시나리오 기반 소프트웨어 아키텍처 분석방법에 대한 비교 연구

윤홍란*, 이서정*, 김유경*, 박재년*
*숙명여자대학교
e-mail:hryun@sookmyung.ac.kr

The comparison of Scenario-based Architecture methods

Hong-Ran Yun*, Seo-Jung Lee*, Yu-Kyung Kim*, Jae-Nyun 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 분석하기 위한 다양한 방법들이 제시되고 있으며, 이 방법들은 비즈니스요구를 반영하고 품질속성에 따른 시나리오를 만들어 그것을 기반으로 소프트웨어아키텍처를 갖추게 된다. 이에 본 논문에서는 시나리오 기반 소프트웨어 아키텍처 분석방법에 대한 비교 연구를 통하여 향후 소프트웨어 아키텍처 분석을 위한 효과적인 방향을 제시해 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 한다. 이는 결과적으로 실질적인 생산성 향상과 품질의 유지를 가능하게 한다. 품질향상을 위한 아키텍처적 접근(Architectural Approach)은 아키텍처 변경에 대한 지속적이고 효과적인 관리를 위해 비즈니스 목표 및 상위수준의 품질속성(성능, 변경용이성, 가용성, 보안, 재사용성 등)을 정의한다. 이 요구사항을 기준으로 시나리오가 생성되고 이 시나리오는 새로운 아키텍처 결정에 영향을 미치게 된다. 이렇게 생성 또는 변경된 아키텍처가 품질 향상을 달성할 수 있는가는 분석 및 평가에 의해 이루어진다.

아키텍처 분석 평가에 대한 많은 연구가 CMU/SEI (Carnegie Mellon University/Software Engineering Institute)에서 이루어지고 있으며 본 논문에서는 QAW, ADD, ATAM, ARID과 CBAM의 특징 및 절차를 정리하고 이들 방법에서 공통적으로 추구하는 시나리오 기반 아키텍처 분석 방법의 비교를 통하여 향후 소프트웨어 아키텍처 분석을 위한 효과적인 연구방향을 제시해 보고자 한다.

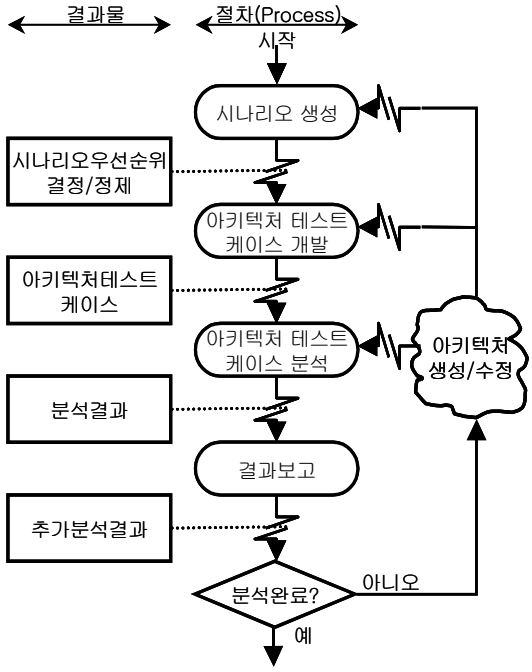
2.관련연구

SAAM을 시작으로 많은 아키텍처 중심 분석 설계 방법들이 제안되어져 왔으며, SAAM은 ATAM,

QAW, CBAM, ARID와 ADD같은 다른 방법들의 생성에 큰 영향을 미쳤다. 이러한 방법들은 각각의 분석 평가의 관점을 갖고 있으며 소프트웨어 라이프사이클전체에 영향을 주고 있다.

2.1 QAW(품질 속성 워크샵)

비즈니스 목표로부터 파생되어진 몇가지 품질속성(가용성, 성능, 보안, 상호운용성, 변경용이성등)에 대하여 시스템의 아키텍처를 분석하기 위한 방법이다. QAW의 경우엔 소프트웨어 아키텍처가 존재하지 않는 상태에서 이루어진다는 것을 전제로 한다. QAW의 세부 절차는 <그림1>과 같다.



