2000 년 한국정보처리학회 춘계 학술발표논문집 제 7 권 제 1 호

다중 사용자 환경에서의 아키텍쳐 기반 컴포넌트 소프트웨어 개발

김상길*, 안치돈*, 왕창종*
*인하대학교 전자계산공학과
e-mail : askid@selab.cse.inha.ac.kr

Architecture-based Component Software Development on Multi-user Environment

O S. K. Kim*, C. D. Ahn*, C. J. Wang*
*Dept. of Computer Science & Engineering, Inha University

요 약

CSCW 시스템은 정보 공유 구조에서 필수적인 기술로 자리잡고 있으며, 이를 위한 사용자간 공동작업 기능과 정보 공유 기능이 목표가 되고 있다. 이 연구에서는 기존의 아키텍쳐 기반 컴포넌트 검색 시스템의 다중 사용자 환경으로의 확장을 고려하였다. 다중 사용자 환경에서 새로운 소프트웨어 개발을 위한 소프트웨어 아키텍쳐 설계 과정에서 개발자들의 서로 다른 관점에서의 의견을 최대한 반영할 수 있는 방법으로 다중 계층 소프트웨어 아키텍쳐 구조를 제시하였다.
다중 사용자 환경에서 사용자간 공동작업과 그룹관리를 위해 세션관리자를 두었으며, 저장소에 저장되어 있는 아키텍쳐와 컴포넌트의 사용자 인터페이스 명세에 버전 정보를 추가함으로써 공동작업에서 사용자들에게 참조를 제공한다. 소프트웨어 아키텍쳐 설계 과정에서 생성된 새로운 아키텍쳐는 아키텍쳐 저장소와 컴포넌트 저장소에 버전 정보와 함께 추가됨으로써 이후의 소프트웨어 개발을 좀 더 효율적으로 이루어질 수 있도록 하였다.

1. 서론

CSCW(Computer-Supported Cooperative Work) 시스템은 정보 공유 구조에서 필수적인 기술로 자리잡고 있으며, 이를 위한 사용자간 공동 작업 기능과 정보 공유 기능이 목표가 되고 있다[1].
이 연구에서는 기존의 아키텍쳐 기반 컴포넌트 검색 시스템[2]을 다중 사용자 환경으로의 확장을 고려한다. 다중 사용자 환경에서 새로운 소프트웨어 개발을 위한 소프트웨어 아키텍쳐 설계 과정은 각각의 개발자 관점에서 설계된 아키텍쳐 정보들을 수집하여 이를 조정할 필요가 있다. 소프트웨어 아키텍쳐 설계 단계에서 가능한 한 모든 개발자의 의견이 반영될 수 있는 복합적인 아키텍쳐를 구성하기 위해 다중 계층의 삼차원 아키텍쳐 구조를 제시하고, 개발자간 의견 충돌 부분을 사이클로 정의하여 이를 관리하는 방안을 제시한다.
아키텍쳐 기반 컴포넌트 검색 시스템을 통해 검색된 아키텍쳐들과 컴포넌트들 또한 사용자들간의 협업을 통해 적절하게 조정되어야 한다. 이를 위해 공동 작업을 위한 그룹 관리 기능을 제공하는 세션 관리자를 두었으며[3], 아키텍쳐와 컴포넌트 명세에 버전 정보를 추가함으로 사용자들간 공동 작업에 참조를 제공한다. 버전 정보를 기반으로 새롭게 구성된 아키텍쳐들은 아키텍쳐 저장소와 컴포넌트 저장소[4]에 추가함으로써 이후의 공동작업이 좀 더 효율적으로 이루어질 수 있도록 한다.

2. 관련 연구 고찰

이 장에서는 기존의 아키텍쳐 기반 컴포넌트 검색 시스템에 대해 간략하게 설명하고, 소프트웨어 개발을 자동화된 방법으로 하기 위한 요구사항과 다중 사용자 환경에서 발생할 수 있는 문제점과 고려사항에 대해 알아본다.

2.1 아키텍쳐 기반 컴포넌트 검색

새로운 소프트웨어 개발을 위한 노력을 최소화할 수 있는 방법으로 소프트웨어의 기능성보다는 구조적인 관점에

서 개발자의 요구 사항과 일치하는 XML 기반의 아키텍처를 검색하는 아키텍처 검색 방법과 검색한 아키텍처에 필요한 컴포넌트를 검색하기 위한 변경된 시그니쳐 일치와 행위 일치 방법을 사용할 수 있다[5]. 그림 1 은 아키텍처 기반 컴포넌트 검색 시스템의 구조를 나타내고 있다.

