아키텍처 기반 소프트웨어의 일반적 요구사항과 이식성의 품질 측정 방안

General Requirements and Portability Measurement Method of Architecture Base

강종업*, 양해술**
서울벤처정보대학원대학교*, 호서대학교 벤처전문대학원**

Jong-Up Kang(kju5311@hanmail.net)*, Hae-Sool Yang(hsyang@office.hoseo.ac.kr)**

요약

소프트웨어 시스템을 구축하는데 아키텍처 기반 메카니즘은 중심에 있다. 성공적인 소프트웨어 시스템의 구축은 얼마나 견고하게 소프트웨어를 정의하는가에 있다. 아키텍처 기반 소프트웨어가 중요한 이유는 이해당사자 사이의 의사소통의 수단으로 사용되며 시스템의 초기 설계 결정 사항을 표현하고 재사용이 가능하기 때문이다. 본 연구에서는 아키텍처의 패턴 및 중요성 및 일반적 요구사항과 이식성의 특성을 분석하였으며 ISO/IEC 9126 및 ISO/IEC 14598을 고려하여 체계 및 메트릭을 제안하였다. 또한, 아키텍처 기반 소프트웨어의 시험절차에 따라 시험한 결함내역과 성능시험을 하고 결과를 분석하였다.

■ 중심어 : | 아키텍처 | 이식성 | 품질 측정 방안 |

Abstract

Architecture base mechanism centers to construct software system. There is construction of successful software system how firmly define software. Reason that architecture base software is important is used and expresses early design decision item of system and is because reusability is possible by means of communication between comprehension person concerned. Analyzed special quality of pattern and importance of architecture and general requirement and portability and consider ISO/IEC 9126 and ISO/IEC 14598 and propose system and Metrik in this research. Also, do defect particulars and efficiency test that test according to examination formality of architecture base software and analyzed result.

■ keyword : | Architecture | Portability | Quality Measurement Method |

Ⅰ. 서 론

아키텍처 관련 이론은 건축분야에서 최초로 등장 했으며 조형예술에 대한 인식이 증대되고 건축물에도 아름다움과 기하학적 견고함과 이에 기반한 정교한 설계가 요구되면서 더욱 주목을 받게 되었다. 이러한 요구들은 단지 건축의 주거기능을 향상시키기 위한 목적에 머무르지 않고 예술, 과학, 산업수준으로까지 건축의 영역을 확대 시키게 되었다. 컴퓨터가 발명되기 전까지는 설계자가 필요한 영역은 건축분야가 유일했기 때문에

* "본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음" (NIPA-2009-(C1090-0902-0032))
접수번호 : #090626-002
접수일자 : 2009년 06월 26일
심사완료일 : 2009년 08월 03일
교신저자 : 양해술, e-mail : hsyang@office.hoseo.ac.kr

아키텍처는 건축분야에만 쓰이는 용어가 되었다. 초기의 컴퓨터가 처리해야 하는 데이터는 그 양이 적었기 때문에, 딱히 설계, 학문적인 견고함 등이 개입될 여지가 적어 당시 소프트웨어 개발자는 단순 기능역할이었으나 컴퓨터가 강력해지고, 점점 더 많은 데이터를 처리하게 되면서, 더욱더 유연하고 강력하게 작동할 수 있도록 설계되어진 소프트웨어의 필요성을 산업전반에서 요구하게 되어 소프트웨어 분야에도 소프트웨어 아키텍처가 사용되어지게 되었다[14][15].

기능공에서 아키텍처 수준으로 발전을 하기 위해서는, 기능구현을 위한 노력외에도 산업자체가 이를 지원할 수 있는 수준에 올라야 됨을 의미한다. 소프트웨어 시스템을 구축하는데 있어서 아키텍처 기반 소프트웨어는 그 중심에 있으며 성공적으로 소프트웨어 시스템을 구축할 수 있느냐 없느냐가 얼마나 견고하게 소프트웨어를 정의하는가에 달려있다고 해도 과언이 아니다. 아키텍처 기반 소프트웨어가 중요한 이유는 이해당사자들 사이의 서로 다른 관점을 이해하고, 협상하며, 합의를 끌어내며, 서로 의사소통하는데 사용될 수 있는 공통적인 언어로서 이해당사자 사이의 의사소통의 수단으로 사용되며 시스템의 초기 설계 결정 사항을 표현하고 재사용이 가능하다는 점이다[9][10].

따라서, 본 연구의 2장에서는 아키텍처의 관련동향과 아키텍처의 패턴 및 중요성 및 일반적 요구사항과 이식성의 특성을 분석하고 3장에서는 ISO/IEC 9126 및 ISO/IEC 14598의 체계 및 메트릭을 구축하였으며 4장과 5장에서는 아키텍처 기반 소프트웨어의 시험절차 및 결함내역, 성능시험을 측정하고 결과를 분석하였다.

II. 관련 연구

1. 아키텍처의 관련동향

최근 정보기술 아키텍처(ITA:Information Technology Architecture)/전사적 아키텍처(EA:Enterprise Architecture) 개념을 도입하여 정보화사업 추진 및 정보자원 관리 기능을 강조하는 법규가 제정되면서, "아키텍처"라는 용어에 대한 관심이 높아지고 있다.

따라서, 체계통합(SI)/컨설팅 업체들이 공공기관의 ITA/EA 구축 사업에 참여하는 빈도가 늘어나면서, 아키텍처 기술과 방법론, 그리고 적용사례 등이 각종 세미나 등에서 다양한 시각을 가지고 발표되고 있다. 본 장에서는 최근 부각되고 있는 아키텍처의 관련 동향을 기술하였다.

