컴포넌트기반 미들웨어에서 효율적인 컴포넌트 재구성

권정호° 김재훈
아주대학교 정보통신전문대학원
jungho0405@dmc.ajou.ac.kr° jaikim@ajou.ac.kr

Efficient Component Reconfiguration in Component-Based Middleware

Jung-ho Kwon° , Jai-Hoon Kim
Graduate School of Information and Communication, Ajou University

요 약

유비쿼터스 컴퓨팅에서의 미들웨어는 주위 환경을 감지하는 센서나 인터넷정보를 이용해서 얻은 모니터링 정보를 이용하여, 상황 변화에 맞게 서비스를 제공할 수 있도록 실행 중에 시스템을 재구성(Reconfiguration)하여 재시작하지 않고 새로운 환경에 적응할 수 있어야 한다. 이를 위해서 분산시스템에서의 동적 재구성(Dynamic Reconfiguration) 기술이 연구되어 왔다. 동적 재구성이 가능한 컴포넌트 기반 미들웨어의 개발은 분산시스템에서 점차 중요성이 높아지고 있다. 미들웨어 내에 존재하는 기존 컴포넌트의 이동, 삭제와 갱신, 새로운 컴포넌트의 삽입 기능을 통한 컴포넌트의 동적 재구성이 가능해짐으로써 시스템을 재시작하지 않고 접속중인 사용자나 환경의 변화와 요구에 맞추어 서비스를 제공할 수 있게 된다. 컴포넌트 동적 재구성이 가능한 미들웨어를 개발하기 위해서 전체 시스템의 컴포넌트간의 의존관계(Dependency)를 파악하는 것이 필요하다. 본 논문은 컴포넌트의 동적 재구성과 컴포넌트간의 의존관계를 설명하고, 컴포넌트의 의존관계를 고려하여 컴포넌트의 동적 재구성을 효율적으로 관리하는 방법을 제안한다.

1. 서론

유비쿼터스 컴퓨팅에서의 미들웨어는 주위 환경을 감지하는 센서나 인터넷정보를 이용해서 얻은 모니터링 정보를 이용하여, 상황 변화에 맞게 서비스를 제공할 수 있도록 실행 중에 시스템을 재구성(Reconfiguration)하여 재시작하지 않고 새로운 환경에 적응할 수 있어야 한다. 이를 위해서 분산시스템에서의 동적 재구성(Dynamic Reconfiguration) 기술이 연구되어 왔다.

최근에는 인터넷에 연결된 컴퓨터뿐만 아니라 많은 전기장치(PDA, 휴대폰, RFID, 자동차의 텔레매틱스, 홈 네트워크에서의 가전제품 등)가 인터넷 등의 네트워크로 연결이 가능해짐에 따라, 미들웨어는 전체 네트워크에 연결된 장치에 필요한 인터페이스와 서비스를 다양하게 제공하여야 한다. 또한 재사용이 뛰어나 컴포넌트 사이에 조립이 가능한 장점을 가진 컴포넌트가 각광을 받게 됨에 따라서 CCM (Corba Component Model)과 EJB (Enterprise Java Bean)와 같은 컴포넌트 기반의 미들웨어가 등장하였다[6]. 이와 더불어 미들웨어의 더욱 안정적이고 효율적인 QoS(Quality of Service)를 제공하기 위해서 동적 재구성이라는 기술이 연구되고 있다. 동적 재구성으로 인하여 미들웨어는 다양한 상황과 사용자의 요구의 변화에 이전보다 더욱 나은 서비스를 제공할 수 있게 되었다. 하지만 앞으로의 시스템은 더욱 거대해지고, 다양한 시스템을 통합하고 다양한 요구에 맞게 서비스해주기 위해서는 보다 높은 유연성과 확장성이 필요하다[5]. 본 논문은 컴포넌트의 동적 재구성과 의존관계를 설명하고, CCM을 통한 컴포넌트간의 의존관계를 고려하여 보다 효율적으로 컴포넌트를 재구성할 수 있는 방법을 제안한다.

