소프트웨어 아키텍처 설계 단계에서 아키텍처 접근법 선정을 위한 평가 방법

고 현 희[†] · 궁 상 환[††] · 박 재 년[†††]

요 약

시스템의 소프트웨어 아키텍처를 설계하기 위해서는 여러 아키텍처 스타일들이 복합적으로 결합하여 시스템의 전체적인 아키텍처를 구성하게 된다. 이 때 다양한 아키텍처 스타일 즉, 아키텍처 접근법들 중 어떤 것을 선택할 것인가는 완성될 시스템이 어떤 기능적, 비 기능적 품질요구사항을 만족시켜야 하는지에 따라 달라지게 된다. 본 논문에서는 아키텍처 접근법 선정을 위한 평가모델을 기반으로 아키텍처 접근법의 기술적인(technical) 부분에 대한 평가를 통해 시스템의 요구사항에 대한 만족도를 결정하여 가장 적합한 아키텍처 접근법을 선정하도록 하는 평가 방법을 정의하고, 메시지 시스템의 아키텍처 접근법을 선정 시 본 평가 방법을 적용하는 사례연구를 통해 본 평가 방법을 검증한다.

키워드 : 소프트웨어 아키텍처, 아키텍처 평가 방법, 아키텍처 접근법, 아키텍처 스타일

Evaluation Method to Choose Architectural Approaches in the Software Architecture Design Phase

Koh Hyon Hee[†] · Kung Sang Hwan[††] · Park Jae Nyon[†††]

ABSTRACT

To design a software system, many architecture styles have to be combined to construct the overall architecture of the system. What to choose among various architecture styles or architectural approaches depends on the fact of what kind of functional or non-functional quality requirements the system should satisfy

In this study, we define the method to choose suitable architectural approaches by the satisfaction level of system requirements that is evaluated through estimation about technical parts of architectural approaches, and verify the evaluation method by the case study that apply the evaluation method to choose architectural approaches for message system.

Key Words : Software Architecture, Architecture Evaluation Method, Architectural Approach, Architectural Style

1. 서 론

복잡하고 방대해진 시스템 개발에서 소프트웨어 아키텍처는 데이터 구조나 알고리즘의 선택보다 중요한 부분이 되고 있다. 즉 다양한 이해 관계자들의 요구사항을 시스템에 정확히 반영해야 하고, 이를 위해 시스템의 품질 속성과 이해관계자들의 이해관계를 반영한 소프트웨어 아키텍처의 설계가 성공적인 프로젝트를 위한 중요한 이슈가 되었다.

이에 따라 성공적인 아키텍처 설계를 위한 다양한 방법과, 설계된 아키텍처의 적합성을 판단하기 위한 평가 방법이 연구되어지고 있다.

시스템의 아키텍처를 설계하기 위해서는 여러 아키텍처 스타일들이 복합적으로 결합하여 시스템의 전체적인 아키텍처를 구성하게 된다. 이 때 다양한 아키텍처 스타일 즉, 아키텍처 접근법들 중 어떤 것을 선택할 것인가는 완성될 시스템이 어떤 기능적, 비기능적 품질요구사항을 만족시켜야 하는지에 따라 달라지게 된다. 즉 아키텍처 선정 시는 그 시스템의 요구사항에 따라 아키텍처를 검증하고 품질 요구사항을 포함하는 요구사항을 만족할 수 있는 아키텍처를 선정해야 한다.

아키텍처 접근법 선정을 위한 방법에는 CBAM(Cost Benefit Analysis Method)에서의 '투자대비효과(Return Of Investment)' 우선순위에 의해 선정하는 방법이 있다. 이 방법은 아키텍처 접근법의 각 시나리오에 대한 효용을 계산하고, 이를 바탕으로 아키텍처 접근법의 이득과 비용을 계산하여 비용대비 이득이 높은 아키텍처 접근법을 선정하는 방식이다[3].

※ 본 연구는 숙명여자대학교 2003년도 교내연구비 지원에 의해 수행되었음.
† 준 회 원 : 숙명여자대학교 컴퓨터과학과 박사과정
†† 정 회 원 : 천안대학교 정보통신학과 조교수
††† 정 회 원 : 숙명여자대학교 이과대학 학장
논문접수 : 2005년 2월 1일, 심사완료 : 2005년 5월 10일

CBAM은 주로 ATAM(Architecture Tradeoff Analysis Method) 평가가 끝난 후 ATAM 평가의 산출물을 기반으로 하여 이루어지는 평가 방법으로 설계자들이 설계 과정 중에 아키텍처의 평가를 위해 활용하기가 적합치 않다[11]. 또한 아키텍처 설계단계에서 설계자가 비용에 대한 고려를 하기가 어렵고, 아키텍처 접근법들 간에 비용대비 효과의 차이가 크지 않을 경우 시스템이 요구하는 품질 속성을 얼마나 충족시키는지에 따른 만족도가 더 중요하다고 볼 수 있다.

본 논문에서 제안하는 아키텍처 접근법 선정을 위한 평가 방법은 설계자들이 아키텍처 설계 단계에서 활용할 수 있는 방법으로, 아키텍처 설계 과정에서 식별된 아키텍처 접근법 대안들에 대해 품질 속성 우선순위에 따라 평가 모델을 만들어 시스템의 요구사항에 대한 만족도를 평가하는 과정을 통해 가장 적합한 아키텍처 접근법을 선정한다.

본 논문의 구성은 다음과 같다.

2절에서는 연구를 수행하기 위한 관련 연구로서 아키텍처 평가 방법인 ATAM과 CBAM에 대해 살펴보고, 3절에서는 본 논문에서 제안하는 소프트웨어 아키텍처 접근법 선정을 위한 모델을 제시한다. 4장에서는 제안한 평가 모델에 의한 상세 절차를 설명한다. 5장과 6장에서는 사례연구를 바탕으로 연구결과에 대해 분석해 보고 마지막으로 7장에서는 결론 및 향후 연구 과제를 제안하도록 한다.

2. 관련 연구

ATAM과 CBAM은 설계과정이 끝난 후 설계 산출물을 가지고 이루어지는 평가 과정으로 평가 내용과 주요 절차는 다음과 같다.

2.1 아키텍처 트레이드오프 분석방법 (ATAM)

ATAM(Architecture Tradeoff Analysis Method) 은 품질 속성 요구사항과 비즈니스 목표달성을 위한 아키텍처 결정 사항들을 평가한다[4]. 즉 ATAM은 시나리오를 중심으로 품질 속성요구사항을 찾아내고 아키텍처가 특정 품질 속성들을 만족하는지를 분석하는데, 다양한 이해 관계자들과 함께 위험요소(risk), 민감점(sensitivity point), 상충점(tradeoff point) 등을 분석한다[4].

