분산 컴포넌트 기반의 소프트웨어 설계 방법

최유희[U] 염근혁
부 산 대 학 교 컴 퓨 터 공 학 과
{yheechoi, yeom}@hyowon.cc.pusan.ac.kr

An Approach to Software Design based on Distributed Components

Youhee Choi[U] Keunhyuk Yeom
Dept. of Computer Engineering, Pusan National University

요 약

현재 새롭게 개발되는 소프트웨어의 추세는 50% 이상이 분산 플랫폼에서 개발되고 있다. 또한 분산 플랫폼을 가능하게 하는 COM, CORBA, EJB와 같은 기술이 급격히 발전하고 있다. 그러나 대부분의 일반적인 컴포넌트 기반 소프트웨어 개발 방법론은 분산 플랫폼에 대한 고려를 체계적으로 다루지 않음으로써 컴포넌트의 분산에 의한 오버헤드를 충분히 극복할 수 있을 만큼의 이점을 얻기 어렵다.

따라서 본 논문에서는 분산 컴포넌트로 구성된 소프트웨어를 설계하는 과정에서 분산 이슈를 실질적으로 다루기 위하여 CORBA 환경을 고려하여 성능, 결함 방지, 안전성, 분산 트랜잭션의 분산 이슈를 명시적으로 다루기 위한 기본적인 지침을 제시한다.

1. 서론

현재 새롭게 개발되는 소프트웨어의 추세는 CORBA, EJB, COM 등의 분산 컴포넌트 아키텍처를 바탕으로 분산 환경에서 수행될 수 있는 분산 컴포넌트로 구성된 소프트웨어를 지향하고 있다. 분산 컴포넌트로 구성된 소프트웨어는 네트워크상의 다양한 형태의 물리적 구성에 배치될 수 있는 별개의 컴포넌트들로 구성된 소프트웨어를 의미한다[1]. 즉 분산 컴포넌트는 다양한 형태의 물리적 구성에 배치될 수 있는 컴포넌트로서 분산 컴포넌트간의 상호작용에 있어서 네트워크를 거치게 된다. 그러므로 수행 속도의 지연에 의한 성능 저하를 극복할 수 있는 이점을 가져야 한다. 분산 컴포넌트 기반의 소프트웨어의 이점 중의 하나가 향상된 가용성(availability) 측면으로 이 측면을 만족하기 위하여 결함 방지에 대한 고려가 필요하다[3]. 또한 분산 컴포넌트간의 네트워크를 통한 상호작용에 따른 안전성에 대한 고려가 요구되며, 분산 환경에서의 트랜잭션 수행에 있어서 하나의 트랜잭션이 여러 분산 컴포넌트에 의해 수행되는 분산 트랜잭션에 대한 고려가 필요하다.

그러나 이러한 여러 고려 사항들이 현재 대부분의 컴포넌트 기반 소프트웨어 개발 방법론에 의해서 체계적으로 다루어지지 않고 있고 하위 레벨인 분산 플랫폼에서 다루어지는 고려 사항으로만 인식되고 있다. 그러나 분산 플랫폼은 단지 결함 방지, 안전성, 분산 트랜잭션 등에 대하여 하위레벨에서의 서비스 제공의 측면을 다루는 것으로 실질적으로 분산 컴포넌트 기반의 소프트웨어의 개발 과정에서 각각을 설계 단계에서 어떻게 다룰 것인가에 대하여 고려하지 않는다. 따라서 설계 단계에서의 명시적인 고려가 없이는 분산 컴포넌트 기반의 실효성과 성능 향상에 대하여 기대할 수 없다. 따라서 본 논문에서는 분산 컴포넌트 기반의 소프트웨어의 기본적인 요소인 분산 플랫폼과 분산에 따른 고려 사항들의 관계를 명시적으로 고려한 설계 방법을 제시한다. 다루고자 하는 분산 플랫폼은 현재의 대표적인 분산 플랫폼인 Sun의 EJB, Microsoft의 COM, OMG의 CORBA 중에서 EJB, COM에 비하여 다양한 프로그래밍 언어와 이기종 플랫폼을 지원하고 향상된 분산 하부 서비스를 지원하는 등의 많은 장점을 가지고 있는 CORBA를 기반으로 한다. 따라서 본 논문에서는 분산 컴포넌트 기반의 소프트웨어의 설계 과정에 대하여 CORBA 환경을 고려하여 성능, 결함 방지, 안전성, 분산 트랜잭션을 다루는 기본적인 지침을 명시적으로 제시한다.

