웹기반 상거래시스템의 품질속성 도출을 위한 사용자 요구사항 분석

윤홍란° 김유경 이서정 박재년
숙명여자대학교 컴퓨터과학과
{hryun°,ykkim,sjlee815,jnpark}@sookmyung.ac.kr

User requirement analysis to derive quality attribute of web-based commerce system

Hongran Yun°, Yukyung Kim, Seojeong lee, Jaenyun Park
Dept. of Computer Science, Sookmyung Women's University

요 약

소프트웨어 아키텍처는 소프트웨어 시스템의 중요한 요소를 정의하고 이들 요소 간의 상호관계를 나타낸 전체 시스템의 기본 구조이다. 시스템의 변경과 확장에 용이하고 시스템의 상호 연동성을 확보하는데 있어 아키텍처기반의 기술 접근은 필수적인 요소이다. 이러한 소프트웨어 아키텍처를 평가하기 위한 다양한 방법들이 제시되고 있으며, 이 방법들은 비즈니스요구를 반영하고 품질속성에 따른 시나리오를 만들어 그것을 기반으로 소프트웨어아키텍처를 평가하게 된다. 이에 본 논문에서는 ISO9126에서 제시한 소프트웨어의 품질을 기반으로 아키텍처 평가에 적용되어질 품질속성을 도출하기 위하여, 웹기반 전자상거래시스템인 B2C에서의 사용자 요구사항을 분석해 보고자 한다.

1. 서 론

소프트웨어 아키텍처는 복잡한 시스템의 구조를 구조적으로 파악할 수 있도록 하는 전체 소프트웨어 시스템의 청사진이라 할 수 있다. 소프트웨어 시스템의 크기와 복잡도가 증가함에 따라 알고리즘이나 자료구조의 선택보다 전체 소프트웨어 시스템의 구조를 결정하는 일이 더욱 중요해졌다. 따라서 잘 정의된 아키텍처는 전통적인 코드 재사용에 비하여 컴포넌트와 아키텍처 디자인에 대한 광범위한 재사용을 가능하게 함으로써 실질적인 생산성 향상과 품질의 유지를 가능하게 한다. 비즈니스변화에 따른 아키텍처 변경의 지속적이고 효과적인 관리가 필요하며, 이를 위해 비즈니스 목표 및 상위수준의 품질요구사항을 기준으로 시나리오가 생성된다. 이 시나리오는 아키텍처적 접근(Architectural approach)을 위한 결정(Architectural decision)에 영향을 미치게 된다. 이에 본 논문에서는 ISO9126에서 제시한 소프트웨어의 품질을 기반으로 아키텍처 평가에 적용되어질 품질속성을 도출하기 위하여, 웹기반 전자상거래시스템인 B2C에서의 사용자 요구사항을 분석해 보고자 한다

2. 관련 연구

2.1 스테이크홀더(stakeholder)

스테이크홀더란 소프트웨어 아키텍처와 이 아키텍처로부터 만들어질 시스템에 관심을 가지고 있는 이해당사자들을 말한다. 스테이크홀더 중에 일부는 coder, integrator, tester, maintainer 등과 같은 개발팀의 일원이 될 수도 있다. 의사 결정권자(project decision maker)도 스테이크홀더가 될 수 있고, 평가 결과에 관심을 가지고 있으며, 프로젝트에 영향을 줄 수 있는 의사결정권을 가지고 있는 사람들인 아키텍트, 컴포넌트 설계자, 프로젝트 관리자, 고객, 스폰서등을 포함한다. 소프트웨어 아키텍처는 스테이크홀더들 간의 원활한 의사소통을 위한 수단으로 사용된다. 또한 아키텍처는 분석, 개발, 개선의 대상이 되는 시스템을 위한 기술적인 청사진(blue print)를 제공한다. 다음의 그림1에서 보는바와 같이 스테이크홀더에게 보이는 시스템의 모습은 그들의 관점에 따라서 다르며 소프트웨어 아키텍처는 스테이크홀더들간의 의사소통을 통해 서로를 이해하고 합의를 도출할 수 있게 해주는 공동적인 시스템의 추상화된 산출물이다.