이렇게 분석된 품질목표들 간의 발생하는 충돌을 프로젝트 초기에 발견함으로써 경제적인 방법으로 문제를 해결할 수 있다.

ATAM은 <표 1>에서 처럼 4개의 그룹과 9개의 스텝으로 이루어져 있으며, 평가팀과 이해관계자들, 프로젝트 의사결정권자들이 모여 아키텍처 전반에 대해 평가를 하게 된다.

ATAM 평가의 결과로는 아키텍처 접근법의 상충점, 위험요소, 민감점에 대한 분석 뿐만 아니라 새로운 아키텍처 접근법의 대안도 제시하게 된다.

2.2 비용과 이득을 고려한 평가 방법 (CBAM)

아키텍처에는 달성해야 하는 품질 속성에 대한 기술적인 측면과 시스템을 구축할 때 드는 비용과 시스템이 제공하는 품질에서 얻을 수 있는 이득에 대한 경제적인 측면이 있다[3].

〈표 1〉 ATAM 평가 절차

그 룹	스 탭
소개 (Presentation)	1. ATAM 소개
	2. 비즈니스 드라이버 소개
	3. 아키텍처 소개
조사와 분석 (Investigation and Analysis)	4. 아키텍처 접근법 식별
	5. 품질속성 유틸리티 트리 만들기
	6. 아키텍처 접근법 분석
시험 (Testing)	7. 브레인스토밍과 시나리오 우선순위 매기기
	8. 아키텍처 접근법 분석
보고 (Reporting)	9. 결과 보고

비용과 이득을 고려한 아키텍처 평가 방법인 CBAM (Cost Benefit Analysis Method)은 아키텍처 접근법을 실현하는 데 필요한 비용과 아키텍처 접근법을 적용했을 때 달성할 수 있는 품질 속성이 가져다주는 이익을 측정한다. 즉, CBAM은 비용과 이익으로부터 '투자대비효과(ROI)'를 계산하여 수익이 최대가 될 수 있도록 의사 결정을 지원한다[2].

CBAM은 주로 ATAM 평가가 끝난 후 새롭게 제시된 대안 아키텍처 접근법 들에 대해 ROI를 계산하여 대안 아키텍처 접근법들 중 ROI가 높은 접근법 대안을 선택할 수 있도록 하고 있다[11].

CBAM의 실행 절차는 <표 2>와 같다[3].

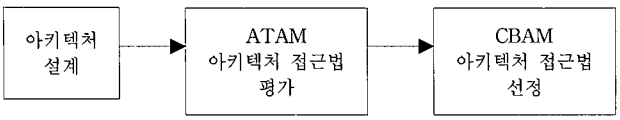
〈표 2〉 CBAM의 실행 절차

시나리오 결정	1. 시나리오를 수집한다.
	2. 시나리오를 정제한다.
	3. 시나리오의 우선순위를 결정한다.
효용-반응값 곡선 작성	4. 선별한 시나리오의 효용-반응값 곡선을 작성한다.
아키텍처 접근법 전체 이익 계산	5. 시나리오를 담당하는 아키텍처 접근법을 찾아서 예상 반응값을 결정한다.
	6. 아키텍처 접근법의 예상효용을 계산한다.
	7. 아키텍처 접근법의 전체 이익을 계산한다.
아키텍처 접근법 선정과 검증	8. 아키텍처 접근법의 ROI를 계산하여 순위를 결정한다.
	9. 비용과 일정을 고려해서 아키텍처 접근법을 선정하고 결과를 검증한다.

2.3 ATAM과 CBAM에 의한 아키텍처 평가 방법의 문제점

(그림 1)에서와 같이 기존의 아키텍처 평가 방법은 설계가 끝난 후 아키텍처 평가 과정을 거치게 된다.

ATAM은 아키텍처 설계 시 사용된 아키텍처 접근법을 찾아서 그 접근법의 위험요소나 상충점들을 분석하고 새로운 아키텍처 접근법 대안을 제시한다. CBAM은 ATAM에서 찾아진 아키텍처 접근법과 분석된 위험요소나 상충 요인들을 고려한 아키텍처 접근법 대안들을 가지고 품질 요구 사



(그림 1) 기존의 아키텍처 설계 및 평가

항에 미치는 효용과 비용을 고려하여 가장 적합한 아키텍처 접근법을 찾게 된다[11].

이때 선정된 아키텍처 접근법이 설계 시 사용한 아키텍처 접근법과 다를 경우 다시 설계 과정을 다시 시작하여야 하기도 하고, 품질 요구사항을 만족시키는 접근법이 없을 경우 다시 아키텍처 동인을 재조정 하여야 하는 과정으로 되돌아와야 한다.

이 과정에서의 문제점은 아키텍처 설계가 완료된 이후에 아키텍처 평가팀을 구성하여 개발조직과의 합의를 통해 아키텍처를 평가해야 하는 과정에 드는 비용과, 다시 설계과정을 재 반복해야 한다는 것이다. 즉 아키텍처 설계자나 개발자가 시스템 전체 차원이 아니라 자신이 맡고 있는 서브시스템이나 모듈 단위로 아키텍처를 검증해 볼 수 있는 소규모 단위의 아키텍처 평가 방법론으로 활용하기 힘들고, 아키텍처 설계가 끝난 후 평가가 이루어져야 하는 문제점이 있는 것이다.

