품질 속성 기반의 소프트웨어 아키텍처 평가 프로세스에 관한 연구

손이경°, 김행곤, 김명수
대구가톨릭대학교 컴퓨터정보통신공학부
{sonlk, hangkon}@c.u.ac.kr, profkms@hanmail.net

A Study on Software Architecture Evaluation Process based on quality attribute

Lee-Kyeong Son°, Haeng-Kon Kim, Myeong-Su Kim
Dept. of Computer Engineering, Catholic University of Daegu

요 약

소프트웨어 아키텍처는 소프트웨어 컴퍼넌트와 이들 컴퍼넌트들 간의 상호 관계를 나타내는 시스템의 전체적인 구조이다. 이때 완성된 시스템이 품질에 대한 요구를 만족시키는 시스템인지의 여부를 결정하는 아키텍처의 평가가 매우 중요하다. 그러나 평가 과정에서 아키텍처에 대한 부적절하거나 모호한 표현으로 인해 광범위한 응용에서는 많은 제약이 따른다. 그러므로 본 논문에서는 소프트웨어 아키텍처를 평가하기 위해 준비하고, 실행하고, 완료하는 세 가지 단계 제시한다. 이들 단계를 수행함에 따라 품질 속성의 획득에 많은 영향을 주는 아키텍처의 설계 결정을 중심으로 체계적인 아키텍처 평가가 이루어질 수 있다.

1. 서 론

오늘날 소프트웨어 개발 기술은 웹과 인터넷의 대중화로 소프트웨어의 품질 개선, 정시 생산, 다양한 요구 변화에 효율적으로 대응할 수 있는 방법을 모색하기 위해 급격히 발전되어 왔다. 소프트웨어 시스템의 주요 쟁점 중 하나인 품질에 대해 특정 시스템의 품질 속성은 주로 아키텍처에 의해 결정되어지며[1], 시스템의 생명주기 초반에 소프트웨어 품질을 평가하면 비용 효율이 높아지게 되고, 품질 요구를 만족하지 못하는 대량의 자원이 시스템 구축에 사용될 수도 있다. 이와 같은 이유로, 시스템이 구축되기 전에 시스템 품질을 만족하는지의 여부를 평가하고 결정하는 것은 매우 중요하다.

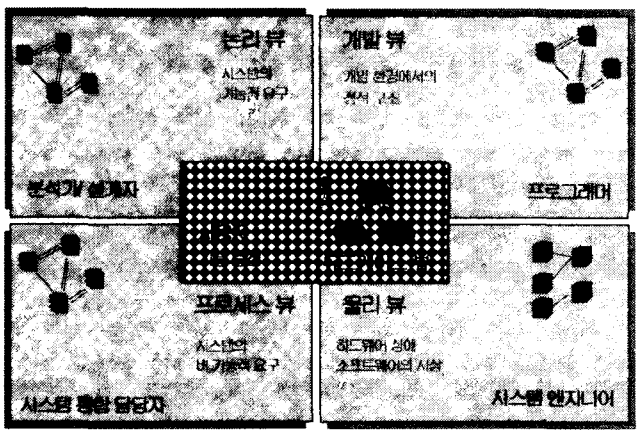
또한 기존의 소프트웨어 아키텍처 평가 방법들은 광범위한 응용에 대해서는 많은 한계를 가지고 있다. 즉, 명세를 생성하고 예측하는데 소프트웨어 공학자의 노력이 많이 요구되거나 아키텍처 설계 과정에서는 얻을 수 없는 시스템에 대한 정보를 요구하기도 한다. 뿐만 아니라, 시나리오 기반 기술들은 시나리오 도출 등의 단계에 많은 불확실성이 존재한다.

그러므로 본 논문에서는 이런 문제들을 해결하기 위해 소프트웨어 아키텍처 평가에 대한 방법론을 제시한다. 이 방법론은 소프트웨어 아키텍처 평가를 준비하고, 평가를 실행하고, 평가를 완료하는 세 가지 단계로 구성되며, 각 단계마다 세부적인 작업을 수행함에 따라 품질 속성의 획득에 많은 영향을 미치는 아키텍처의 설계 결정들을 중심으로 한 체계적인 아키텍처 평가가 이루어질 수 있다. 또한 이러한 평가는 소프트웨어 아키텍트가 아키텍처에 대한 질적인 평가를 가능하게 하며 이는 양적인 평가나 이론적 평가보다 더욱 효과적이다[2,3].