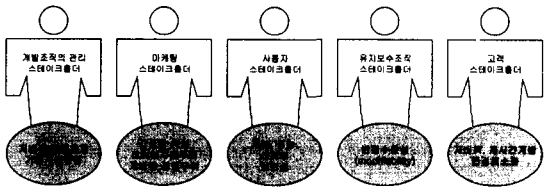

그림 1. 아키텍처에 영향을 주는 스테이크홀더

이 산출물은 시스템의 품질 속성에 큰 영향을 주며 성능과 보안, 유지보수성과 신뢰성, 비용절감에 기여한다. 이 소프트웨어 아키텍처는 여러 시스템들을 위해 재사용되는 시스템의 추상적 모습이며 시스템이 어떻게 구성되어 있고, 시스템의 조각들이 어떻게 결합되어 있는지를 보여주고 있는 모델이다. 이 모델은 너무 크거나 복잡해서는 안되며, 개발자가 충분히 인식할 수 있을 정도여야 한다. 이러한 조건에 맞는 모델(아키텍처)이라면, 비슷한 요구사항을 갖고 있는 다른 시스템을 위해 재사용될 수 있다.

2.2 품질 속성(Quality Attribute, QA)

품질속성은 시스템이 만족해야 하는 품질에 대한 요구사항을 의미한다. 품질에 대한 요구사항이 요구사항 명세에 자세히 정의되는 것이 이상적이나 아키텍처 정의 초기에는 완벽한 품질 속성이 도출되지 않으며 따라서 아키텍처는 요구사항 명세에 정의되어 있지 않은 품질에 대한 요구사항이 있을 수 있다. 아키텍처 분석 평가의 최우선 과제는 아키텍처를 분석 평가하기 위한 품질 속성을 정확하게 도출해 내는 것이다. 품질속성은 아키텍처 평가를 위한 근간이 되며 품질속성은 절대적인 수치가 아니라 목표하고 있는 상황(context)을 함께 제시해 주어야 한다.

ISO9126에서는 소프트웨어 평가를 위한 품질 속성을 6가지로 분류하고 있다. 이것은 단지 소프트웨어의 평가뿐만 아니라 품질에 대한 요구사항을 정의하는데도 사용되어진다. 이러한 소프트웨어 품질 특성모델은 다음 그림2와 같다.

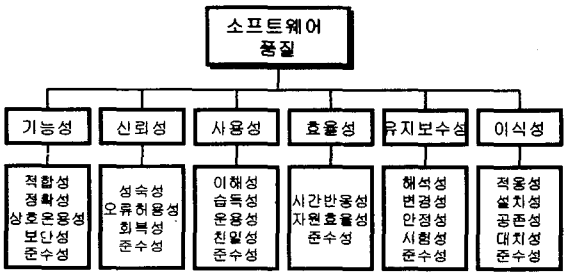

그림 2. 소프트웨어 제품 품질 특성 모델

기능성(Functionality)은 규정된 기능 및 성능을 정확하게 충족하는지를 평가하며 신뢰성(Reliability)은 소프트웨어가 특정조건에서 사용할 때 요구되는 성능수준을 유지하는지를 평가한다. 사용성(Usability)은 소프트웨어를 사용시 사용자가 쉽게 이해하고 학습할 수 있는지를 평가하며 효율성(Efficiency)은 소프트웨어 운용시 메모리, 저장장치등의 자원을 적절하고 효율적으로 운영하는지를 평가한다. 유지보수성(Maintainability)은 소프트웨어의 수정, 개선등을 쉽게 수행할 수 있는지를 평가하고 이식성(Potability)은 다양한 사용환경(H/W, OS등의 변경)에서 사용할 수 있는지를 평가한다.