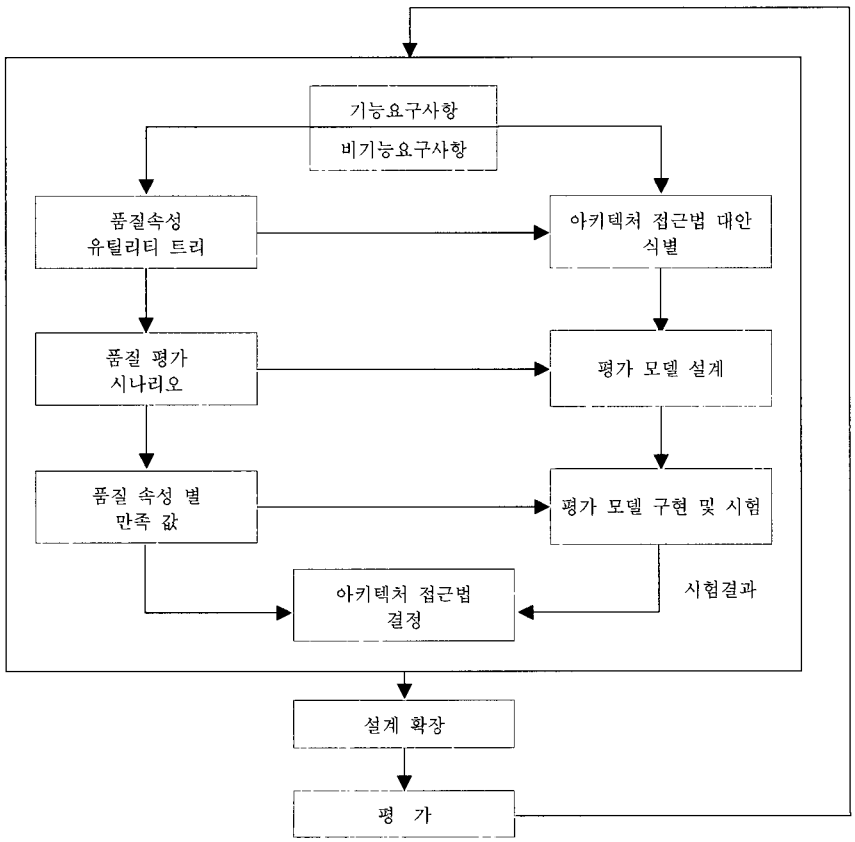
따라서 설계 단계에서 품질 요구사항을 고려한 아키텍처 접근법 대안에 대한 고려를 하고 아키텍처 접근법을 선정할 때 충분한 평가 과정을 거쳐 가장 적합한 아키텍처 접근법을 찾는 다면 설계된 아키텍처의 완전성과 신뢰성이 높아질 뿐만 아니라 설계 이후의 평가과정에서 새로운 아키텍처가 선정되는 사례도 감소할 것이다.

3. 아키텍처 접근법 선정을 위한 평가 모델

아키텍처 설계는 시스템이 만족해야 하는 기능요구사항과 품질 속성요구사항을 만족시키도록 설계되어야 한다. 아키텍처 설계를 위한 아키텍처 접근법은 품질 속성 우선 순위에 따라 우선순위가 높은 속성을 만족시키는 아키텍처 접근법을 먼저 식별해 나가게 된다. 즉 하나의 품질 속성 요구사항을 만족시키기 위한 아키텍처 접근법을 식별하고 다음으로 다른 품질 속성 요구사항을 만족시키는 아키텍처 접근법을 찾는 과정을 반복함으로써 아키텍처 설계를 확장해 나가게 된다[12].

본 논문에서는 설계 과정 중에 식별된 아키텍처 접근법 대안들에 대한 품질 평가 모델을 만들고, 이를 가지고 달성하고자 하는 품질 속성 요구사항에 대한 객관적 검증을 통해 아키텍처 접근법을 선정하고 설계를 발전시켜 나가는 설계 단계에서의 아키텍처 접근법 선정을 위한 평가 방법을 정의한다.

설계 단계에서의 아키텍처 접근법 선정을 위한 모델은 다음의 (그림 2)와 같다.



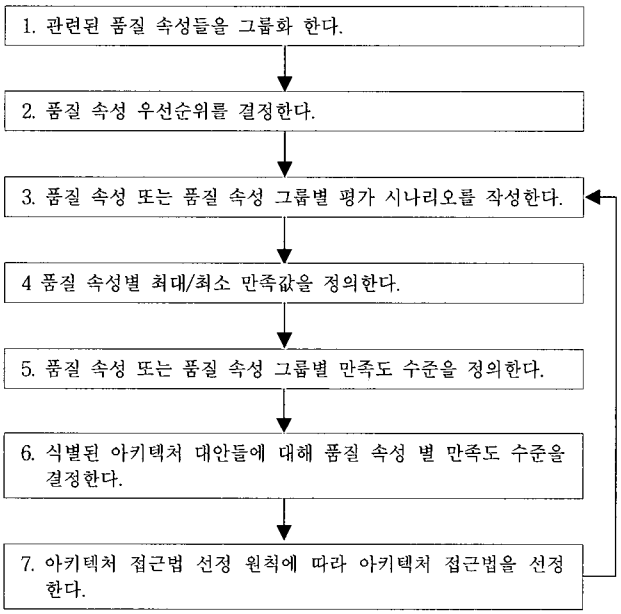
(그림 2) 아키텍처 접근법 선정을 위한 평가 모델

시스템의 기능 요구사항과 비 기능 품질 요구사항으로부터 아키텍처 동인을 얻어 아키텍처 접근법을 식별하게 되고, 아키텍처 접근법이 여러 개의 대안이 존재할 경우 각 대안에 대한 평가 과정을 거쳐 해당 품질 속성에 가장 적합한 하나의 아키텍처 접근법을 선정하게 된다. 이 과정을 만족해야 하는 품질 속성들에 대해 반복하여 아키텍처 접근법들을 선정해 나가고, 이렇게 선정된 아키텍처 접근법에 기능을 할당하여 아키텍처를 확장 설계하고, 설계된 아키텍처에 대해서는 다시 한번 평가 과정을 거쳐 최종적으로 설계를 완성하게 된다.

4. 아키텍처 접근법 선정 절차

아키텍처 접근법 선정을 위한 평가 모델에 의한 평가 과정은 (그림 3)과 같다.

각 단계별 세부 실행 내용은 다음과 같다.



(그림 3) 아키텍처 접근법 선정을 위한 평가 절차

〈표 3〉 {확장성, 성능} 품질 속성 그룹의 만족도 수준 표

품질 속성 그룹 \ 수준	수준 1	수준 2	수준 3	수준 4
{확장성, 성능}	{10,000,1500}	{10,000,1,000}	{10,000, 500}	{10,000, 200}
{확장성, 성능}	{5,000, 3500}	{5,000, 2,500}	{5,000, 2000}	{5,000, 1000}

※ 수준1: 매우만족, 수준2: 만족, 수준3: 충분, 수준4: 부족

4.1 관련 품질 속성 그룹화

품질 속성들 간에는 상호 종속적이 될 수 있다. 예를 들어 확장성과 성능의 경우, 확장성을 높이기 위한 설계가 성능을 저하시킬 수 있고, 성능을 높이기 위한 설계는 확장성을 저하 시킬 수 있다. 이 경우 하나의 품질 속성은 결국 다른 품질 속성 값의 결정에 영향을 줄 수 있다. 따라서 각각의 품질 속성에 대해 평가되기 보다는 두개의 품질 속성을 하나로 그룹화 하여 두 품질 속성간의 적정 값으로 선정되도록 평가되어야 한다. 예를 들어 최대 동시 사용자 수가 10,000명인 경우, 성능은 초당 1,000개의 트랜잭션을 처리할 수 있고, 최대 동시 사용자 수가 5,000명인 경우, 성능은 초당 2,500개의 트랜잭션을 처리해야 하는 만족도 수준을 가지고 있는 시스템의 경우 <표 3>의 만족도 수준 표와 (그림 4)의 그래프로 만족도 수준을 나타낼 수 있다.

