CBD 프로세스 전개를 위한 설계 패턴을 이용한 컴포넌트 아키텍쳐

차정은[O*] 양영종* 김행곤**
* 한국전자통신연구원, 소프트웨어공학부 컴포넌트 S/W 연구팀
** 대구가톨릭대학교 컴퓨터공학과 소프트웨어공학 연구실
{mary2743, yangyj, shinsk}@etri.re.kr, hangkon@cuth.cataegu.ac.kr

Component Architecture Using Design Pattern for Evolution of CBD Process

Jung-Eun Cha[O*] Young-Jung Yang* Hang-Kon Kim**
* Software Engineering Department ETRI-Computer & Software Technology Laboratory
** Dept. of Computer Engineering, Catholic University of Daegu

요 약

CBD(Component Based Development)가 이미 완성된 소프트웨어 모듈인 컴포넌트들을 특정 목적을 위해 배치하고 조립함으로써 품질 보증된 고생산성의 재사용을 가능하게 됨에 따라 , 최근 IT 비즈니스 솔루션 도출을 위한 핵심적인 기술 전략으로 인식되고 있다. 따라서 영역에 적절한 컴포넌트의 추출과 개발 및 응용 시스템으로의 전개를 위한 컴포넌트들의 배치와 조립 등을 포함하는 일련의 CBD 프로세스를 실제화 하기 위해서는 아키텍처 기반의 접근과 제어가 매우 중요하다. 하지만 대부분의 CBD 연구는 비즈니스 로직들의 기능성을 그룹화한 컴포넌트의 생산에 초점을 두고 있지 컴포넌트간의 상호작용을 명시하는 아키텍쳐 정보의 관리를 간과함으로써 컴포넌트의 조립과 통합을 통한 CBD 시스템으로의 전개는 극히 어려운 실정이다.

따라서, 본 논문에서는 CBD 프로세스 상에서 아키텍쳐 접근의 중요성과 전개 방법을 살펴보고, 컴포넌트 프레임워크(Component Framework)를 위한 아키텍처의 계층을 새롭게 재정의 하며 그 의미를 설명한다. 이를 위해 영역 분석 및 설계 정보를 선언적으로 명시하는 수단이며, 도메인 내에서 특정 서비스를 제공하는 컴포넌트간의 일반적 결합 방식의 규정하기 위한 수단으로서 설계 패턴을 이용하여 서비스 아키텍쳐 패턴를 정의하고 실 예로 적용한다.

1. 서론

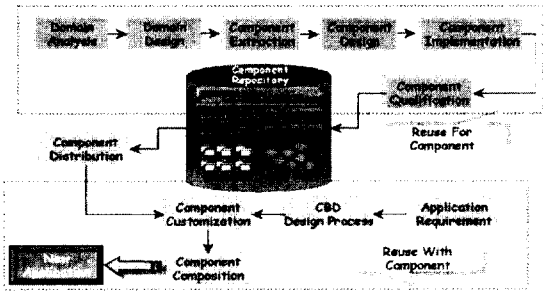
컴포넌트 기반 소프트웨어 개발(CBD : Component Based Development)은 기능적인 구현 함수들을 조작하고 전역 변수를 관리하는 프로그래밍 개념이 아니라 이미 완성된 소프트웨어 부품들을 조립, 통합함으로써 품질 보증된 고생산성의 재사용을 추구한다. 따라서 의미있는 로직들을 잘 정의된 인터페이스를 통해 패키징하고 이것들을 별개의 목적으로 조립함으로써 특정 목적을 갖는 응용 시스템으로 전개하는 것이다. 또한 CBD 프로세스는 다양한 형상의 기존 컴포넌트들의 조립과 개조 등과 같은 일련의 가공을 통해 또 다른 컴포넌트 프로덕트를 생산해 내는 모든 과정으로, 컴포넌트의 생산과 품질보증, 개조 및 조립, 커스터마이징의 단계로 구성된다.