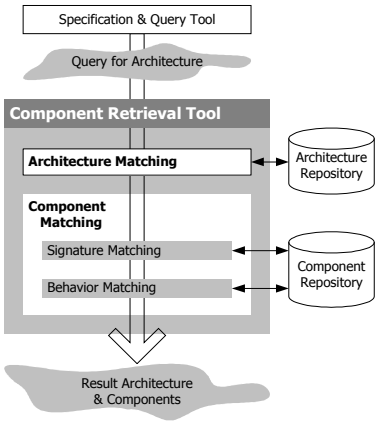

그림 1 아키텍쳐 기반 컴포넌트 검색 구조

그림 1 에서 개발자는 자신이 개발하고자 하는 소프트웨어 아키텍쳐를 질의 명세서의 형태로 입력하게 되면, 아키텍쳐 일치 모듈에서 아키텍쳐 일치를 수행하게 된다. 일치 수행 결과 후보 아키텍쳐들이 선정되면, 선정된 아키텍쳐 내의 컴포넌트 명세를 컴포넌트 일치 모듈로 전송하여 각각의 컴포넌트들을 시그니쳐 일치와 행위 일치 방법을 이용하여 검색하게 된다. 만일 개발자가 아키텍쳐 일치 결과 선정된 후보 아키텍쳐 중에서 원하는 아키텍쳐가 존재하면 검색을 종료할 수 있다.

2.2 소프트웨어 개발 자동화 도구의 요구사항

최근에 많은 조직들은 시스템 개발 환경을 향상시키기 위해 다양한 CASE 도구들을 사용하고 있다. CASE 는 시스템 개발 과정의 전체, 또는 일부를 자동화 시켜주는 도구로 정의한다. ISO(International Standards Organization)는 소프트웨어의 품질 평가를 위한 6 가지 주된 특징들을 정의하고 있다. 기능성(functionality), 신뢰성(reliability), 유지보수성(maintainability), 이동성(portability), 효율성(efficiency), 그리고 이용성(usability)이 이러한 평가 기준들이다[6]. CASE 도구들을 위해서도 이러한 품질 평가 기준들은 고려되어야 한다. 기능성은 시스템 개발 단계라기보다는 설계 분석 단계에서 고려되므로 CASE 도구를 위한 고려사항에서는 제외된다.

CASE 도구를 사용하면 결함율(defect ratio)을 감소시켜 신뢰성을 향상시킬 수 있으며, 유지 보수를 위한 노력을 감소시킬 수 있다. 그리고 개발의 효율성을 증대시킬 수 있으며, 시스템을 배우기 쉽게 한다. 서로 다른 플랫폼과 운영 체제간 이동성 또한 보장되어야 한다.

2.3 다중 사용자 환경에서의 고려사항

분산 환경에서의 소프트웨어 어플리케이션 개발을 위해서는 고려해야 할 사항이 많이 존재한다. 독립적으로 구현된 어플리케이션 컴포넌트의 경우 대부분 이 서로 다른 하드웨어와 플랫폼에서의 운영을 목적으로 하고, 서로 다른 프로그래밍 언어들로 개발되기 때문에 이질적인 경향을 가지고 있다. 이러한 컴포넌트들이 네트워크 상에 분산되어 있다면 위치 투명성, 보안, 동시성 제어, 동기화, 데이터 무결성, 그리고 트랜잭션을 위한 로킹(locking) 기법 등이 고려되어야 한다[7].

3. 다중 사용자 환경에서 아키텍쳐 기반 컴포넌트 소프트웨어 개발의 고려사항

이 장에서는 다중 사용자 환경에서 새로운 소프트웨어 개발을 위해 필요한 요소들과 개발 과정에 대해 알아본다. 그리고 개발자간 의견 조정을 위한 고려사항과 다중 계층 소프트웨어 아키텍쳐 구조에 대해 설명하고, 공동 작업을 위해 개발자들에게 참조를 제공하는 버전 정보 관리에 대해 알아본다.