1.1 아키텍처의 기본개념

아키텍처란 대상물을 형성하고 있는 구성품(Component)의 구성(Structure)과 그들간의 관련성(Relationship)이라고 정의될 수 있으며, 그들의 설계와 진화과정에 적응되는 원칙과 지침을 포함한다. 따라서, 대상물이 무엇이냐에 따라 시스템 아키텍처, 소프트웨어 아키텍처 등으로 불릴 수 있다. 아키텍처의 개념은 시스템공학 또는 소프트웨어공학분야에서 정하는 시스템 또는 소프트웨어 개발 절차에서도 이미 반영되어 있다.

1.1.1 아키텍처 프레임워크

체계의 아키텍처를 구분하고 정의하는 방식을 아키텍처 프레임워크(Framework)라고 하는데, Zachman 프레임워크를 기반으로 부서/기관별로 다양한 형태로 발전하고 있다[3][4].

Zachman 프레임워크는 세로열과 가로열로 구성되는 표 형태로 정의되며, 각 셀에 아키텍처를 설명하는 산출물을 설정하게 된다. 가로열은 관점에 대한 구성요소로서, What(Data), How(Function), Where(Network), Who (People), When(Time), Why(Motivation)로 구분하며, 세로열은 사용자/작성자 측면에서 계획자, 소유자, 설계자, 개발자, 구축자 등으로 구분한다[5].

최근에는 아키텍처 프레임워크를 아키텍처 개발방법론으로 간주하는 경향이 있는데, 이를 위해서는 아키텍처 산출물을 정의하는 아키텍처 기반의 체계설계를 위한 절차까지를 정의하게 된다.

1.1.2 개발지원도구 및 리포지토리

소프트웨어 개발방법론의 경우와 마찬가지로 아키텍처 기반의 체계설계를 위해서는 방법론이라고 할 수 있는 아키텍처프레임워크를 지원하는(정해진 아키텍처

산출물을 작성하기 위한) 지원도구가 필요하며, 작성되어진 산출물 전체 또는 산출물이 담고 있는 정보를 리포지토리로 관리할 수 있는 또 다른 지원도구가 필요하게 된다.

기존의 소프트웨어공학 지원도구(CASE Tool)가 사용될 수 있으나, 체계공학 또는 아키텍처 개발 차원의 별도의 전용 도구를 사용할 수도 있다. 한편, 대부분의 개발도구는 독자적인 메타모델을 기반으로 리포지토리를 구성하게 되므로 이를 활용하여 아키텍처 관리체계로 사용할 수도 있으나, 리포지토리를 기반으로 다양한 응용기능을 제공하는 별도의 아키텍처 관리체계를 구성할 수도 있다[6].

1.2 아키텍처의 개발 절차 및 패턴

아키텍처 기반 소프트웨어의 개발 절차는 [그림 1]과 같이 6단계로 구분할 수 있는데 첫째, 사업정의로는 아키텍처 개발용도, 아키텍처 범주, WBS 작성, 팀 구성 및 일정계획 수립 등이 있으며 둘째, 아키텍처 전략정의로는 아키텍처 환경분석, 요구사항 및 동기분석, 정보기술 전략, 아키텍처 전략, 아키텍처 산출물 정의, 통합자료 사전준비 등이 있으며 셋째, 아키텍처 분석으로는 현재 아키텍처 자료수집, 현재 아키텍처 산출물 정의, 현재 아키텍처 분석, 개선요소 방향도출 등이 있으며, 넷째 목표 아키텍처 정의로는 운용관점 요구사항, 상세설계, 운용구조체계 구조연결성정의, 기술구조설계, 상호운용성 소요정의 등이 있고 다섯째, 적용계획 수립으로는 구현 및 전환 계획 수립, 개발단계 전환준비 등이 있으며 마지막 아키텍처 운용 및 관리로는 아키텍처 자료 승인, 아키텍처 활용, 아키텍처 리포지토리 관리 등이 있다[7][8].

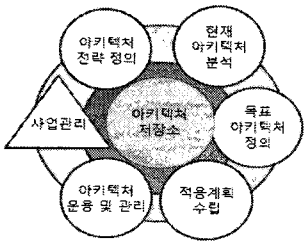

그림 1. 아키텍처 개발절차

그리고, 아키텍처 기반 소프트웨어의 패턴이란 여러 문제들이 경쟁하는 상황에서 특정 문제에 대한 반복해서 나타나는 해결책을 문서화한 것이며 설계 생산성을 높이고 문제와 해결책을 문서화해서 문제 영역에서 쓰는 어휘와 언어를 정의한다. 또한, 언어는 개별 패턴으로는 해결하기 힘든 복잡한 문제를 해결하고자 패턴들을 일정한 규칙에 따라 모아 놓은 것으로 패턴과 패턴 언어는 소프트웨어 공학의 기본 어휘와 언어로 소프트웨어 공학의 거의 모든 분야에 패턴이란 개념이 적용된다. [그림 2]는 아키텍처 패턴의 하나로 POSA(Pattern Oriented Software Architecture)의 패턴의 예를 보여주고 있다[11][12].

