제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

컴포넌트 기반 아키텍처 설계 사례

조진희*, 나희동**, 김진삼*, 김태우**
*한국전자통신연구원
**투이컨설팅㈜
e-mail : chojh@etri.re.kr

A Case of Component-based Architecture Design

Jin-Hee Cho*, Hee-Dong Na**, Jin-Sam Kim*, Tae-Woo Kim**
* Eletronics and Telecommunications Research Institute
**2e Consulting Co. Ltd.

요 약

컴포넌트 기반 개발환경이 성숙되면서 아키텍처는 재사용 기반을 제공하는 핵심 기술요소로 인식되고 있으며, 이의 체계적인 설계와 관리를 위한 소프트웨어 아키텍처 설계개념이 중요시 되고 있다. 그러나 최근 객체지향 개발 프로세스에서 클래스 단위의 하위수준 모듈에서 시작하여 컴포넌트를 도출해가는 상향식(bottom-up)설계방식이나 컴포넌트 개발에서 강조하는 패턴중심 설계기법은 비즈니스 관점의 전략적 아키텍처 설계가 불가능한 단점이 있다.

이에 본 논문에서는 기존의 아키텍처 설계 기법의 단점을 보완하기 위해 한국전자통신연구원에서 개발한 컴포넌트기반 시스템 개발 방법론인 마르미-III 에서 채택하고 있는 아키텍처 설계기법을 소개하고 이를 적용한 사례를 소개한다.

1. 서론

컴포넌트 기반 개발방법론은 단순한 소프트웨어 프로세스와는 달리 체계적인 아키텍처와 잘 정제되고 요구변화에 강건한 컴포넌트 설계기술을 전제로 한다. 또한 잘 정의된 아키텍처는 전통적인 코드 재사용에 비하여 컴포넌트와 아키텍처 디자인에 대한 광범위한 재사용을 가능하게 함으로써 실질적인 생산성 향상과 품질 수준 유지를 가능하게 한다. 그 동안 하위의 클래스에서 상위의 컴포넌트를 도출해가는 상향식(bottom-up) 설계방법 및 패턴중심의 설계기법이 제시되어 왔다. 그러나 이와 같은 기법은 기술적인 부분에는 중점적으로 고려하였으나 비즈니스 전략 및 시스템 변화요건과 같은 상위수준의 요구사항을 아키텍처 설계에 반영할 수 있는 체계적인 지원이 미흡하였으며 결과적으로 비즈니스 변화에 따른 아키텍처 변경에 대한 지속적이고 효과적인 관리에 어려움이 많았다.

ATAM(architecuture trade-off analysis method)은 비즈니스 목표 및 상위수준의 품질속성(재사용성, 성능, 보안, 유지보수성, 확장성 등) 요구사항을 기준으로 아키텍처 결정사항에 대한 분석·평가를 실시하는 대표적인 아키텍처 분석 방법으로 CMU/SEI 에서 제시하고 있는 방법이다.[18] ATAM 은 시스템 초기에 아키텍처와 관련한 위험요인을 도출하고, 이에 대한 위험완화계획 수립 및 조치를 가능케 함으로써 성공적인 시스템 개발을 위한 최적의 아키텍처를 결정할 수 있도록 한다.

이에 따라 본 논문에서는 ATAM 을 적용한 새로운 소프트웨어 아키텍처 설계 기법으로 마르미-III의 아키텍처 설계방안을 제시하고 이에 대한 적용사례를 제시한다.[3, 5, 6, 8]

2. 컴포넌트 기반 아키텍처 개발

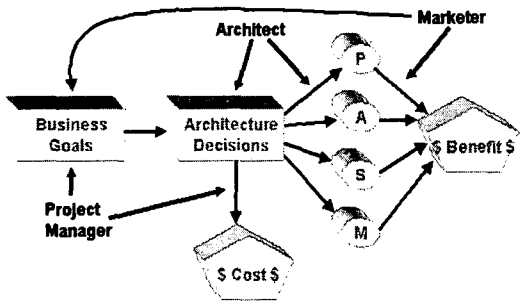
2.1 아키텍처 트레이드오프 분석 방법 (Architecture Tradeoff Analysis Method)

ATAM 은 비즈니스 목표를 감안하여 아키텍처 디자인 결정사항을 도출하고 이를 변경 용이성(modifia-bility), 보완성(security), 성능(performance) 및 가용성(availability)과 같은 품질 속성(quality attribute) 요구사항에 비추어 아키텍처를 분석·평가하는 아키텍처 분석 방법이다.

