품질속성의 일관성 유지를 위한 아키텍처 설계 방법

이정아*
*고려대학교 컴퓨터정보통신대학원 소프트웨어공학과
e-mail:gracelee@korea.ac.kr

Architecture Design Method for the Consistency Preservation of Quality Attribute

JungA Lee*
*Graduate School of Computer and Information Technology, Korea University

요 약

소프트웨어 아키텍처는 다양한 이해관계자들의 관점을 반영한 뷰들로 구성된다. 품질속성은 아키텍처의 구조를 결정하므로 아키텍처에 일관성 있게 반영될 수 있도록 설계되어야 한다. 품질속성 기반 아키텍처 설계 방법과 아키텍처 설계 산출물에서의 일관성을 위한 활동을 규정한 표준에도 불구하고 설계 과정에서 이를 실현하기 위한 구체적인 기법에 대한 제시가 없었다. 따라서 실제 설계 과정에서는 아키텍처 뷰들 간에 불일치 요소가 발생하며 특히 품질속성에 대한 일관성이 깨짐으로 재작업 발생과 시스템 품질 저하의 문제점이 나타나고 있다. 본 논문에서는 뷰 설계 산출물에 품질속성을 명시하고 뷰 품질속성 테이블을 활용함으로 아키텍처 설계에서 품질속성에 대한 일관성을 유지할 수 있는 기법을 제시한다. 제시된 기법은 품질속성에 대한 일관성을 유지함으로서 불일치성으로 인한 재작업 방지와 고품질의 소프트웨어 시스템 개발을 가능하게 한다.

1. 서론

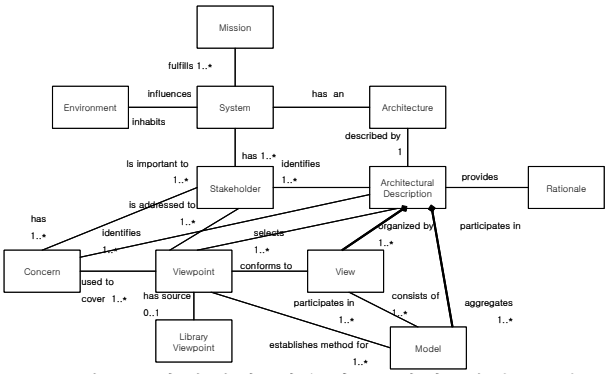
소프트웨어 아키텍처는 품질속성을 달성하는 시스템의 전체적인 구조이며 다양한 이해관계자들의 관점을 반영한 뷰들로 만들어진다[1][2]. 분석단계 산출물로부터 도출된 품질속성이 아키텍처에 일관성 있게 반영될 수 있도록 설계 과정이 진행되어야 한다. 지금까지는 품질속성 기반 아키텍처 설계 방법과 다중 뷰들에 대한 기술로 이루어진 아키텍처 설계 산출물에서의 일관성을 위한 활동을 규정한 표준이 제시되었다[1][2][3]. 이러한 표준에도 불구하고 설계 과정에서 이를 실현하기 위한 구체적인 기법에 대한 제시가 없었다. 따라서 실제 설계 과정에서는 아키텍처 뷰들 간에 불일치 요소가 발생하며 특히 품질속성에 대한 일관성이 깨짐으로 재작업 발생과 시스템 품질 저하의 문제점이 나타나고 있다. 본 논문에서는 뷰 설계 산출물에 품질속성을 명시하고 뷰 품질속성 테이블을 활용함으로 아키텍처 설계에서 품질속성에 대한 일관성을 유지할 수 있는 기법을 제시한다. 제시된 기법은 품질속성에 대한 일관성을 유지함으로서 불일치성으로 인한 재작업 방지와 고품질의 소프트웨어 시스템 개발을 가능하게 한다.

본 논문의 구성은 다음과 같다. 2장에서는 연구 배경으로서 소프트웨어 아키텍처와 품질 속성 기반 아키텍처 설계 방법을 소개한다. 3장에서는 품질속성의 일관성 유지를 위한 아키텍처 설계 기법을 제시하고 4장에서는 예제를 통해 검증한다. 5장에서는 결론과 향후 연구 과제를 제시한다.