2. 관련연구

현재 체계적인 설계 방법에 대한 제시는 없으나 분산에 따른 여러 고려 사항들과 여러 실패 사례를 통하여 추출된 문제점에 대해서 인식하고 있고[2,5], 이러한 여러 고려 사항들과 문제점을 분산 플랫폼(CORBA, EJB 등)을 이용하여 해결하고자 하는 연구들이 있다. 이러한 연구 중에서 CORBA를 기반으로 결함 방지와 실시간 측면의 문제를 해결하기 위해서 각 분산 컴포넌트의 인터페이스를 정의하는 관점에서의 연구가 있다[4]. 그러나 현재 대부분의 분산 플랫폼을 이용한 소프트웨어 개발 방법과 관련된 연구는 주로 분산 플랫폼에서의 각 분산 이슈 자체의 특징을 해결하기 위한 측면이기 때문에 분산 컴포넌트 기반의 소프트웨어 설계를 위한 측면을 제시하고 있지 않다.

3. 분산 컴포넌트 기반의 소프트웨어 설계

본 논문에서 제시하는 설계 방법은 UML의 사용을 기반으로 하고 CORBA를 분산 플랫폼으로 하여 성능, 결함 방지, 안전성, 분산 트랜잭션을 다룬다. 분산 컴포넌트 기반의 소프트웨어를 개발하는데 필요한 요구 사항 추출과 분석 단계는 UML을 사용하는 컴포넌트 기반 소프트웨어 개발 방법론인 Unified process[6]를 바탕으로 수행한다.

설계 방법의 전체 작업은 분산 컴포넌트 기반의 소프트웨어의 기본 단위인 분산 컴포넌트를 생성하는 작업과 추출된 분산 컴포넌트에 대해 분산 이슈를 고려하여 클래스를 설계하는 작업으로 구성된다.

3.1. 분산 컴포넌트 추출

대표적 분산 구조의 형태인 3-계층 구조와 분석 클래스를 바탕으로

분산 컴포넌트를 추출한다. 이 때 각 분석 클래스는 요구 사항 추출 단계와 분석의 산출물인 use case diagram과 analysis package를 중심으로 추출된다. 각 3-계층에 따라서 분산 컴포넌트를 추출하는 기준은 다음과 같다.

◆ 사용자와의 상호작용을 다루는 계층의 경우, 사용자와의 상호작용을 모델링하는 boundary class가 분산 컴포넌트로 추출될 수 있다. Boundary class는 use case의 사용 목적 및 방식에 따라서 달라진다. 따라서 동일한 사용 목적 및 방식에 의한 여러 boundary classes가 분산 컴포넌트에 해당된다.

◆ 비즈니스 로직을 다루는 계층의 경우, 비즈니스 로직을 모델링하는 control class가 분산 컴포넌트로 추출될 수 있다. 독립적으로 분리될 수 있는 관계 측면에서 하나의 분산 컴포넌트로 추출될 수 있는 control classes간의 관계는 다음과 같다.

① association 관계가 있는 control classes
② 동일 analysis package(or 동일 use case)와 관련 있는 control classes
③ 수행상의 조건 만족 여부에 따라 수행되는 control classes
④ aggregation 관계의 control classes의 경우 'whole-part'의 part에 해당되는 control classes

◆ 지속적으로 유지되는 정보를 다루는 계층의 경우, 지속적으로 유지되는 정보를 모델링하는 entity class가 분산 컴포넌트로 추출될 수 있다. entity class와 control class의 상호작용을 고려했을 때, 동일한 control class와 관련 있는 여러 entity classes를 하나의 분산 컴포넌트로 추출한다.