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 이해당사자간의 의사소통을 증가시킴으로써 시스템의 품질 및 생산성을 향상시킬 수 있다.

아래 [그림 1]은 ATAM 의 구조를 나타낸다.



[그림 1] ATAM 구조

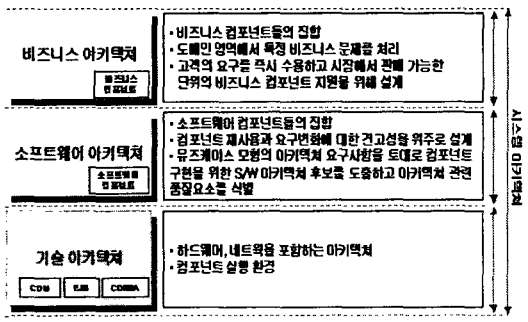
이러한 ATAM 의 세부 절차는 아래 [표 1]과 같다.

[표 1] ATAM 의 세부절차

단 계	상 세 작 업
제시 (Presentation)	ATAM 절차 소개
	비즈니스 드라이버 식별
	아키텍처 제시 및 설명
조사 및 분석 (Investigation & Analysis)	아키텍처 의사결정요소 식별
	품질속성 시나리오 도출과 유틸리티 트리 작성
	아키텍처 의사결정요소 분석
테스트(Testing)	시나리오를 구분하고 우선순위를 부여하고 의견을 수렴
	아키텍처 의사결정요소 분석
보고(Report)	최종 결과를 작성하여 보고

2.2 ATAM 을 적용한 아키텍처 설계방안

아키텍처에는 여러 관점의 뷰를 정의할 수 있다. 본 논문에서는 시스템 아키텍처를 비즈니스 아키텍처, 소프트웨어 아키텍처 및 기술 아키텍처로 나누었으며 아래 [그림 2]에서와 같다.



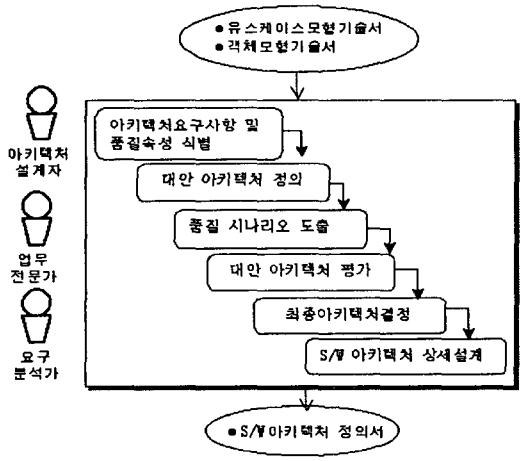
[그림 2] 아키텍처 분류

소프트웨어 아키텍처는 기술 아키텍처 위에서 비즈니스 컴포넌트의 수행을 지원하는 역할을 수행한다. 기술 아키텍처는 하드웨어, 네트웍을 포함하며 컴포넌트의 실행환경을 제공한다.

이러한 소프트웨어 아키텍처 구성상에서 재사용성이 높은 비즈니스 컴포넌트를 개발하기 위해서는 범용성이 있으면서 확장가능하고 운영체제와 독립적인 소프트웨어 아키텍처 위에서 구현해야 한다. 비즈니스 아키텍처는 시스템의 특성과 업무영역에 따라 다양한 구조로 구성되게 하며, 이에 따라 소프트웨어 아키텍처 또한 도메인에 따라 표준적인 구성이 달라지게 된다.