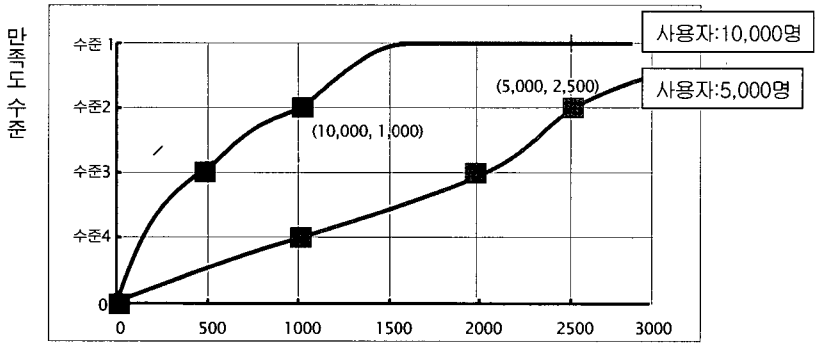
4.2 품질 속성 우선 순위 결정

품질 속성 우선순위는 요구사항 분석단계에서 산출된 품질 속성 유틸리티 트리에서 결정된다고 볼 수 있다. 만약 결정이 되지 않았다면 이해관계자들에 의해 시스템이 만족시켜야 할 품질 속성의 우선순위를 결정하도록 한다.

이렇게 결정된 우선순위에 따라 아키텍처 설계를 위한 아키텍처 접근법 식별이 이루어지게 된다. 즉 가장 중요한 품질 속성을 만족시키기 위한 아키텍처 접근법이 먼저 선정이 되고, 다음으로 다른 품질 속성들을 위한 아키텍처 접근법들을 선정해 나가면서 아키텍처 설계를 발전 시켜 나가게 된다.

4.3 평가 시나리오 작성

시스템이 만족해야 할 품질 속성과 품질 속성 그룹별로



(그림 4) 품질 속성 그룹의 만족도 수준

평가 시나리오를 작성한다. 비 기능 즉 품질 요구사항 시나리오를 참조하여 작성하고, 이 때 작성된 평가 시나리오는 평가 모델을 만들기 위한 기반이 된다.

예를 들어 성능을 평가하기 위해 시험해보아야 할 내용으로 즉 단위 시간당 메시지 처리량, 또는 트랜잭션 처리량 등을 시나리오로 작성할 수 있다.

4.4 품질 속성별 최대, 최소값 결정

품질 속성 별로 만족해야 하는 최대 값과 최소 값을 결정한다. 이 값은 해당 시스템이 만족해야 하는 품질 속성의 최대, 최소 한계 값으로 아키텍처 접근법의 시험 결과 값이 이 한계 값을 넘는 경우 해당 아키텍처 접근법은 선정될 수 없다.

품질 속성값의 결과를 최대/최소로 결정 할 수 없고, 만족/불만족으로 결정되는 경우도 있다. 이 경우 시험 결과 값이 불만족 값을 갖는 경우 그 아키텍처 접근법도 선정 대상에서 제외한다.

4.5 만족도 수준 정의

품질 속성 또는 품질 속성 그룹에 대해 평가 시나리오의 결과 값에 대한 만족도 수준을 정의한다. 만족도 수준 결정에는 설계자와 시스템과 관련된 이해관계자들의 의견을 반영하여 결정한다. 이해관계자들의 의견이 완전히 같을 수는 없지만, 시스템의 목적에 대한 이해도가 유사하다고 볼 수 있으며, 조정과정을 통해 만족도 수준을 결정한다.

예를 들어 다음의 <표 4>와 같이 정의할 수 있다.

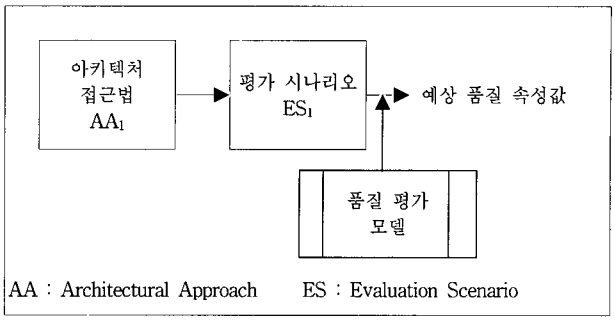
4.6 아키텍처 접근법의 예상 만족도 수준 결정

이 단계에서는 품질 속성들에 대한 아키텍처 접근법의 예상 만족도 수준을 결정한다. 아키텍처 접근법의 예상 품질 속성 값은 해당 품질 평가 시나리오에 대한 시험 모델의 구현과 시험을 통해 시험 결과를 가지고 시스템의 결과값을 예측해 볼 수 있다. 이 때 아키텍처 설계자와 개발자가 아키텍처 접근법의 시험 모델을 만들고 구현하여 그 결과를 가지고 가장 객관적인 값을 예측하도록 한다.

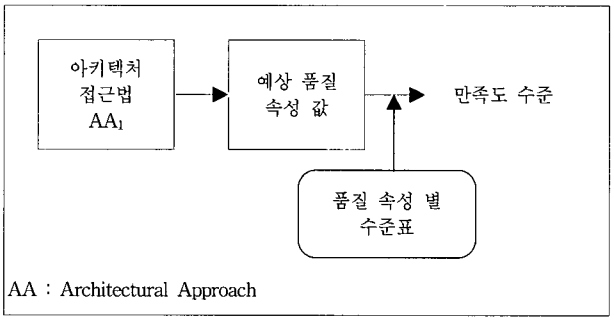
이렇게 예측된 값을 가지고 각 아키텍처 접근법 대안들의 품질 속성에 대한 만족도 수준을 결정한다.

4.7 아키텍처 접근법 선정 원칙

위의 모든 과정을 거쳐 아키텍처 접근법 대안들의 만족도



(그림 5) 예상 품질 속성 값 결정



(그림 6) 아키텍처 접근법의 만족도 수준 결정

수준을 결정했으면 그 결과를 가지고 아키텍처 접근법을 선정하여야 한다. 기본 원칙은 품질 속성별 결과 값이 최대/최소값 한계를 넘는 경우 선택에서 제외하며, 만족도가 가장 높은 아키텍처 접근법을 선택하도록 한다. 그러나 아키텍처 접근법의 만족도 결과에 관계없이 반드시 있어야 하는 아키텍처 접근법이 있을 수 있다. 예를 들어, 개발 환경의 특성이나 개발 언어의 특징에 의해 만족도가 낮더라도 개발의 편의성이나 시스템의 안전성을 위해 선택하여야 하는 아키텍처 접근법들이다. 이런 아키텍처 접근법들을 식별하여 우선 선정한다.