2.2 ATAM(Architecture Trade-off Analysis Method)

ATAM은 비즈니스 목표를 감안하여 아키텍처 요구사항을 도출하고 이를 성능(performance), 보안(security), 변경용이성(modifiability), 가용성(availability)등의 품질속성에 비추어 아키텍처를 분석 평가하는 방법이다. 이러한 ATAM의 세부 절차는 그림3과 같다.

아키텍처 분석 평가의 가장 큰 장점은 아키텍처를 개선하고 발전시킬 수 있다는 것이다. 또한 비즈니스 전략을 지원할 수 있는 품질 속성에 대한 요구사항을 보다 명확히 정의할 수 있도록 하며, 소프트웨어 개발에 참여하는 다양한 스테이크홀더간의 의사소통을 증가시킴으로써 시스템의 품질 및 생산성을 향상시킬 수 있다. ATAM은 품질속성에 비추어 시나리오를 반복적으로 평가함으로서 성숙한 아키텍처를 도출하는 것에 중점을 둔다.

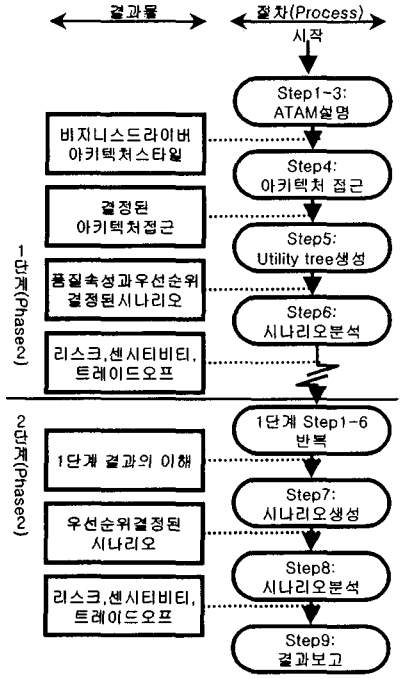

그림 3. ATAM절차

3. 품질 속성 도출

3.1 웹기반 상거래시스템에서 사용자 중심의 품질속성

ISO9126에서 정의한 여섯 가지 품질 속성을 사용자 측면에서 살펴보면, 시공을 초월한 웹을 기반으로 하므로 언제 어디서나 접속이 항상 가능해야 한다는 면이 사용자 중심측면의 신뢰성으로 볼 수 있다. 또한 새로운 카테고리의 상품이 추가된 경우에 사용자의 기존 인터페이스를 유지하면서 변경이 가능해야 하는 유지보수성을 볼 수 있으며 사용하는 사용자는 처리시간이 오래 걸리는 시스템에 대한 인내심이 부족하므로 성능에 대한 고려가 있어야 한다. 웹기반으로 상거래가 일어나게 되므로 결재 정보나 개인 정보에 대한 보안이 중요하며 사용자가 얼마나 쉽게 시스템에 익숙해지고 따라서 거래로 연결되도록 하는 사용성이 필요하다.

3.2 웹기반 상거래시스템의 사용자 요구사항

웹기반 상거래시스템인 B2C의 경우에 사용자의 요구사항은 그 시스템이 가져야할 품질에 대한 요구사항으로 그

대로 적용될 수 있다.

사용자는 다양한 많은 요구사항을 가질 수 있으나 일반적인 경우에 고려될 수 있는 요구사항을 분류하여 정리하면 다음과 같다.

① 사용 및 탐색의 용이성

R1. 일관적인 인터페이스를 제공해야 한다.

R2. 제품에 대한 목록이나 카탈로그를 논리적인방법으로 구성해야 한다.

R3. 고객이 원하는 제품의 위치를 모르는 경우에 제품을 검색할 수 있도록 해야 한다.

② 빠른 성능

R4. 사용자가 동작을 요청한 후 결과를 보기 전에 기다려야 하는 응답시간을 최소화해야 한다.

③ 익명 구매

R5. 사용자가 회원가입을 하기 전에 원하는 상품에 대한 정보가 있는지 없는지 찾고 쇼핑바구니에 담을 수 있도록 해야 한다.

