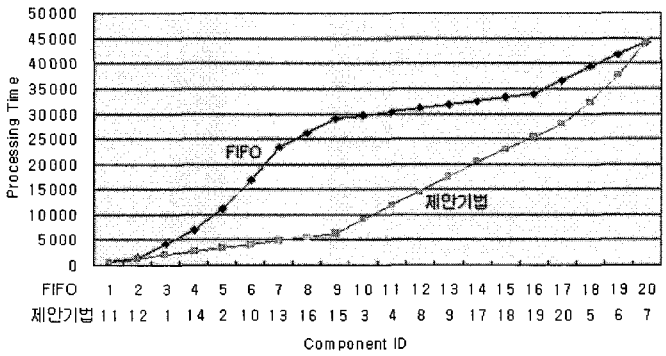
	테스트 컴포넌트									
	1	2	3	4	5	6	7	8	9	10
컴포넌트 처리시간 (ms)	701	704	2754	2804	4256	5568	6590	2864	2884	704

	테스트 컴포넌트									
	11	12	13	14	15	16	17	18	19	20
컴포넌트 처리시간 (ms)	629	688	704	702	714	705	2886	2889	2890	2893

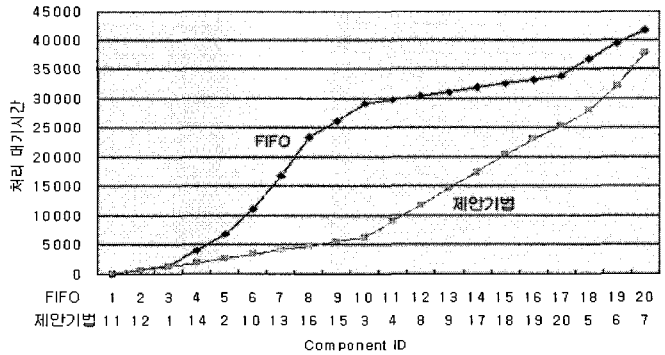
록을 포함한 20개의 컴포넌트 수행에 대한 총 처리시간이 45623ms가 소요된다. 반면, 본 제안 커넥터에 의해 수행된 결과는 총 수행시간 46011 ms로서 추가적인 우선순위 처리로 인하여 기존의 FIFO 방식에 비하여 388ms가 더 소요된다. 그러나 단위시간별 컴포넌트의 처리결과는 우선순위 커넥터 방식이 더 효율적으로 나타났다. 즉, 그림 7과 같이 기존의 FIFO 방식으로 합성되어 운용된 결과는 각 컴포넌트의 특성과는 상관없이 요청된 순서대로 처리됨을 알 수 있다. 그러나 제안된 우선순위 커넥터는 컴포넌트의 처리시간 값에 따라 처리순서가 다르게 적용됨을 알 수 있다. 이는, 처리시간이 짧은 컴포넌트들을 먼저 처리함으로서 초기의 단위시간당 많은 컴포넌트들을 처리할 수 있었다.

또한, 컴포넌트들의 처리대기시간은 그림 8과 같이 나타났다. 즉, 커넥터에서 컴포넌트 처리에 대한 평균대기시간은 기존의 FIFO방식이 23323.1 ms이며, 제안기법은 12731.27ms로서 제안기법이 컴포넌트들의 처리를 위한 대기시간에서 더 짧은 것으로 나타났다.

결과적으로 본 제안기법이 우선순위 알고리즘 처리에 의한 요구처리 소요시간이 더 필요하지만



〈그림 7〉 우선순위 커넥터의 적용결과 비교



〈그림 8〉 컴포넌트 처리 대기시간

전체 시스템의 운용측면에서는 초기에 보다 많은 컴포넌트의 요구처리를 수행할 수 있었으며, 커넥터에서 처리를 기다리는 각 컴포넌트들도 처리 대기 시간이 짧게 나타남으로서 커넥터에 연결된 컴포넌트들이 많아지게 될 경우 효율적인 컴포넌트 처리 운용이 가능하다.

5. 결 론

블록 맞추기와 같이 소프트웨어 모듈인 컴포넌트를 합성하여 새로운 시스템으로 구축하려는 컴포넌트 기반 개발방법에서는 연결된 컴포넌트들의 상호작용을 담당하는 커넥터의 역할이 매우 중요하다. 본 연구에서 제안된 우선순위 커넥터는 연결 컴포넌트들의 처리시간 값을 고려하여 처리하도록 하였으며 정형화된 명세를 위하여 Wright기반의 아키텍처를 이용함으로서 커넥터의 연결구조와 행위를 분리하여 표현하였다. 제안된 커넥터의 적용을 위하여 처리시간을 다르게 가지는 테스트 컴포넌트 20개를 구현하였으며, 이를 기존의 FIFO방식과 우선순위방식에 따라 적용 후 그 결과를 비교하였다. 이러한 결과로서 기존의 방식에서 추가된 우선순위처리에 의하여 제안된 커넥터 방식이 전체적인 처리시간에서 388ms 더 소요되었다. 그러나 제안된 우선순위 커넥터는 처리시간이 짧은 컴포넌트들부터 우선적으로 처리함에 따라 시스템 운영초기에 많은 컴포넌트들이 처리되어 단위 시간당 처리되는 컴포넌트의 개수가 많음을 알 수 있었다. 또한, 커넥터에서 컴포넌트 처리대기시간은 기존의 FIFO방식이 23323.1ms이며, 제안기법은 12731.27ms로 나타남으로서 컴포넌트들의 처리를 위한 대기시간에서 더 짧은 것으로 나타났다.