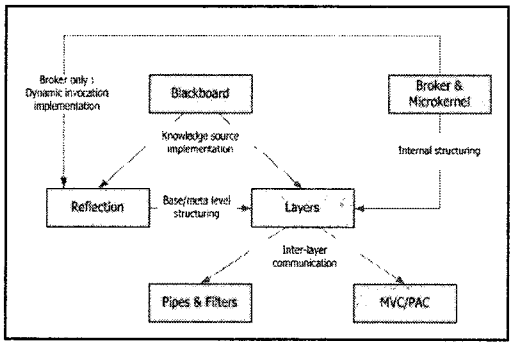

그림 2. 아키텍처 패턴의 예(POSA)

1.3 국제표준의 이식성에 관한 품질특성

ISO/IEC 9126 표준에서 이런 특성 체계를 기반으로 평가하는데 ISO/IEC 9126 표준은 일반적인 소프트웨어 품질평가를 위한 지침이니 아키텍처 소프트웨어에 직접적으로 적용할 수 없어 국제표준의 품질특성 부특성

표 1. ISO/IEC 9126의 이식성의 체계

이식성	
적응성	상이한 환경에 적응하는데 필요한 최소한의 조치만으로 이식될 수 있는 능력
설치성	지정된 환경에 설치될 수 있는 능력
공존성	다른 독립적인 소프트웨어와 공동자원을 사용해야 하는 공동 환경에서 공존할 수 있는 능력
대체성	동일 환경에서 같은 목적으로 사용되는 다른 소프트웨어를 대신하여 사용될 수 있는 능력
준수성	이식성에 관한 표준, 관례 등을 따르는 능력

성 체계를 수용해서 객관성을 최대한 제고하되 아키텍처 소프트웨어의 품질평가에도 최적화해야 하기 때문에 아키텍처 소프트웨어의 특성 및 요구사항을 분석하고 국제표준의 이식성 품질특성과 연계하여 분석함으로서 국제표준의 품질특성을 수용한 아키텍처 소프트웨어의 이식성 평가방법을 구축하고자 하였다.

III. 아키텍처 소프트웨어의 시험모듈

1. 국제 품질 표준

소프트웨어 관련 국제 품질표준은 ISO/IEC 9126과 ISO/IEC 14598, ISO/IEC 12119등이 있으며 현재, [그림 3]과 같이 ISO/IEC 25000시리즈로 통합을 추진하고 있다.

표 2. ISO/IEC 9126의 품질특성의 체계

이식성		사용성	
적응성	상이한 환경에 적응하는데 필요한 최소한의 조치만으로 이식될 수 있는 능력	이해성	사용자가 소프트웨어를 활용하기 위한 방법이나 조건, 적정성 등을 파악할 수 있게 하는 능력
설치성	지정된 환경에 설치될 수 있는 능력	습득성	사용자가 소프트웨어의 응용을 배울수 있게 하는 능력
공존성	다른 독립적인 소프트웨어와 공동자원을 사용해야 하는 공동 환경에서 공존할 수 있는 능력	운용성	사용자가 소프트웨어를 운영하고, 제어할 수 있도록 하는 능력
대체성	동일 환경에서 같은 목적으로 사용되는 다른 소프트웨어를 대신하여 사용될 수 있는 능력	친밀성	소프트웨어가 사용자에게 호감을 갖게 하는 능력
준수성	이식성에 관한 표준, 관례 등을 따르는 능력	준수성	사용성에 관한 표준, 규정, 관례, 스타일 등을 따르는 능력
기능성		효율성	
적합성	사용자의 목적하는 바에 따라 적절한 기능을 제공하는 능력	시간 반응성	주어진 조건에서 기능을 수행할 때 적절한 응답시간, 처리시간, 처리율을 제공하는 능력
정확성	올바른 혹은 동의된 효능 결과를 제공할 수 있는 능력	자원 효율성	주어진 조건에서 기능을 수행할 때 적절한 양과 종류의 자원을 사용하는 능력
상호 운용성	하나 이상의 지정된 타 시스템과 서로 작동하는 능력		
보안성	인가되지 않은 사람이나 시스템의 액세스를 방지하여 정보 및 데이터를 보호하는 능력	준수성	효율성에 관한 표준, 관례 등을 따르는 능력
준수성	그 응용에 관란 표준, 규정, 관례 등을 따르는 능력		
신뢰성		유지보수성	
성숙성	프로그램이나 데이터의 결함으로 인한 기능장애를 피할 수 있는 능력	분석성	소프트웨어의 약점이나 장애 원인을 진단하고 변경될 부분을 식별할 수 있게 하는 능력
결함 허용성	결함이나 인터페이스 문제 발생시에도 지정된 수준의 성능을 유지하는 능력	변경성	지정된 변경사항이 구현될 수 있게 하는 능력
회복성	장애발생시 지정된 수준의 성능을 회복하고 데이터를 복구하는 능력	안정성	변경에 따른 예상 밖의 결과를 최소화 하는 능력
		시험성	변경되었을 경우 검증 받을 수 있는 능력
준수성	신뢰성에 관한 표준, 규정, 관례등을 따르는 능력	준수성	유지보수성에 관한 표준, 관례 등을 따르는 능력

1.1 ISO/IEC 9126

소프트웨어 품질관리를 위해 4개의 부분으로 나누어 문서를 추진하고 소프트웨어 품질의 외부 특성과 내부 특성 사용에 있어서의 품질에 대한 연구 결과를 수록한 국제 표준 문건이다. 즉, 소프트웨어 제품평가에 관한 국제표준인 ISO/IEC 9126에서 정의하고 있는 각 품질 특성의 체제를 살펴보면 [표 2]의 이식성은 한 환경에서 다른 환경으로 전이될 수 있는 소프트웨어 제품의 능력을 평가하는 것이며, 이식성의 부특성으로는 적응성, 설치성, 공존성, 대체성, 준수성을 참조하여 주특성인 이식성을 측정할 수 있다. 현재 품질요구사항, 품질평가 모델 수립 방안 및 소프트웨어 품질 측정 기법 그리고 이들을 총괄하는 지침서 등의 표준화 진행을 SQuaRE 프로젝트라는 이름으로 진행하고 있다.

1.2 ISO/IEC 14598

소프트웨어 제품 평가를 위한 측정 기준과 메트릭 선정을 위한 요구사항과 지침을 제공하고 있다. 현재, ISO/IEC 9126과 ISO/IEC 14598 시리즈의 불일치성을 인지하여 표준을 체계적으로 적용하고 더욱 정량적인 평가가 가능한 방안을 마련하기 위해 SQuaRE 프로젝트(ISO/IEC 25000시리즈)를 진행하고 있다.