2. 관련 연구

2.1 소프트웨어 아키텍처

소프트웨어 아키텍처는 소프트웨어 컴포넌트, 이들 컴포넌트의 가시적인 속성, 컴포넌트들 사이의 관계로 구성된 시스템의 전체적인 구조이다. 아키텍처는 다양한 스테이크 홀더들 간의 원활한 의사소통의 수단이며 프로젝트 초기의 설계 결정사항을 기재해 놓은 산출물의 역할을 한다. 이러한 아키텍처에 대해 좋은 명세를 가지는 것은 아키텍처의 평가 성공에 결정적인 역할을 하게 되므로 완벽하고 명확한 문서가 필수적이다[4]. Kruchten



<그림 1> 4+1 뷰 모델

에 의해 제시된 4+1 뷰 모델은 아키텍처를 구축하고 분석하는 동안 스테이크 홀더 간의 개념 분리가 가능하여 아키텍처에 대한 이해를 향상시킬 수 있다. 이 모델은 4개의 뷰로 각 스테이크 홀더에게 적절한 설계 결정을 획득하게 하고 다섯 번째 뷰를 통해 각각을 설명하고 확인한다[5].

2.2 품질 속성(Quality Attribute)

품질 속성이란 시스템의 품질에 대한 요구사항을 의미하는 것으로 주로 시스템의 아키텍처에 의해 결정되어진다. 시스템의 생명주기 초반에 완벽한 품질에 대한 요구가 결정되기 어렵지만 품질 속성을 만족하지 못하는 자원이 시스템 구축에 사용됨으로써 시스템 전체가 품질에 대한 요구를 만족하지 못하는 경우가 발생 할 수 있다. 이러한 이유로 시스템이 구축되기 전에 사용자의 품질에 대한 요구를 만족하는지의 여부를 평가하고 결정하는 것은 매우 중요하며 이는 소프트웨어 아키텍처의 평가에도 많은 영향을 미친다.

소프트웨어의 품질에 대한 요구를 정의하거나 평가하기 위한 품질 속성은 크게 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성 등 6가지로 분류될 수 있다.

2.3 소프트웨어 아키텍처 평가를 위한 기존 방법론

기존의 소프트웨어 아키텍처 평가 방법들은 <표 1>과 같다[2].

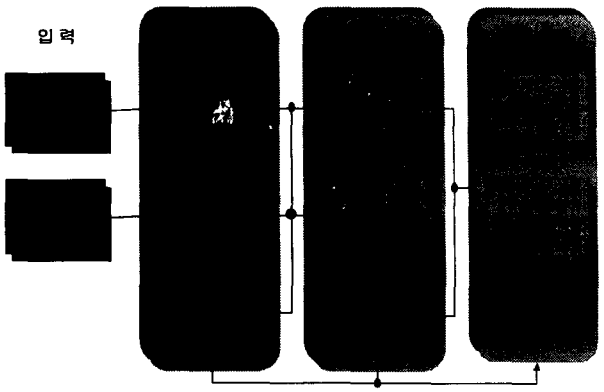
<표 1> 기존 소프트웨어 아키텍처 평가 방법

특정한 품질 속성을 평가	시뮬레이션이나 프로토타입 사용	시나리오 기반의 평가
-통계적인 방법 • Markov Chain Model • Queuing Model -언어를 통해 정보 표현 • ADL(Architecture Description Language) -소프트웨어공학자가 명세를 생성하고 예측하는데 많은 노력 필요 -표현 능력 한계	-아키텍처의 주요 컴포넌트들이 먼저 구현 -기타 컴포넌트는 어느 정도 실행 가능 시스템에서 시뮬레이트 -개발 전 시스템에 대한 정보 요구 -평가 목적의 상세한 시뮬레이션 프로토타입 고가	-품질 요구의 실제 의미를 구체적으로 얻어내기 위해 시나리오를 개발 -아키텍처의 특성에 초점-평가 단계에 대하여 시나리오를 어느 정도 세부적으로 표현해야 하는지와 같은 모호함이 존재