3.1 소프트웨어 개발 과정

새로운 소프트웨어 개발을 위해 적합한 아키텍쳐와 컴포넌트를 검색하기 위해서는 우선 질의를 생성해야 한다. 다중 사용자 환경에서 질의를 생성하기 위해서는 사용자로부터의 요구사항들을 분석하여 적절하게 조정하는 과정이 필요하다. 의견 조정을 위해 이 연구에서는 사용자 요구사항들을 비교하고 공통 부분을 추출하여 하나의 질의를 생성하는 방법을 사용하였다. 사용자 요구사항 질의가 생성되면 이는 적절한 아키텍쳐와 컴포넌트 검색을 위해 검색시스템에 입력된다.

그림 2 는 이러한 아키텍쳐 기반 컴포넌트 소프트웨어 개발 과정을 그림으로 나타내었다.

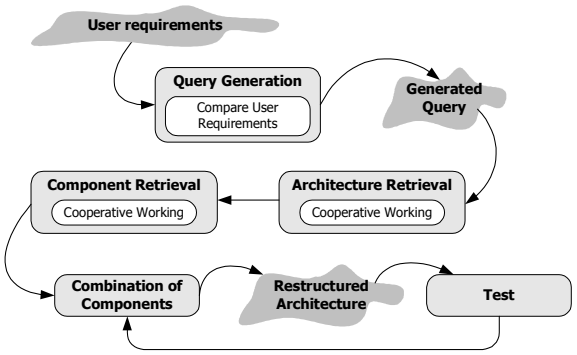

그림 2 소프트웨어 개발 과정

각각의 아키텍쳐와 컴포넌트는 버전 정보를 가지고 있으며, 사용자들은 이를 기반으로 원하는 아키텍쳐와 컴포넌트를 선택할 수 있다. 아키텍쳐 검색 과정을 통해 아키텍쳐들이 검색되면 사용자들은 적절한 아키텍쳐를 선택한다. 사용자들이 선택한 아키텍쳐가 서로 다를 경우 이에 대한 조정이 또한 필요하고, 이를 위해 선택된 아키텍쳐들을 비교하는 루틴이 필요하다. 메시지 기반의 아키텍쳐 비교 루틴은 이미 검색 시스템에 포함되어 있으므로 버전 정보를 사용하여 하나의 아키텍쳐를 생성해야 한다.

사용자들의 협의 과정을 통해 선택된 최종 버전의 아키텍쳐가 선택되면 아키텍쳐 재구성을 위한 컴포넌트 검색 루틴을 사용한다. 아키텍쳐 검색 과정에서처럼 컴포넌트 검색에서 또한 사용자들이 검색된 컴포넌트를 선택하여 적용한 새로운 아키텍쳐가 서로 다를 수 있으므로 이에 대한 조정과정이 필요하게 된다. 최종적인 아키텍쳐를 테스트하는 과정에서 사용자가 원하는 결과와 다른 결과가 생

성된 경우 다른 컴포넌트를 적용시켜 테스트하기 위해 컴포넌트 검색 과정으로의 피드백할 수 있어야 한다.

3.2 개발자간 의견 조정에서의 고려사항

다중 사용자 환경에서 소프트웨어 개발에 필요한 아키텍처 설계에 참여하는 개발자들은 해당 분야에서 전문가로 볼 수 있다. 그리고 이러한 각각의 개발자는 개발에 필요한 소프트웨어 아키텍처 구성에 대해 서로 다른 관점을 가지고 있을 수 있다.

서로 다른 관점의 개발자간 의견 조정을 위해 어떠한 협의 과정을 통해 하나의 소프트웨어 아키텍처를 구성할 수는 있지만, 구성된 아키텍처가 소프트웨어 개발에 적합하다고 할 수는 없다. 이러한 문제점은 단일 개발자에 의한 소프트웨어 아키텍처 설계에서도 동일하지만 개발자가 해당 분야에서 전문가라는 가정 하에서 크게 고려되진 않아왔다.

다중 사용자 환경에서 해당 분야의 여러 개발자들이 동시에 소프트웨어 아키텍처 설계 작업에 참여하게 되면 개인간의 견해 차이나 여러 가지 이유로 인하여 의견 충돌이 발생하게 된다. 이러한 경우 서로 다른 관점에서 설계된 아키텍처 설계의 특정 부분은 반영하고 다른 부분은 배제하는 방식의 접근 방법은 개발자들의 일반적인 의견을 반영하기 보다는 특정 개발자의 관점에만 집중될 수 있는 문제점을 가진다.