④ 사용자 프로필 관리

R6. 사용자가 시스템에 다시 방문 할 경우 이전에 방문했을때의 정보를 유지 관리해야 한다.

⑤ 강력한 보안

R7. 안심하고 구매가 일어날 수 있는 보안에 대한 신뢰가 있어야 한다.

R8. 인증되지 않은 사용자가 민감한 정보에 접근할 수 없도록 해야 한다.

3.3 사용자 요구사항과 품질속성의 매핑(mapping)

ISO9126의 6가지 품질속성 중에서 유지보수성과 이식성의 경우 소프트웨어의 내부적인 품질 속성으로 볼 수 있다. 따라서 정리된 사용자요구사항(R1~R8)은 사용자의 시스템에 대한 외부 품질에 관련된 4가지속성과 그에 따른 부품질 속성들을 기준으로 아래 표1과 같이 관련지을 수 있다.

표 1. 요구사항별 품질속성

분류	사용자 요구사항	품질속성			
		기능성	신뢰성	사용성	효율성
사용 및 탐색의 용이성	R1			○	○
	R2	○		○	
	R3			○	○
빠른 성능	R4			○	○
익명 구매 가능	R5	○	○		
사용자프로필 관리	R6	○			○
강력한 보안	R7	○	○		
	R8	○	○		

표1을 보면 상거래시스템이 가져야 하는 기능적인 면과 보안이 중요한 품질속성이 되므로 기능성이 중요하다. 그리고, 사용자가 얼마나 쉽게 거래시스템이 제공하는 서비스를 받을 수 있느냐 하는 사용성이 역시 중요한 품질이 된다. 웹의 특성상 어느곳에서도 접속이 가능하며 사이트가 항상 접속 가능한 상태라야 사용자로 하여금 신뢰성을 얻을 수 있으므로 웹 기반 거래시스템의 중요한 품질 속성으로 신뢰성(Reliability)도 매우 중요하다는 것을 볼 수 있다.

4. 향후 연구방향 및 결론

소프트웨어 아키텍처는 시스템을 이해하기 위한 최선의 요소가 되며 스테이크홀더 간에 의사소통 수단으로 사용된다. 또한, 소프트웨어 아키텍처는 초기 설계 결정사항을 기재해 놓은 설명으로써 조직을 구성하고 시스템을 설계하기 위한 전략을 세울 수 있도록 도와준다. 소프트웨어 아키텍처는 재사용할 수 있는 모델로써 동일한 아키텍처를 사용하는 미래의 시스템을 이해하는데 도움을 준다. 소프트웨어 아키텍처를 분석 평가하는 방법들은 현재 도입단계이며 아키텍처 분석 평가에 있어서 중요한 점은 시스템 요구사항에 맞는 품질 속성을 잘 도출하는 것이다. 이것은 향후 평가를 위한 중요한 기준이 되기 때문이다. 본 논문에서는 소프트웨어 아키텍처가 만들어지기 전에, 사용자에 의한 시스템의 요구사항을 분류하고 중요한 품질 속성을 찾아내는 방법을 제안하였다. 사용자 요구사항에 대한 품질속성의 매핑이 품질속성을 도출하기 위한 아키텍트의 직관에 많이 좌우되는 점을 볼 때, 이를 위한 객관적이 기준마련에 대한 연구가 필요하다.

참 고 문 헌

[1] ISO9126, "Software Quality characteristics and *metrics*", 1997

[2] Mario R.Barbacci, Robert Ellison, Anthony J. Lattanze, Judith A.Stafford, Charles B.Weinstock and William G.Wood, "Quality Attribute Workshops(QAWs)", 2003

[3] Rick Kazman, Mark Klein, Paul Clements, "ATAM: A Method for Architecture Evaluation", CMU/SEI, 1999

[4] P.Clements, R.Kazman and Klein, "Evaluating S/W Architectures", Addison Wesley, 2002

[5] L.Bass, P.Clements and R.Kazman, "Software Architecture in Practice", Addison Wesley, 1998

[6] David Garlan, Mary Shaw, "An Introduction to Software Architecture", CMU, 1994