그리고 각 분산 컴포넌트에 대하여 다른 분산 컴포넌트에 의존관계가 있는 클래스를 각 분산 컴포넌트의 인터페이스로 정의한다.

3.2. 분산 이슈를 고려한 분산 컴포넌트와 클래스 설계

3-계층 구조를 바탕으로 추출된 분산 컴포넌트에 대해서 성능, 결함 방지, 안전성, 분산 트랜잭션의 분산 이슈를 CORBA를 바탕으로 고려하여 분산 컴포넌트와 클래스를 설계한다.

◆ 성능에 대한 고려

분산 컴포넌트 기반의 소프트웨어에서 성능에 영향을 미치는 것은 분산 컴포넌트간의 상호작용에 있어서 네트워크를 거치게 되는 원거리 호출에 따른 전송 지연과 마셜링 오버헤드라고 할 수 있다. 따라서 성능 향상을 위해서는 원거리 호출을 줄이기 위한 설계가 요구된다. 원거리 호출을 줄이기 위한 접근 방법으로 분산 컴포넌트간의 상호작용을 고려하여 두 가지 측면으로 나눌 수 있다.

① 분산 컴포넌트를 재구조화하는 측면: 그림 1에서처럼 각 분산 컴포넌트의 클래스에 대하여 수행 과정상에서 다른 분산 컴포넌트에 속하는 클래스에 요청을 하고 응답을 받아야만 다음 과정의 수행이 이루어지는 원거리 호출의 관계를 class diagram과 collaboration diagram을 이용하여 추출한다. 그러한 관계에 해당되는 각 클래스를 각 클래스가 속한 분산 컴포넌트 중 하나의 분산 컴포넌트에 옮긴다. 옮기는 클래스를 선택하기 위해서는 각각이 속해 있는 분산 컴포넌트 내부의 다른 클래스들간의 관계에서 다른 분산 컴포넌트로 이동되었을 때를 가정하여 마셜링 오버헤드를 비교한다. 비교하여 마셜링 오버헤드가 상대적으로 작은 클래스를 옮긴다.

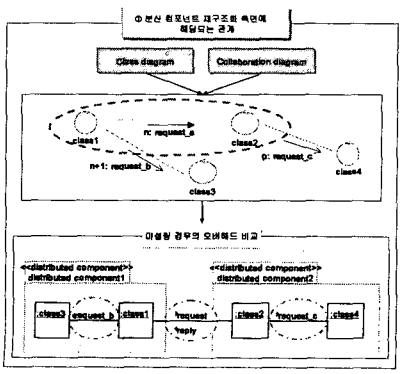

그림 1 분산 컴포넌트의 재구조화 측면

② 분산 컴포넌트의 인터페이스를 설계하는 측면: 각 분산 컴포넌트에 존재하는 클래스들간의 기능 호출 관계가 존재할 경우 해당되는 기능에 대하여 인터페이스의 오퍼레이션을 어떻게 정의하느냐에 따라서 원거리 호출의 수가 달라질 수 있다. 그림 2(a)에서처럼 다른 분산 컴포넌트에 속한 클래스의 여러 오퍼레이션(op1(), op2())이 하나의 기능 호출관계에 의한 것일 경우를 추출한다. 이러한 경우에 그림 2(b)에서처럼 여러 오퍼레이션(op1(),op2())을 포함할 수 있는 하나의 오퍼레이션(Wrapping_op()) 형태로 다른 분산 컴포넌트와 관계가 있는 인터페이스를 정의한다.