<그림 1>QAW절차

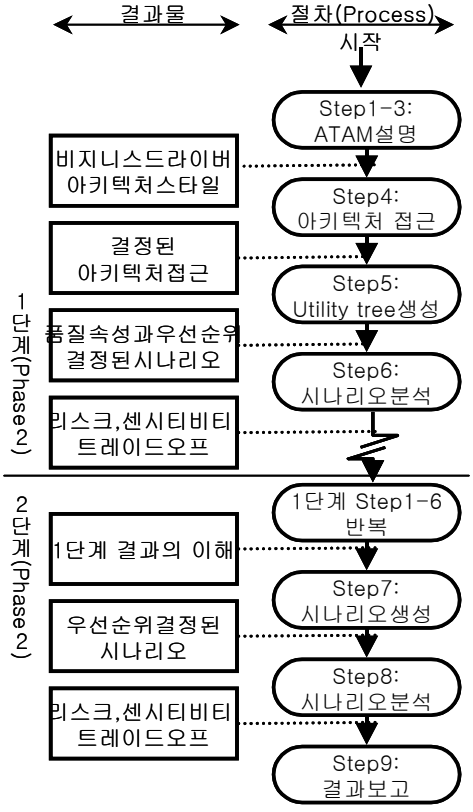
<그림1>에서 보는 바와 같이 QAW는 비즈니스 드라이버에 따른 요구사항을 분석하고 시나리오를 생성한다. 그 시나리오에 대한 우선순위를 정하고 반복적인 리뷰를 통하여 시나리오를 정제하는 과정을 거치게 된다. 이렇게 정제된 시나리오에 따라 적절한 테스트케이스를 만들어내고 각각의 상황에 따라 시스템이 어떻게 반응할 것인지를 고려한다. 그리고 어떤 아키텍처뷰(시스템, 테크니컬, 절차, 행위, 구조 등)가 각각의 시스템을 가장 잘 표현해 낼지를 찾아내는 테스트케이스분석을 거쳐 아키텍처를 만들어내게 된다. 결과에 대한 보고를 통하여 분석이 완료되게 되며 지금까지의 과정이 반복되면서 성숙된 소프트웨어 아키텍처를 갖추게 된다.

2.2 ADD(Attribute-Driven Design)

ADD는 소프트웨어가 수행해야만 하는 품질속성을 기반으로 소프트웨어 아키텍처를 정의한다. 시스템에 대한 제약조건과 기능적인 요구사항, 품질에 대한 요구사항의 이해를 필요로 하며 아키텍처의 분해를 포함하여 모듈분해(module decomposion)도, 병행(concurrency)도, 배치(deployment)도를 결과물로 만들어낸다. 세부 절차를 살펴 보면 분해할 모듈을 선택하고 구체적인 품질 시나리오와 기능적 요구사항으로부터 아키텍처적 드라이버를 선택하고 이것을 만족시키는 아키텍처적 패턴을 찾아낸다. 이어서 다양한 관점으로 유스케이스로부터 기능을 분배시키고 하부모듈과의 인터페이스를 정의하고 유스케이스와 품질 시나리오를 정제하는 과정을 거치게 된다.

2.3 ATAM(Architecture Trade-off Analysis Method)

ATAM은 비즈니스 목표를 감안하여 아키텍처 요구사항을 도출하고 이를 성능, 보안, 변경용이성, 가용성 등의 품질속성(Quality Attribute)에 비추어 아키텍처를 분석 평가하는 아키텍처 분석 방법이다.