* 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 지원에 의한 것임

2. 관련 연구

2.1 CCM의 컴포넌트

CCM에서 Facets, Receptacle, Attribute, Event source/sinks 4가지를 포트(port)라고 하고, 이 포트 메카니즘으로 컴포넌트의 특성(capability)이 정해진다. CCM은 포트를 이용하여 이전의 코바 객체 모델에 비하여 컴포넌트의 재사용성이 강화된다[3]. Facet는 다른 컴포넌트에 인터페이스를 제공해준다. Receptacle은 다른 컴포넌트가 요구하는 특정 인터페이스를 제공해주기 위하여 facet를 받아들이는 역할을 한다. Attribute는 컴포넌트의 속성을 의미하고, Event source/sinks는 비동기의 이벤트를 모니터링 함으로써 에러 없이 상호작용 시켜주는 기능을 한다.

2.2 동적 재구성

재구성은 컴포넌트와 오브젝트, 자바빈(Java Beans)과 같은 재구성 가능한 엔티티를 추가, 삭제, 이동, 삭제 하는 것을 말한다. 동적 재구성은 변화된 환경이나 사용자의 요구에 맞게 시스템의 중단 없이 시스템을 발전시키는 것을 가능하게 한다. 또한 시스템을 투명하게 재구성함으로써 안정적인 서비스를 제공하도록 해야 한다.

그림1은 동적 재구성을 간소화 한 모델이다. 그림1은 현재의 구성정보에서 새롭게 변화되어야 할 디자인 정보를 가지며 재구성 관리자는 현재의 구성정보와 재구성디자인을 통하여 현재 실행중인 시스템을 발전시키고, 시스템은 재구성한 구성정보를 가지고 계속적으로 실행된 상태를 유지한다. 동적 재구성을 통하여 시스템은 높은 가용성(high availability), 적응성(adaptability)과 지속성(maintainability)을 가질 수 있게 된다[2].

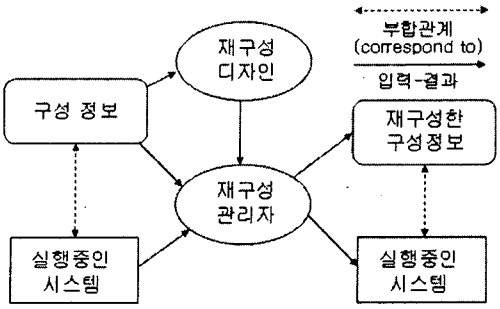

그림1. 동적 재구성 모델

또한 동적 재구성 이후에도 시스템의 컴포넌트가 올바르게 실행되어야 하므로, 재구성을 할 때에는 다음의 세 가지 조건을 만족해야 한다. 아래의 조건을 만족시키면 시스템은 올바른 동적 재구성이 실행되고 있다고 말할 수 있다[1].

- 재구성은 전체 시스템에 영향을 미칠 수 있으므로, 시스템은 조직적으로 통합이 되어야 한다.
- 시스템의 엔티티들은 상호간에 일관적인 상태에 있어야 한다. 엔티티들은 각각 다른 엔티티의 서비스를 필요로 하는 의존관계를 가질 수 있기 때문이다.
- 재구성 전 후의 시스템을 이용하는 애플리케이션의 상태 또한 변하지 않아야 한다.

2.3 컴포넌트 의존관계

컴포넌트가 다른 컴포넌트의 서비스를 필요로 하여 호출(invocation)하면, 해당 컴포넌트는 다른 컴포넌트에 의존한다고 정의한다. 컴포넌트 시스템은 개별적인 컴포넌트들로 구성된다. 정적인 의존관계에서 컴포넌트 시스템을 디자인 할 때에는 컴포넌트의 추가, 삭제, 갱신의 3가지 형태로 이루어진다[2]. 이와 다르게 동적 재구성이 가능한 컴포넌트 시스템은 추가, 갱신, 삭제, 이동의 4가지의 형태를 수행한다. 다음은 의존관계를 고려한 동적재구성의 수행 형태이다.