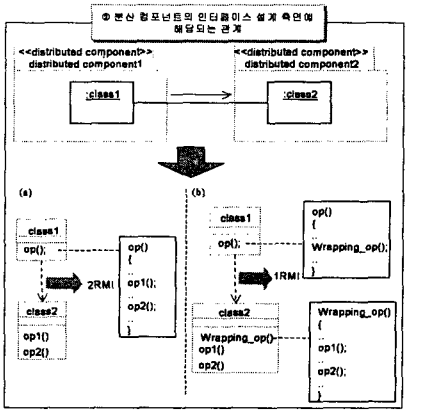

그림 2 분산 컴포넌트의 인터페이스 설계

◆ 결함 방지에 대한 고려

결함 방지의 기본적인 요소인 결함 탐지의 단위에 따라서 복제(replication)의 타입과 단위, 결함의 타입에 영향을 줄 수 있다. 따라서 설계 단계에서 고려될 수 있는 결함 탐지의 레벨을 분산 컴포넌트 레벨과 호스트 레벨로 나누어 결함의 타입과 복제의 타입을 분류한다.

① 호스트 레벨: 주요 결함은 호스트 자체의 결함과 네트워크를 거치는 원거리 호출에서의 결함이다. 이러한 종류의 결함은 수행을 중단시키는 결함인 fail-stop 타입에 해당되고 복제의 타입으로 passive 타입이 해당될 수 있다.

② 분산 컴포넌트 레벨: 주요 결함은 분산 컴포넌트에 포함되는 클래스의 오퍼레이션 수행상의 의미적인 결함이다. 이러한 종류의 결함은 수행을 중단시키지는 않으나 잘못된 수행을 하는 byzantine 타입에 해당되고 복제의 타입으로 active 타입이 해당될 수 있다.

또한 3-계층 구조와 데이터의 타입에 따라 결함 발생시 동기화가 필요한 상태를 분류하면 다음과 같다.

① 첫 번째 계층과 두 번째 계층의 경우, 각 분산 컴포넌트의 기능 수행 관계에서 다른 분산 컴포넌트의 기능 수행에 영향을 주는 데이터에 대한 변경이나 생성이 되는 경우의 상태

② 세 번째 계층의 경우, 데이터베이스와 같은 저장소에 저장된 데이터에 대해서 수정이 가해지는 경우의 상태

결함 방지에 대한 표현은 UML의 tagged value '{replication=' '}'를 이용하여 결함 방지가 필요한 레벨에 대하여 해당되는 복제 타입을 명시한다.

◆ 안전성에 대한 고려

안전성을 다루기 위하여 CORBA 환경에서 설계시에 고려하여야 할 사항은 어떤 정보에 대해서 안전성을 다루고 어떤 원칙을 적용할 것인지에 대한 것으로 3-계층 구조에 따라서 다루는 데이터의 타입과 안전성 정책을 분류하여 제시하면 다음과 같다.

① 첫 번째 계층의 경우, 사용자의 입력 또는 요구와 사용자에게 제공하는 출력 또는 응답의 초기값이 변경되지 않도록 하는 무결 보호(integrity protection)정책과 사용자가 누구인지를 증명하는 인증(authentication)정책이 사용될 수 있다.

② 두 번째 계층의 경우, 분산 컴포넌트간의 네트워크를 통하는 모든 메시지에 대하여 기밀성 보호(confidentiality protection)정책과 사용자의 권한이 각 분산 컴포넌트에 전달될 수 있도록 하는 권한 위임(delegation of privilege)정책이 사용될 수 있다.

③ 세 번째 계층의 경우, 데이터베이스에 접근 권한을 다루는 접근 원

책(access policy)과 권한 부여(authorization)정책이 사용될 수 있다.

분류된 안전성 정책이 적용되는 레벨에 따라서 tagged value '{security=' ')'를 이용하여 어떤 안전성 정책이 사용될 수 있는지를 명시한다.