다음으로는 먼저 선정된 아키텍처 접근법과의 의존성을 파악하여 이미 선정된 아키텍처 접근법과의 의존성이 높은 경우 그 아키텍처 접근법 또한 우선적으로 선정하여야 한다.

위의 두 가지 제약에 의한 선정을 마친 후 나머지 아키텍처 접근법 대안들 중 높은 만족도를 나타내는 접근법을 선정한다.

위의 선정 절차를 다음과 같이 정리할 수 있다.

〈표 4〉 품질 속성 별 수준 표 예

수준 / 품질 속성	수준 1	수준 2	수준 3	수준 4
신뢰성	90	80	70	50
변경성	0.2M/M	0.5M/M	1M/M	2M/M
사용성	90	70	50	30
{확장성, 성능}	{10,000,1500}	{10,000,1,000}	{10,000, 500}	{10,000, 200}
{확장성, 성능}	{5,000, 3500}	{5,000, 2,500}	{5,000, 2000}	{5,000, 1000}

※ 수준 1: 매우만족(100), 수준 2: 만족(80), 수준 3: 충분(60), 수준 4: 부족(30)

① **반드시 있어야 하는 접근법 선택**
아키텍처 접근법의 만족도에 상관 없이 반드시 있어야 하는 접근법을 선택한다.

② **아키텍처 접근법들 간의 의존성 파악**
이미 선택된 아키텍처 접근법과의 의존도가 높은 경우 선택한다.

③ **최대, 최소값 한계를 넘는 접근법 제외**
품질 평가 시나리오의 시험 결과가 최대/최소값 한계를 넘는 경우 해당 접근법은 선정대상에서 제외한다.

④ **만족도가 높은 접근법 선택**
높은 만족도를 나타내는 접근법을 선택한다

5. 사례 연구

위의 평가 과정 적용을 위한 사례연구로 메시지 시스템의 품질 요구사항 중 우선순위가 높은 성능과 확장성 요구사항을 만족시키기 위한 동시성 뷰의 아키텍처 접근법 대안들 중에서 가장 적합한 접근법을 선정하는 과정을 보이고자 한다. 기타 메시지 시스템의 다른 요구사항과 다른 설계 뷰의 아키텍처 접근법에 대해서는 본 사례연구에서는 다루지 않는다.

메시지 시스템의 성능 요구사항은 서버에서 클라이언트로의 메시지 전달 속도가 일정하고 빨라야 한다는 것이고, 확장성 요구사항은 대규모 사용자를 고려할 때 동시사용자 수가 5000명 이상이어야 하고 동시 사용자 수에 상관없이 일정한 성능을 유지하기 위해 일정한 서버 자원을 유지하여야 한다는 것이다[1].

동시성 뷰의 아키텍처 접근법은 메시지 발행이나 구독을 위한 클라이언트로부터 요청이 들어올 때마다 복수의 클라이언트의 요구를 동시에 처리하기 위해 각 클라이언트에게 스레드를 할당하는 멀티 스레드 패턴을 참조한 (그림 7)의 아키텍처 접근법을 선택한다.

멀티 스레드 패턴은 멀티 스레드의 생성이나 처리 방법에 따라 여러 대안이 나올 수 있다. 따라서 동시성 뷰의 아키텍처 접근법에 대한 대안으로 다음과 같은 3개의 아키텍처 접근법을 식별할 수 있다.

(1) 출판이나 구독을 위한 클라이언트로부터 요청이 들어올 때마다 메시징 서버에서 클라이언트에게 서비스를 담당할 스레드를 하나씩 할당하는 클라이언트 별 스레드 할당(Thread-per-Client) 아키텍처[1]
(2) 여러 개의 스레드를 미리 가동해 두고 워커 스레드로서 기다리게 하는 스레드 풀(Thread Pool) 아키텍처[1]
(3) 서버측의 단일 스레드를 사용하는 단일 스레드(Single Thread) 아키텍처[6]

5.1 관련 품질 속성 그룹화

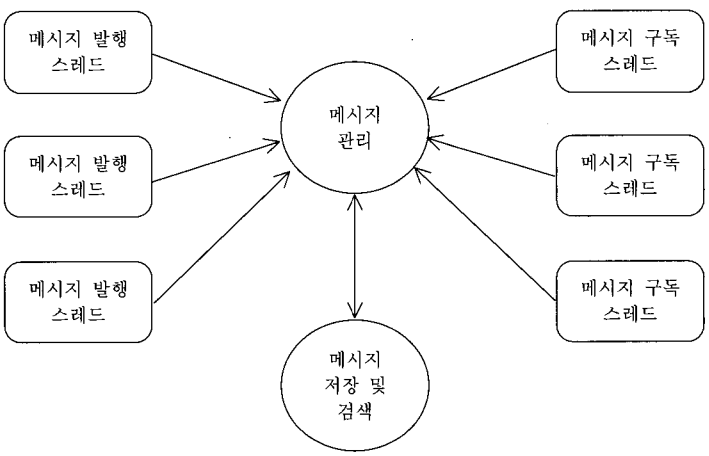
본 사례연구에서 다루고자 하는 성능과 확장성은 동시 사용자 수에 따라 성능이 영향을 받을 수 있는 속성이므로 {확장성, 성능}을 하나의 관련된 품질 속성으로 그룹화 할 수 있다.

5.2 품질 속성 우선 순위 결정

메시지 시스템의 품질 속성 요구사항은 본 사례연구에서 다루고자 하는 성능과 확장성 이외에 신뢰성, 가용성, 보안 등의 요구사항이 있다. 우선순위는 요구사항 분석단계에서 품질 속성 유틸리티 트리에 의해 결정이 되었다고 보고, 본 사례연구에서는 우선순위 결정을 위한 과정을 생략한다. 본 사례연구에는 우선순위가 높은 확장성과 성능 품질 속성에 대해 평가 과정을 적용한다.

5.3 평가 시나리오 작성

{확장성, 성능} 요구사항을 평가하기 위해서는 연결된 송신자, 수신자의 수에 따른 메시지 전달 속도를 측정해 보아야 한다. 연결된 송신자와 수신자는 메시징 서버의 스레드를 기동시키므로 서버측의 스레드 기동에 드는 시간과 서버 자원의 결과를 가지고 확장성을 예측 해볼 수 있고, 성능은 메시징 서버에서 기동된 스레드가 메시지를 생성해서 전달하는 속도를 가지고 예측해 볼 수 있다.



