아키텍처기반 설계 방식에 대한 평가기능이 통합된 소프트웨어 설계 방법론

(A Software Design Methodology Integrating Evaluation Method of Architectural Design Approaches)

고 현 희 † 궁 상 환 †† 박 재 년 †††
(Hyon Hee Koh) (Sang Hwan Kung) (Jae Nyon Park)

요 약 소프트웨어가 대형화되고 복잡해짐에 따라 소프트웨어 아키텍처의 설계가 성공적인 프로젝트를 위한 중요한 이슈가 되었다. 아키텍처 설계의 기반이 되는 아키텍처 접근법 선정은 아키텍처 설계와 후속설계 작업의 성공을 위해 무엇보다 중요하다. 본 연구에서는 아키텍처 접근법 대안들을 설계단계에서 식별하고 시스템의 품질 속성을 만족시키는 아키텍처 접근법을 선정하기 위한 평가 방법과 평가 방법을 통합한 설계 방법을 제안함으로써 설계 산출물의 신뢰도를 높이고 재설계에 따른 비용의 절감 및 검증된 아키텍처 접근법을 기반으로 아키텍처 설계를 완성함으로써 후속 설계 작업의 성공을 높이고자 한다.

키워드 : 소프트웨어 아키텍처, 아키텍처 설계, 아키텍처 평가

Abstract Due to the software is getting complex and large, software architecture design is issued for success of project. In the design phase, selection of the suitable architectural approach is very important. In this thesis, we identify the architectural approach alternatives in the design phase. And the evaluation method to choose architectural approaches and the architecture design methodology integrated with that evaluation method are proposed. In the proposed architecture evaluation method and the architecture design methodolody, we intend to raise reliability, completeness of design and reduce development costs.

Key words : Software Architecture, Architecture Design, Architecture Evaluation

1. 서 론

소프트웨어 아키텍처는 소프트웨어 엔지니어링에서 중요한 역할을 담당하며 시스템에 대한 높은 수준의 뷰를 개발한다[1]. 소프트웨어 아키텍처 설계는 복잡한 소프트웨어를 시각화 및 단순화시킴으로 개발자와 요구자 간의 이해도를 향상시킬 수 있을 뿐만 아니라, 개발 시 예상되는 변화에 대한 대안들을 고려할 수 있어 대안의 선택에 사용할 수 있다. 전체적인 요구사항의 일관성, 품질요소 등을 검증, 분석하고, 소프트웨어 개발 비용 및 일정 예측의 근거자료로 활용할 수 있으며, 위험요소의 예측 및 관리방안을 계획하는 데 활용할 수 있다. 또한 향후 유사 소프트웨어 개발 시 재사용의 근거를 제시할 수 있어 중복개발에 대한 비용을 절감할 수 있고, 재사용 잠재력이 많은 컴포넌트를 찾아 낼 수도 있다.

아키텍처 설계의 기반이 되는 아키텍처 접근법(Architectural Approaches)은 품질 속성을 달성하기 위한 아키텍처 모델 또는 아키텍처 스타일의 선택을 통해 식별될 수 있다. 소프트웨어 엔지니어는 아키텍처 모델에 근거하여 시스템 설계의 모든 측면을 토의하게 되고, 아키텍처 모델을 개발할 때 내린 결정들은 나머지 설계 작업에 대하여 지대한 영향을 미친다[1]. 또한 아키텍처 설계가 변경되면 개발 조직의 변경 및 후속 설계 작업이 모두 재설계되어야 하는 위험이 따른다. 따라서 초기에 시스템의 중요한 품질 요구사항을 만족시키는 가장 적합한 아키텍처 접근법을 선정하는 것이 이러한 위험을 해결하고 설계의 성공 여부를 좌우하는 가장 중요한 일이라고 할 수 있다.

본 논문에서는 설계 단계에서 아키텍처 접근법을 선정하기 위한 평가 방법과 이 평가 방법을 통합한 설계

† 학생회원 : 숙명여자대학교 컴퓨터과학과
hhkoh@sookmyung.ac.kr
†† 정 회 원 : 백석대학교 정보통신학부 교수
kung@cheonan.ac.kr
††† 정 회 원 : 숙명여자대학교 컴퓨터과학과 교수
hhkoh@sookmyung.ac.kr
논문접수 : 2006년 2월 24일
심사완료 : 2007년 6월 4일

방법을 제안하고자 한다. 논문의 구성은 2장에서 관련 연구로서 소프트웨어 아키텍처 설계와 평가 방법에 대해 소개하고, 3장에서는 본 논문에서 제시하는 평가를 통합한 아키텍처 설계 방법과 절차를 설명한다. 4장에서는 사례 연구를 통해 제안하는 설계 방법을 적용하여 보고, 5장에서는 연구 내용 및 사례 연구 결과를 바탕으로 평가 방법과 설계 방법에 대해 평가하며 마지막으로 6장에서는 본 논문의 결론을 맺는다.

2. 관련연구

2.1 아키텍처 설계 방법

아키텍처 설계 방법은 미국 Carnegie Mellon 대학의 SEI에서 연구 개발한 아키텍처 기반 설계 방법(Architecture Based Design)과 품질 속성에 기반 한 아키텍처 설계 방법(Attribute Driven Design)을 대표적으로 볼 수 있다.

ABD는 전체 시스템의 분해 기법에 기반을 두고 있다. 분해 작업을 통해 디자인 요소를 만들어 나간 뒤 디자인 요소 내에서 논리적 뷰를 정의하고 동시성 뷰와 배치 뷰를 정의한 후 검증과정을 거쳐 다시 논리적 뷰로 피드백 되는 프로세스로 구성되어 있다[2].

1. 기능을 분할 한다.
2. 아키텍처 스타일을 선택한다.
3. 기능을 스타일에 할당한다.
4. 템플릿을 정제한다.
5. 기능을 검증한다.
6. 동시성(Concurrency) 뷰를 생성한다.
7. 배치(Deployment) 뷰를 생성한다.
8. 품질 시나리오를 검증한다.
9. 제약사항을 검증한다.