이 연구에서는 다중 사용자 환경에서의 새로운 소프트웨어 개발을 위한 아키텍처 설계 단계에서 개발자들의 의견 조정을 위한 의견 조정 관리 방식을 제안하고, 제안된 방식에 의해 생성된 아키텍처를 새로운 버전의 아키텍처로 정의하는 방안은 제시한다. 따라서 다른 개발자들간 공동작업 과정에서 발생할 수 있는 의견 충돌을 해결함으로써 모든 개발자의 의견이 최대한 반영된 소프트웨어 아키텍처를 설계할 수 있도록 한다.

3.3 다중 계층의 소프트웨어 아키텍처

다중 사용자 환경에서 새로운 소프트웨어를 개발하기 위해 각각의 개발자들은 우선 개발에 적합한 소프트웨어 아키텍처를 설계하게 된다.

이 연구에서 제시하는 설계 과정에서의 소프트웨어 아키텍처는 그래프 형태로 표현된 이차원 평면들이 계층적 구조를 갖는 삼차원 구조를 가진다.

최상위 계층의 평면은 컴포넌트와 컴포넌트간 연결자에 의해 설계된 소프트웨어 아키텍처를 나타낸다. 소프트웨어 아키텍처를 구성하는 각각의 컴포넌트는 하부단 평면에 있는 컴포넌트들의 조합인 복합 컴포넌트로 표현될 수 있다. 따라서 최상위 계층의 소프트웨어 아키텍처를 설계하기 위해서는 아키텍처가 하부단의 복합 컴포넌트에 대한 연결 정보를 가져야 한다. 이러한 연결 정보는 아키텍처들과 컴포넌트들의 명세가 XML 을 기반으로 명세되어 있으므로 쉽게 유지될 수 있다.

그림 3 은 이 연구에서 제시하는 다중 계층의 소프트웨어 아키텍처의 구조를 보여주고 있다.

설계 과정에서의 소프트웨어 아키텍처는 단면적으로는 트리의 형태를 가지며, 평면적으로는 그래프의 형태를 가지는 다차원 그래프로 정의될 수 있다. 소프트웨어 아키텍처에서 각 계층의 의미는 상위 계층을 구성하는 컴포넌트들은 하부 계층의 컴포넌트들의 조합인 복합 컴포넌트로 구성된다는 사실만을 표현하므로 재구성을 통해 하나의 평면에 관계를 표현할 수도 있다.

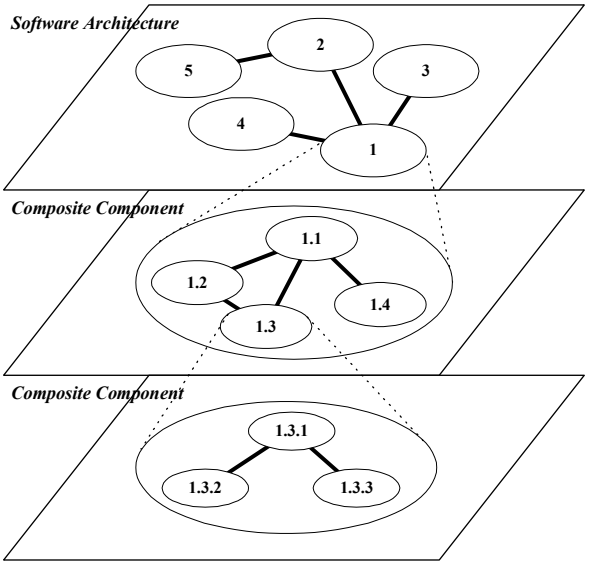

그림 3 다중 계층 소프트웨어 아키텍처

소프트웨어 아키텍처가 평면화 되면 개발자 단위의 아키텍처 설계 과정은 종료하게 된다. 각각의 개발자는 평면 형태의 소프트웨어 아키텍처를 하나씩 생성하고, 생성된 아키텍처들은 시스템에 의해 하나의 소프트웨어 아키텍처로서 표현됨으로써 서로 다른 관점에서 설계된 아키텍처들을 정의할 수 있어야 한다. 즉 각각의 개발자 관점에 따라 설계된 아키텍처간 이동을 지원하는 연결된 형태의 유일한 소프트웨어 아키텍처로 구성되어야 한다.

그림 4 는 그림 3 에 나타낸 다중 계층의 소프트웨어 아키텍처를 평면화한 아키텍처를 나타낸 것이다.