◆ 분산 트랜잭션에 대한 고려

CORBA 환경에서 분산 트랜잭션은 2PC protocol을 사용하므로 2PC protocol 사용에 의한 오버헤드를 줄이기 위하여 동일 분산 컴포넌트 내에서 2PC protocol이 수행되도록 한다. 따라서 3-계층 구조에서 세 번째 계층에 해당되는 분산 컴포넌트의 entity class를 다루는 오퍼레이션 단위로 트랜잭션을 다루는 per-operation transaction model을 이용한다. 또한 결함 방지와 관련하여 per-operation transaction model을 사용함으로써 상태를 동기화할 필요가 없는 복제 타입이 적용될 수 있다.

4. Case Study

본 연구에서 제안한 설계 방법을 이용하여 지능형 교통 시스템(ITS: Intelligent Transportation System)의 첨단 교통 정보 시스템의 자가 운전자를 위한 여행전 정보 제공 시스템을 실제 적용분야로 선택하였다.

그림 3과 그림 4는 '3.1. 분산 컴포넌트 추출'의 수행결과 중 3-계층 구조의 두 번째 계층에 해당되는 분산 컴포넌트로 추출된 예이다.

그림 3은 association 관계가 있는 control class들에 대하여 추출된 것으로 'Route calculator', 'Static traffic info. acquisition', 'Traffic info. acquisition' control class들은 association 관계가 존재한다. 이러한 association 관계가 존재한다는 것은 상호작용이 존재한다는 것이므로 association에 따라서 분산 컴포넌트 단위로 추출된다.

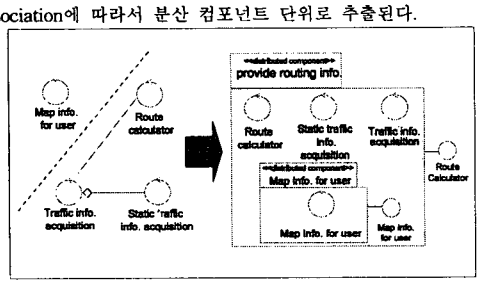

그림 3 분산 컴포넌트 추출 예

그림 4는 aggregation 관계가 있는 control class에 대하여 추출된 것으로 whole-part 관계에서 'Traffic info. acquisition' control class에 대하여 독립적인 구성요소인 part에 해당되는 'Static traffic info. acquisition'이 독립적으로 분산될 수 있으므로 분산 컴포넌트의 단위로 추출된다.

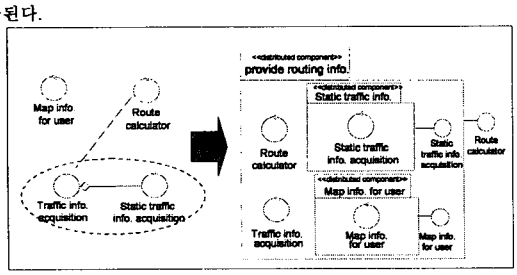

그림 4 분산 컴포넌트 추출 예

그림 5는 성능에 대한 고려에서 분산 컴포넌트를 재구조화하는 측면에 대한 예이다. 그림 5에서 'User's role for routing info.' 분산 컴포넌트의 'Routing info. request UI' boundary class에서 맵 정보를 얻어오기 위하여 다른 분산 컴포넌트의 'Map info. for user' control class의 기능을 사용하므로 원거리 호출이 존재하게 된다. 그러나 'Map info. for user' control class는 'provide routing info.' 분산 컴포넌트에 속해 있는 control class들과 관계가 없기 때문에 'Map info. for user' control class가 다른 분산 컴포넌트에 존재한다고 해도 다른 오버헤드는 없다. 이런 경우에 'Map info. for user' control class를 원거리 호출 관계가 있는 'User's role for routing info' 분산 컴포넌트에 옮김으로써 원거리 호출을 줄인다.