그림 1 ABD 방법론의 수행단계

ADD 방법론은 우선 품질 속성 달성방안을 고려해서 아키텍처 스타일을 결정하고, 결정한 아키텍처 스타일에 따라 분할을 수행한다[3]. 분할에 의해 구성 요소와 구성 요소들 사이의 관계가 결정되면 기능요구사항을 만족시킬 수 있도록 구성요소들을 구체화한다. 이 작업을 ADD로 더 이상 분해할 것이 없을 때까지 반복한다.

2.2 아키텍처 평가 방법

아키텍처 평가는 설계가 가져올 수 있는 프로젝트의 재앙을 피할 수 있는 가장 경제적인 방법이다. 아키텍처 평가의 결과는 아키텍처가 시스템에 적절하게 설계되었는지, 시스템을 위해서 가장 적절한 아키텍처가 무엇인지, 즉 아키텍처의 적합성을 판단하는 것이다[4]. 이러한 아키텍처 평가방법에는 평가 방법이나, 평가 대상, 평가의 구체성에 따라 여러 방법들이 나와 있다.

단계 1 : 분해할 모듈을 선택한다.
단계 2 : 선택한 모듈을 분해하고 정제한다.
  단계 2a. 아키텍처 동인을 결정한다.
  단계 2b. 아키텍처 스타일을 선택한다.
  단계 2c. 모듈을 실체화하고 기능을 할당한다.
  단계 2d. 하위 모듈의 인터페이스를 정의한다.
  단계 2e. 유스케이스와 품질 요소를 정제하고 검증해서 하위 모듈의 제약사항으로 바꾼다.
단계 3 : 모듈을 분해할 필요가 없을 때까지 반복한다.

그림 2 ADD 방법론의 수행단계

본 논문에서는 다양한 평가 방법 중 가장 많이 활용되고 있는 ATAM(Architecture Tradeoff Analysis Method)과 아키텍처 접근법 선정을 위한 평가 방법인 CBAM(Cost Benefit Analysis Method)의 문제점을 보완하고 발전시켜 적용하고 있다.

ATAM은 품질 속성 요구사항과 비즈니스 목표달성을 위한 아키텍처 결정사항들을 평가한다[5]. 즉 ATAM은 시나리오를 중심으로 품질 속성 요구 사항을 찾아내고 아키텍처가 특정 품질 속성들을 만족하는지를 분석하는데, 다양한 이해관계자들과 함께 위험요소, 민감점, 상충점 등을 분석한다[4].

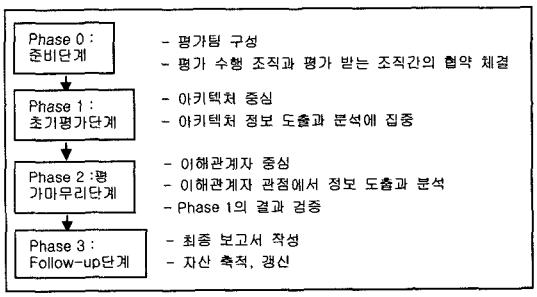

그림 3 ATAM 실행단계

비용과 이득을 고려한 아키텍처 평가 방법인 CBAM은 아키텍처 접근법을 실현하는 데 필요한 비용과 아키텍처 접근법을 적용했을 때 달성할 수 있는 품질 속성이 가져다 주는 이익을 측정한다. 즉, 비용과 이익으로부터 '투자대비효과(ROI)'를 계산하여 수익이 최대가 될 수 있도록 의사결정을 지원한다[5]. CBAM은 주로 ATAM 평가가 끝난 후 새롭게 제시된 대안 아키텍처 접근법들 중에서 ROI가 높은 접근법 대안을 선택 할 수 있도록 하고 있다[6].

기존의 아키텍처 설계방법과 평가방법은 주로 설계가 끝난 후 아키텍처 평가 과정을 거치게 된다. 이 과정에서의 문제점은 평가 후 선정된 아키텍처 접근법이 설계 시 사용한 아키텍처 접근법과 다를 경우 다시 설계 과정을 다시 시작하여야 한다. 이러한 문제점을 해결하기 위해서는 아키텍처 설계 단계에서 시스템의 기능, 비기능 요구사항을 가장 잘 만족시키는 아키텍처를 설계 하는 것이고, 이를 위해서는 설계단계에서 가장 적합한 아키텍처 접근법이 선정될 수 있도록 해야 한다. 그러나

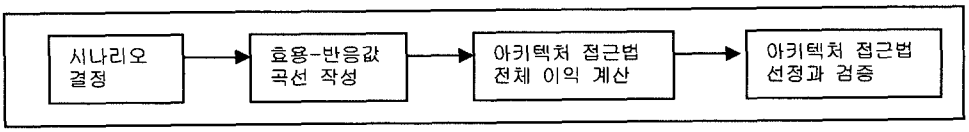

그림 4 CBAM 실행단계

ABD나 ADD는 아키텍처 스타일의 선택시 설계자의 경험과 주관에 의존하고 있고, 설계단계에서 식별된 아키텍처 스타일 또는 접근법의 적합성 검증이나 대안의 고려 및 선정에 대한 평가방법이 제시되지 않고 있다.

또한 ATAM 평가 방법의 경우는 평가를 위해 팀을 조직하고 다양한 이해관계자들과 개발 조직간에 협상을 거쳐 평가를 시작하는데 수주일이 걸리는 등 소규모의 개발 프로젝트에 적용하기는 힘든 면이 있고, 현실적으로 아키텍처 설계 이후에 이러한 평가과정을 거치는 사례가 드물다.

CBAM 평가의 경우도 주로 조직에서 가지고 있는 데이타를 이용하여 비용과 일정을 예측하고 있다, 즉 비용과 일정에 대한 객관적 평가방법이 제시되지 못하고 있고, 비용이 크게 차이가 나지 않는 경우의 아키텍처 접근법 선정에 대한 고려를 하고 있지 않다.

