UML을 이용한 소프트웨어 아키텍처 모델링에 관한 연구

문 성 준(文 聖 俊), 김 병 기(金 炳 基)
전남대학교 전산학과
전화 : (062) 530-0107 / 팩스 : (062) 530-0108

A Study on Software Architecture Modeling using UML

Sung Jun Mun, Byung Ki Kim
Dept. Computer Science of Chonnam National University
E-mail : sjmun@superse.chonnam.ac.kr, bgkim@chonnam.chonnam.ac.kr

Abstract

In this paper, we propose the method to express various factors for software architecture modeling effectively using UML extension mechanism. Furthermore, we show usefulness of proposed method by presenting several modeling techniques and their examples.

I. 서론

소프트웨어 아키텍처에 대한 연구는 컴포넌트 기반 소프트웨어 개발(CBSD : Component Based Software Development) 방법론의 등장과 함께 새로운 국면을 맞이하게 되었다. 소프트웨어가 대형화되고 복잡해짐에 따라서 컴포넌트 기반 소프트웨어 개발은 시스템을 컴포넌트들의 조합만으로 완성함으로써 소프트웨어 개발의 새로운 패러다임으로 생각되어 지고 있다. CBSD를 이루는 관련 기술들 중 하나인 소프트웨어 아키텍처[1]는 시스템을 구성하는 컴포넌트와 그들 사이의 상호작용을 기술해 줌으로써 컴포넌트들로 조립된 전체 시스템이 원활히 역할을 수행하는데 필수적인 요소라고 생각된다. 소프트웨어 아키텍처는 전체 시스템의 구성을 개발의 초기 단계에서 기술함으로써 설계 단계의 결정(design decision)을 위한 지침을 제공하고, 개발자들의 시스템 이해를 도모하고, 시스템 구성에 대한 분석을 가능하게 한다. 이러한 소프트웨어 아키텍처를 표현하기 위한 다양한 방법들 중 대표적인 것이 아키텍처 기술언어(ADL : Architecture Description Language)[2]와 객체지향 소프트웨어 개발 모델링 방법의 표준이라 할 수 있는 UML(Unified Modeling Language) 표기법을 사용하는 것이다.

현재까지 많은 아키텍처 기술언어들이 제안되었으나, 각각의 기술언어들은 각각의 적용 영역에 따라 고유한 특성을 가지며 발전해 왔고, 이들을 이용한 아키텍처 기술은 많은 경우 정형 명세언어에 기반하고 있기 때문에, 정형기법에 관한 지식이 부족한 일반 개발자의 경우 명세에 어려움을 가질 수 있다. 이러한 것을 해결하기 위해서 보다 범용적이고 시각적인 기술언어의 제안이 요구된다[3].

최근 들어, 컴포넌트 및 아키텍처 명세의 표기법으로 UML을 사용하려는 연구가 활발히 일어나고 있다. 이는 UML이 제공하는 시각적 표기법을 통한 명세의 가독성과 표준화 등의 장점에 기인한 것이다.

본 논문의 구성은 우선 UML 및 UML 확장 메커니즘, 소프트웨어 아키텍처 모델링을 위한 핵심요소 및 세부 구성요소, 핵심요소 표현과 관련된 UML 다이어그램에 대해 2장에서 관련연구로서 알아보고, 3장에서는 이러한 소프트웨어 아키텍처 모델링을 위한 다양한 요소들을 UML 확장 메커니즘을 활용하여 효과적으로 표현하기 위한 방법과 더블어 몇 가지 모델링 기법 및 예제를 보임으로써 제안된 방법의 효용성을 보인다.

마지막으로 4장에서는 본 연구를 정리하여 결론을 맺고 추후 연구방향을 제시한다.

II. 관련연구

1. UML 및 UML 확장 메커니즘

UML은 97년 OMG(Object Management Group)에 의해 객체지향 모델링 언어의 산업 표준으로 승인되어 현재 널리 사용되고 있는데, 배우기 쉽고 확장하기도 쉬우며 다양한 표기법을 지니고 있어서 여러 방법론의 특징을 잘 표현할 수 있다. 또한, 표준화된 언어이기 때문에 분석, 설계, 구현 과정에서 발생하는 개발자간의 의사소통의 불일치를 해소할 수 있으며, 모델링에 대한 표현력이 강하고 비교적 모순이 적은 논리적인 표기법을 가진 언어이다. 하지만, 소프트웨어 모델링에서 모든 상황에서의 어떤 의미를 표현하기 위해서는 하나의 언어로는 부족하다. UML 또한 마찬가지로 어떤 미묘한 의미의 차이를 표현하기에는 부족한 면이 있다. 이를 위해 UML에서는 분석·설계 정보를 보다 정확하게 전달하고 약간의 변형이나 확장으로 보다 나은 의사 소통을 가능하게 하기 위한 목적으로 세가지 확장 메커니즘(extension mechanism)을 제공한다.