(그림 7) 메시지 시스템의 동시성 뷰 - 멀티 스레드 패턴

{확장성, 성능} 품질속성에 대한 평가 시나리오
ES1. 메시징 서버의 스레드 생성과 기동 시간 측정한다. ES2. 클라이언트 수에 따른 메시지 생성 및 전달 속도를 측정한다. −3개의 "one-to-one" 테스트를 실행한다. (10/10/10, 100/100/100, 1000/1000/1000) −테스트 메시지는 10MB의 파일을 512Bytes 메시지 단위로 전송하는 Byte형태의 메시지로 테스트한다. −메시지 전송 결과를 측정하는 과정을 5번 반복하고 그 평균값을 최종 결과로 한다.

따라서 평가 시나리오(Evaluation Scenario)는 다음과 같이 만들 수 있다.

"one-to-one" 테스트는 클라이언트가 짝을 이루는 즉 송신자/수신자가 짝을 이루어 하나의 토픽(topic)으로 메시지를 교환하는 것이다[10]. 예를 들어 "10/10/10"(10개의 송신자(publisher), 10개의 수신자(subscriber), 10개의 토픽) 테스트는 10개의 출판과 구독 쌍이 10개의 토픽으로 메시지를 교환하는 것이다.

5.4 품질 속성별 최대/최소값 결정

품질 속성별 최대/최소값은 각 이해관계자들이 모여 결정하여야 하나, 본 사례연구를 위해 {확장성, 성능} 품질 속성에 대한 최소값은 {1000, 200}으로 정한다. 즉 최소 1000명의 동시 사용자를 수용할 수 있어야 하고, 이때 메시지 전달 속도는 최소 200Msgs/sec를 전달하여야 한다.

5.5 만족도 수준 정의

만족도 수준도 이해 관계자들이 모여 결정하여야 하나, 대규모 사용자를 대상으로 할 때 메시지 서버의 동시 접속자 수가 5000명 이상이어야 하는 요구사항을 고려하고, 상용화된 메시지 시스템의 성능을 참조하여 본 사례연구에서는 <표 6>과 같이 정한다.

5.6 아키텍처 접근법의 예상 만족도 수준 결정

아키텍처 접근법 대안들의 예상 만족도를 결정하기 위해 먼저 평가 시나리오에 따라 평가 모델 작성과 구현이 이루어져야 하고 구현된 평가모델에 의한 평과 결과를 가지고 만족도를 결정한다.

〈표 6〉 만족도 수준 표

(단위 {명, Msgs/sec})

수준 / 품질 속성 그룹	수준 1	수준 2	수준 3	수준 4
{확장성, 성능}	{5000,8000}	{5000,6000}	{5000, 3000}	{5000, 1000}
{확장성, 성능}	{2000, 5000}	{2000, 3000}	{2000, 1000}	{2000, 500}

평가시나리오 ES1.에 대한 시험은 먼저 스레드를 생성시키고 start()를 호출하여 실제 스레드가 동작되기까지의 시간 소요를 프로그램으로 구현하여 실시하였다[1]. 그 결과는 다음과 같다[1].

(1) 1000개의 문자열을 만드는데 소요되는 시간 : 0.006 milisecond
(2) 1000개의 스레드를 만드는데 소요되는 시간 : 0.066 milisecond
(3) 1000개의 스레드를 만듬과 동시에 실행시키는 시간 : 2.433 milisecond

위의 결과로부터 메시지 시스템에서 스레드를 실행시키는 시간이 성능저하에 중요한 요인이 될 수 있음을 알 수 있다[1]. 즉 접속자 수가 늘어나 스레드 할당이 계속 증가할 경우 성능이 저하될 수 있다.

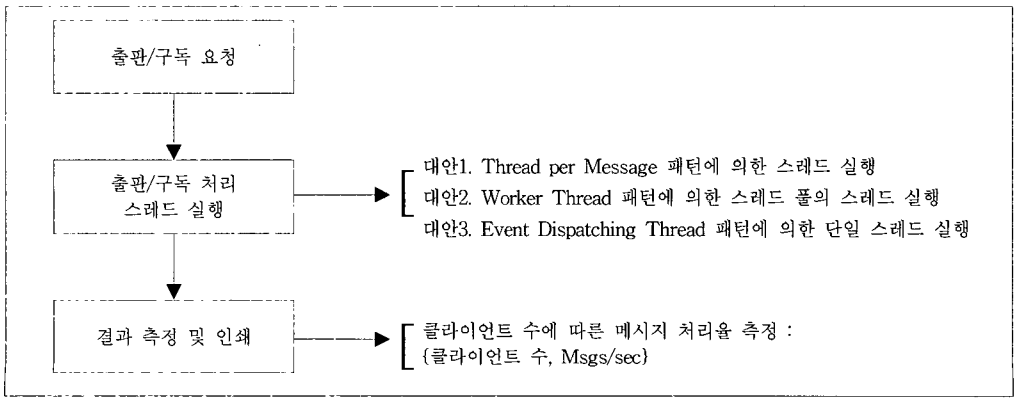
평가 시나리오 ES2.에 의해 {확장성, 성능} 품질 속성 그룹을 평가하기 위한 모델은 (그림 8)과 같고, 평가를 위한 구현은 각 참조 패턴을 활용하여 구현하였다.

구현된 평가 모델에 의한 시험 결과는 <표 7>과 같다.

〈표 7〉 메시지 전송 테스트 결과

아키텍처 접근법 / P/S/T	MsgSize (Bytes)	Msgs/sec		
		클라이언트별 스레드 할당	스레드 풀	단일 스레드
10/10/10	512	913	624	103
100/100/100	512	1406	4338	76
1000/1000/1000	512	1240	2145	18

(P/S/T : Publisher/Subscriber/Topic)



(그림 8) 평가 시나리오 ES2.에 의한 평가 모델

위의 표에 나타난 시험 결과를 볼 때 접속자 수가 작을 때는 클라이언트별 스레드 할당(Thread per Client) 아키텍처가 좋은 성능을 보여주고 있으나, 접속자의 수가 많아져서 스레드 할당이 많아지는 경우 스레드 풀(Thread Pool) 아키텍처의 성능이 좋아지는 걸로 볼 수 있다. 본 사례연구에서 스레드 풀의 스레드 수를 100개로 하였으나, 스레드 풀의 스레드 수를 다르게 할 경우 다른 결과를 보일 수도 있다. 스레드 풀의 스레드 개수를 100개로 한 경우 100/100/100의 상황에서 좋은 성능을 보여주고 접속자 수가 늘어남에 따라 성능이 저하되는 것을 볼 수 있다. 따라서 스레드 풀 아키텍처의 경우 스레드 풀의 스레드 수를 정하는 것이 중요한 요소라 볼 수 있다. 클라이언트별 스레드 할당(Thread per Client) 아키텍처의 경우는 접속자 수가 증가함에 따라 성능의 큰 차이는 보이지 않지만, 성능의 감소하는 추세를 보이는 것으로 보아 접속자 수가 많아짐에 따라 스레드 수가 증가하여 성능이 감소하고 있음을 예측할 수 있다.