따라서 본 논문에서 제시하고 있는 방법론에 따라 아키텍처 설계자나 개발자가 시스템 전체 차원이 아니라 자신이 맡고 있는 서브 시스템이나 모듈 단위로 아키텍처를 검증해 볼 수 있는 소규모 단위의 아키텍처 평가 방법에 의해 설계 단계에서 아키텍처 접근법을 검증하고 대안들을 평가하여 선정한다면 설계된 아키텍처의 완전성과 신뢰성이 높아질 뿐만 아니라 아키텍처 설계 이후의 평가 과정에서 새로운 아키텍처 접근법이 선정되는 사례도 감소할 것이다.

3. 평가를 통합한 아키텍처 설계 방법

본 장에서는 그림 5에서 처럼 아키텍처 설계과정 중에 식별된 아키텍처 접근법 대안들을 평가하여 달성하고자 하는 품질 속성 요구사항에 대한 객관적인 검증을 통해 아키텍처 접근법을 선정하고 설계를 발전시켜 나가는 아키텍처 설계 방법론을 정의한다.

그림 6은 본 논문에서 제안하는 아키텍처 접근법 선

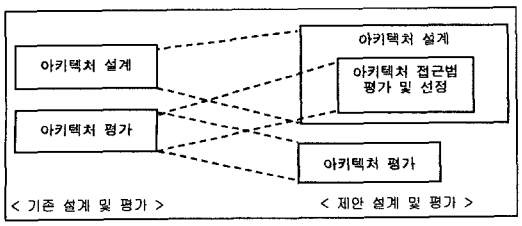

그림 5 아키텍처 설계 과정에 대한 비교

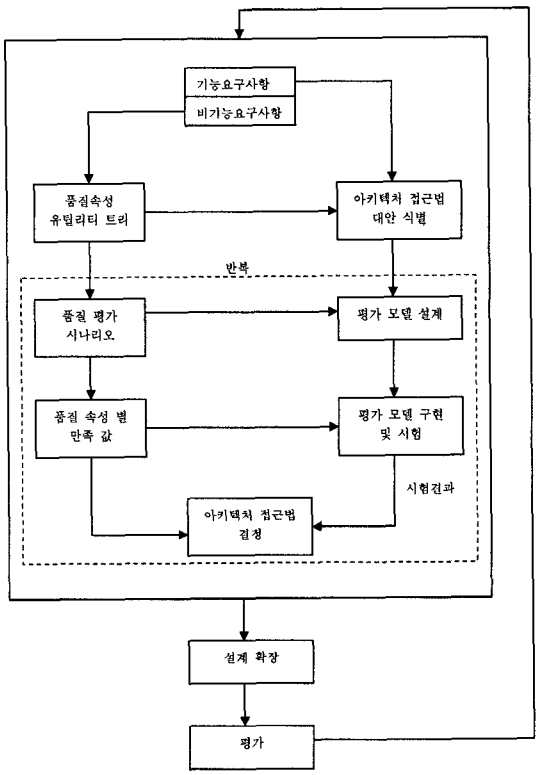

그림 6 아키텍처 설계 및 평가 통합 모델

정을 위한 평가를 통합한 설계방법의 모델이다. 먼저 시스템의 기능 요구사항과 비기능 요구사항으로부터 아키텍처 동인을 얻어 아키텍처 접근법을 식별한다. 식별된 아키텍처 접근법이 여러 개의 대안이 존재할 경우 각 대안에 대한 평가과정을 거쳐 가장 적합한 아키텍처 접근법을 선정하게 된다. 이렇게 선정된 아키텍처 접근법에 기능을 할당하여 아키텍처를 확장 설계한다.

3.1 아키텍처 설계 절차

아키텍처 설계 절차는 그림 7과 같다. 각 단계별 세부 실행 내용은 다음과 같다

(1) 단계 1: 요구사항 정의 - 이 단계에서는 시스템의 기능/비기능 요구사항을 시나리오 형태로 표현하고 정제하는 과정을 거친다.

(2) 단계 2: 아키텍처 동인 결정 - 아키텍처 동인을 선정할 때는 아키텍처에 가장 많은 영향을 미치는 시나리오나 유스케이스를 찾는다. 이 과정은 시스템이

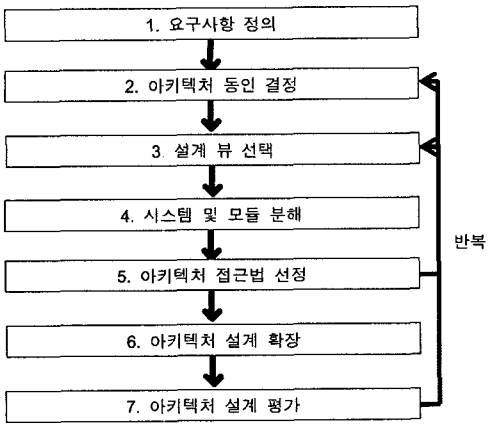

그림 7 아키텍처 설계 단계

만족해야 하는 모든 품질 속성에 대한 시나리오를 트리 형태로 나타낸 품질 속성 유틸리티 트리를 작성하는 과정을 통해 실행한다[5]. 이를 통해 품질 속성 우선순위를 쉽게 결정할 수 있다.

(3) 단계 3: 시스템 및 모듈 분해 - 아키텍처 설계를 하기 위해 우선 시스템을 적절한 서브시스템으로 나눈다. 각 서브시스템은 개발 단위의 모듈로 분해되고, 이 과정은 적절한 크기의 모듈로 분해될 때까지 반복하게 된다. 이렇게 분해된 모듈은 아키텍처 설계의 단위가 되며, 각 팀 또는 조직으로 할당되는 단위가 된다.