2. 연구 배경

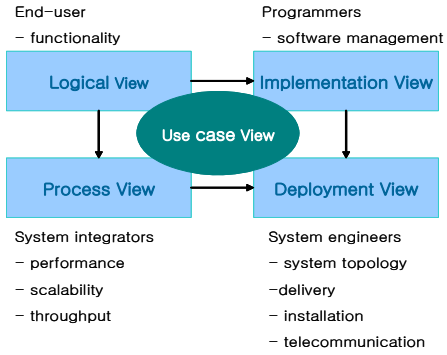
2.1 소프트웨어 아키텍처

소프트웨어 아키텍처는 소프트웨어 컴포넌트, 이들 컴포넌트들의 가시적인 속성, 그리고 컴포넌트들 사이의 관계로 구성된 시스템의 구조로서, 시스템을 설계하고 발전시키기 위한 지침과 원리로 정의할 수 있다[2]. 소프트웨어 아키텍처는 개발 초기 설계 단계의 결정사항에 대한 산출물로서 시스템과 프로젝트에 많은 영향을 미치며, 특정 시스템의 품질속성은 주로 아키텍처에 의해 결정된다. IEEE Std 1471-2000은 아키텍처 개발에 관련된 best practices에 기반 한 국제표준으로, (그림 1)과 같이 아키텍처가 표현해야 하는 내용 및 이들 간의 관계를 제공하고 있다. 제시된 표준에서 아키텍처는 모델로 구성되어진 뷰들로 조직되어 아키텍처 기술서로 표현하는 메타모델로 설명되어진다[2].



(그림 1) 아키텍처 기술서를 위한 개념 모델

아키텍처는 다양한 이해관계자들의 관심사항과 이에 따른 관점을 반영한 여러 뷰 모델로 만들어진다. 대표적인 뷰 모델인 4+1 뷰에서는 4개의 분리된 구조로 구성되는 (그림 2)와 같은 아키텍처 개념을 제시하고 이들 4개 뷰의 검증과 확인을 위해 유스케이스를 사용할 것을 제안하였다[4]. <표 1>은 뷰 모델에서 제공되어지는 유용한 정보를 보여준다.



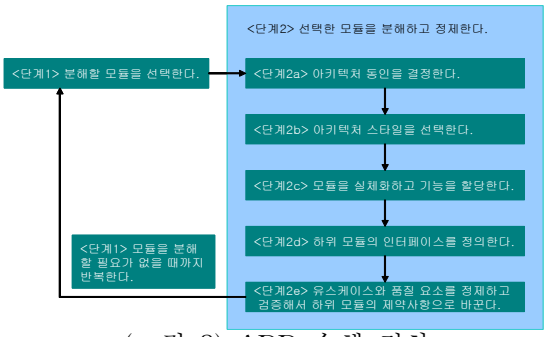
(그림 2) 4+1 뷰 아키텍처 모델

<표 1> 4+1 뷰 모델의 아키텍처 정보

뷰	아키텍처 정보
Use case view	- 시스템의 주요 목적 - 주요 목적 수행과 관련된 외부의 행동 흐름
Logical view	- 설계에 대한 개념적 모델 - 정적 구조와 동적인 상호작용
Implementation view	- 개발 환경에서의 소프트웨어 정적 구조 - 구현 모듈과 그것들간의 상호관계 - 컴포넌트의 그룹화 및 분리,가시적 인터페이스 정의
Process view	- 동시성 및 동기화 - 자원의 사용,병렬 수행,비동기적 이벤트의 처리 - 작업 그룹과 복제단위로서의 프로세스 분할 - 작업 단위간의 상호작용 메커니즘 - 메시지 흐름 및 프로세스의 부하
Deployment view	- 소프트웨어 구성 요소의 하드웨어로의 배치관계 - Logical,Process,Implement view에서 결정된 요소들의 처리 노드로의 매핑관계

2.2 품질 속성 기반 아키텍처 설계 방법(ADD)

ADD(Attribute Driven Design)는 소프트웨어가 달성해야 할 품질속성에 기반 한 소프트웨어 아키텍처 설계 방법이다. (그림 3)은 ADD 수행 절차를 보여준다. 품질속성 달성방안을 고려하여 아키텍처 스타일을 결정하고 결정한 아키텍처 스타일에 따른 모듈 분할을 수행한다. 분할로 구성요소와 이들 구성요소들 간의 관계가 결정되면 기능 요구사항을 만족시킬 수 있도록 설계요소들을 구체화한다[1][3].



(그림 3) ADD 수행 절차

아키텍처 기반에 관한 연구와 품질속성에 기반 한 아키텍처 설계 방법이 제시되었다[1][3]. 그러나 아키텍처가 반영해야 할 다양한 뷰들을 고려하고 설계 결정사항인 품질속성에 대한 이들 간에서의 일관성 유지를 위한 연구는 미흡하다. 아키텍처 기술서는 뷰들의 집합으로 포함해야 할 내용으로 이들 간의 불일치 요소를 파악과 기록을 규정하고 있지만 구체적인 실현방안이 제시된 바는 없다. 본 논문에서는 이를 해결하기 위해 아키텍처 설계에서 품질속성에 대한 일관성 유지를 위한 기법을 제시한다.