3. 품질 속성 기반의 소프트웨어 아키텍처 평가 프로세스

3.1 개 요

본 논문에서 제시하는 방법론은 크게 평가를 준비, 실행, 완료하는 세 가지 단계로 구성된다. 먼저 준비 단계에서는 아키텍처 평가의 범위와 목표를 정하고 초기 아



<그림 2> 아키텍처 평가 프로세스

키텍처를 제공한다. 이때 제공되는 아키텍처는 UML을 이용한 4+1 뷰 모델을 이용하며 기능적인 요구와 품질 요구를 분리하여 취급하며 품질 속성들은 품질 요구에 의해 더욱 명확하게 특성화된다. 두 번째 평가를 실행하는 단계에서는 아키텍처의 설계 결정을 찾고 이들을 분석하며 각 품질 속성에 대해 설계 결정들 간의 관계를 명확히 표현한다. 세 번째 완료 단계에서는 이제까지의 결과물을 자신의 아키텍처의 품질 속성과 연결하기 위해 구조적인 형태로 통합하고 잠재적으로 문제점이 있는 아키텍처 설계 결정에 대해 위험을 기술한다.

이러한 과정을 거쳐 소프트웨어 아키텍처에 대한 품질 속성을 예측하거나 품질 요구를 충족시킬 수 있도록 도와주며 품질 속성의 획득에 많은 영향이 있는 아키텍처 설계결정이 체계적으로 발견되어 명확히 표현되며 이들 설계 결정에 근거하여 아키텍처 위험이 기술된다. <그림 2>는 본 논문에서 제안된 아키텍처 평가를 위한 핵심적인 활동을 나타낸 것이다. 입력단계는 초기소프트웨어 아키텍처와 응용소프트웨어 요구사항을 정제하여 입력하는 단계이다.

<표 2> 평가 템플릿

목표 속성	품질 속성의 이름(예 : 수행능력, 용이성 …)	
Contexts	수행능력	QR1, QR2, QR3, …
	…	…
기능적 요구 (FRs)	P1 : 주요 기능 1 P2 : 주요 기능 2 …	1(우선 순위) 2 …
	M1 : 필수 기능 1 M2 : 필수 기능 2 …	P1 P2 …
	O1 : 선택 기능 1 O2 : 선택 기능 2 …	P1 P1 …
품질 요구 (QRs)	QR1 : 품질 요구 1 QR2 : 품질 요구 2 …	
QRs-FRs 관계	QR1	M1, M2, M4, O1, …
	QR2	M1, M3, M4, O2, …
	…	…

3.2 준비 단계

Step 1. 평가 범위와 목표 결정

이 단계는 명확한 아키텍처 평가 범위와 목표가 결정되는 단계로써 먼저 시스템의 기능적 요구와 품질요구를 분리하여 문서화 할 수 있는 간단한 템플릿을 정의한다 <표 2>. 또한, 주요 기능들을 상대적 중요도에 따라 정렬함으로써 기능적 요구들에 대한 평가의 범위를 결정하고(FRs), 품질 요구와 기능 요구간의 관계를 기술함으로써 품질 요구에 대한 평가 범위를 결정한다(QRs). 마지막으로 각각의 품질 요구에 관련되어 있는 품질 속성을 식별한다.

Step 2. 품질 속성 명세화

본 논문의 방법은 품질 속성에 초점을 맞춘 평가이므로 각 품질 속성에 대한 명확하고 유용한 특성을 관리하는 것이 중요하다. 품질 속성에 대한 특성을 잘 표현할 수 있는 템플릿을 <표 3>과 같이 정의하였다.

<표 3> 품질 속성 특성표

속성	임의의 품질 속성 이름		
요소	**이벤트**	**아키텍처 설계 결정**	**반응**
구체적 속성 (선택적)	아키텍처의 응답을 유발시키는 환경	<step 5,6>에서 결정	'요소'에 관해 관찰될 수 있는 결과

Step 3. 아키텍처 표현

아키텍처 분석에 근거한 품질 평가는 시스템 명세의 일자성에 따라 품질 평가 규모에 많은 차이를 보이므로 4+1 뷰 모델은 광범위한 애플리케이션에 대해 적절한 수준의 추상화를 가진 아키텍처를 기술하는데 도움이 되는 모델이다. <표 4>는 4+1 모델에서 얻을 수 있는 아키텍처 정보를 보여준다. 각 뷰들마다 아키텍처 스타일과 패턴을 표현할 수 있으며 아키텍처와 레벨에 대한 정보의 일치로 더욱 명확한 평가가 가능하게 된다.