이로써 소프트웨어 개발에서 새로운 이슈들(issue)을 표현하기 위해 새로운 구성요소를 추가하고, 새로운 속성과 의미를 지정할 수 있다.

다음의 (표 1)에는 UML에서 제공하는 세가지 확장 메커니즘에 대한 설명이 나와 있다.

종 류	의 미
스테레오 타입 (Stereotypes)	• UML 어휘의 확장 • 새로운 종류의 구성요소를 생성
꼬리표 값 (Tagged Values)	• UML 속성을 확장 • 구성요소의 명세서에 새로운 정보를 추가 생성
제약 (Constraints)	• UML 의미를 확장 • 새로운 규칙의 추가 및 기존 규칙 변경

(표 1) UML 확장 메커니즘

따라서, 새로운 구성요소를 모델링하는 과정은 우선 기본 구성요소로 표현 가능한가를 파악한다. 만약 사용할 구성요소가 없을 경우 스테레오 타입으로 표현하고, 표현하려는 기본 요소에 정의된 이상의 공통 속성과 의미에 대하여 꼬리표 값과 제약들의 집합으로 정의한다.

2. 소프트웨어 아키텍처 모델링을 위한 핵심요소 및 세부 구성요소

소프트웨어 아키텍처를 모델링하기 위해서는 (표 2)에 나와 있는 바와 같이 컴포넌트, 커넥터(connector), 전체 시스템 구성(configuration)과 같은 세가지 핵심적인 요소들에 대한 기술 및 이들의 세부 구성요소에 대한 기술을 포함해야만 한다.

핵심 요소	세부 구성요소	의 미
컴포넌트	인터페이스 (interface)	컴포넌트가 제공하는 서비스들을 명세
	타입 (types)	기능을 재사용 가능한 블럭들로 캡슐화
	의미 (semantics)	컴포넌트 행위의 고수준 (high-level) 모델
	제약사항 (constraints)	시스템 또는 시스템의 부분에 대한 속성 또는 단언, 받아들일 수 없는 시스템의 위반사항 기술
	진화 (evolution)	컴포넌트 속성들의 변경
	비기능적 속성 (non-functional properties)	컴포넌트 행위의 명세로부터 직접 유도될 수 없는 비기능적 속성
커넥터	인터페이스	컴포넌트들의 올바른 연결과 그들의 상호작용을 가능하게 함
	타입	컴포넌트 커뮤니케이션, 조정, 중재 결정들을 캡슐화(encapsulate)함
	의미	커넥터 행위의 고수준 (high-level) 모델
	제약사항	상호작용 프로토콜 엄수 보장, 내부 커넥터 의존성 확립, 사용 범위 강요
	진화	커넥터 속성들의 변경
	비기능적 속성	커넥터 의미의 명세로부터 완전히 끌어낼 수 없는 정확한 커넥터 구현을 위한 요구사항 표현
전체 시스템 구성	이해 가능한 명세 추적 가능성 상세화, 이식성 확장성, 발전성 다양성, 제약사항 비기능적 속성	적절한 컴포넌트들이 연결되었는지, 그들의 인터페이스는 맞는지, 커넥터들이 올바른 통신을 가능케 하는지, 그들의 결합된 의미가 바라는 행위로 귀착되는지를 결정하는데 필요한 정보

(표 2) 핵심요소 및 세부 구성요소

결국, 소프트웨어 아키텍처를 모델링하기 위해서는 컴포넌트, 커넥터, 전체 시스템 구성의 기술을 비롯하여 의미·제약사항·비기능적 속성 등을 다루는 컴퓨테이션 패러다임, 커뮤니케이션 패러다임, 그리고 한 컴포넌트가 전체 아키텍처의 어디에 있는지 파악할 수 있는 계층적 컴포지션 등이 기초가 되어야 한다.

3. 핵심요소 표현과 관련된 다이어그램

UML에서 시스템의 서로 다른 모형을 나타내기 위해 제공하는 여러 다이어그램 중 아키텍처 모델링을 위한 핵심요소의 표현과 특히 관련 깊은 다이어그램들에 대한 설명이 다음의 (표 3)에 나와 있다.