<그림2>ATAM절차

<그림2>의 ATAM에서는 비즈니스 요구사항을

파악하고 이에 따라 품질속성(modifiability, availability, performance, maintainability, scalability, flexibility 등)을 식별하여 품질속성별로 구체적인 시나리오를 작성한다. 여기에는 시스템에 대한 기능 요구사항(use case)뿐만 아니라 비즈니스 요구사항 및 예측 가능한 업무환경 변화(change case) 등이 반영되며 시나리오를 작성한 뒤에는 시나리오에 대하여 우선순위를 부여하고 이를 유틸리티 트리(utility tree)로 작성한다. 다음으로 아키텍처 설계자는 아키텍처 요구사항을 만족할 수 있는 소프트웨어 아키텍처 후보를 도출하여야 한다. 이 때 프레임워크 및 아키텍처 스타일을 적용할 수 있으며, 레퍼런스 아키텍처 리파지토리가 구축되어 있는 경우 해당 정보를 이용할 수 있다. 소프트웨어 아키텍처 후보가 도출된 뒤에는 사전에 만들어진 유틸리티 트리에 따라 품질속성(시나리오)별로 아키텍처에 대한 분석 평가를 실시한다. 이 때 품질속성간의 트레이드 오프를 분석하는 것이 중요하다. 이를 통하여 특정 품질속성을 만족하기 위한 아키텍처 결정이 보다 우선순위가 높은 품질속성에 대한 요구사항을 만족할 수 없도록 하는 것을 방지할 수 있다. 아키텍처 분석 평가가 끝난 뒤에는 평가 결과를 반영하여 최종 소프트웨어 아키텍처를 정의 한다.

2.4ARID(ActiveReviewsforIntermediateDesigns)

ARID는 ATAM처럼 스테이크홀더들이 사용성(usability)에 대한 설계를 테스트하는데 사용하는 시나리오를 생성하도록 한다. ARID방법은 초기에 일부만 설계되어 있을 경우 평가하기에 적합하다.

ARID방법에는 초기 시나리오와 존재하는 아키텍처적 디자인 문서를 필요로 하며 아래의 세단계 절차를 따른다.

(1) 디자인 소개

선임디자이너가 디자인에 대한 개요을 소개하고 예를 보여준다. 이 단계에서 디자인이 완료되고 생산에 대한 준비를 하기 전에 직면하게 되는 잠재적인 문제들에 대한 정리를 하게 된다.

(2) 브레인스토밍과 우선순위결정

참가자들은 직면하게될 문제를 해결하는데 사용되어질 시나리오를 제안하고 각각의 시나리오를 분석하고 투표를 통해 적합한 시나리오를 결정한다.

(3) 시나리오 적용

투표에서 가장 많은 표를 받은 시나리오를 시작으로 시나리오에 의해 제안된 문제를 해결하기위해 적용해본다.

2.5CBAM(Cost-BenefitAnalysisMethod)

CBAM은 시스템의 스테이크홀더들이 디자인이나 유지보수단계에서 시스템을 향상시키기 위한 아키텍처적 대안들 사이에 선택해야 되는 상황을 효과적으로 도와준다. CBAM은 ATAM등의 다른 방법들이 아키텍처를 구축한 상태에서 트레이드오프를 다루는 것과 달리, 아키텍처를 유지하고 업그레이드하는 단계에서 위험을 최소화하고 효율을 극대화할 수 있는 의사결정을 지원하는 방법이다.

이 방법의 컨셉은 아키텍처적 전략이 달라짐에 따라 시스템의 품질속성에 영향을 주게되고 그 결과는 스테이크홀더에게 이익을 제공하므로 전략을 제대로 선택해야 한다는 것이다. 상대적 중요도를 고려한 이익(Benefit Bi)를 계산하고, 비용을 고려하여 ROI(Return on Investment)를 구한다. 이 과정은 9단계로 구성된다.

(1) 시나리오 수집
(2) 시나리오 가공
(3) 시나리오 우선순위화
(4) 사용가능도 부여
(5) 시나리오의 아키텍처적 전략 개발 및 품실속성 응답수준 결정
(6) 값보정을 통한 사용가능도 기대치결정
(7) 아키텍처적 전략에 따른 전체 이익 계산
(8) 비용제약에 따라 ROI전략 선정
(9) 결과 확정

3.소프트웨어아키텍처분석방법비교

위에서 제시한 소프트웨어 아키텍처 분석 평가방법들은 모두 시나리오 기반이고 이 시나리오에 중점을 두어 소프트웨어 아키텍처를 찾아가게 된다.

소프트웨어 개발의 생명주기별과 분석목적측면에서로 나누어 소프트웨어 아키텍처 분석 방법을 비교해 보고자 한다.