ISO/IEC 14598은 평가 프로세스를 규정한 표준으로, 평가는 [표 3]과 같은 특성을 가져야 한다[13].

표 3. ISO/IEC 14598의 표준

특성	원칙
반복성 (Repeatability)	어떤 하나의 제품에 대해 동일 평가자가 동일 사양에 따라 평가 했을 때 동일하다고 여길 수 있는 결과가 나와야 한다.
재생산성 (Reproducibility)	어떤 하나의 제품에 대해 다른 평가자가 동일 사양에 따라 평가 했을 때 동일하다고 여길 수 있는 결과가 나와야 한다.
공평성 (Impartiality)	평가는 특정 결과에 편향되어서는 안된다.
객관성 (Objectivity)	평가 결과는 평가자의 감정이나 의견에 의해 영향을 받아서는 안된다.

ISO/IEC 14598에서 사용하는 품질 모델은 ISO/IEC 9126에서 규정한 표준에 따르고, 품질 평가를 위한 측정 기술, 측정결과의 해석 방법 등에 대한 내용은 규정하지 않으며 [표 4]와 같은 내용으로 구성되어 있다[16].

표 4. ISO/IEC 14598의 세부사항

표준 구분	내 용
ISO/IEC 14598-1	일반사항
ISO/IEC 14598-2	기획 및 관리
ISO/IEC 14598-3	개발자를 위한 프로세스
ISO/IEC 14598-4	구매자를 위한 프로세스
ISO/IEC 14598-5	평가자를 위한 프로세스
ISO/IEC 14598-6	평가모듈

1.3 ISO/IEC 12119

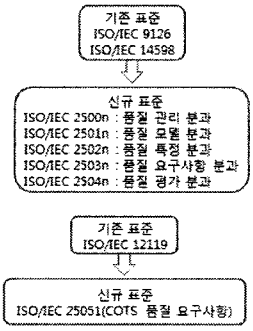

그림 3. 국제품질표준의 통합 추진 방향

ISO/IEC 12119는 패키지 소프트웨어에 대한 품질 요구사항과 시험에 관한 국제 표준 문건이며 [그림 3]은 기존의 국제 표준인 ISO/IEC 9126과 ISO/IEC 14598, ISO/IEC 12119등이 ISO/IEC 25000시리즈로 통합을 추진하고 있음을 보여준다[17].

2. 아키텍처 기반 소프트웨어 품질수준 메트릭

본 연구에서는 아키텍처 기반 소프트웨어의 일반적 요구사항과 이식성의 메트릭을 제안하여 아키텍처 기반 소프트웨어의 일반적 요구사항과 이식성을 시험하기 위한 방법을 모색하였다. 일반적 요구사항의 서비스는 다른 서비스나 프로그램에 의해 사용될 수 있으며 서비스들 간의 관계는 상호작용에 기반하고 있으므로, 서비스 상호 간에 식별이 가능해야 한다. 이는 서비스 명세(service description)를 통해 가능하다. 아키텍처 기반 소프트웨어의 일반적 요구사항의 시험사례 구축은 제품 설명서에 사용자가 아키텍처 기반 소프트웨어 제품에 대해 기능을 파악하고 자신의 요구사항에 일치함을 식별하기에 충분한 정보가 제공되는가, 서비스들 간의 관계는 상호작용에 기반하고 있으므로, 서비스 상호 간에 식별이 가능해야 하며 서비스 명세(service description)를 통해 서비스들을 식별하고 있는가, 소프트웨어가 제조 및 패키징 도중 바이러스에 감염되지 않았는지를 소프트웨어 제품평가를 위한 국제표준인 ISO/IEC 9126과 소프트웨어 시험에 관한 지침인 ISO/IEC 12119를 기반으로 [표 5]와 같이 구축하였다[1][2].

표 5. 아키텍처 소프트웨어 일반적 요구사항의 메트릭

구분	메트릭명	계산식	결과영역
일반적 요구사항	제품정보 제공(PIP)	제품정보제공(PIP) = (Y로 측정되는 항목 수) / (적용 가능한 데이터 항목 수)	0 ≤ 제품정보제공(PIP) ≤ 1
	서비스 상호간 식별	서비스 상호간 식별 = B(서비스 명세를 제공하고 있는 서비스의 수) / A(제공되고 있는 모든 서비스의 수)	0 ≤ 서비스 상호간 식별 ≤ 1
	바이러스 감염 여부	바이러스 감염 여부(VII) = (SOA 기반 소프트웨어의 바이러스 감염 여부)	바이러스 감염 여부(VII) = Y or N or NA

아키텍처 기반 소프트웨어의 이식성이란 한 환경에서 다른 환경으로 전이될 수 있는 소프트웨어 제품의 능력으로 여기서 환경이란 조직, 하드웨어 혹은 소프트