- 추가 : 실행 중에 사용자의 요구로 인하여 동적으로 시스템에 새로운 컴포넌트를 적재한다. 적재 후 바로 전체시스템의 의존관계를 검사한다.
- 갱신 : 기존에 있는 컴포넌트가 새로운 버전으로 갱신된다. 갱신된 컴포넌트에 새로운 서비스가 추가될 수도 있으므로, 추가와 마찬가지로 갱신 후 다른 컴포넌트와의 의존관계를 검사한다.
- 이동 : 컴포넌트가 현재의 위치에서 새로운 위치로 이동한다. 이후 전체 시스템의 의존관계를 검사한다. 다른 의존관계에 있는 컴포넌트와의 연결지점을 이동하는 것이다.
- 삭제 : 컴포넌트의 서비스를 사용하지 않게 될 경우 해당 컴포넌트를 시스템에서 삭제한다. 컴포넌트 삭제하기 전에 컴포넌트가 의존관계에 있는 다른 컴포넌트에 사용되고 있는지 반드시 전체 시스템을 검사해야 한다.

그림2의 (A)는 컴포넌트 A는 B에 의존관계에 있고, B는 C에 의존관계에 있음을 나타낸다. A는 컴포넌트 B의 클라이언트이고 C는 B의 서버이고 B가 삭제된다고 가정한다면, 클라이언트 A가 B의 컴포넌트를 호출할 수 없으므로 시스템은 더 이상 올바른 수행을 할 수 없다. 그림2의 (B)는 컴포넌트간의 의존관계가 네스티드(nested)인 경우이다. 서로 의존관계가 아닌 컴포넌트 A, B 사이에 A에 호출되고, C를 호출하는 B컴포넌트가 추가된다면, A와 B와 C는 서로 네스티드 동적 의존관계(Nested dynamic dependencies)에 있게 된다. 컴포넌트 재구성은 컴포넌트의 네스티드 의존관계도 고려해야 한다.

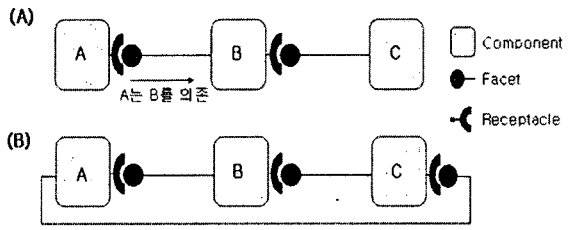

그림 3. 컴포넌트의 의존관계

3. 제안 모델

분산컴포넌트시스템에서는 실행 중에 시스템의 진화와 유지가 가능하지만 시스템을 재구성하는 시간동안 사용자들은 일시적인 인터럽트로 인해 서비스를 제공받지 못하게 되므로 QoS가 떨어지게 된다[1]. 본 논문에서 제안하는 모델을 통하여 시스템의 재구성하는 횟수를 감소시켜서 사용자들의 인터럽트 횟수를 줄이고 QoS를 높이고자 한다.

그림3에서와 같이 컴포넌트의 효율적인 의존관계의 유지를 위하여 두 단계의 모델을 구성하였다. 단계1은 자신 컴포넌트에 의존을 요구하는 컴포넌트가 없고, 자신이 다른 컴포넌트에 의존하는 경우의 컴포넌트만을 분류하여 구성하였고, 단계2는 자신 컴포넌트에 의존하고 있는 다른 컴포넌트가 있어서 재구성시에 전체 시스템에 영향을 끼칠 수 있는 컴포넌트들을 따로 분류하여 구성하였다.