3. 품질속성의 일관성 유지를 위한 아키텍처 설계 방법

3.1 품질속성의 일관성 유지를 위한 아키텍처 설계 개요

품질속성과 관련된 정보를 효율적으로 관리함으로서 품질속성의 일관성을 유지할 수 있는 아키텍처

설계 방법을 제시하고자 한다. 본 논문에서 제시하는 아키텍처 설계 절차의 주요흐름은 (그림 4)와 같다.

(그림 4) 아키텍처 설계 절차의 주요흐름

3.2 아키텍처 설계 절차

- 단계 1. 요구사항 정의

설계 대상이 될 모듈을 선택하고 모듈의 기능 요구사항, 품질 요구사항, 설계전략, 제약사항 등을 고려하여 아키텍처 요구사항을 정의한다.

- 단계 2. 아키텍처 동인 결정

아키텍처 동인(driver)은 요구사항이 구체화된 것으로 아키텍처에 크게 영향을 미치는 우선순위에 따라 결정되어진다. 선택한 아키텍처 동인은 모두 분할한 모듈에 반영되어야 한다. 시스템은 아키텍처 동인의 선택에 따라 모듈로 분할되어진다.

- 단계 3. 아키텍처 뷰 설계

본 논문에서는 [4]에서 제시한 뷰를 설계뷰로 선택한다. 시스템에 따라 설계뷰는 선택할 수 있다. 논리뷰, 구현뷰, 프로세스뷰, 배치뷰로 아키텍처를 만들고 유스케이스뷰로 4개의 뷰를 검증하고 확인한다. 뷰 모델은 적절한 수준의 추상화를 가진 아키텍처를 기술하는데 도움을 준다. 뷰 설계 시 산출물에 품질속성을 명시한다. 뷰 품질속성 테이블에 산출물과 품질속성의 연관관계를 표시한다.

- 단계 4. 아키텍처 스타일 선택

아키텍처 동인들을 달성할 수 있는 품질속성 달성 방안을 선택하고 이것들을 실현할 수 있는 아키텍처 스타일을 선택한다. 품질속성 달성방안들이 조화를 이룰 수 있도록 조율한다.

- 단계 5. 모듈 분할과 정제

모듈 분할 과정의 처음 시작은 일반적으로 전체 시스템이다. 전체 시스템을 적절한 수준의 서브시스템으로 분할하고, 다음 단계에서는 서브시스템을 적절한 수준의 모듈로 분할하는 방법으로 분할을 반복한다.

- 단계 6. 모듈을 실체화하고 기능을 할당한다.

단계 4에서 선택한 아키텍처 스타일이 정의한 모듈들을 목표 시스템에 필요한 모듈들로 실체화한다. 상위 모듈의 유스케이스를 기반으로 하위 모듈에 기능을 할당한다.

- 단계 7. 하위 모듈의 인터페이스 정의

각 하위 모듈이 제공하는 서비스와 제공받아야 하는 서비스를 정의한다. 설계뷰를 기반으로 하위 모듈을 문서화하고 분석해서 모듈 사이의 상호작용을 찾는다.

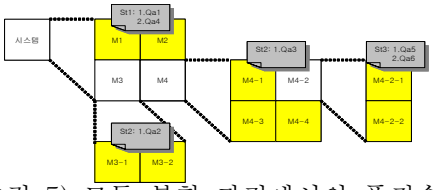
- 단계 8. 뷰 품질속성 테이블 정의

설계 뷰와 유스케이스, 먼저 설계되었던 뷰들간의 품질속성에 대한 정의를 확인하고 완성한다. 정의된 테이블을 통해 뷰 품질속성 관리가 가능하다.

3.3 주요 설계 요소

3.3.1 뷰 모델 품질속성 명시

모듈 분할과 정제의 결정사항인 품질속성을 확인하여 각 산출물에 분할 단계와 품질속성의 우선순위를 명시하고 (그림 5)와 같이 진행한다. 뷰 모델의 각 다이어그램과 명세서에 품질속성의 우선순위를 기록한다. 이들 산출물의 명시적인 정보는 상세 설계에서 참조하여 품질속성의 일관성을 유지하기 위해 쉽게 활용되어질 수 있다.

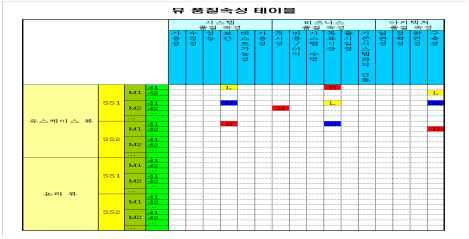


(그림 5) 모듈 분할 과정에서의 품질속성 명시

3.3.2 뷰 품질속성 테이블

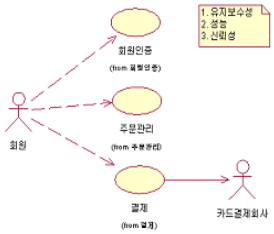
뷰 설계 시 각 산출물에 명시한 품질속성을 테이블로 <표 2>와 같이 만들어진다. 이 테이블을 통하여 설계 과정에서의 품질속성에 대한 정보를 검증하여 일관성을 유지할 수 있다.