<표 4> 4+1 뷰 모델에서의 아키텍처 정보

Views	Architectural Information
유즈케이스 뷰	- 주요 요구에 대한 추상화 - 시스템의 최우선 목표
논리 뷰	- 추상의 집합과 그들의 관계 - 시스템의 논리적 구조 - 논리적인 상호연결 메카니즘
개발 뷰	- 실제 소프트웨어 요소들의 구조 - 컴포넌트들의 인터페이스
프로세스 뷰	- 동시성과 동기화
물리 뷰	- 소프트웨어 컴포넌트의 물리적인 분배

3.3 실행 단계

Step 4. 아키텍처 스팟 식별

이제까지 정의된 기능적 요구와 아키텍처를 바탕으로 평가 계약으로부터 아키텍처 스팟들을 찾아내기 위해 다음의 활동이 이루어진다.

- 시스템의 주요 목적을 서술한 use case를 식별
- 기능적 요구에 관련된 품질 속성들로부터 뷰 결정
- 뷰로부터 use case에 관련된 아키텍처 스팟 식별

Step 5. 아키텍처 설계 결정 검색 및 분석

아키텍처 설계 결정이 품질 속성의 획득에 많은 영향을 미치므로 아키텍처를 평가하는 동안 아키텍처 설계 결정의 검색과 분석이 매우 중요하다. 각 아키텍처 설계 결정들은 <표 5>와 같이 표현되며 다음과 같은 활동에 의해 검색된다.

- 품질 속성 특성표의 이벤트에 대한 하나 또는 그 이상의 설계 이슈들을 바탕으로 설계의 문제점을 표현하는 결정 변수를 정의
- 각 문제점들에 대한 해결책인 결정 값을 선택하고 설계에 의해 얻어진 지식이나 아키텍처들 간의 경쟁을 통해 대안 설정

<표 5> 아키텍처 설계 결정

Decision:ADD#	**결정 변수**	**결정 값**	**대안**
설계 결정	설계 문제점	설계 해결책	대안 해결책
원리	추론 문		
아키텍처 Spots : AS1, AS2, ...	아키텍처 표현(Fig #)		

이렇게 얻어진 설계 결정들은 각각 독립적으로 분석하여야 하며 분석 결과에 따라 특정한 품질 속성이 설계 결정에 미치는 효과와 원리에 대해 기술한다.

Step 6. 아키텍처 설계 결정간의 의존성 결정

각각의 결정들은 다른 설계 결정들에게 어떻게 영향을 미치는가에 따라 식별되어지며 이 단계를 통해 각 결정들 간의 의존성이 결정되어 <표 6>과 같은 아키텍처 프로필이 생성될 수 있다. 즉, 주체의 결정이 특정 품질 속성에 관하여 어떻게 객체의 결정에 영향을 미치는지를 보여주며 품질 속성들 간의 trade-off를 결정하고 아키텍처 설계를 변경함으로써 야기될 수 있는 효과를 이해하도록 도와준다. 여기에서 QAi는 특정 품질 속성을 말하며 +는 QA에서의 긍정적 효과를, -는 부정적 효과를 의미한다.

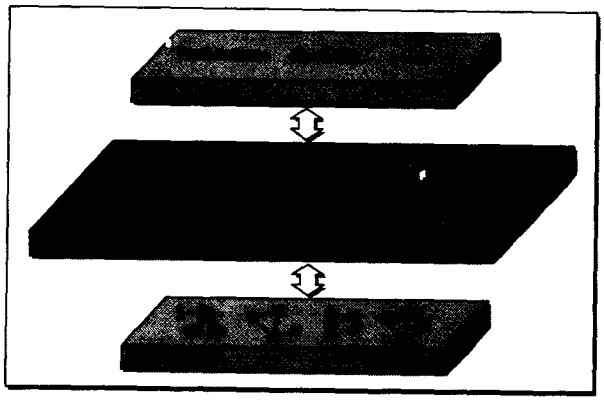
<표 6> 아키텍처 설계 결정들 간의 의존 관계

아키텍처 설계 결정		객체			
		ADD1	ADD2	ADD3	…
주체	ADD1		(QA1,+)		
	ADD2	(QA2,-) (QA3,+)			
	ADD3				(QA1,+)
	…				