핵심 요소	관련 다이어그램	의 미
컴포넌트	컴포넌트 다이어그램	소프트웨어를 구성하는 모듈 및 수행물에 대한 정적인 형상을 보여준다
	패키지 다이어그램	클래스들의 그룹과 그들간의 의존관계를 보여준다
	클래스 다이어그램	컴포넌트가 수행하는 기능을 보여준다
	시퀀스 다이어그램	컴포넌트의 동적인 행위를 표현한다
전체 시스템 구성	유스케이스 다이어그램	시스템 외부와 시스템간의 상호작용을 보여준다
	배치 다이어그램	하드웨어 노드(node)에 배치된 컴포넌트들의 물리적인 레이아웃을 보여준다

(표 3) 핵심요소 표현과 관련된 UML 다이어그램

III. UML 확장 메커니즘을 활용한 소프트웨어 아키텍처 모델링 요소 표현방법

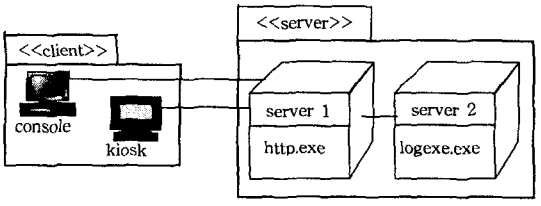
어떤 소프트웨어 시스템을 모델링하고자 할 때, UML에서 제공하는 기본 표기법만을 사용해야 한다면 모델링하고자 하는 요소들의 개념이나 의미 등을 표현하는데 한계가 있을 수 있다. 이러한 경우에 UML 표기법을 확장하여 새로운 구성 요소를 추가하거나 새로운 속성과 의미를 지정함으로써 모델링하고자 하는 요소들을 효과적으로 표현할 수 있게 된다.

여기에서는 (표 4)에 나와 있는 바와 같이 UML 확장 메커니즘을 활용하여 소프트웨어 아키텍처 모델링을 위한 요소들을 효과적으로 표현하기 위한 방법을 제안하고, 몇 가지 모델링 기법 및 예제를 보인다.

모델링 요소	표현방법
컴포넌트	스테레오 타입 <<component>> 정의
커넥터	스테레오 타입 <<connector>> 정의
인터페이스	스테레오 타입 <<interface>> 정의
타입	컴포넌트의 타입을 표현하기 위해 스테레오 타입 <<controller>>, <<link>>, <<computation>>, <<memory>> 정의 커넥터의 타입을 표현하기 위해 스테레오 타입 <<implicit invocation>>, <<procedure call>>, <<dataflow>>, <<instantiation>>, <<shared data>>, <<message passing>> 정의
의미	꼬리표 값 {semantics} 활용
제약사항	오퍼레이션 행위의 검증을 위해 사용될 수 있도록 불변조건, 선조건, 후조건을 나타내는 스테레오 타입 <<invariant>> <<precondition>>, <<postcondition>> 활용
진화	형상관리를 위해 version을 표시하는 등의 경우를 위해 꼬리표 값 활용
비기능적 속성	스테레오 타입 <<requirement>> 활용

(표 4) UML 확장 메커니즘을 활용한 소프트웨어 아키텍처 모델링 요소 표현방법

[기법1] 클라이언트/서버 시스템을 모델링하기 위해 스테레오 타입 <<client>>, <<server>>를 정의한다.