위의 결과를 가지고 <표 8>과 같이 만족도 수준을 예측해 볼 수 있다.

〈표 8〉 만족도 수준 예측 결과

(단위: 명, Msgs/sec)

품질 속성 그룹 \ 아키텍처 접근법	클라이언트별 스레드 할당	스레드 풀	단일 스레드
{확장성, 성능}	{5000,800}	{5000,1500}	{5000, 10}
{확장성, 성능}	{2000, 1200}	{2000, 2000}	{2000, 20}

클라이언트별 스레드 할당(Thread per Client) 아키텍처의 경우 시험결과 접속자의 수가 증가할수록 성능이 조금씩 감소하는 결과를 보였다. 스레드 풀 아키텍처의 경우는 풀에 생성된 스레드의 수를 100개로 한 결과에 의해 위와 같은 예측 결과가 나왔으나 스레드의 수를 다르게 할 경우 더 좋은 성능을 보일 수 도 있다. 단일 스레드 아키텍처의 경우는 접속자의 수가 많아질수록 스케줄링에 의한 부하가 많아 성능이 감소하는 결과를 보이고 있다.

따라서 세가지 아키텍처 접근법 대안 중 스레드 풀 아키텍처가 가장 좋은 만족도를 보이고 있고, 풀에 생성된 스레드의 수를 다르게 하면 만족도가 더 높아질 수도 있다.

5.7 아키텍처 접근법 선정

테스트 환경이나 영속성(Durability), 신뢰성(Persistent)등의 속성에 따라 성능에 영향을 미칠 수 있으므로 결과는 달라질 수 있다. 그러나 본 사례연구에서는 확장성과 성능만을 고려하고 있으므로 평가모델에 의한 만족도 수준의 결과에 따라 다음과 같이 아키텍처 접근법을 선정하였다.

우선 반드시 있어야 하는 접근법으로 선정되어야 하는 아키텍처 접근법은 3가지 대안 중 존재하지 않고, 이전 평가 과정에서 선정된 접근법 대안도 없으므로 접근법간의 의존성으로 인해 선정되어야 하는 대안도 해당 사항이 없다. 세 번째 선정원칙인 최대, 최소값 한계를 넘는 대안으로 Single Thread 아키텍처가 해당된다. 따라서 단일 스레드 아키텍처는 선정대상에서 제외하고 나머지 두 개의 대안 중에서 높은 만족도를 보이는 아키텍처 접근법을 선정한다.

이와 같은 선정 원칙에 따라 본 사례연구에서는 높은 만족도를 보이는 스레드 풀 아키텍처 접근법을 선정한다.

다른 품질 속성에 대해서도 우선순위에 따라 아키텍처 접근법 선정과정을 반복하여 메시지 시스템의 전체 아키텍처를 설계해 나갈 수 있다.

6. 분석 및 평가

아키텍처 접근법 선정을 위한 평가 방법인 CBAM은 아키텍처 설계 산출물 또는 ATAM 평가 이후 제시된 아키텍처 접근법 대안들을 가지고 시스템의 모든 품질 요구 사항에 대해 아키텍처 접근법의 효용성을 전체적으로 평가하고 있다. 따라서 아키텍처 설계자나 개발자가 시스템 전체 차원이 아니라 자신이 맡고 있는 서브시스템이나 모듈 단위로 설계 과정 중에 아키텍처를 검증해 볼 수 있는 소규모 단위의 아키텍처 평가 방법론으로 활용하기 힘들고, 아키텍처 설계가 끝난 후 평가가 이루어져야 하는 문제점이 있다.

또한 ATAM이나 CBAM에 의한 아키텍처 접근법 평가는 설계가 끝난 이후 이루어짐으로써 이때 선정된 아키텍처 접근법이 설계 시 사용한 아키텍처 접근법과 다를 경우 다시 설계 과정을 다시 시작하여야 하여야 한다.

본 연구에서는 설계단계에서 평가과정을 거쳐 가장 적합한 아키텍처 접근법을 선정함으로써 아키텍처 설계의 신뢰성을 높이고, 설계 이후에 새로운 접근법이 선정되어 설계 단계를 재 반복하는 사례를 감소시키기 위한 평가 방법을 정의하였다.

본 논문에서 제안 하고 있는 평가 방법과 기존의 평가 방법을 비교하면 <표 9>와 같다.

우선 평가시점은 ATAM의 경우 설계가 끝난 후 이루어지고, 그 이후에 CBAM을 통해 아키텍처 접근법을 선정하게 된다. 그러나 본 연구에서 정의한 평가 방법은 설계 과

〈표 9〉 평가 방법에 대한 비교표

평가방법 \ 비교내용	평가 시점	평가 대상	평가자	평가의 객관성
ATAM	설계 후	아키텍처 접근법	아키텍처 평가자, 이해관계자	정성적
CBAM	설계 후	아키텍처 접근법	아키텍처 평가자, 이해관계자	정량적(ROI 계산)
본 연구	설계 단계	아키텍처 접근법, 설계 산출물	설계자	정량적(만족도 수준 계산)

정 중에 아키텍처 접근법을 선정하기 위해 사용된다.

평가 대상은 ATAM이나, CBAM 모두 아키텍처 접근법을 대상으로 하고 있지만, 본 연구의 평가 방법은 아키텍처 접근법뿐만 아니라, 설계 산출물에 대해서도 평가하고 있다.

평가자는 ATAM과 CBAM의 경우 아키텍처 평가 팀을 구성하여 독립된 평가자와 여러 이해 관계자들이 하고 있지만, 본 연구의 방법에서는 설계 과정 중에 이루어지므로 설계자 또는 개발자들에 의해 평가가 진행된다.