<표 2> 뷰 품질속성 테이블



4. 전자상거래 예제를 통한 검증

기본적인 형태의 전자상거래 예제를 통하여 적용한 결과로 본 논문에서 제안하는 설계 기법에 대한 유용성을 검증한다. 요구사항 정의와 아키텍처 동인 결정 단계에서 (그림 6)의 유스케이스와 <표 3>의 품질속성이 도출되었다.



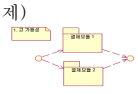
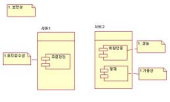
(그림 6) 전자상거래 유스케이스

<표 3> 전자상거래 품질속성

품질속성	유스케이스	근거
유지보수성	주문 관리	비즈니스 로직의 변경이 유연해야 함
성능	회원인증	PKI기반 인증을 사용하므로, 응답지연발생가능 비즈니스 로직 변경가능성 적음
가용성	결제	결제 모듈의 중단은 매출에 큰 영향 미침 특별한 가용성 관리 필요

전체 아키텍처 설계를 통한 주요 산출물과 설계사항은 <표 4>로 정리하였다.

<표 4> 뷰 설계 산출물과 설계사항

뷰	뷰 산출물(다이어그램)	설계사항
논리뷰		- 시스템 전체에 요구되는 보안품질 속성을 만족시키기 위해 2계층 모델로 설계 - 기능 및 품질속성 만족을 고려하여 3개의 서브시스템으로 구성하고, 서브시스템에 필요한 품질속성을 명시
프로세스뷰	(주문관리)  (회원인증)  (결제) 	- 논리적 서브시스템을 구성하는 실제 컴포넌트들을 설계. 전체 시스템의 특징보다 서브시스템의 품질속성이 먼저 고려 - 주문관리 : 주문관리는 비즈니스 트랜잭션을 담당하는 서브시스템으로서, 유지보수 품질속성이 가장 먼저 고려 - 회원인증 : PKI인증을 이용함으로써, 로직의 변경가능성이 적은 반면, 인증을 위한 응답시간이 지연될 수 있으므로, 성능 품질속성 우선순위 - 결제 : 가용성이 매출과 직결되므로 모듈을 중복 구성
배치뷰		- 유지보수성이 고려된 주문관리 컴포넌트들을 배치 - 성능과 가용성이 중요한 회원인증, 결제 컴포넌트들은 네트워크를 이중화

기존 아키텍처 설계에서는 뷰 모델의 작성이 완료된 뒤 평가단계에서 불일치 요소가 발견되어 재작업이 발생할 수 있다. 제시된 방법은 설계 과정에서 각 뷰 설계 산출물에 품질속성이 명시됨에 따라 전체 설계 과정에서 품질속성의 일관성을 유지하여 불필요한 재작업 방지 및 아키텍처 품질 향상을 기대할 수 있다.

5. 결론 및 향후 연구과제

본 논문에서는 아키텍처 설계에서의 뷰들 간의 품질속성의 일관성 유지를 위한 기법으로서 뷰 모델 산출물에 품질속성을 명시하고 뷰 품질속성 테이블을 활용하는 아키텍처 설계 방법을 제시하였다. 예제 검증을 통하여 시스템의 품질속성 달성방안으로서의 아키텍처의 일관성을 보장할 수 있는 유용한 방법임을 보였다. 기존 아키텍처 설계에서는 품질속성으로 구조를 결정한 후 이후 설계 과정에서는 고려하지 않는다. 그러나 제시된 방법에서는 아키텍처 설계 진행 과정에서 품질속성 관련 정보가 관리되므로 여러 아키텍처 뷰에서 일관성을 유지할 수 있다.

향후에는 이러한 기법을 활용하여 많은 사례에 적용하고 설계 과정 이후 단계에서도 품질속성의 일관성을 유지할 수 있는 방법에 대한 연구가 필요하다.

참고문헌

[1] Len Bass, Paul Clements, Rick Kazman, Software Architecture in Practice Second Edition, Addison Wesley, 2003

[2] IEEE, Recommended Practice for Architectural Description of Software_Intensive Systems(IEEE Std 1471-2000), New York, NY, 2000

[3] Len Bass, Mark Klein, Felix Bachmann, "The Quality Attribute Design Primitives and the Attribute Driven Design Method", 4th International Workshop on Product Family Engineering Bilbao, Spain, 3-5 October 2001

[4] P. Kruchten, "The 4+1 View Model of Architecture", IEEE Software, vol.12, no.6, pp. 42-50, Nov 1995

[5] 컴포넌트비전(주), 실전 CBD project, 영진닷컴, 2004