3.1생명주기(Life-Cycle)비교

아키텍처 중심의 소프트웨어 개발 방법에 있어서의 생명주기(life-cycle)는 우선 비즈니스 요구사항과

제약조건의 이해하고 이에 따른 요구사항 도출과 수집단계를 거쳐 아키텍처 설계하고 설계된 아키텍처를 기반으로 구현(implementation), 테스팅(testing), 배포(deployment), 유지보수(maintenance)의 단계를 거치는 것이 일반적이다. 아래 <표 1>은 생명주기 단계별 방법들의 연관관계를 보여준다.

I(input)/O(output)

생명주기단계	QAW	ADD	ATAM	CBAM	ARID
비즈니스요구사항 및 제약조건이해	I	I	I	I	
요구사항분석	I/O	I	I/O	I/O	
아키텍처디자인		O	I/O	I/O	I
상세디자인					I/O
구현					
테스팅					
배포					
유지보수				I/O	

<표 1> 방법과 생명주기단계

<표 1>에서 보면 아키텍처디자인은 ADD의 결과로써보여지고 ARID의 입력임과 동시에 ATAM과 CBAM에서는 입력과 출력형태로 적용된다. 아키텍처 중심의 소프트웨어 개발의 생명주기를 볼 때 QAW는 생명주기의 초기 단계에 이루어지며 QAW 결과는 ATAM의 초기단계에 응용되어져 비즈니스 요구사항을 반영하고 품질 속성을 고려한 소프트웨어 아키텍처를 정의 할 수 있다. QAW는 비즈니스 시나리오를 도출해내는 것에 중점을 두는 반면 ATAM은 품질속성에 비추어 시나리오를 반복적으로 평가함으로서 성숙한 아키텍처를 도출하는 것에 중점을 둔다.

3.2분석목적비교

분석목적은 시나리오 도출, 위험분석, 사용성 평가 그리고 비용분석 등으로 구별된다. 분석 목적측면에서 보면 QAW는 요구사항분석을 통해 효과적인 시나리오 생성을 목표로 한다. ADD는 전체시스템을 작은 모듈로 분해하기 위한 제약조건과 기능적인 요구사항, 품질속성의 요구사항을 수렴하여 아키텍처의 분해를 이루는 것이 목표이다. ATAM의 경우엔 품질속성의 요구사항을 만족시키기 위한 결정을 하게 되는 원인을 파악하여 리스크에 대한 정보와 트레이드오프 포인트에 대한 결과를 최종산출물로 내놓게 된다. ARID의 경우 부분적으로 완료된 디자인을 검사하기 위한 기술이다. 사용성(usability)측면에서 디자인을 테스트하는데 사용될 수 있는 시나리오를 생성해서 디자인의 성공적인 사용을 막는 문제들을 찾아낸다. 이 방법은 ATAM의 일부로 포함될 수 있다. CBAM의 경우엔 아키텍처 기반의 비용적인 분석을 하게한다. 시스템의 디자인과 유지보수 단계에서 시스템을 향상시킬 수 있도록 하기위한 선택의 경우 스테이스홀더의 판단을 도와주는 역할을 한다. 비용과 이익을 평가하여 아키텍처적 전략을 세우고 그에 따른 리스크를 찾아내게 된다.

4. 향후연구방향및결론

소프트웨어 아키텍처 분석 방법으로 사용되는 방법들은 대부분 비즈니스 요구사항을 반영하는 품질속성에 따라 시나리오 기반으로 시스템의 아키텍처를 분석하게 된다. 이러한 소프트웨어 아키텍처를 분석하는 방법들은 현재 도입단계이며 실무에서는 전체과정보다는 일부 필요한 부분을 선택하여 적용해가고 있다. 소프트웨어 개발의 생명주기상에 각 단계에서 적용가능한 방법들을 적절히 활용하는 것이 중요하며 이는 적절한 시나리오 도출에도 영향을 미친다.

향후과제는 품질속성에 따른 시나리오의 트레이드오프를 평가하는 정형적인 방법을 제시하여 아키텍트의 경험 의존도를 낮추어 보고자 한다.

참고문헌

[1] Rick Kazman, Robert L.Nord, Mark Klein, "A Life-Cycle View of Architecture Analysis and Design Methods", CMU/SEI, 2003

[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