(4) 단계 4: 설계 뷰 선택 - 설계 뷰는 논리 뷰(기능관점, 기능요구사항), 동시성 뷰(동시성/병행처리 관점, 성능요구사항), 배치 뷰(하드웨어상의 소프트웨어 설치 관점)로 구분된다. 선택된 뷰에 따라 참조할 아키텍처 스타일이 달라지게 된다.

(5) 단계 5: 아키텍처 접근법 선정 - 아키텍처 동인에 따른 아키텍처 접근법은 하나 또는 여러 개의 대안이 나올 수 있다. 두개 이상의 접근법 대안이 있을 경우 평가과정을 통해 가장 적합한 접근법을 선정한다. 아키텍처 접근법 선정을 위한 평가 절차는 다음절에서 설명한다.

(6) 단계 6: 뷰 타입별 아키텍처 설계 - 각 모듈별로 선택된 아키텍처 스타일에 기능을 할당하여 실체화함으로써 아키텍처 설계를 확장하여 목표 시스템의 아키텍처를 설계한다.

(7) 단계 7: 아키텍처 설계 평가 - 선택된 아키텍처 접근법에 따라 설계된 아키텍처를 평가한다.

3.2 아키텍처 접근법 선정을 위한 평가 방법

본 논문에서 제안하는 설계단계에서의 아키텍처 접근법 선정을 위한 평가 과정은 그림 8과 같다.

각 단계별 세부 실행 내용은 다음과 같다

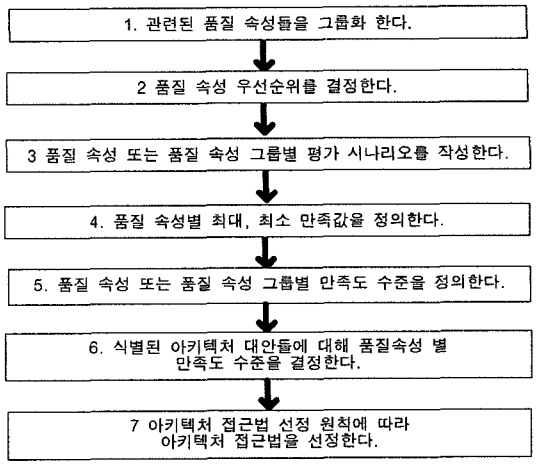

그림 8 아키텍처 선정을 위한 평가 과정

(1) 관련 품질 속성 그룹화

품질 속성들 간에는 상호 종속적이 될 수 있다. 예를 들어 확장성을 높이기 위한 설계가 성능을 저하시킬 수 있고, 성능을 높이기 위한 설계는 확장성을 저하시킬 수 있다. 이 경우 각각의 품질 속성에 대해 평가되기 보다는 두개의 품질 속성을 하나로 그룹화하여 두 품질 속성간의 적정 값으로 선정되도록 평가 되어야 한다.

(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.

(3) 평가 시나리오 작성

품질 속성과 품질 속성 그룹별로 평가 시나리오를 작성한다. 이 때 작성된 평가 시나리오는 평가 모델을 만들기 위한 기반이 된다.

(4) 품질 속성별 최대/최소값 결정

품질 속성 별로 만족해야 하는 최대값과 최소값을 결정한다. 이 값은 해당 시스템이 만족해야 하는 품질 속성의 최대/최소 한계 값으로 아키텍처 접근법의 시험 결과값이 이 한계값을 넘는 경우 해당 아키텍처 접근법은 선정될 수 없다.

(5) 만족도 수준 정의

품질 속성 또는 품질 속성 그룹에 대해 평가 시나리오의 결과값에 대한 만족도 수준을 정의한다. 만족도 수준 결정에는 설계자와 시스템과 관련된 이해관계자들의 의견을 반영하여 결정한다.

(6) 아키텍처 접근법의 예상 만족도 수준 결정

이 단계에서는 품질 속성들에 대한 아키텍처 접근법

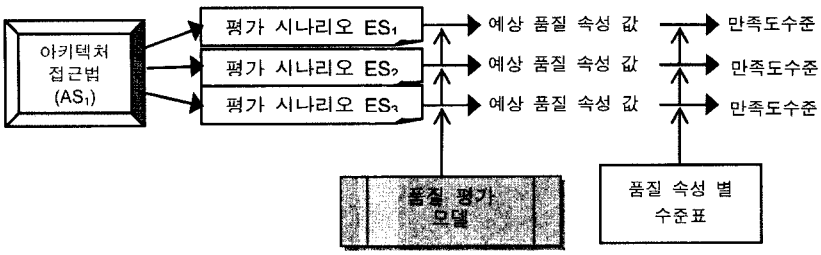

그림 9 아키텍처 접근법의 예상 품질 속성값 및 만족도 수준 결정

의 예상 품질 속성값을 결정한다. 아키텍처 접근법의 예상 품질 속성값은 해당 품질 평가 시나리오에 대한 시험 모델의 구현과 시험을 통해 시험 결과를 가지고 시스템의 결과값을 예측해 볼 수 있다. 이 값을 기반으로 그림 9와 같이 각 아키텍처 접근법 대안들의 품질 속성에 대한 만족도 수준을 결정한다.

(7) 아키텍처 접근법 선정

위의 만족도 수준 결과를 가지고 아키텍처 접근법을 선정한다. 기본 원칙은 품질 속성별 결과값이 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 접근법 들을 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.

4. 사례 연구

사례연구에서는 인터넷 활용 뿐만 아니라, 엔터프라이즈 어플리케이션 통합에서도 활용되고 있는 메시지 시스템을 대상으로 하고자 한다.

본 연구에서 정의한 평가모델을 기반으로 메시지 시스템의 아키텍처 설계 단계에서 아키텍처 접근법 선정 과정까지를 보이고, 설계의 확장은 다루지 않는다. 아키텍처 동인 중 우선순위가 높은 성능과 확장성 품질 속성에 대한 평가과정을 적용하고 그 외 다른 속성들에 대해서 적용하는 반복과정과 모듈 분해에 따른 반복과정은 다루지 않는다.