평가의 객관성 측면을 살펴보면 ATAM의 경우 주로 검토와 투표 방식에 의한 정성적 방법으로 이루어지지만, CBAM의 경우는 전체이득과 비용 즉, ROI를 계산하여 아키텍처 접근법을 선정하는 정량적, 즉 보다 객관적인 방법이 사용되고 있다. 본 연구의 방법에서 사용하고 있는 평가도 아키텍처 접근법의 품질 속성에 대한 만족도를 계산하여 평가하는 정량적 방법에 의해 평가가 이루어진다.

또한 품질 속성들에 대한 평가시에 관련된 품질 속성들을 고려하여 품질 속성 그룹으로 정의 하고 만족도 수준을 정의함으로써 아키텍처 접근법 선정시 관련된 품질 속성들이 상충(tradeoff)요소들을 가지고 있는 경우 품질 속성 그룹에 대해 정의된 만족도 수준에 따라 아키텍처 접근법을 선정할 수 있도록 하고 있다.

따라서 본 평가방법에 의해 설계 단계에서 시스템의 중요한 품질 속성요구사항을 만족시키는 아키텍처 접근법을 선정함으로써 설계 이후 새로운 아키텍처 접근법이 선정되는 사례를 줄일 수 있고, 재설계에 따른 비용의 절감효과도 기대할 수 있다.

7. 결론 및 향후 연구과제

본 논문에서는 소프트웨어 아키텍처 접근법 대안들을 설계단계에서 식별하고 품질 속성 우선 순위에 따라 평가 과정을 반복함으로써 설계 단계에서 가장 적합한 아키텍처 접근법을 선정하기 위한 방법을 제안하였다.

아키텍처 접근법 선정을 위한 평가과정을 설계 단계에 도입하여 설계의 신뢰성과 완성도를 높이고 있고, 평가 모델의 구현 및 시험을 통해 결과를 비교 평가함으로써 기존의 검토로 이루어지던 평가 과정보다 객관적인 평가가 이루어지도록 하여 선택된 아키텍처 접근법에 대한 신뢰도를 높이고 있다.

앞으로 본 논문에서 제안한 평가 모델을 많은 실 사례에 적용하여 타당성을 검증하여야 하고, 적용 시 나타나는 문제점을 보안하는 연구가 이루어져야 한다.

참 고 문 헌

[1] 궁상환, "모바일 인터넷 환경에서 Dynamic Scalable 메시지 성능에 관한 연구", 한국 전자 통신 연구원, 2002

[2] Jayatirtha Asundi, Rick Kazman, Mark Klein, "Using Economic Considerations to Choose Among Architecture Design Alternatives" Technical Report CMU/SEI, pp.1-17, 2001.

[3] Len Bass, Paul Clements, Rick Kazman, "Software Architecture in Practice Second Edition", Addison Wesley, pp.307-324, 2003.

[4] Paul Clements, Rick Kazman, Klein, "Evaluating Software Architectures : Methods and Case Studies", Addison Wesley, pp.43-84, 2002.

[5] Richard Monson-Haefel, David Chappell, "Java Message Service" O'Reilly, 2000.

[6] Yuki Hiroshi 저 , 조해미 역 "Java 언어로 배우는 디자인 패턴 입문-멀티 스레드 편" 영진출판사, pp.247-292, 2003.

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

[8] Douglas C. Schmidt, Michael Stal, Hans Rohnert, "Pattern-Oriented Software Architecture : Patterns for Concurrent and Networked Objects Volume 2", Wiley, 2000.

[9] Jahming Technologies, "SonicMQ 4.0과 IBM MQSeries 5.2에 대한 벤치마크 비교", Jahming Technologies 2001.

[10] Sonic Software Corporation, "JMS Performance Comparison : Publish/Subscribe Messaging", Sonic Software Corporation, pp.4-5, 2003

[11] Robert L. Nord, Mario R. Barbacci, Paul Clements, Rick Kazman, Mark Klein, Liam O'Brien, James E. Tomayko "Intergrating the Architecture Tradeoff Analysis Method (ATAM) with the Cost Benefit Analysis Method(CBAM)" Technical Report CMU/SEI, 2003 .

[12] SEI "Attribute Driven Design(ADD) Method Tutorial" SEI 2004.

[13] Mario R. Barbacci, S. Jeromy Carriere, Peter H. Feiler, Rick Kazman, Mark H. Klein, Howard F. Lipson, Thomas A. Longstaff, Charles B. Weinstock, "Steps in an Architecture Tradeoff Analysis Method : Quality Attribute Models and Analysis", CMU/SEI, 1998.

고 현 희

e-mail : hhkoh@sookmyung.ac.kr

1992년 숙명여자대학교 전자계산학과 (이학사)

2000년 숙명여자대학교 컴퓨터과학과 (이학석사)

1991년~1993년 코오롱정보통신 연구원

1993년~1999년 LG산전연구소 주임연구원

2000년~2002년 LG전자 정보통신연구소 선임연구원

2002년~2002년 미국 카네기 멜론 대학 SE 과정 수료

2001년~현재 숙명여자대학교 컴퓨터과학과 박사과정

관심분야 : 소프트웨어 설계 방법론, 소프트웨어 아키텍처, 설계 패턴

궁 상 환

e-mail : kung@cheonan.ac.kr

1977년 숭실대학교 전자계산학과(이학사)

1983년 고려대학교 전자정보처리학과 (경영학석사)

1998년 충북대학교 전자계산학과 (이학박사)

1977년~1981년 제2군수지원사령부 제원처리실 프로그램장교

1981년~1998년 한국전자통신원 책임연구원

1996년~1997년 미국 스탠포드 연구소(SRI) 초빙연구원

1998년~2000년 중소기업청 정보화지원과 전산사무관

2000년~2000년 미국 카네기 멜론 대학 SE 과정 수료

2000년~현재 천안대학교 정보통신학과 조교수

관심분야 : 소프트웨어 구조, 분산시스템

박 재 년

e-mail : jnpark@sookmyung.ac.kr

1966년 고려대학교 물리학과(이학사)

1969년 고려대학교 대학원(이학석사)

1972년 독일 함부크르대학 전산학 전공

1981년 고려대학교 대학원 물리학 전공 (이학박사)

1970년~1972년 독일 국립 입자 가속기 연구소(DESY) 연구원

1972년~1979년 고려대학교 전자계산소 총간사

1978년~1983년 전남대학교 계산통계학과 교수

1987년~1998년 숙명여자대학교 전산원 원장

1991년~1991년 숙명여자대학교 사무처장

1994년~1996년 정보과학회 이사

1994년~1996년 전산교육 연구회 운영위원장

1983년~현재 숙명여자대학교 정보과학부 교수

2002년~현재 숙명여자대학교 이과대학 학장

관심분야 : 시스템 분석 및 설계, 정보구조 모델링, 컴포넌트 기반 개발 방법론