한편 소프트웨어 아키텍처는 컴포넌트의 생산과 배포, 획득 조립 등의 컴포넌트 라이프사이클에 관련된 CBD 행위들에 대한 표준 규칙과 지침을 제시한다. 컴포넌트의 조립으로 완성된 시스템의 품질은 보증된 경험적 시나리오가 바탕이 되느냐에 따라 큰 차이를 나타낸다. 영역 전문가와 시스템 숙련가에 의해 검증된 공통적인 솔루션을 정규화한 패턴은 CBD에서 컴포넌트간의 상호작용을 표현할 수 있다. 또한 컴포넌트간의 상호작용은 소프트웨어 아키텍처의 컨넥터로서 구체화 될 수 있다. 그러므로 영역 분석 및 설계 정보의 구체적인 규칙들을 구현 단계의 가공물로의 유연한 연계가 가능하도록 설계 패턴을 통해 아키텍처 상에서 정의하고, 이 패턴들은 특정 영역을 위한 CBD 시스템의 아키텍처로서 대체할 수 있다. 그러므로 CBD 프로세스 전개를 위해서는 종래의 소프트웨어 아키텍처의 개념을 확장하여 설계 패턴을 기반으로 컴포넌트간의 상호작용을 정의한 새로운 접근이 필요하다. 본 논문에서는 CBD 전개를 위한 아키텍처 기술의 적용 방향을 살펴보고 컴포넌트 프레임워크 관점에서 서비스 지향의 새로운 아키텍처를 설계 패턴을 이용하여 정의한다.

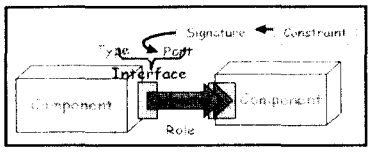
2. 관련 연구

2.1 CBD 프로세스

CBD 프로세스는 기존 컴포넌트들을 다양한 형상으로 조립하고 개조(Adaptation)하며 연계(Wrapping)함으로써 구축될 수 있는 모든 가공물로의 접근을 위한 전역적인 관점을 표현한다[1]. 본 논문에서는 (그림 1)과 같이 합성(Composition)에 의해 컴포넌트를 생산하는 과정과 컴포넌트를 이용하여 응용을 구축하는 두 가지 과정으로 세분화하였다. 이는 또한 재사용의 관점에서 컴포넌트 통합 라이브러리 시스템인 저장소를 중심으로 CBD를 위한 재사용(Reuse For CBD)과 CBD에 의한 재사용(Reuse With CBD)으로 분류된다.



(그림 1) CBD 프로세스



(그림 2) 아키텍처의 개념

2.2 소프트웨어 & 컴포넌트 아키텍처

소프트웨어 아키텍처는 영역에 대해 보편적인 관점에서 설정된 정형화된 구조적 틀을 제시하고, 이들 틀 내에 포함되는 구성 요소들의 상호 관계성을 규정하는 기술들의 총체적인 의미이다[2]. 따라서 (그림 2)와 같이 아키텍처의 개략적인 구성 개념도는 구조적인 위상에 대한 모델을 이루는 컴포넌트와, 타입 및 한정성을 포함하는 인터페이스를 통해 컴포넌트간의 관계성을 행위적 관점에서 모델링하는 컨넥터로 구조화된다. 한편 컴포넌트 아키텍처는 기본 인터페이스를 공유하고 하위 계층 컴포넌트 조합에 의한 상위 계층의 독립적 응용과 그룹 컴포넌트 형성으로 응용 구축을 위한 비즈니스 프로세스 제공함으로써 컴포넌트의 생산과 배포 조립의 지침을 마련한다. 따라서 컴포넌트 아키텍처 상에서 추상성에 따른 수평적 컴포넌트 계층과 동일 영역의 규모에 따른 수직적 컴포넌트의 범주가 형성된다. 요약하면, 소프트웨어 아키텍처는 간단히 컴포넌트와 이들의 적절한 배치를 위한 규칙의 집합으로 고려될 수 있으며 컴포넌트 아키텍처는 CBD 응용을 위한 아키텍처 스타일로서 이해할 수 있다.

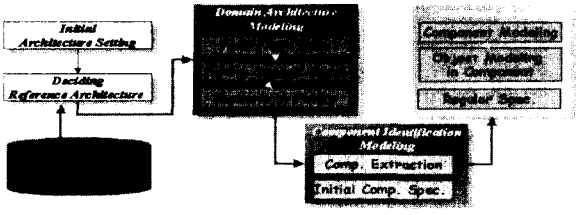
2.3 설계 패턴

재사용의 플랫폼 환경과 응용 도메인의 한정성에 대한 문제는 패턴을 통해 극복할 수 있다. 특히 설계 문제의 추상화와 특정 영역에 대해 검증되어진 공통적인 해결책을 구성 요소간의 관련성을 통해 정의함으로써 공통 도메인의 응용 구축에 대한 자동화된 아키텍처 생성을 가능하게 하는 설계 패턴은 신뢰성 있는 설계 규칙들을 확보하고 이를 바탕으로 개별 개발자의 구현 특성을 제공한다. 이는 추상화된 재사용의 개념에서 자신의 의도를 반영하고 자동화 개발 프로세스 사용에 의한 아키텍처 공유로 표준화된 결과물들을 획득하도록 한다[3].