표 6. 아키텍처 기반 소프트웨어의 이식성의 메트릭

구분	메트릭명	계산식	결과 영역
이식성	데이터 구조 적응 정보 제공	데이터구조 적응 정보제공 = A(데이터구조 적응에 관한 정보 제공 여부)	데이터구조 적응 정보제공 = Y or N or NA
	데이터 구조 적응률	- 데이터구조 적응률 (DAR) = B(각 항목별 테스트케이스 성공률의 합)/A(평가할 데이터구조 적응시킬 데이터 항목 수) - B = $\sum_{i=1}^{A} \frac{Success\_TC_i}{Total\_TC_i}$ - Success_TC : i 번째 데이터 항목 확인을 위해 수행한 테스트케이스 중 성공한 건 수 - Total_TC : i 번째 데이터 항목 확인을 위해 수행한 테스트케이스 수	0 ≤ 데이터구조 적응률 ≤ 1
	설치 정보 제공	설치 정보제공 = (Y로 측정되는 항목 수) / (데이터 항목 수)	0 ≤ 설치 정보제공 ≤ 1
	설치 가능률	설치 가능률 = B(성공한 설치(설치 재시도) 횟수)/A(시도한 설치(설치 재시도) 횟수)	0 ≤ 설치 가능률 ≤ 1
	데이터 지속 정보 제공	데이터 지속 정보제공 = A(데이터 대체에 관한 정보 제공 여부)	데이터 지속 정보 제공 = Y or N or NA
	데이터 지속 가능률	- 데이터 지속가능률 = B(각 항목별 테스트케이스 성공률의 합)/A(평가할 대체 가능한 데이터 형식 항목 수) - B = $\sum_{i=1}^{A} \frac{Success\_TC_i}{Total\_TC_i}$ - Success_TC : i 번째 항목 확인을 위해 수행한 테스트케이스 중 성공한 건 수 - Total_TC : i 번째 항목 확인을 위해 수행한 테스트케이스 수	0 ≤ 데이터 지속 가능률 ≤ 1
	공존 가능 정보 제공	공존 가능 정보 제공 = A(소프트웨어 제품을 사용자가 자주 사용하는 다른 소프트웨어와 동시에 사용할 수 있는지에 대한 정보제공 여부)	공존 가능 정보 제공 여부 = Y or N or NA
	공존 가능률	- 공존 가능률 = B(각 항목별 테스트케이스 성공률의 합)/A(평가할 공존 프로그램 항목 수) - B = $\sum_{i=1}^{A} \frac{Success\_TC_i}{Total\_TC_i}$ - Success_TC : i 번째 공존 프로그램 확인을 위해 수행한 테스트케이스 중 성공한 건 수 - Total_TC : i 번째 공존 프로그램 확인을 위해 수행한 테스트케이스 수	0 < 공존 가능률
	이식 표준 준수 정보 제공	이식 표준 준수 정보 제공 = A(이식 관련 표준 준수 정보 제공 여부)	이식 표준 준수 정보 제공 = Y or N or NA

웨어 환경을 의미한다. 부특성으로는 고려 대상인 소프트웨어에서 이 목적으로 제공되는 것 이외의 활동 혹은 수단을 적용하지 않고 다른 명세된 환경으로 변경될 수 있는 소프트웨어 제품의 능력을 가지며 내부 용량(화면 영역, 테이블, 트랜잭션 크기, 보고서 형식 등)의 확장성을 포함한다. 그리고 소프트웨어가 최종 사용자에 의해 개작된다면 개별화를 위한 적합성에 대응되면, 운용성에 영향을 줄 수 있는 적응성, 명세된 환경에 설치될 수 있는 소프트웨어 제품의 능력을 가지며 최종사용자에 의해 설치되면 적합성과 운영성에 영향을 줄 수 있는 설치성, 공통 자원을 공유하는 공동 환경에서 다른 독립적인 소프트웨어와 공존할 수 있는 공존성, 동일한 환경에서 동일한 목적으로 다른 지정된 소프트웨어 제품을 대신하여 사용될 수 있는 대치성, 이식성과 관련된 표준 및 관례를 고수하는 준수성이 있다. 소프트웨어 제품평가를 위한 국제표준인 ISO/IEC 9126과 소프트웨어 시험에 관한 지침인 ISO/IEC 12119를 기반으로 [표 6]과 같이 구축하였다.

Ⅳ. 아키텍처 기반 소프트웨어의 시험

이식성은 소프트웨어가 특정 조건에서 사용될 때, 명시된 요구와 내재된 요구를 만족하는 기능을 제공하는 능력으로서 본 연구에서는 아키텍처 기반 소프트웨어의 일반적 요구사항과 이식성을 시험하기 위한 환경을 구성하였다.

1. 시험 환경 구축

시험대상 아키텍처 기반 소프트웨어는 계정 및 용량 관리, 부서 및 직책 관리, 게시판 관리 등의 메일 서버 관리, 메일 송수신, 수신확인, 바이러스 차단 및 스팸 필터링 등의 메일 기본 기능, 주소록 관리, 일정관리, 게시판 및 자료실 등의 부가기능을 처리할 수 있는 소프트웨어이다. 평가대상 제품의 소프트웨어를 시험하기 위해 메일서버에 설치한 프로그램은 DBMS : MySQL v5.0.51a이며 클라이언트에 설치한 프로그램으로는 웹 브라우저: Internet Explorer 7.0과 일반 응용프로그램인 MS-Office 2007, 한글 2005, 백신 프로그램 등을 사용하였으며 네트워크는 10/100Mbps 스위칭 허브를 사용하였다. 성능측정도구로 메일서버에 설치한 성능측정도구로는 TeamQuest Manager 10.1, 클라이언트에

설치한 성능측정도구로는 TeamQ uest View 10.1 을 설치하여 시험하였다.

2. 일반적 요구사항과 이식성의 시험

2.1 일반적 요구사항과 이식성의 시험결과

2.1.1 일반적 요구사항과 이식성의 결함내역

품질특성 및 결함속성별로 아키텍처 기반 소프트웨어에 대한 일반적 요구사항과 이식성의 결함 건수 및 내역 등을 정리하면 [표 7]과 같다.

표 7. 일반적 요구사항과 이식성의 결함 내역

품질특성	결함 수	결함 정도	결함내역	최종결함 수
일반적 요구사항	4	M	고객 인도 항목 미제공	0
			시스템 정보 미제공	0
			제품 정보 미제공	0
			제품 지원 정보 미제공	0
이식성	1	M	설치 및 삭제 정보 미제공	0
합계	5			0

적용대상 아키텍처 기반 소프트웨어의 일반적 요구사항에 대해 시험한 결과 [표 7]과 같이 고객 인도 항목 미제공, 시스템 정보 미제공, 제품 정보 미제공, 제품 지원 정보 미제공등 총 4건의 결함이 발생하였으나 수정 보완 및 회귀시험 과정을 거친 후 최종적으로 정확한 정보가 제공됨을 확인하였고 이식성의 결함을 시험한 결과 설치 및 삭제 정보를 제공하지 않는 오류가 발생하였으나 수정 보완 및 회귀시험 과정을 거친 후 최종적으로 이상이 없음을 확인하였다.