(1) 요구사항 정의

메시지 시스템은 네트워크에서 클라이언트와 클라이언트간에 주고 받는 메시지와, 이를 처리하는 서비스 제공자로 구성되며, 일대일 접속을 위한 지점간 연결(point-to-point) 모델과, 일대다 또는 다대일 접속 형태를 지원하는 발행/구독(publish-and-subscribe) 모델로 볼 수 있다[7].

메시지 시스템의 기능요구사항은 그림 10과 같이, 비기능 요구사항에 대한 품질 속성 유틸리티 트리는 그림 11과 같이 결정하였다[8].

1. 메시지가 한 시스템에서 다른 시스템으로 네트워크 타고 비동기식(Asynchronous)으로 전달 된다.
2. 일대다(브로드케스팅) 전송이 가능하다.
3. 일대일(p2p) 전송이 가능하다.
4. JMS 클라이언트는 생산자와 소비자 둡 다 될 수 있다.
5. 메시지 전달에 대한 확인 응답을 한다.
6. 토픽을 구독하는 모든 클라이언트는 그 토픽으로 발행된 메시지에 대해 자신만의 복사본을 갖는다. 한명의 발행자에 의해 생산된 하나의 메시지가 수백 또는 수천개의 구독자에게 복사되어 전달 될 수 있다.
7. 메시지는 무조건 한번만 전달된다.
8. 클라이언트는 지속적 구독(durable subscription)을 할 수 있다.

그림 10 메시지 시스템의 기능 요구 사항

(2) 아키텍처 동인 결정

엔터프라이즈 애플리케이션의 성능, 확장성, 신뢰도는 실제 배치 환경에서 가장 중요하게 고려되어야 한다[7]. 또한 메시지 전달 속도와 관련된 성능이나 동시 연결자 수는 메시지 시스템의 평가에 중요한 요소 이므로 아키텍처 동인으로 정할 수 있다. 정확한 메시지 전달 즉 신뢰성도 메시지 시스템의 중요한 요소이다. 그러나 가용성의 경우 비동기식 메시징 시스템은 밀접하게 결합된 RPC와는 달리 메시지 시스템의 아키텍처 자체가 각 하위 시스템과 다른 시스템을 분리하여 각자 메시징 서버를 통해 통신하고 있기 때문에 한곳의 장애가 다른 곳의 운영에 영향을 주지 않는다. 따라서 가용성의 경우는 소프트웨어 아키텍처에서 더 이상 고려하지 않아도 된다. 보안의 경우 어떻게 구현하느냐는 벤더가 정하며, 각 벤더들은 JMS클라이언트간에 서로 인증하고 권한을 부여하고, 안전한 통신을 할 때 사용할 수 있는 기술을

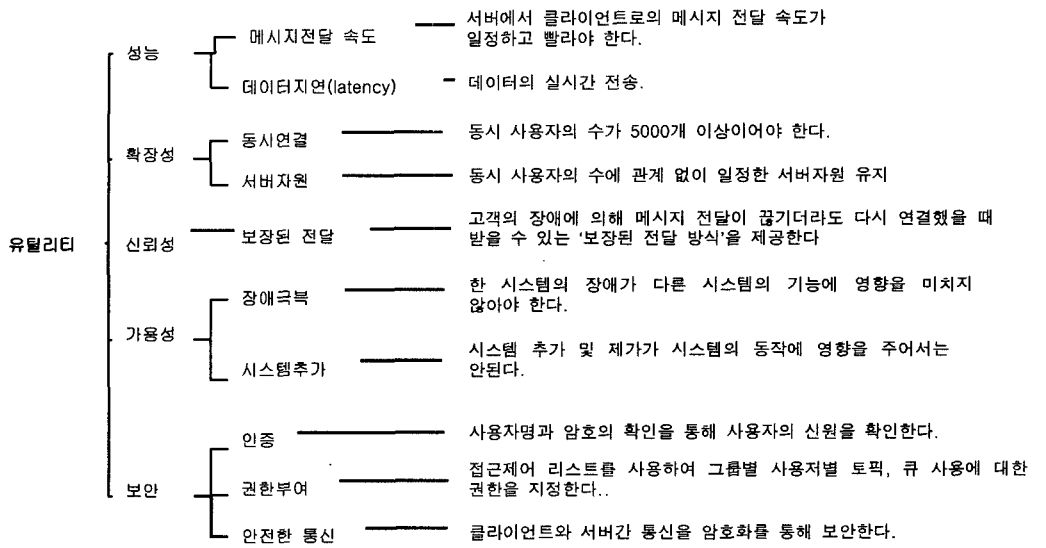

그림 11 비기능 요구사항에 대한 품질 속성 유틸리티 트리

나름대로 조합하여 사용하고 있으므로, 여기서 아키텍처 설계를 위한 동인으로 성능, 확장성, 신뢰성 시나리오를 선택하도록 한다.

(3) 설계 뷰 선택

배치 뷰의 경우 하드웨어 환경과 사용자의 요구에 따라 달라질 수 있으므로 본 사례 연구에서는 논리 뷰와 동시성 뷰에 대해 설계 하도록 한다.

(4) 시스템 및 모듈 분해

메시지 시스템은 먼저 메시지 송신 역할을 하는 발행 부분과 메시지 수신 역할을 하는 구독 부분, 그리고 메시지 발행자와 메시지 구독자를 관리하는 메시지 관리부, 그리고 메시지의 저장과 검색을 지원하는 부분으로 구분된다.

(5) 메시지 시스템의 아키텍처 접근법 선정

① 관련된 품질 속성 그룹화

성능과 확장성은 동시 사용자 수에 따라 성능이 영향을 받을 수 있는 속성이므로 {확장성, 성능}을 하나의 관련된 품질 속성으로 그룹화한다.

② 품질 속성 우선 순위 결정

우선순위는 요구사항 분석 단계에서 품질 속성 유틸리티 트리에 의해 결정이 되었다고 보고, 본 사례 연구에서는 우선순위 결정을 위한 과정을 생략한다. 본 사례 연구에는 우선 순위가 높은 확장성과 성능 품질 속성에 대해 평가 과정을 적용한다.