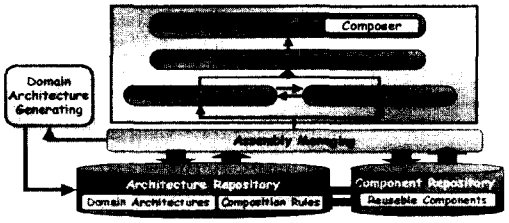
3. CBD에서의 아키텍처 기술의 접근

3.1 개념

CBD 관점에서 소프트웨어와 컴포넌트 아키텍처의 통합 개념은 설계에 국한된 문제가 아닌 시스템의 전체 라이프사이클을 대상으로 고려, 적용된다. 그리고 컴포넌트 라이프사이클이 소프트웨어 개발 라이프사이클의 대체 개념으로 인식되는 CBD 환경을 위한 소프트웨어 아키텍처 스타일에 중점을 둔다. 또한 컴포넌트 재사용을 위한 통합 아키텍처로서 컴포넌트 아키텍처 프레임워크로의 전개를 목표로 한다. 이는 소프트웨어 아키텍처가 CBD 기반의 컴포넌트 아키텍처가 결합한 개념으로 즉, 컴포넌트들이 서비스 기반의 규칙들을 기반으로 컴포넌트 시스템 구축을 동적으로 통합, 조립될 수 있는 통합 환경이다[4,5].



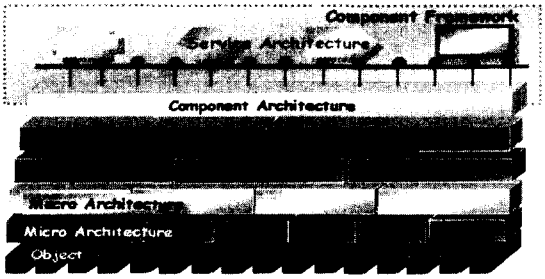
(그림 3) For Reuse CBD를 위한 아키텍처 접근



(그림 4) Reuse With CBD

3.2 기술 분류

아키텍처 기술은 CBD 프로세스의 재사용 관점에 따라 기반 기술과 활용 기술로 구분되며 각각 (그림 3)과 (그림 4)와 같은 접근 체계를 가진다. For Reuse 접근에서는 도메인 공학과 재공학을 바탕으로, 공통성 유지 및 특정 요소의 상반 관계를 만족시키는 컴포넌트 시스템을 위한 아키텍처 추출, 명세화, 그리고 카탈로깅을 중심으로 확장, 정규화 작업의 반복 처리가 핵심이 된다. With Reuse의 아키텍처 접근에서는 아키텍처 정보를 구현 시스템으로 전개하기 위한 컴포넌트의 조립에 초점을 둔다. 따라서 도메인 한정적인 아키텍처를 생성, 관리하고 특정 영역 아키텍처 내에서 컴포넌트들을 배치하고 합성하고 개조하는 규칙을 정의하며 이를 바탕으로 도메인 지향적인 CBD 시스템 생산을 목적으로 한다.