2.1.2 결함정도별 분류

결함 정도의 분류는 강한결함(H)인 경우 기능이 정상적으로 동작하지 않거나, 하드웨어 시스템 혹은 프로그램이 비정상적으로 종료되는 등의 치명적인 결함이 발생하는 경우로 분류하며, 중간결함(M)은 프로그램 운영에는 문제가 없으나, 기능이 정확하게 동작하지 않거나 사용자의 혼란을 야기하는 정도의 결함이 발생하는 경우로 분류할 수 있으며, 약한결함(L)은 프로그램 운영에 문제가 없고, 기능도 정확하게 동작하나 권고 사항 수준의 경미한 결함이 발생하는 경우를 들 수 있다.

본 연구의 일반적 요구사항과 이식성의 결함정도는 이식성에서의 중간결함이 1건 발생하였고 일반적 요구사항은 중간결함이 4건 발생하였으나 수정 보완 및 회귀시험 과정을 거친 후 최종적으로 이상이 없음을 확인하였다.

2.2 아키텍처 기반 소프트웨어의 결함 분석

아키텍처 기반 소프트웨어의 일반적 요구사항과 이식성에서의 결함을 분석해 보면 이식성의 결함인 프로그램의 설치 및 삭제시 프로그램 사용자에게 설치 및 삭제 정보를 제공하지 않은 설치 및 삭제 정보 미제공의 결함이 나타났으며 일반적 요구사항의 결함중에서는 제품정보나 제품지원정보, 시스템정보를 제공하지 않는 정보 미제공 결함이 다수 나타남을 확인 하였다. 향후 개발하는 아키텍처 기반 소프트웨어 프로그램에서는 이에 대한 개선방안이 필요하다고 본다.

V. 성능시험 및 결과

성능시험 시나리오는 [표 8]과 같이 [시나리오 1, 2]로 나누어 [시나리오 1]에서는 시험대상 제품을 구동하고, [시나리오 2]에서는 메일 내용에 한글 500자를 입력하고 1MB, 10MB, 20MB, 30MB 파일을 첨부하며, [시나리오 3]에서는 27명의 동시사용자가 5분 동안 450통의 메일을 송신 및 수신하여 측정하였다.

표 8. 성능시험 시나리오

시나리오 구분	내 용
시나리오-1	시험대상 제품을 구동
시나리오-2	메일 송신 및 수신 -메일 내용에 한글 500자를 입력하고 1MB,10MB, 20MB, 30MB 파일을 첨부
시나리오-3	27명의 동시사용자가 5분동안 450통의 메일을 송신 및 수신 -메일 내용에 한글 500자를 입력하고 1MB 파일을 첨부

성능시험 결과는 [표 9]와 같이 자원효율성과 시간효율성을 측정하였으며, 자원효율성의 시험결과는 CPU

사용률과 메모리 사용률을 측정하였으며 시간효율성은 응답시간을 측정하였다.

표 9. 성능시험 측정 항목

항목		단위	내용	
자원 효율성	CPU 사용률	%	%usr	사용자모드로 스레드를 실행하는데 소비하는 시간의 백분율
			%sys	시스템모드로 스레드를 실행하는데 소비하는 시간의 백분율
			%idle	시스템이 idle 상태에 있었던 시간의 백분율
	메모리 사용량	MB	cachedmem	시스템에서 캐시되어 사용된 평균 메모리 양
			usedmem	시스템에서 사용된 평균 메모리 양
시간 효율성	응답 시간	초	시스템에 조회나 요구 등을 입력한 직후부터 해당 명령의 처리가 완료된 시점까지 소요된 시간	

1. 자원 효율성

1.1 CPU 사용률

[그림 4]와 같이 아키텍처기반 소프트웨어의 제품을 구동하는 경우, 메일서버의 CPU 사용률이 일시적으로 최고 6.98%까지 올라갔으나 평균 CPU 사용률은 1%로 안정적으로 유지되고 있다.

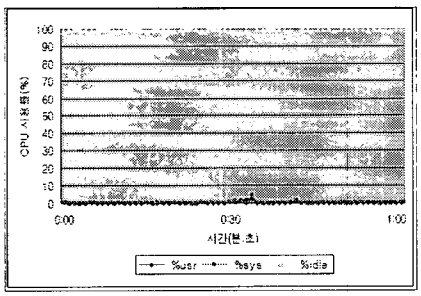

그림 4. 메일서버의 CPU 사용률

메일 내용에 한글 500자를 입력하고 각각 1MB, 10MB, 20MB, 30MB의 파일을 첨부한 후 메일을 송신하는 경우 메일서버의 평균 CPU 사용률은 각각 3%, 12%, 12%, 15%로 적절함을 알 수 있었으며 메일 내용에 한글 500자를 포함하고 각각 1MB, 10MB, 20MB, 30MB의 파일이 첨부된 메일을 수신하는 경우 메일서버의 평균 CPU 사용률은 각각 1%, 4%, 6%, 12%로 적절하였다. 27명의 동시 사용자가 메일 내용에 한글 500자를 입력하고 1MB의 파일을 첨부하여 5분간 450통의 메일을 송신하는 경우 메일서버의 평균 CPU 사용률은 42%로 적절하였고 메일 내용에 한글 500자를 포함하고 1MB의 파일이 첨부된 메일을 5분간 450통 수신하는 경우 [그림 5]와 같이 메일서버의 평균 CPU 사용률은 15%로 적절하였다.