안정적인 비즈니스 아키텍처와 소프트웨어 아키텍처를 구축하기 위해서는 지속적인 반복개발과 경험축적이 필요하고 소프트웨어 아키텍처 설계 절차는 비즈니스 전략 및 향후 발생 가능한 변경 요구사항을 고려할 수 있어야 한다.

따라서 본 논문에서는 주요 이해당사자가 참여하고 이해할 수 있는 아키텍처 설계 절차로써 품질 시나리오 작성 및 아키텍처 평가 등과 같은 실제적인 접근 방법을 제시하고 있으며 그 흐름도는 아래 [그림 3]과 같다.



[그림 3] 소프트웨어 아키텍처 설계 절차

첫째, 아키텍처 요구사항을 파악 및 품질속성(modifiability, availability, performance, maintainability, scalability, flexibility 등)을 식별하고 품질속성별로 구체적인 품질 시나리오를 작성한다. 이 때, 품질 시나리오에는 시스템에 대한 기능 요구사항(use case)뿐만 아니라 비즈니스 요구사항 및 예측 가능한 업무환경 변화(change case) 등이 포함되어야 한다. 품질 시나리오를 작성한 뒤에는 품질 속성 및 품질 시나리오에 대하여 우선순위를 부여하고 이를 유틸리티 트리(utility tree)로 작성한다.

둘째, 아키텍처 설계자는 아키텍처 요구사항을 만족할 수 있는 소프트웨어 아키텍처 후보를 도출하여야 한다. 이 때 프레임워크 및 아키텍처 스타일을 적

용할 수 있으며, 아키텍처 리파지토리가 구축되어 있는 경우 해당 정보를 이용할 수 있다.

셋째, 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 의사결정이 보다 우선순위가 높은 품질속성에 대한 요구사항을 만족할 수 없도록 하는 것을 방지할 수 있다.

넷째, 아키텍처 분석· 평가가 끝난 뒤에는 평가 결과를 반영하여 최종 소프트웨어 아키텍처를 정의한다.

3. 컴포넌트 기반 아키텍처 설계 사례

3.1 사례개요

A 은행의 전산 시스템 중 고객정보관리 시스템을 컴포넌트 기반 소프트웨어 개발방법을 적용하여 웹기반으로 재구축하는 것으로, 기존의 클라이언트-서버 아키텍처의 분산된 고객정보를 통합하여 고객과 은행 사이에 이루어지는 모든 활동에 필요한 정보를 통합 관리함으로써 효율적인 고객관리 및 서비스 제공을 지원하는 시스템이다.

3.2 상세 적용사례

사례 프로젝트의 소프트웨어 아키텍처를 설계하기 위하여 2.2 절에서 제시한 설계방법을 준수하고 각 단계별로 필요한 활동을 수행하였다. 다음은 사례 프로젝트에서 수행한 작업 및 결과를 구체적으로 나타낸 것이다.

● 아키텍처 요구사항 및 품질속성 식별

첫번째 단계로서 아키텍처에 대한 요구사항을 도출하고 아울러 품질 속성을 식별하였다. 고객정보의 중요성이 강조됨에 따라 " 보안(security)" 이 가장 중요한 아키텍처 요구사항으로 도출되었다. 또한, 업무 프로세스가 제도변경, 비즈니스 환경 변화에 따라 수시로 바뀔 수 있으므로 재사용성과 확장성이 중요한 요구사항으로 제기되었다.

● 대안 아키텍처 정의

사례 프로젝트에서 웹 어플리케이션을 도입함으로써 기본적인 성능 및 보안의 요구사항을 수용하도록 하였다. 그러나 재사용성, 확장성, 유지보수성 및 대체성을 만족시키기 위해서는 보다 구조화된 아키텍처에 대한 필요성이 제기되었다. 이에 따라 순수 계층형 아키텍처(layered architecture)에서의 재사용성을 높이는 장점은 수용하고 어플리케이션에서 불필요하게 객체간의 호출이 증가하여 발생하는 성능 저하 문제는 보완하여 변형된 계층형 아키텍처를 작성하였다.