그러나 본 제안기법은 우선순위 알고리즘의 수행시간에 따른 추가적인 시간이 소요됨으로서 전체적인 처리시간의 지연을 가져왔다. 따라서 향후 연구과제로는 이러한 처리시간의 지연을 감소시킬 수 있는 처리방법이 요구된다.

참 고 문 헌

[1] Dennis Heimbigner and Alexander L. Wolf, "Intrusion Management Using Configurable Architecture Models", Technical Report CU-CS-929-02, University of Colorado Department of Computer Science, April 2002.

[2] Marlon E. R. Vieira Marcio S. Dias Debra J. Richardson, "Describing Dependencies in Component Access Points", Proceedings of the 4th Workshop on Component Based Software Engineering, 23rd International Conference on Software Engineering, May 2001.

[3] Miguel Goulão, Fernando Brito e Abreu, "Towards a Component Quality Model". In Proc. of Work in Progress Session at the 28th EUROMICRO Conference, September, 2002.

[4] Bent Bruun Kristensen, "Component Composition and Interaction", In Proc. of International Conference on Technology of Object-Oriented Languages and Systems, 1996.

[5] Ioana Sora, Pierre Verbaeten, Yolande Berbers, "Using Component Composition for Self-customizable Systems", Component-based Software Engineering Workshop: Composing Systems from Components ECBS 2002, April 2002.

[6] Dušan Bálek, "Connectors in Software Architectures", Ph.D Thesis, Faculty of Mathematics and Physics Department of Software Engineering, Charles University, Mar 2002.

[7] F. Brosard, D. Bryan, W. Kozaczynski, E. S. Liongorari, J. Q. Ning, A. Ola-

fsson, and J. W. Wetterstrand, "Toward Software Plug-and-Play", in Proc. of the 1997 Symposium on Software Reusability, 1997.

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

[9] Robert J. Allen, "A Formal Approach to Software Architecture", Ph.D Thesis, CMU-CS-97-144, School of Computer Science Carnegie Mellon University, May 1997.

[10] Marija Rakic, Nenad Medvidovic, "Increasing the Confidence in Off-the-Shelf Components: A Software Connector-Based Approach", Technical Report USC-CSE-2000-518, University of California, Irvine, November 2000.

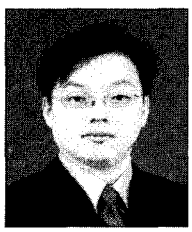
[11] Nenad Medvidovic, Marija Mikic-Rakic, Nikunj Mehta, Sam Malek, "Software Architectural Support for Handheld Computing", Computer, IEEE Computer Society, September 2003.

[12] A. Ramdane-Cherif, L. Hazem, N. Levy, "Knowledge Repository Concerning Architectural Styles For Building Component-Based Systems". CoLogNET'02: Colognet Joint Workshop on Component-Based Software Development and Implementation Technology for Computational Logic Systems. September, 2002.

[13] Allen R. "A Formal Approach to Software Architecture" Ph.D. Thesis, CMU-CS-97-144, Carnegie Mellon University, 1997.

[14] ROBERT ALLEN and DAVID GARLAN, "A Formal Basis for Architectural Connection", ACM Transactions on Software Engineering and Methodology, Vol. 6, No. 3, July 1997, Pages 213-249.

◐ 저 자 소 개 ◑



정 화 영 (Hwa-Young Jeong)

1991년 목원대학교 수학교육학과 졸업(학사)
1994년 경희대학교 전자계산공학과 졸업(석사)
2004년 경희대학교 대학원 전자계산공학과 졸업(박사)
2000년~2003년 예원예술대학교 전자상거래학과 전임강사
2003년~2005년 예원예술대학교 멀티미디어디자인학과 조교수
2005년~현재 경희대학교 교양학부 전임강사
1994년~1998년 아주시스템(주) 부설연구소 S/W개발팀 전임연구원
1998년~1999년 CNA Research(주) S/W 개발팀 전임연구원
2003년~2004년 (사)한국콘텐츠학회 학회지 편집위원
2003년~현재 (사)한국정보과학회 프로그래밍언어연구회 운영위원
2003년~현재 (사)한국디지털컨텐츠학회 학술위원
2004년~현재 (사)한국인터넷정보학회 논문지 편집위원
관심분야 : 소프트웨어 공학, 컴포넌트 기반 개발기법, 컴포넌트 조립/합성, 웹 엔지니어링.
E-mail : hyjeong@khu.ac.kr