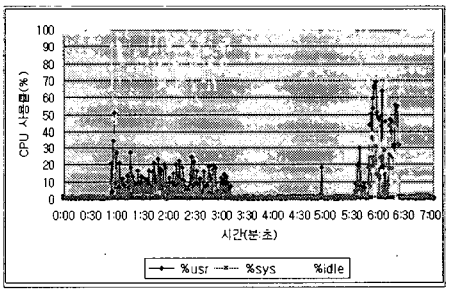

그림 5. 메일서버의 평균 CPU 사용률

1.2 메모리 사용량

제품을 구동하는 경우, 메일서버의 메모리 사용량은 평균 928MB로 안정적으로 유지되었고 메일 내용에 한글 500자를 입력하고 각각 1MB, 10MB, 20MB, 30MB의 파일을 첨부한 후 메일을 송신하는 경우 메일서버의 평균 메모리 사용량은 각각 1,187MB, 1,252MB, 1,447MB, 1,610MB로 적절하였다. 메일 내용에 한글 500자를 포함하고 각각 1MB, 10MB, 20MB, 30MB의 파일이 첨부된 메일을 수신하는 경우 메일서버의 평균 메모리 사용량은 각각 1,605MB, 1,620.79MB, 1,713MB, 1,880MB로 적절하였으며 27명의 동시 사용자가 메일 내용에 한글 500자를 입력하고 1MB의 파일을 첨부하여 5분간 450통의 메일을 송신하는 경우 메일서버의 평균 메모리 사용량은 1,670MB로 적절하였다. 메일 내용에 한글 500자를 포함하고 1MB의 파일이 첨부된 메일을 5분간 450통 수신하는 경우 메일서버의 평균 메모리 사용량은 1,309MB로 적절하였다.

2. 시간 효율성

2.1 응답시간

메일 내용에 한글 500자를 입력하고 각각 1MB, 10MB, 20MB, 30MB의 파일을 첨부한 후 메일을 송신

하는 경우 [그림 6]과 같이 클라이언트의 평균 응답시간은 각각 0.91초, 1.19초, 1.72초, 2.29초로 적절함을 알 수 있다.

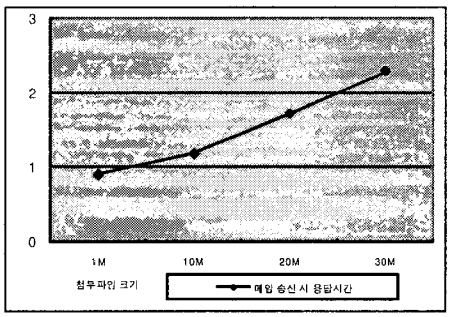

그림 6. 클라이언트의 평균 응답시간

메일 내용에 한글 500자를 포함하고 각각 1MB, 10MB, 20MB, 30MB의 파일이 첨부된 메일을 수신하는 경우 클라이언트의 응답시간은 약 7초, 9초, 12초, 19초로 적절하였다.

3. 평가방법의 비교분석

[표 10]의 ISO/IEC 9126과 ISO/IEC 12119 기반의 품질평가에 대한 장단점과 기존의 평가 방법에 대해 기술하고 비교하였다. ISO/IEC 9126은 ISO/IEC 9126-2의 외부매트릭에 의한 평가와 ISO/IEC 9126-3의 내부메트릭에 의한 평가로 분류할 수 있다. 외부메트릭에 의한 평가는 국제표준을 기반으로 하여 상대적으로 높은 객관성을 가지며 실행 프로그램의 평가에는 적합하지만 라이프사이클 전반에 적용할 수 없다. 내부메트릭에 의한 평가는 높은 객관성을 가지며 실행 프로그램에 한정되지 않고 소프트웨어 개발 전 과정의 중간산출물 대상으로 하여 소프트웨어 라이프사이클 전반에 걸쳐 적용할 수 있지만 중간산출물의 품질 측정을 통해 최종 소프트웨어 제품인 실행 프로그램의 품질을 예측하는 수준에 그칠 뿐 확신할 수 없다는 단점이 있다.

ISO/IEC 12119 기반의 품질평가 방법의 경우에는 국제표준을 기반으로 하여 객관성을 확보할 수 있으며 소프트웨어의의 다수를 차지하는 패키지 소프트웨어의 평가에 적합하지만 기본적인 표준만으로는 일반적인 사무용 패키지 소프트웨어 중심으로서 다양한 소프트웨어 분야에 적용하기 쉽지 않다.

본 연구의 평가 방법은 ISO/IEC 9126과 12119를 기반으로 아키텍처 기반 소프트웨어의 이식성과 일반적 요구사항에 초점을 맞추어 핵심적이고 최적화된 평가가 가능하지만 범용적인 품질평가 표준을 기반으로 하여 아키텍처 기반 소프트웨어의 특성을 수용하여 구체화하였으므로 아키텍처 기반 소프트웨어 고유의 특성에 대한 반영이 미흡할 수 있으므로 향후, 아키텍처 기반 소프트웨어의 관련 표준을 프레임워크로 한 품질평가 방법에 대한 연구가 추진되어야 할 것으로 생각된다.