● 품질 시나리오 도출

앞에서 도출된 아키텍처 요구사항 및 품질 속성에 대하여 주요 이해당사자와의 인터뷰를 통하여 중요도를 파악하고 최종적으로 품질 속성에 대한 우선순위와 시나리오는 다음과 같다.

(1) 보안성
- 내부 및 외부의 정보보호 및 통신망 보호 규정을 준수할 수 있는 높은 수준의 보안을 제공해야 한다.

(2) 재사용성
- 제도변경에 따른 모듈 또는 업무 로직 변경시 기존에 작성한 컴포넌트 또는 모듈을 재사용할 수 있어야 한다.

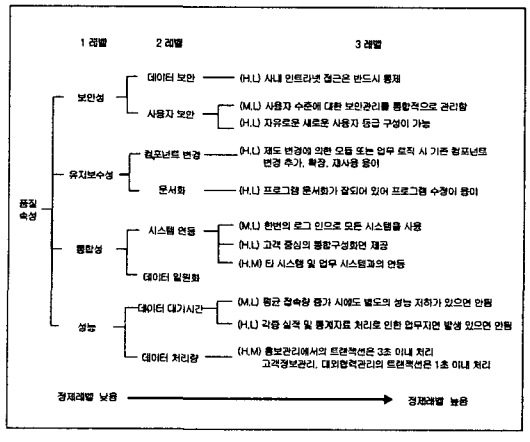
(3) 확장성
- 여신 상품 추가 등과 같은 새로운 업무의 추가 및 업무 규모가 커지는 경우 모듈이나 컴포넌트 추가, 필요한 경우 시스템 확장이 쉽게 가능해야 한다.

(4) 유지보수성
- 업무간의 의존성을 적게 설계하여 변경에 대하여 해당 컴포넌트만 수정할 수 있도록 한다.

(5) 대체성
- 이자계산 모듈의 변경, 여신상품개발, 제도변경 등에 의하여 기존에 작성한 컴포넌트를 새로운 컴포넌트로 교체해야 하는 경우 별도의 코드수정 없이 새로운 컴포넌트로 바꿀 수 있어야 한다.

(6) 성능
- 하나의 트랜잭션은 1 초이내에 처리되어야 한다.

이러한 품질 속성 및 품질시나리오를 토대로 유틸리티 트리를 작성하였다.[그림 4]



[그림 4] 사례 프로젝트에서의 유틸리티 트리

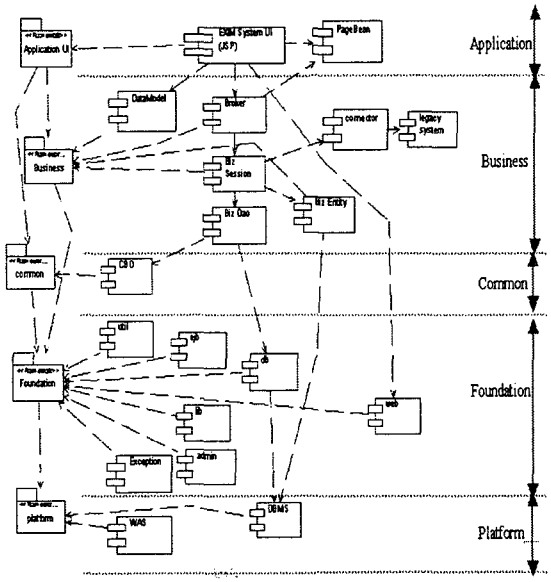
● 대안 아키텍처 평가

계층간의 인터페이스를 정의하여 각 계층을 서브시스템(subsystem)화 하기에는 특정 UI 컴포넌트 및 확장된 웹 어플리케이션 서버와 같은 컴포넌트의 기

능을 수용하는데 어려움이 있어 일정부분만 기본적인 계층형 구조를 유지하도록 정의한 변형된 아키텍처에 대하여 품질 속성별로 평가를 실시하였다.