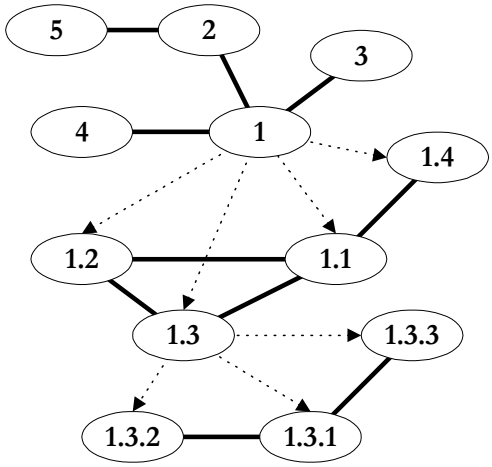

그림 4 평면화된 소프트웨어 아키텍처

3.4 의견 조정 과정

다중 사용자 환경에서의 각각의 개발자 관점에 따라 설계된 소프트웨어 아키텍처들을 개발자들의 의견을 존중한 하나의 복합 소프트웨어 아키텍처로 구성하기 위해서는 상위 계층과 하위 계층간 순서, 또는 다중 단계에 걸쳐 순서가 바뀐 모든 관계들을 고려해야 한다. 이 연구에서는

이러한 계층간 순서가 바뀌어 정의되는 관계를 사이클이라 정의한다.
소프트웨어 아키텍쳐 설계 과정에서 이러한 사이클이 발생했을 경우 상위 계층과 하위 계층간의 순서 관계를 파악하기 힘든 경우가 발생할 수 있다. 따라서 이 연구에서는 이러한 사이클이 발생한 경우 저작자의 의견을 모두 수렴하여 복합적인 형태의 소프트웨어 아키텍쳐를 구성한다.
그림 5 는 두 명의 개발자 관점에서 설계된 서로 다른 소프트웨어 아키텍쳐를 보여주고 있다.

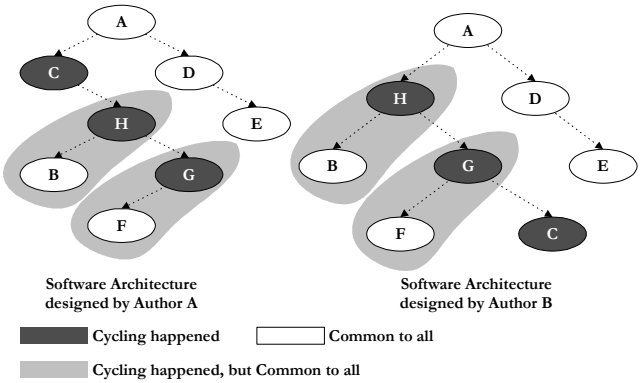

그림 5 서로 다른 관점에서의 소프트웨어 아키텍쳐 설계

각각의 개발자의 설계된 소프트웨어 아키텍쳐에서 공통적으로 정의된 부분은 연관 관계가 일치하는 하나의 평면으로 구성한다. 사이클이 발생하는 부분은 서로 다른 평면으로 분리하고, 공통적으로 정의된 평면 계층에 연결 정보를 추가 정의함으로써 개발자들의 의견이 모두 반영될 수 있다. 따라서 개발자들의 아키텍쳐 설계 과정을 모두 수용할 수 있는 하나의 소프트웨어 아키텍쳐를 구성할 수 있다.
그림 6 은 그림 5 의 서로 다른 관점에서 설계된 두 개의 아키텍쳐를 하나의 소프트웨어 아키텍쳐로 구성된 예를 그림으로 보여주고 있다.