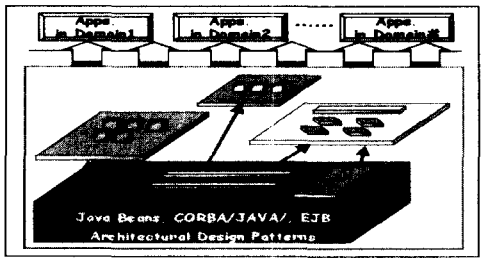


(그림 5) CBD 환경에서의 아키텍처 단계

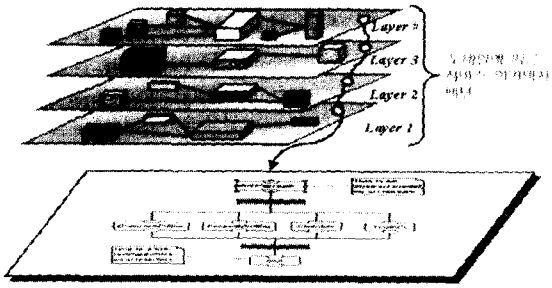
4. 컴포넌트 아키텍처 프레임워크

4.1 개념

CBD 환경에서 설계 수준의 영역 정보의 획득을 목적으로 하는 종래의 아키텍처 개념과, 로직들을 포함한 컴포넌트의 조립에 의한 구현 단계 생산물의 창출 개념이 결합함에 따라 본 논문에서는 (그림 5)와 같은 수정된 아키텍처 단계를 정의하였다. 상위 2계층인 컴포넌트 아키텍처와 상위의 서비스 아키텍처는 전형적인 소프트웨어 아키텍처에 컴포넌트의 적용에 의해 도출한 것이다. 이 계층에서 컴포넌트들의 영역별 조립 시나리오를 제공함으로써 컴포넌트들 간의 합성, 개조, 연계 등을 정규화된 정보 규칙을 바탕으로 보증할 수 있다.



(그림 6) 컴포넌트 아키텍처 프레임워크의 개념



(그림 7) 패턴 정보에 의한 컴포넌트 결합 아키텍쳐의 개념도

즉, 컴포넌트 아키텍쳐 프레임워크의 핵심은 미리 정의된 컴포넌트 구성 설계 정보를 수행이 준비된 형태로 지원함으로써 사용자는 단지 각 컴포넌트들의 결합 방식을 결정하면 된다. 그러므로 컴포넌트 아키텍쳐 프레임워크에서는 준비된 아키텍쳐 즉, 영역 아키텍쳐를 구성하고 명세화 하기 위한 기술과 개별 컴포넌트들을 특정 영역에 따라 합성하고 조립하는 커스터마이징 기술이 중심이 된다.

(그림 6)은 컴포넌트 아키텍쳐 프레임워크의 개념을 도식화한 것이다.

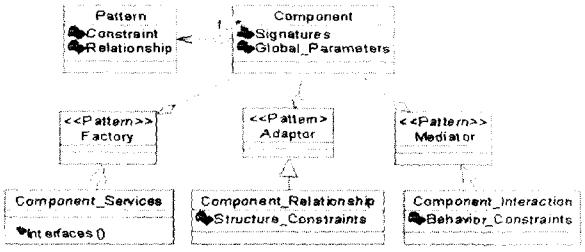
4.2 설계 패턴의 적용

컴포넌트 결합을 위한 영역별 아키텍쳐를 패턴으로 제공하고 이를 배치하고 계층화함으로써 CBD 시스템으로의 전개를 위한 개념을 (그림 7)에 나타내었다. 본 논문에서는 이를 위해 3가지 종류의 패턴 유형으로 정의하였다:

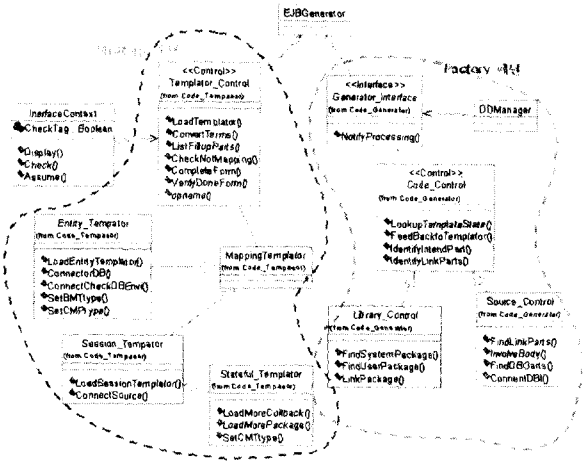
① **메타 패턴 : 영역 아키텍쳐 패턴의 서술 방법 제공**
② **아키텍쳐 패턴 : 영역 내의 공통적인 시스템 구조 정의**
③ **상호연결 패턴 : 영역 내에서 특정 목적에 대한 솔루션을 제공하기 위한 컴포넌트들의 상호작용 시나리오 제공**

본 논문에서는 특정 영역에서 특정 목적을 수행하기 위한 시나리오를 제공하기 위한 패턴을 "서비스 아키텍쳐 패턴"이라고 명명하였다. 따라서 위의 3가지 타입의 패턴들이 특정 영역에서 컴포넌트들의 특정 결합 양식을 나타내는 시나리오로서 서비스 아키텍쳐 패턴을 구성한다.