● 최종 소프트웨어 아키텍처

아키텍처 평가 결과에 따라 대안 아키텍처중에서 최종 소프트웨어 아키텍처를 결정하고 컴포넌트 모형, 클래스 모형 등을 포함하여 다음 [그림 5]와 같은 소프트웨어 아키텍처를 기술하였다.



[그림 5] 최종 소프트웨어 아키텍처

4. 결론

본 논문에서는 아키텍처의 기술적인 요소를 중심으로 한 기존의 아키텍처설계 기법의 단점을 보완하기 위하여 ATAM 을 적용한 새로운 소프트웨어 아키텍처 설계기법을 제시하였다. 본 연구에서 제시한 아키텍처 설계기법을 적용하기 위해서 프로젝트 이해당사자의 적극적인 참여와 이해, 경험 있는 아키텍처 설계자가 필요하다는 것을 알 수 있다. 비록 이러한 요소들이 모두 준비되는 것이 어렵다고 하더라도 우리는 다음과 같은 이점을 얻을 수 있다.

첫째, 비즈니스 전략 및 향후 시스템에 대한 변경 요건(change case)을 고려함으로써 비즈니스 환경 및 사용자 요구사항 변화에 유연하게 대응할 수 있는 아키텍처를 설계할 수 있다.

둘째, 품질 속성 식별 및 품질 시나리오 작성시 중요도에 따라 우선순위를 부여하고 또한 품질속성간 트레이드오프를 분석·평가함으로써 시스템 개발 초기 지속적으로 아키텍처에 대한 위험관리를 가능하게 한다.

셋째, 컴포넌트 중심 개발 프로세스에 적용함으로써 광범위한 재사용을 통한 생산성 및 효율성 향상을 기대할 수 있다.

참고문헌

[1] 나희동 " 컴포넌트 기반 개발 프로세스 성숙도 모형설계" , 한국 SI 학회 2002.7.

[2] 나희동, 박현철, 김정아, 조은숙, " 알기쉬운 컴포넌트" , 한국소프트웨어컴포넌트컨소시엄, 2002.2.

[3] 나희동, " Architecture driven Component Development" : SoftEXPO, 2001.11.

[4] 박준성, " CBD 도입전략" , 2001. 한국소프트웨어컴포넌트컨소시엄.

[5] 박창순 외, 컴포넌트기반 시스템 개발방법론 마르미-III v1.0, http://www.component.or.kr/, 2001.8.

[6] 박창순 외, 마르미-III v2.0 절차서, 2002.6.

[7] 오영배, 나희동, 박준성, 백두권, " 컴포넌트 기반 개발프로세스" , 한국정보과학회 소프트웨어공학회지 2002. 5.

[8] 조진희 외, " 컴포넌트 기반 시스템 개발 방법론 마르미-III" , 프로젝트관리기술 논문집, 2001. 11.

[9] Gartner Group, "2001 Hype Cycle of Emerging Trends and Technologies", 2001.07.02

[10] Bass, L.; Clements, P.; & Kazman, R. Software Architecture in Practice. Reading, MA: Addison Wesley, 1998.

[11] Clements, P. & Northrop, L. " A Framework for Product Line Practice." Version 2.0, July 1999.

[12] Kazman, R.; Barbacci, M.; Klein, M.; Carriere, S.J.; & Woods, S.J." Experience with Performing Architecture Tradeoff Analysis." Proceedings of ICSE99. Los Angeles, CA, May 1999.

[13] R.Kazman, " The Architecture Tradeoff Analysis Method"

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

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

[16] D.Garlan and M.Shaw, "An Introduction to Software Architecture", Advances in Software Engineering and Knowledge Engineering"

[17] M.Shaw and D.Garlan, "Software Architecture : Perspectives on an Emerging Discipline", Prentice Hall, 1996

[18] L.Bass and R.Kazman, " Architecture-Based Development" , Technical Report CMU/SEI-99-TR-007, April 1999