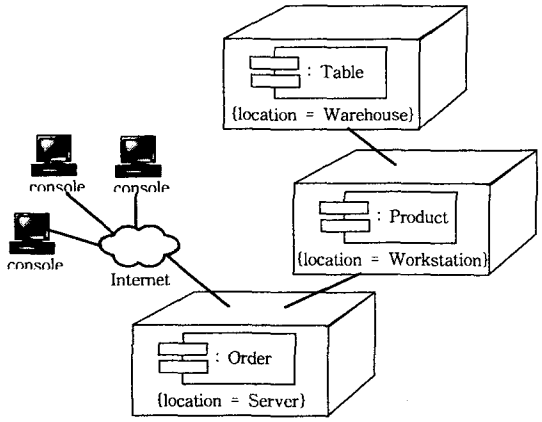


(그림 1) [기법1]을 설명하는 모델링 예제

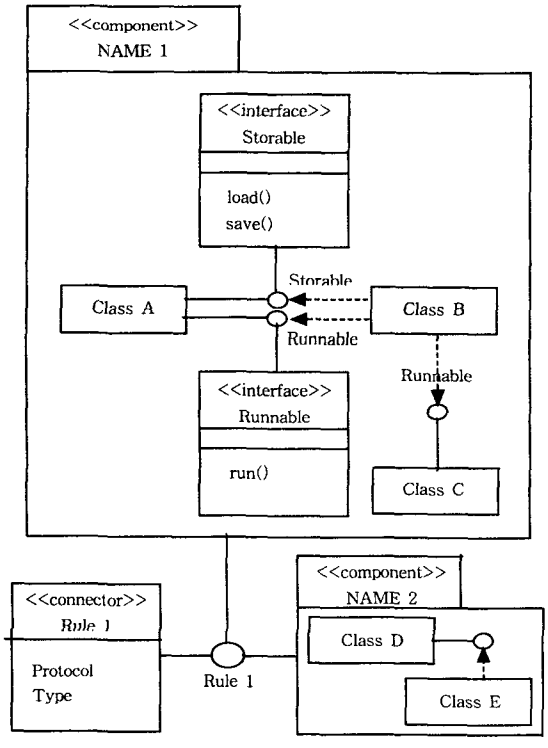
[기법2] 분산 시스템을 모델링하는 경우, 어떤 node에서 다른 node로 이동할 수 있는 컴포넌트들의 물리적인 분산을 고려하여 컴포넌트의 위치를 꼬리표 값 {location}을 사용하여 표현한다.

[기법3] 컴포넌트는 주어진 요구 기능들을 구현하기 위하여 하나 이상의 클래스들이 상호작용하는 개발 단

위를 구성하므로, 클래스들의 그룹과 그들간의 의존관계를 보여주는 패키지 다이어그램에 스테레오 타입 <<component>>를 정의하여 컴포넌트를 표현한다. 또한, 소프트웨어 아키텍처 모델링의 중요한 요소인 인터페이스와 커넥터를 보다 명확하게 표현하기 위해 이들을 스테레오 타입 <<interface>>, <<connector>>로 정의하여 사용한다.

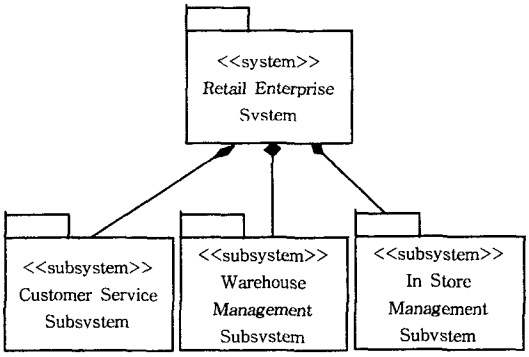


(그림 2) [기법2]를 설명하는 모델링 예제



(그림 3) [기법3]을 설명하는 모델링 예제

[기법4] 복잡한 시스템의 아키텍처를 모델링하기 위해 필요하면 시스템을 몇 개의 서브시스템으로 분해하고, 이러한 경우를 위해 스테레오 타입 <<system>>, <<subsystem>>을 정의한다.



(그림 4) [기법4]를 설명하는 모델링 예제

Ⅵ. 결론 및 추후 연구

본 논문에서는 소프트웨어 아키텍처 모델링을 위한 핵심요소 및 세부 구성요소에 대해 분석한 후, UML의 확장 메커니즘을 활용하여 이러한 다양한 요소들을 효과적으로 표현하는 방법에 대한 연구를 하였다. 또한, 몇 가지 모델링 기법을 제시하고 적용 예제를 보임으로써 본 연구의 효용성을 보였다.

추후 연구방향으로 보다 다양한 모델링 요소들을 UML의 확장을 통해 효과적으로 표현하는 방법과 그러한 표현방법을 지원하는 모델링 자동화 툴 개발에 관한 연구를 계획하고 있다.

참고문헌

[1] D. Garlan and M. Shaw : "An Introduction to Software Architecture" Advances in software Engineering and Knowledge Engineering, Vol. 1, River Edge, NJ : World Scientific Publishing Company, 1993.

[2] Nenad Medvidovic and Richard N. Taylor : "A Classification and Comparison Framework for Software Architecture Description Languages", IEEE Transactions on Software Engineering, Vol. 26, No. 1, January 2000.

[3] 김태효, 정인복, 배두환, 차성덕 : "소프트웨어 아키텍처의 연구동향", 소프트웨어공학회지 제12권 제3호, 1999.