③ 아키텍처 접근법 대안

먼저 논리 뷰의 아키텍처 접근법은 각 모듈간에 서비스를 제공하고 이용하는 사용 스타일(Uses Style)을 참조한 그림 12의 아키텍처 접근법을 선택한다.

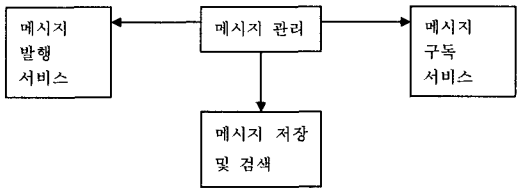

그림 12 메시지 시스템의 논리 뷰 - 사용 스타일

메시지 시스템의 성능 요구사항은 서버에서 클라이언트로의 메시지 전달 속도가 일정하고 빨라야 한다는 것이고, 확장성 요구사항은 대규모 사용자를 고려 할 때 동시 사용자 수가 5000명 이상이어야 하고 동시 사용자 수에 상관없이 일정한 성능을 유지하기 위해 일정한 서버 자원을 유지하여야 한다는 것이다[9]. 이러한 요구사항을 고려하여 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 활용한 아키텍처 접근법을 선택한다[8]. 멀티 스레드 패턴은 멀티 스레드의 생성이나 처리 방법에 따라 여러 대안이 나올 수 있다. 본 논문에서는 이러한 사항들을 고려하여 다음과 같은 3개의 아키텍처 접근법 대안을 식별하였다[8].

(1) 출판이나 구독을 위한 클라이언트로부터 요청이 들어올 때마다 메시징 서버에서 클라이언트에게 서비스를 담당할 스레드를 하나씩 할당하는 "클라이언트별 스레드 할당" 아키텍처

(2) 여러 개의 스레드를 미리 가동해 두고 워커 스레드

로서 기다리게 하는 "스레드 풀" 아키텍처

(3) 서버측의 단일 스레드를 사용하는 "단일 스레드)" 아키텍처

④ 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.

따라서 평가 시나리오는 다음과 같이 만들 수 있다.

⑤ 품질 속성 별 최대/최소값 결정

품질 속성별 최대/최소값은 각 이해관계자들이 모여 결정하여야 하나, 본 사례 연구를 위해 참조한 메시지 시스템의 요구사항 및 상용 메시지 시스템의 성능 평가 자료 등에 의해 {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.

⑥ 만족도 수준 정의

만족도 수준도 이해관계자들이 모여 결정하여야 하나, 대규모 사용자를 대상으로 할 때 메시지 서버의 동시 접속자 수가 5000명 이상이어야 하는 요구사항을 고려하고, 상용화된 메시지 시스템의 성능을 참조하여 본 사례 연구에서는 표 1과 같이 정한다.

⑦ 아키텍처 접근법의 예상 만족도 수준 결정

평가 시나리오 ES1.에 대한 시험은 먼저 스레드를 생성시키고 start()를 호출하여 실제 스레드가 동작되기까지의 시간 소요를 프로그램으로 구현하여 실시하였다[9]. 그 결과는 다음과 같다.

(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 성능 저하에 중요한 요인이 될 수 있음을 알 수 있다[9]. 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 13과 같고, 평가를 위한 구현은 각 참조 패턴을 활용하여 구현하였다[8].

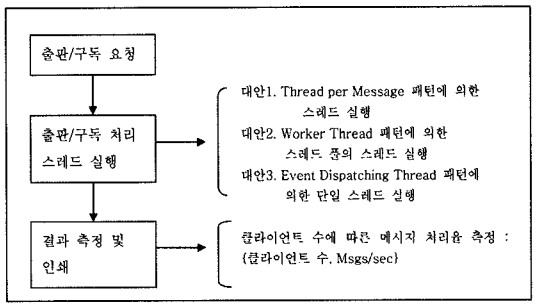

그림 13 평가 시나리오 ES2.에 의한 평가 모델

구현된 평가 모델에 의한 시험 결과는 표 2와 같다.

표 2에 나타난 시험 결과를 볼 때 접속자 수가 작을 때는 클라이언트별 스레드 할당 아키텍처가 좋은 성능을 보여주고 있으나, 접속자의 수가 많아져서 스레드 할당이 많아지는 경우 스레드 풀 아키텍처의 성능이 좋아지는 걸로 볼 수 있다.

본 사례 연구에서 스레드 풀의 스레드 수를 100개로 하였으나, 스레드 풀의 스레드 수를 다르게 할 경우 다른 결과를 보일 수도 있다. 스레드 풀의 스레드 개수를

표 1 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}

표 2 메시지 전송 테스트 결과 (P/S/T : Publisher/Subscriber/Topic)

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

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 아키텍처의 경우는 접속자 수가 증가함에 따라 성능의 큰 차이는 보이지 않지만, 성능의 감소하는 추세를 보이는 것으로 보아 접속자 수가 많아 짐에 따라 스레드 수가 증가하여 성능이 감소하고 있음을 예측할 수 있다.

위의 결과를 가지고 표 3과 같이 만족도 수준을 예측해 볼 수 있다.

표 3 만족도 수준 예측 결과 (단위 {명, Msgs/sec})

아키텍처 / 품질 속성 그룹 / 접근법	클라이언트별 스레드 할당	스레드 풀	단일 스레드
{확장성, 성능}	{5000,800} 수준 4	{5000,1500} 수준 4	{5000, 10}
{확장성, 성능}	{2000, 1200} 수준 3	{2000, 2000} 수준 2	{2000, 20}

클라이언트별 스레드 할당 아키텍처의 경우 시험 결과 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 수도 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.