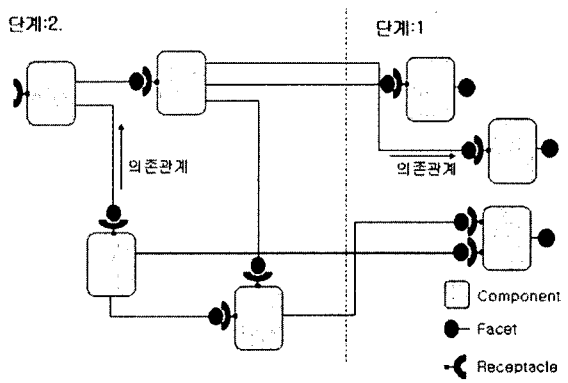

그림 3 의존관계에 따라 이원화 한 컴포넌트 모델

이와 같이 컴포넌트 모델을 두 단계로 나눔으로써 시스템의 효율적성을 증대시켰다. 단계2에 있는 컴포넌트를 재구성할 때에는 전체시스템에 대하여 재구성을 하고, 단계1에

있는 컴포넌트가 재구성될 때에는 단계2의 재구성은 고려하지 않고, 바로 단계1의 컴포넌트만 재구성시키면 된다.

처음 컴포넌트를 생성할 때에는 단계1에 컴포넌트를 배치(deploy)하고, 이후에 단계1에 있는 컴포넌트 중에서 다른 컴포넌트가 자신 컴포넌트에 의존하게 될 경우에만 그 컴포넌트를 단계2로 이동 시키는 방법을 이용한다. 왜냐하면 처음부터 단계2에 컴포넌트를 배치하게 되면 전체 시스템을 재구성해야 하는데 단계1에 배치시키면 전체 시스템에 영향을 끼치지 않기 때문이다. 서비스 결국 전체 시스템이 동적 재구성을 하는 횟수를 줄일 수 있게 되어 사용자의 입장에서도 일시적인 인터럽트의 횟수와 시간을 줄일 수 있다.

아래는 제안된 모델에서 컴포넌트의 재구성을 위한 각각의 수행별 절차이다.

1) 추가
- 클라이언트의 요구로 인하여 새로운 컴포넌트를 단계1에 추가시킨다.
- 구성자(configurator)는 새로 추가된 컴포넌트와 다른 컴포넌트 사이의 의존관계를 알기 위하여 전체 시스템의 구성을 파악한다. 새로운 컴포넌트에 다른 컴포넌트가 의존하게 되면, 단계2로 컴포넌트를 이동시키고 전체 시스템을 재구성한다. 재구성 중에 사용자들이 접속 중이면 오류가 날 수 있으므로, 일시적으로 인터럽트 시킨다.
- 만약 새로 추가된 컴포넌트에 다른 컴포넌트가 의존하지 않으면, 단계1에 컴포넌트를 추가만 하여 재구성한다. 새로운 컴포넌트에 의존하는 컴포넌트가 없는 시스템의 단계2는 영향을 받지 않으므로 사용자는 일시적인 인터럽트가 필요 없다.

2) 삭제
- 단계2에서의 컴포넌트 삭제는 전체의 시스템에 영향을 끼치므로, 전체 시스템을 재구성하게 된다. 단계1에 있는 컴포넌트의 삭제에는 바로 그 컴포넌트만 삭제하면 된다.

3) 이동
- 단계2에 있는 컴포넌트가 이동할 때에서 전체시스템에 대하여 재구성을 하고, 단계1안에서 컴포넌트가 이동할 때에는 단계1만 재구성을 한다.
- 단계2에 있는 컴포넌트에 의존하는 다른 컴포넌트가 삭제, 갱신 등의 이유로, 그 컴포넌트에 의존하는 컴포넌트가 없어지면 그 컴포넌트는 단계1로 이동한다.

4) 갱신
- 단계2의 컴포넌트의 갱신에는 기존의 재구성 방법과 같다.
- 단계1에서 재구성시에는 본인 컴포넌트에 의존관계가 없으면 전체시스템과 상관없이 본인 컴포넌트만 재구성한다.