표 10. 품질평가 방법의 비교

구분 / 평가방법		장점	단점	비고
ISO /IEC 9126 기반의 평가 방법	외부 메트릭 기반	국제표준을 기반으로 하여 상대적으로 높은 객관성을 가지며 실행 프로그램의 평가에 적합	실행 프로그램을 대상으로 평가하는 데 한정되므로 라이프사이클 전반에 적용할 수 없음	
	내부 메트릭 기반	높은 객관성을 가지며 실행 프로그램에 한정되지 않고 S/W 개발 전 과정의 중간산출물 대상으로 함	중간산출물의 품질로 실행 프로그램의 품질을 예측하나 확신할 수 없음	
ISO/IEC 12119 기반의 품질평가 방법		국제표준을 기반으로 하여 높은 객관성을 가지며 S/W의 다수를 차지하는 패키지 S/W 평가에 적합	일반적인 사무용 패키지 S/W 중심으로서 다양한 S/W 분야에 적용하기 쉽지 않음	평가대상 S/W의확대를위한 연구활발
본 연구의 평가방법		ISO/IEC 9126과 12119를 기반으로 아키텍처 기반 S/W의 이식성과 일반적 요구사항에 초점을 맞추어 핵심적이고 최적화된 평가 가능	범용적인 품질평가 표준을 기반으로 아키텍처 기반 S/W의 특성을 수용하여 구체화하였으므로 고유의 특성에 대한 반영이 미흡할 수 있음	

VI. 결 론

소프트웨어 제품의 품질이 중요한 관건으로 대두된 지 오래이며 소프트웨어 제품 품질에 대한 인증의 중요성이 높아짐에 따라 다양한 소프트웨어 유형에 따른 품질시험 및 인증 방법에 대한 연구가 활성화되고 있다.

본 연구에서는 아키텍처 기반 소프트웨어 제품의 품

질 수준을 파악할 수 있는 지표를 도출하여 지표산식을 정의하고 지표의 결과를 산출하기 위해 필요한 수집항목을 선정하며 수집과 분석을 통해 실질적으로 어떤 결함 유형들이 주로 발생하고 있는지를 확인하였으며 아키텍처 기반 소프트웨어에 대한 시험 평가 모델을 개발하여 시험 사례를 분석하였다. 향후 실질적인 활용을 통해 고품질 아키텍처 기반 소프트웨어의 개발을 촉진함으로써 높은 부가가치를 창출하고 경쟁력을 갖춘 제품의 개발 지원이 가능하다고 본다.

참 고 문 헌

[1] ISO/IEC 15288, Systems Engineering - Systems Life Cycle Processes, Nov. 2002.

[2] ISO/IEC 12207 Amendment 1, Information technology - S/W Life Cycle Processes, 2002.

[3] Zachman, John A., "The Physics of Enterprise Architecture," Enterprise Architectures Conference, Zachman Int'l. 2000.

[4] US CIO Council, "Federal Enterprise Architecture Framework version 1.1," 1999.

[5] Christopher D. Kolenda, "Transforming How We Fight : A concepture Approach," Naval War College Review, 2003.

[6] DoD Architecture Framework Version 1.0 Volume I, 2003.

[7] DoD Instruction 5000.2, "Operation of the Defense Acquisition System," 2001.

[8] Ken Williams, Operational Relevance obtained from Architecture Integration, U.S. Joint Forces Command, 2006.

[9] DoD OASD, DoD Architecture Framwork Ver. 1.0, 2004(2).

[10] CJCSI 6212.01C, "Joint Capavilities Integration and Development System," 2003.

[11] DoD OASD, "C4ISR CADM Version 2.0 Final Report," 1998.

[12] 정보통신기술협회, "공공부문 전사적 아키텍처 프레임워크 표준(TTAS. KO-10.01535)", 2003(12).

[13] 국방부(국과연), "국방아키텍처프레임워크 (MND-AF Ver. 1.0 사용자지침서)", 2005(2).

[14] 김정호, 송재하, "아키텍처 기반 소프트웨어 이론과 실제", 에이콘, 2007.

[15] 전병선, "객체지향 CBD 개발 방법론", 영진com, 2004.

[16] 양해술, "소프트웨어 시험평가 모듈 개선 연구", ETRI 컴퓨터 · 소프트웨어기술연구소 위탁과제, 최종보고서, 2001(11).

[17] 진종현, "국방 통합아키텍처관리체계(ICAMS) 개발", 정보과학회지, 제24권 제9호, 2006(9).

저 자 소 개

강 종 업(Jong-Up Kang) 정회원

- 1990년 8월 : Bernadean 대학교 경영학과 졸업(학사)
- 2006년 8월 : 서울벤처정보대학원대학교 정보경영학과 졸업(석사)
- 2006년 8월 : 서울벤처정보대학원대학교 정보경영학과 박사과정
- 1976년 5월 ~ 1991년 11월 : 위례정보산업고, 예일여자고, 거창중앙고 교사
- 1999년 9월 ~ 2002년 2월 : 마산대학 인터넷비즈니스과 겸임교수
- 2001년 4월 ~ 2005년 12월 : (주)용두종합건설 회장
- 1991년 11월 ~ 현재 : 해룡한방민간요법연구소 소장

<관심분야> : 정보처리, 재무관리 및 경영지도, 전자상거래, 품질경영

양 해 술(Hae-Sool Yang) 정회원

- 1975년 2월 : 홍익대학교 전기공학과 졸업(학사)
- 1978년 8월 : 성균관대학교 정보처리학과 졸업(석사)
- 1991년 3월 : 日本 오사카대학 정보공학과 S/W공학 전공(공학박사)
- 1975년 5월 ~ 1979년 6월 : 육군중앙경리단 전자계산실 시스템분석장교
- 1980년 3월 ~ 1995년 5월 : 강원대학교 전자계산학과 교수
- 1986년 12월 ~ 1987년 12월 : 日本 오사카대학교 객원연구원
- 1995년 6월 ~ 2002년 12월 : 한국소프트웨어품질연구소 소장
- 1999년 11월 ~ 현재 : 호서대학교 벤처전문대학원 교수

<관심분야> : S/W공학(특히, S/W 품질보증과 품질평가, 품질감리 및 컨설팅, OOA/OOD/OOP, SI), S/W 프로젝트관리, 품질경영