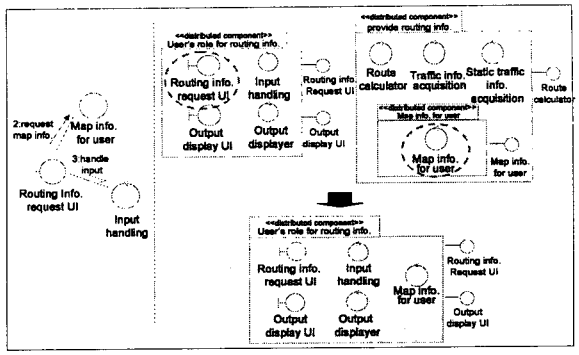

그림 5 성능에 대한 고려

그림 6은 결함 방지와 관련하여 분산 컴포넌트 단위에서의 결함 탐지일 경우에 대한 결함의 타입과 복제 타입에 대한 예이다. 'routing calculator' class는 경로를 계산하는데 필요한 'weight' 요소와 경로, 총시간과 거리 등의 값을 생성하게 된다. 이 클래스가 가지고 있는 attribute들은 기능 수행에 영향을 주는 생성된 데이터에 해당되므로 동기화가 필요한 상태에 해당된다. 또한 이러한 계산을 수행하는 여러 오퍼레이션들이 존재하는데 오퍼레이션이 잘못 구현될 경우 틀린 값이 계산될 수 있다. 즉, 이러한 오퍼레이션들이 수행을 중단시키지는 않으나 잘못된 수행을 하는 byzantine 타입의 결함을 생성할 수 있다. 이러한 타입의 경우 active 타입의 복제가 요구될 수 있으므로 그림 6에서처럼 '{replication=ACTIVE}'의 tagged value로서 나타낸다. 그림 7은 안전성과 관련하여 3-계층 구조의 첫 번째 계층에 해당되는 클래스의 설계에 대한 예이다. 그림 7에서 사용자 인증 기능을 수행하는 'login UI' class와 'ID checker' class에는 인증 정책이 사용될 수 있다. 또한 사용자에게 제공하는 응답이 전달 중에 변경되지 않도록 하기 위해서 응답을 전달하는 과정의 오퍼레이션에는 무결 보호 정책이 사용될 수 있음을 나타낸다.

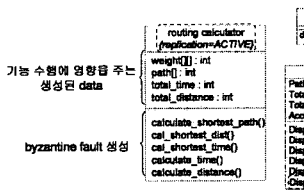

그림 6 결함 방지에 대한 고려

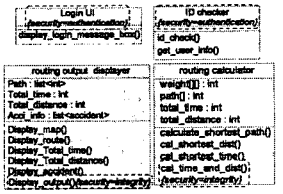

그림 7 안전성에 대한 고려

5. 결론 및 향후 연구 과제

본 논문에서는 분산 컴포넌트 기반의 소프트웨어의 설계 과정에 대하여 UML의 사용을 기반으로 하고 CORBA 환경을 고려하여 성능, 결합 방지, 안전성, 분산 트랜잭션을 다루는 기본적인 지침을 명시적으로 제시하였다. 그리고 지능형 교통 시스템의 교통 정보 시스템 분야에 적용함으로써 제시한 방법의 효용성을 평가하였다. 향후 연구 과제로는 실질적인 구현을 통하여 분산 이슈를 다루기 위한 지침을 보완한다.

6. 참고 문헌

[1] Microsoft, "Understanding How Distributed Components Affect Performance", http://msdn.microsoft.com/library, 1999.

[2] Sessions, R., "Ten Rules for Distributed Object Systems", http://www.objectwatch.com/magart5.htm.

[3] Chelliah, M. and Ahamad, M., "System mechanisms for distributed object-based fault-tolerant computing", Proceedings of IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems, 1995.

[4] Yau, S.S. and Bing Xia, "Object-oriented distributed component software development based on CORBA" COMPSAC '98. Proceedings., 1998.

[5] Uzun, U., "Towards distributed object design", University of Warwick, March 1998.

[6] Jacobson, I., Booch, G., and Rumbaugh, J., "The Unified Software Development Process", Addison-Wesley, January 1999.