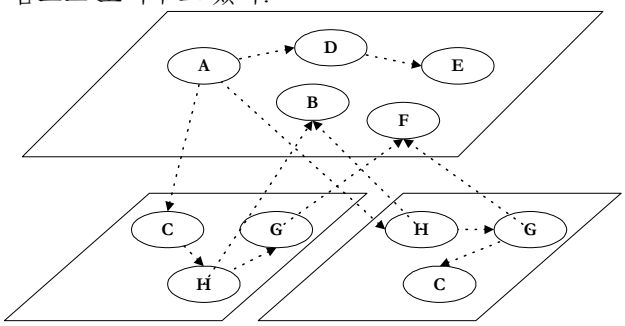

그림 6 복합 소프트웨어 아키텍쳐

3.5 아키텍쳐 버전 관리

다중 사용자 환경에서 각각의 개발자의 협의 과정을 통해 하나 이상의 소프트웨어 아키텍쳐가 설계될 수도 있다. 이 때 설계된 서로 다른 아키텍쳐들을 따로 관리한다는 것은 크게 의미를 지니지 못하며, 저장 공간의 낭비 또한 크게 된다. 따라서 이 연구에서는 아키텍쳐를 다중 계층의 평면으로 관리하는 방법과 사이클 발생에 대한 해결책을 제시하였다.
모든 개발자의 의견이 모두 반영된 소프트웨어 아키텍쳐의 관리를 위해 이 연구에서는 아키텍쳐와 컴포넌트 저장소의 사용자 인터페이스 명세에 버전 정보를 추가, 관리함으로써 사용자들의 공동 작업에 참조를 제공할 수 있도록 한다.
XML 로 표기되는 아키텍쳐 및 컴포넌트 명세서는 버전 정보 관리를 위해 명세서를 작성한 사람을 표기하는 <author> 태그와 명세서가 작성된 날짜를 표기하는 <date> 태그를 가진다. 새로운 버전 정보를 가진 소프트웨어 아키텍쳐는 아키텍쳐 저장소와 컴포넌트 저장소에 추가됨으로써 이후의 소프트웨어 개발을 위해 사용될 수 있도록 한다.

4. 시스템

이 장에서는 다중 사용자 환경에서 새로운 소프트웨어 개발을 위한 간단한 시스템을 구성하고, 사용자 그룹 관리를 위한 세션 관리자에 대해 설명한다.

4.1 시스템 구성

다중 사용자 환경에서 새로운 소프트웨어를 개발하기 위해서는 우선 사용자의 요구사항들을 수집해야 한다. 하지만 각각의 사용자의 의견은 서로 다를 수 있으며 이러한 의견들을 하나의 소프트웨어 개발을 위해 조절하는 과정은 필수적이라 할 수 있다. 이러한 과정을 통해 생성된 하나의 요구사항이 질의 형태로 주어지면 소프트웨어 개발을 위한 적당한 아키텍쳐와 컴포넌트를 검색할 수 있는 루틴이 필요하다.
아키텍쳐 기반 컴포넌트 검색 시스템은 기존에 이미 개발된 시스템을 사용하였다. 검색된 소프트웨어 아키텍쳐에 대한 적절한 선택을 위해 사용자들의 협의과정이 필요하게 되며, 아키텍쳐 재구성을 위한 컴포넌트 검색에서도 동일한 의견 조절 과정은 필요하다. 전체적인 과정을 통해 생성된 아키텍쳐는 최종적인 테스트 과정을 통해 검증받게 된다. 이 연구에서 제안하는 다중 사용자 환경을 고려한 소프트웨어 개발 시스템 구조를 그림 7 을 통해 나타내었다.

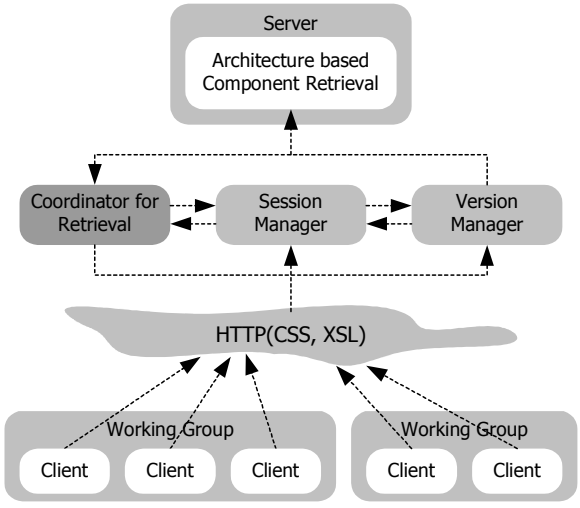

그림 7 시스템 구조

4.2 세션 관리자

새로운 소프트웨어 개발을 위한 공동작업을 위해서는 사용자를 그룹별로 관리할 수 있다. 각각의 사용자 그룹 관리를 위해 이 연구에서는 서버측의 검색 시스템에 세션