3.4 완료단계

Step 7. 결과를 체계화

아키텍처와 품질 속성들간의 관계를 체계적으로 분류하는 것은 설계에서의 위험 영역을 식별하는데 도움을 줄 뿐 아니라 품질 속성을 이해하고 제어하는 능력을 향상시켜준다. <그림 3>과 같이 3가지 계층으로 이루어지며 단계적인 평가가 가능하고 평가 영역의 확장에 의한 복잡성을 제어할 수 있게 된다.



<그림 3> 품질 요구와 아키텍처의 연결 구조

Step 8. 아키텍처 위험 요소 기술

마지막 과정으로써 잠재적으로 문제가 될 수 있는 아키텍처 설계 결정을 <표 7>과 같이 원인과 결과, 아키텍처 위험 요소로 표현한다. 아키텍처 위험을 명확한 문서로 표현하는 것은 소프트웨어 아키텍트가 위험 요소를 인식하여 위험을 완화시킬 수 있는 계획을 세우도록 지원하기 위함이다.

<표 7> 아키텍처 위험 명세

RISK : RSK #	위험의 종류
아키텍처 설계 결정	ADD #
관계있는 속성들	수행 능력, …
원인	현재 야기되고 있는 문제
결과	품질 속성에 미치는 영향
아키텍처 Spots	AS1, AS2, …

4. 결론 및 향후 연구

본 논문에서는 양질의 소프트웨어 개발을 가능하게 하는 아키텍처 평가에 대한 과정을 제시하였다. 아키텍처의 평가는 소프트웨어에 대한 요구와 초기 아키텍처에 의해 시작되어 평가 과정을 수행하는 동안 평가 계약, 품질 속성 특성표, 아키텍처 설계 결정, 아키텍처 프로필 등과 같은 유용한 중간 결과물들을 체계적으로 정의 또는 생성하였으며 이러한 중간 결과물들을 근거로 아키텍처와 품질요구를 연결하는 종합 구조와 아키텍처 위험 요소들이 최종적으로 제공하였다.

본 논문의 방법론은 기존의 방법론에 비해 평가 과정이 잘 정의되어 있으며 소프트웨어 요구나 소프트웨어 아키텍처에서 어느 정도의 상세화가 적절한지, 언제 평가가 이루어지고 평가된 대상 등과 같은 모호함을 줄이기 위해 UML의 뷰 모델을 사용하였다. 또한, 평가 결과가 구체적으로 문서화 되어 단계가 진행되는 동안 그 결과물들이 구체적으로 명세 되었으며 이는 개념적인 흐름을 이해하는데 도움을 줄 뿐 아니라 아키텍처에 대한 명확한 통찰력을 얻음으로써 위험을 이해할 수 있게 한다. 마지막으로 수행능력, 유지 보수성, 보안성, 신뢰성 등 뷰 모델에서 언급된 다양한 품질 속성들이 광범위하게 평가되어질 수 있다. 향후에는 소프트웨어 요구에서의 모호성 처리와 좋지 않은 아키텍처 설계 결정의 처리와 같은 이슈들에 대해 관심 있게 논의할 것이다.

참고 문헌

[1] P. Clements, R. Kazman, M. Klein, Evaluating Software Architecture: Methods and Case Studies, Addision-Wesley, 2002

[2] Bosch, J., Design and Use of Software Architecture, Addison-Wesley, 2000

[3] Stephan Kurpjuweit, Ph D. Thesis, "A Family of Tools to Integrate Software Architecture Analysis and Design", Final Draft Version, to be published 2002

[4] Mugurel T. Ionita, Dieter K. Hammer, Henk Obbink, Scenario-Based Software Architecture Evaluation Methods: An Overview, http://www.win.tue.nl/oas/architecting/aimes/papers

[5] Kruchten, P., "The 4+1 View Model of Software Architecture", IEEE Software, Vol. 12, No.6 pp.42-50. November 1995