⑧ 아키텍처 접근법 선정

테스트 환경이나 영속성(Durability), 신뢰성(Persistent) 등의 속성에 따라 성능에 영향을 미칠 수 있으므로 결과는 달라질 수 있다. 그러나 본 사례 연구에서는 확장성과 성능만을 고려하고 있으므로 평가 모델에 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 해당사항이 없다. 세번째 선정 원칙인 최대/최소값 한계를 넘는 대안으로 단일 스레드 아키텍처가 해당된다. 따라서 단일 스레드 아키텍처는 선정 대상에서 제외하고 나머지 두 개의 대안 중에서 높은 만족도를 보이는 아키텍처 접근법을 선정한다. 이와 같은 선정 원칙에 따라 본 사례 연구에서는 높은 만족도를 보이는 스레드 풀 아키텍처 접근법을 선정한다.

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

5. 연구 결과의 분석

본 연구에서 제안하는 설계 방법론은 설계 단계에서 아키텍처 접근법 선정을 위한 평가 과정을 거쳐 가장 적합한 아키텍처 접근법을 선정하여 설계의 완성도와 신뢰성을 높임으로써 이후의 설계 결과의 성공을 높이도록 하고 있다. 뿐만 아니라, 설계 이후에 평가가 이루어짐으로써 발생할 수 있는 설계 단계로의 회귀를 감소시킬 수 있도록 하고 있다.

본 논문에서 제안 하고 있는 평가 방법과 기존의 평가 방법을 비교해 보면 표 4와 같고, 표 5에서 아키텍처 설계 방법인 ABD, ADD와 본 연구에서 제안한 설계 방법을 비교하고 있다.

[4]에서 제시된 중규모 시스템에 대한 ATAM 평가에 소요되는 시간을 참조하여 기존 설계 및 평가 방법과, 본 연구에서 제안하는 평가를 통합한 설계 방법에 소요되는 예상 시간을 표 6에 나타내었다.

본 연구에서 제안하는 설계 방법에 의해 설계를 한 경우, 아키텍처 접근법과 설계 산출물에 대해 평가를 거

표 4 평가 방법에 대한 비교표

비교 내용 / 평가 방법	평가 시점	평가 대상	평가자	평가의 객관성
ATAM	설계 후	아키텍처 접근법	아키텍처 평가자 이해관계자	정성적
CBAM	설계 단계 설계 후	아키텍처 접근법	아키텍처 평가자 이해관계자	정량적 (ROI 계산)
본 연구	설계 단계	아키텍처 접근법 설계 산출물	설계자	정량적 (만족도 수준 계산)

표 5 설계 방법에 대한 비교표

비교 내용 / 설계 방법	아키텍처 동인 식별	아키텍처 동인 변경	기능 분해	아키텍처 접근법 대안 식별	아키텍처 접근법 검증	아키텍처 접근법 대안 평가
ABD	O	×	O	×	×	×
ADD	O	×	O	×	×	×
본 연구	O	O	O	O	O	O

표 6 기존 방법과 제안방법의 비용 비교표 (중규모 시스템)

	설계	ATAM	CBAM	합계
기존방법	설계시간	70 person-day	α person-days	설계시간 + α + 70 person-days
제안방법	설계시간 + 8 person-days	0/42/70 person-day	0/α person-days	설계시간 + α + 8/50/78 person-days
비고	8 - 아키텍처 접근법 선정을 위한 평가 시간	0 - ATAM 평가를 하지 않는 경우 42 - Phase 2 에서 이해관계자의 동의만 얻는 경우 70 - ATAM을 실행하는 경우	0 - CBAM을 실행하지 않는 경우 α - CBAM 실행에 소요되는 시간	8 - ATAM 평가를 하지 않는 경우 50 - Phase 2에서 이해관계자의 동의만 얻는 경우 78 - ATAM을 실행하는 경우

친 상태이므로 ATAM 평가를 생략할 수 있다. 단 이해관계자들의 동의를 얻는 Phase 2의 과정만을 실행함으로써 ATAM 평가와 동일한 효과를 기대할 수 있다.

또한 본 연구에서 제안하는 평가 방법은 비용을 고려하지 않는 평가 방법으로써 비용에 대한 데이타를 가지고 있지 않거나, 비용을 고려할 수 없는 조직, 또는 비용이 중요하지 않은 프로젝트인 경우 본 평가 방법에 의해 아키텍처 접근법을 선정하고 CBAM 평가를 생략할 수 있다.

이러한 내용들을 고려한 결과가 표 6의 결과이다. 결과에서 보는 것처럼 최선의 경우에는 본 연구에서 제안하는 설계와 평가 방법이 훨씬 좋은 효과를 보이고 있고, 최악의 경우, 즉 설계 후 ATAM과 CBAM 평가를 모두 거치는 경우라 할 지라도 기존 방법과 유사한 결과를 보이고 있다. 소규모 시스템의 경우도 같은 결과를 보이고 있다.

그러나 설계 및 평가에 소요되는 시간이나 비용보다 평가 이후 문제점 발견 시 또는 새로운 아키텍처 접근법 선정시 재설계 과정으로 회귀하여야 하는데 따르는 비용이 가장 큰 위험 요소라 할 수 있다. 따라서 설계 단계에서 적합한 아키텍처 접근법을 선정하는 것이 무엇보다 중요하다고 할 수 있고, 본 연구에서 제안하는 설계 방법은 이러한 위험 요소를 감소시킬 수 있는 방법이라 할 수 있다.

6. 결론 및 향후 연구 과제

본 논문에서는 소프트웨어 아키텍처 접근법 대안들을 설계 단계에서 식별하고 품질 속성 우선 순위에 따라 평가 과정을 반복함으로써 설계 단계에서 가장 적합한 아키텍처 접근법을 선정하기 위한 방법을 정의하고, 본 평가 방법을 통해 설계 단계에서 가장 적합한 아키텍처 접근법을 선정함으로써 설계 산출물의 신뢰도를 높일 수 있는 설계 방법론을 제안하였다.