시스템을 1,2 두 단계로 나눔으로써 단계1에서만 재구성을 할 때에 단계2는 계속 현 시스템을 유지하여 사용자에 인터럽트를 주지 않고 계속적으로 서비스를 할 수 있다. 때문에 시스템은 사용자가 보다 안정적이고 꾸준한 서비스가 가능해짐으로 인하여 QoS가 향상된다.

본 논문에서는 각각의 컴포넌트 재구성을 위해 두 단계로 나누어 관리하였다. 또한 점차 시스템이 거대해지고 복잡해지므로 다양한 방법을 통해 더욱 효율적으로 동적 재구성을 하는 방법을 연구해야 한다. 컴포넌트의 특성인 Name, Version, Owner, Vender, Data Imported 등을 이용하여 컴포넌트를 그룹으로 분류하여 관리할 수 있다. 예를 들어, 같은 벤더에 구매한 컴포넌트 들은 모두 특정 컴포넌트에 의존해야 하는 경우에 그 컴포넌트들을 한번에 관리하면 된다. 위의 장점으로 해당 그룹을 동적 재구성 할 때에 다른 그룹은 상관없이 계속된 서비스를 제공한다. 앞으로 그룹 컴포넌트를 고려한 컴포넌트 동적 재구성에서도 특정 조건하에서 전체 시스템을 재구성하지 않고 특정 그룹만 재구성할 수 있는 방법을 연구해야 한다. 또한 구성자의 개수를 여러 개로 나눌 수 있다. 본 논문의 관리를 두 단계로 나눈 시스템에서 구성자가 두 개 존재할 경우에 시스템의 단계1만 재구성한다면, 단계2와 전체시스템은 단계1의 재구성에 영향을 받지 않는다. 때문에 단계 2의 구성자는 단계 1의 재구성에 상관없이 계속 자신이 관리할 영역의 구성정보만 유지하면 된다.

4. 결론 및 향후 과제

컴포넌트기반시스템 내에서 컴포넌트는 각기 다른 컴포넌트와 의존관계에 있고, 인터페이스를 공유한다. 그리고 동적 재구성을 통하여 사용자에 꾸준한 서비스를 제공해준다. 본 논문은 이러한 의존관계를 고려하여 두 가지 레벨로 컴포넌트 시스템을 분리하여 더욱 효율적인 서비스가 가능할 것으로 예상되는 컴포넌트 동적 재구성 모델을 제안하였다. 향후 개발 예정인 컴포넌트 기반의 상황 인지 미들웨어에 본 논문에서 제시한 동적 재구성 모델을 응용하여 구현할 것이다.

5. 참고 문헌

[1] Maarten Wegdam, "Dynamic Reconfiguration and Load Distribution in Component Middleware," Publisher. Telematica Institut, pp. 71-112, 2003.

[2] Xuejun Chen, "Dependence management for dynamic reconfiguration of component-based distributed systems," Proceedings. ASE 2002, pp. 279-284, Sept 2002.

[3] Wang, Schmidt, O'Ryan, "Overview of the CORBA Component Model," Component-based software engineering: putting the pieces together, pp.557-571., 2001.

[4] Fabio Kon and Roy H.campbell, "Dependence Management in Component-Based Distributed System," Proceedings. IEEE [see also IEEE Parallel & Distributed Technology] Volume 8, Issue 1, pp. 26-36, Jan-March 2000.

[5] Marlon Vieria, Debra Richardson, "Analyzing Dependencies in Large Component-Based System," Proceedings. ASE 2002, pp. 241-244, Sept 2002.

[6] Emmerich, W. Kaveh, N, "Component technologies: Java beans, COM, CORBA, RMI, EJB and the CORBA component model," Proceeding. ICSE 2002, pp. 691-692, May 2002.