일반적으로 참조되는 분석, 설계 패턴에는 인터페이스를 중심으로 컴포넌트들간의 독립적인 상호 연계를 위한 패턴들, 즉, Bridge 패턴, Factory Method 패턴, Adapter 패턴, Mediator 패턴 등이 존재한다. 그러므로 본 논문에서는 이들 참조 패턴들을 기본 템플리트로 하고, 패턴 형식 중 솔루션 부분을 상세히 서술함으로써 구현 영역으로의 전개를 용이하도록 영역 컴포넌트들 간의 상호작용을 위해 패턴을 확장, 조합하였다. 먼저, 컴포넌트들간의 상호작용을 위한 패턴의 구성 및 표현 양식이 (그림 8)과 같이 정의하였다.



(그림 8) 컴포넌트 상호작용을 위한 아키텍쳐 패턴의 구성 및 표현



(그림 9) 서비스 아키텍쳐 패턴의 상호작용 패턴 예 (EJB 생성기)

패턴들은 자체의 시그네이처와, 연관 컴포넌트들 사이의 전역 변수를 가지는 다수 개의 컴포넌트로 구성된다. 그리고 컴포넌트는 Facade 패턴을 통해 제어, 조절됨으로써 단일의 추상화된 인터페이스를 제공한다. 또한 구조적 제약 조건을 가지는 Component_Relationship 클래스와 영역 내의 동적 통합에 대한 제한을 갖는 Component_Interaction 클래스를 포함하며 이들은 각각 Strategy와 Mediator 패턴을 확장하고 그 특성을 구현한다. 그 다음으로, 개별 영역에 맞는 컴포넌트들의 조합, 즉, 영역의 특정 서비스를 제공하기 위해 프로세스 패턴(Process Pattern) 아키텍쳐 스타일의 개념과 골격의 개념을 구체화하였다. (그림 9)는 컴포넌트의 결합 시나리오를 나타내는 한 예를 표현한 것으로 (그림 8)에서 정의한 아키텍쳐 패턴에 실제 EJB 컴포넌트 코드[6]의 자동 생성 시스템을 적용, 상호작용 패턴의 예로서 생성한 것이다.

5. 결론

본 논문에서는 CBD 관점에서 소프트웨어와 컴포넌트 아키텍쳐가 통합된 개념으로 컴포넌트 아키텍쳐 프레임워크를 정의하였다. 이는 설계에 국한된 문제가 아닌 시스템의 전체 라이프사이클을 대상으로 확장되고, 컴포넌트들이 특정 목적을 위해, 특정 영역의 서비스 규칙에 따라 통합할 수 있는 개념이다. 또한 CBD를 위한 새로운 컴포넌트 아키텍쳐 단계를 정의하고, 컴포넌트 프레임워크 상에서 컴포넌트들의 상호작용 규칙들을 설명하기 위해 서비스 아키텍쳐 패턴을 정의하고, 나아가 실제 예를 적용하였다. 본 논문을 통해 제시한 것과 같이 영역별 정보 단위로서 기존 패턴을 확장, 재정의 한 서비스 아키텍쳐 패턴을 이용함으로써 컴포넌트 조립을 위한 지침을 마련할 수 있을 수 있으리라 기대된다.

향후 동적인 컴포넌트 결합을 명시하기 위해 프레임워크 측면에서 다형성이 고려된 자동화된 CBD 전개로의 접근이 요구된다.

[참고문헌]

[1] Peter Herzum, *Business Component Factory*, John Wiley&Sons, 2000
[2] Mary Shaw, Software Architecture, Prentice Hall, 1996
[3] James O. Coplien, *Pattern Languages of Program Design, Vol2*, Addison-Wesley, 1996
[4] Wolfgang Pree, *Design Patterns for Object-Oriented Software Development*, Addison-Wesley, 1995
[5] Stephen S. Yau, "Integration in CBSD Using Design Patterns", *Computer Software & Applications Conference COMPSAC 2000*, pp.369 -374, 2000
[6] *J2EE Developer's Guide (for J2EE SDK version 1.2)*, Sun Micro System, 2000, URL : http://java.sun.com/j2ee/docs.html