본 논문에서 제안하고 있는 아키텍처 설계 방법의 특징 및 장점은 첫째 기존의 아키텍처 설계 및 평가 방법을 수정, 보완하여 활용하고 있고, 둘째 아키텍처 평가를 설계 단계로 통합하고 있다. 셋째 객관적인 평가 방법 활용하고 있고, 넷째 관련 품질 속성 또는 상충 요소를 가지고 있는 품질 속성들을 그룹화하여 평가하고 있다.

이 연구를 통해 중요한 품질 요구 사항을 실현하는 아키텍처 접근법이 설계 단계에서 결정됨으로써, 설계 이후의 평가 과정에서 새로운 아키텍처 접근법이 선정되는 사례가 감소할 수 있고 재설계에 따른 비용의 절감 또한 기대할 수 있으며, 검증된 아키텍처 접근법을 기반으로 아키텍처 설계를 완성함으로써 후속 설계 작업의 성공을 높이고 있다.

뿐만 아니라 본 연구에서 제안하는 평가 방법은 현실적으로 평가팀을 구성하고 조직간의 협의를 통해 진행하는 ATAM과 같은 평가 방법을 적용하기 힘든 소규모의 개발 프로젝트의 경우나, 평가시 비용을 고려하기 힘든 조직 또는 설계 단계에서 아키텍처 설계의 적합성 여부를 평가해 보고자 하는 설계자들이 쉽게 활용할 수 평가 방법이라고 할 수 있다.

향후 연구 과제로는 본 연구에서 제안한 평가 방법과 설계 방법을 보다 많은 실 사례에 적용하여 보고, 나타나는 문제점들에 대한 보완 작업이 이루어져야 한다.

참 고 문 헌

[1] 최은만, "객체지향 소프트웨어 공학", 사이텍미디어, pp.194-195, 2005.

[2] Felix Bachmann, Len Bass, Gary Chastek, Patrick Donohoe, Fabio Peruzzi, "The Architecture Based Design Method," Technical Report CMU/SEI, pp.3, 2000.

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

[4] Paul Clements, Rick Kazman, Mark Klein, "Evaluating Software Architectures : Methods and Case Studies," Addison Wesley, pp.44-68, 2002.

[5] Len Bass, Paul Clements, Rick Kazman, "Software Architecture in Practice Second Edition," Addison Wesley, pp.271-304, pp.308-315, 2003.

[6] Robert L. Nord, Mario R. Barbacci, Paul Clements, Rick Kazman, Mark Klein, Liam O'Brien, James E. Tomayko "Intergrating the Architecture Trade-off Analysis Method(ATAM) with the Cost Benefit Analysis Method(CBAM)," Technical Report CMU/SEI, p.3, p.13, 2003.

[7] Richard Monson-Haefel, David Chappell, "Java Message Service," O'Reilly, pp.7-8, pp.126, 2000.

[8] 고현희, 궁상환, 박재년, "설계 패턴을 활용한 메시지 시스템의 소프트웨어 아키텍처 접근법 식별 및 평가", 한국인터넷정보학회 논문지 제6권 제4호, pp.11-18, 2005.

[9] 궁상환, "모바일 인터넷 환경에서 Dynamic Scalable 연결에 관한 연구", 한국 전자 통신 연구원, p.74, p.78, 2001.

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

[11] Mary Show, David Garlan, Software Architecture : Perspectives on an Emerging Discipline, Pierson Education, 1996.

[12] Mario R. Barbacci, Mark H. Klein, Charles B. Weinstock "Principles for Evaluating the Quality Atributes of a Software Architecture," Technical Report CMU/SEI, 1997.

[13] Mark Klein, Rick Kazman, "Attribute-Based Architectural Styles," Technical Report CMU/SEI, 1999.

[14] Gregor Hohpe, Bobby Woolf, "Enterprise Integration Patterns : Designing Building, and Deploying Messaging Solutions," Addison-Wesley, 2003.

[15] Rick Kazman, Len Bass, "Toward Deriving Software Architectures From Quality Attributes," Technical Report CMU/SEI, 1994.

[16] Rick Kazman, Len Bass, Gregory Abowd, Mike Webb, "SAAM : A Method for Analyzing the Properties of Software Architectures," International Conference on Software Engineering, 1994.

[17] Mugurel T. Ionita, Dieter K. Hammer and Henk Obbink, "Scenario-Based Software Architecture Evaluation Methods: An Overview," International Conference on Software Engineering, 2002.

고 현 희

1992년 숙명여자대학교 전자계산학과(이학사). 2000년 숙명여자대학교 컴퓨터과학과(이학석사). 2005년 숙명여자대학교 컴퓨터과학과(이학박사). 1991년~1993년 코오롱정보통신 연구원. 1993년~1999년 LG산전연구소 주임연구원. 2000년~2002년 LG전자 정보통신연구소 선임연구원. 2002년~2002년 미국 카네기 멜론 대학 SE 과정 수료. 관심분야는 소프트웨어 설계 방법론, 소프트웨어 아키텍처, 설계 패턴

궁 상 환

1977년 숭실 대학교 전자계산학과(이학사). 1983년 고려대학교 전자정보처리학과(경영학석사). 1998년 충북대학교 전자계산학과 졸업(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 백석대학교 정보통신학부 조교수. 관심분야는 소프트웨어 구조, 분산시스템

박 재 년

1966년 고려대학교 물리학과(이학사). 1969년 고려대학교 고에너지 물리학과(이학석사). 1972년 독일 함부크르대학 전산학과. 1981년 고려대학교 고에너지 물리학과(이학박사). 1970~1972년 독일 국립입자 가속기 연구소(DESY) 연구원. 1972년~1979년 고려대학교 전자계산소 총간사. 1978년~1983년 전남대학교 계산통계학과 교수. 1987년~1998년 숙명여자대학교 전자계산소 원장. 1983년~현재 숙명여자대학교 정보과학부 교수. 2002년~현재 숙명여자대학교 이과대학 학장 관심분야는 시스템 분석 및 설계, 정보구조도, 컴포넌트 기반 개발 방법론