관리자를 두고 있다. 세션 관리자는 질의 생성을 위해 사용자들의 요구사항들을 분석하는 단계와 그리고 아키텍쳐 검색 루틴, 컴포넌트 검색 루틴에 각각 위치하게 된다.
세션 관리자는 공동 작업을 위한 그룹에 쉽게 참여하거나, 정보를 얻을 수 있는 환경을 제공함으로써 새로운 응용 개발에 쉽게 접근할 수 있는 방법을 제공한다. 사용자는 세션 관리자를 통해 생성된 그룹들에 대한 정보를 쉽게 얻을 수 있다. 현재 어떤 그룹이 생성되어 있는지, 임의의 한 그룹에 누가 참여하고 있는지 등에 대한 정보를 얻을 수 있으며, 그룹을 생성하거나, 참여한 그룹을 떠나거나, 생성한 그룹을 삭제할 수 있다.
세션 관리자는 사용자의 세션에 대한 참여, 탈퇴 기능 등을 쉽게 해주는 도구이다. 따라서 여러 참여자에 대한 정보를 참여자가 그룹에 참여하는 동안 계속 유지할 수 있어야 한다. 세션 관리자는 참여자에 대한 정보를 개별적으로 유지한다. 세션 관리자는 이러한 정보들을 일관성 있게 관리하여야 한다. 세션 관리자는 외부 모듈과의 인터페이스로 사용된다. 새로운 그룹의 생성이나 그룹의 소멸 등의 요구에 대해 그룹 관리자를 관리하며, 외부의 요구를 해석하여 통지 모듈을 통해 사건을 등록하고, 통지 모듈을 통해 전송된 외부 요구를 처리한다.

5. 결론 및 향후 연구 과제

이 연구에서는 기존의 아키텍쳐 기반 컴포넌트 검색 시스템을 다중 사용자 환경으로의 확장을 위해 고려하여야 할 점을 제시하였다. 다중 사용자 환경에서의 새로운 소프트웨어 개발을 위한 소프트웨어 아키텍쳐 설계 단계에서의 개발자들의 의견을 최대한 반영할 수 있는 방법으로 다중 계층의 소프트웨어 아키텍쳐 구조를 제시하였다. 다중 계층의 소프트웨어 아키텍쳐설계에서 개발자간 의견 충돌이 발생한 부분은 다른 계층의 평면으로 구성하여 공통적인 평면으로의 연결과 순서를 유지시킴으로써 일반적으로 적용할 수 있는 소프트웨어 아키텍쳐를 구성할 수 있다.
전체 시스템에서의 다중 사용자 환경에 필수적인 개발자간 공동작업과 공동작업에서의 그룹 관리를 위해 세션 관리자를 검색 시스템에 추가하여 협업을 위한 새로운 환경을 구성하였다.
그리고 저장소에 저장되어 있는 아키텍쳐와 컴포넌트의 사용자 인터페이스 명세에 버전 정보를 추가함으로써 개발자간 이후의 공동 작업을 위한 참조를 제공한다. 협업을 통한 새로운 소프트웨어 아키텍쳐가 설계되면 이를 아키텍쳐 저장소와 컴포넌트 저장소에 버전 정보와 함께 추가함으로써 이후의 소프트웨어 개발 과정에 사용할 수 있도록 하였다.

참고문헌

[1] S. Konomi, Y. Yokota, K. Sakata, Y. Kambayashi, “Cooperative View Mechanisms in Distributed Multiuser Hypermedia Environments,” *Proc. 2nd IFCIS International Conference on Cooperative Information Systems*, 1997
[2] Y. S. Lee, K. S. Yoon, C. J. Wang, “Component Retrieval Based on Architecture for Reuse,” *Proc. International Conference on Software*, 2000, 게재 예정
[3] 김남용, 이승근, 왕창종, “CORBA 기반 멀티미디어 응용을 위한 공동작업 서비스 설계,” *한국정보처리학회 정보처리논문지 제 5 권 1 호, PP. 72*, 1998
[4] C. D. Ahn, S. K. Kim, C. J. Wang, “XML based Component Specification Repository,” *Proc. International Conference on Software*, 2000, 게재 예정
[5] 이윤수, 윤경섭, 왕창종, “재사용을 위한 XML 기반 소프트웨어 아키텍쳐 명세 언어,” *한국정보처리학회 정보처리논문지*, 제 7 권 4 호, 2000 게재 예정
[6] G. Low, V. Leenanuraksa, “Software Quality and CASE Tools,” *Proc. Software Technology and Engineering Practice*, 1998
[7] W. Emmerich, N. Roodyn, “Distributed objects,” *Proc. International Conference on Software Engineering*, 1999