韓國컴퓨터情報學會 論文誌
第14卷 第3號, 2009. 3.

2009-14-3-4-2

서비스지향 아키텍처 소프트웨어의 신뢰성 평가 모델

최 인 용*, 양 해 술**

Reliability Evaluation Model for Service-Oriented Architecture Software

In-Yong Choi *, Yang Hae Sool **

요 약

현재 서비스지향 아키텍처 소프트웨어의 중요성이 인식되면서 국내외 서비스지향 아키텍처 소프트웨어 시장이 급격히 증가하고 있는 추세이다. 이에 따라 서비스지향 아키텍처 소프트웨어에 대한 고신뢰성과 고품질 소프트웨어의 요구가 증대되고 있다. 본 연구에서는 서비스지향 아키텍처 소프트웨어의 신뢰성 품질을 평가하기 위해 품질을 시험하여 측정하고 그 결과를 적절한 기준에 따라 판정하는 방법에 대해 연구를 수행하고 평가 사례를 제시하여 평가 방법을 명확히 제시하였다. 본 연구를 통해 서비스지향 아키텍처 소프트웨어의 신뢰성 품질 향상을 유도하고 국제 표준을 수용하는 전략기술 개발을 통해 객관성과 활용도를 높일 수 있을 것으로 기대한다.

Abstract

Nowadays, as the importance of service-oriented architecture software is recognized, the market of service-oriented software is getting bigger. In response to this, the requirements of high reliability and quality about service-oriented architecture software is getting increased. In this research, we clearly suggested the evaluation method by giving a specific evaluation example to evaluate the reliability quality of service-oriented architecture software. It is expected to raise the objectivity and the utilization by inducing the reliability quality improvement from this research.

▸ Keyword : 서비스지향 아키텍처(Service-Oriented Architecture), 신뢰성(Reliability),품질평가모델(Quality Evaluation Model), 품질특성(Quality Characteristics)

• 제1저자 : 최인용
• 투고일 : 2008. 9. 3, 심사일 : 2008. 9. 11, 게재확정일 : 2009. 1. 20.
* 서울벤처정보대학원대학교 컴퓨터응용기술학과 ** 호서대학교 벤처전문대학원 정보경영학과 교수
※ 본 연구는 지식경제부와 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음"
(IITA-2009-(C1090-0902-0032))

Ⅰ. 서 론

최근 들어 소프트웨어 기술 시장은 기존의 문제점 즉, 소프트웨어 기술 간의 상호운용성 문제와 서비스들 간의 융합 문제를 지원하기 위해 데이터 전송 기술의 표준화와 함께 다양한 이기종의 애플리케이션들에 대한 서비스화 기술 즉, 서비스지향 아키텍처(Service Oriented Architecture) 기술의 등장으로 소프트웨어 시장의 재편을 예고하고 있다.

이는 서비스지향 아키텍처 기반의 차세대 소프트웨어 기술들의 등장은 기존 소프트웨어 시장의 침체된 상황을 정리하고 신규 시장의 확대와 시장의 활성화를 유도할 것으로 예상된다[1].

최근의 비즈니스 환경은 과거 독립적인 조직 및 프로세스에 의해 주도되는 수직적 통합에서 고객, 공급자, 파트너 등 다수 기업과의 관계적 협업 관계가 중시되는 수평적 통합 환경으로 변화하고 있다[2]. 이러한 관계적 협업이 중시되는 비즈니스 환경에서 경쟁력 있는 기업은 급변하는 시장요구에 민첩하게 대응할 수 있어야 한다.

이러한 비즈니스 요구에 효율적으로 대응하기 위한 기업 IT 아키텍처로 대표되는 것이 서비스지향 아키텍처이다. 서비스지향 아키텍처는 전통적인 프로그램 중심의 설계/개발 방식에서 비즈니스 프로세스 관점에서 재활용 가능한 단위로 서비스를 설계/개발하게 함으로써 특정 프로세스나 서비스 변경 또는 내부/외부 시스템과의 비즈니스 통합 시 효율적이고 빠른 대응이 가능하다는 점에서 그 의미가 깊다.

서비스지향 아키텍처는 특정 기술이나 플랫폼에 종속되지 않고 느슨한 결합(Loosely Coupled)을 가지고 상호 연동할 수 있는 서비스들의 조합으로 어플리케이션 개발을 가능하게 하는 정보시스템 아키텍처이다[3]. 즉, 한 덩어리의 방대한 코드로 이루어진 어플리케이션들을 각각 개발하는 대신 각각의 비즈니스 기능을 수행하는 서비스를 구성하고, 이 서비스를 조합하거나 분리함으로써 비즈니스 프로세스들을 구현할 수 있게 하는 정보시스템 구축을 목표로 한다[15]. 이상과 같은 서비스지향 아키텍처 소프트웨어의 상용화가 급격히 진전되고 있는 시점에서 이에 따른 서비스지향 아키텍처 소프트웨어의 품질평가 요구에 대응하기 위해, 본 연구에서는 소프트웨어 품질평가 표준[4, 5, 6]을 기반으로 하여 서비스지향 아키텍처 소프트웨어의 신뢰성 특성을 분석함으로써 서비스지향 아키텍처 소프트웨어의 신뢰성 품질을 시험하여 측정하고 평가하는 모델을 개발하고자 한다.

본 연구의 2장에서는 서비스지향 아키텍처 소프트웨어의 관련 현황 및 전망을 소개하고 3장에서는 서비스지향 아키텍처 소프트웨어의 신뢰성 평가모델을 개발하기 위해 서비스지향 아키텍처 소프트웨어의 품질시험 및 평가를 위한 품질특성을 분석하고 4장에서는 서비스지향 아키텍처 소프트웨어의 신뢰성에 관한 품질특성 모델에 대해 기술하였다. 5장에서는 서비스지향 아키텍처 소프트웨어의 품질평가에 관한 전반적인 방법을 종합한 시험모듈에 대해 기술하고 6장에서는 개발된 시험모듈을 적용한 평가사례를 제시하였으며 마지막으로 결론과 향후 연구과제를 제시하였다.

Ⅱ. 서비스지향 아키텍처 관련 현황

1. 국내외 서비스지향 아키텍처 기술 현황

근래 들어 많은 국가들이 전자정보를 구축하기 위해 노력하고 있으며 특히 미국, 캐나다, 유럽의 여러 국가들은 가장 모범적인 구축 사례를 보여주고 있다.

미국 정부는 전자정부의 아키텍처를 개별 부처 및 기관별 서비스로부터 공통서비스로 전환하는 것을 목표로 하고 있다. 또한 미국 정부는 OMB에서 배포한 EA 평가프레임워크 2.0을 통해 서비스지향 아키텍처를 IT개발운영관리 구조로서 추진하고 있다. 여기에서는 해외 각국이 전자정부를 실현하기 위해 채택하고 있는 대표적 사례들을 통해 전자정보 기반 구조에 서비스지향 아키텍처를 적용한 예를 소개하였다.

1.1 덴마크

덴마크 전자정부의 주된 관심사는 공공부문 IT 시스템들에 상호운영성을 지원해 주어야 한다는데 있다. 이를 위해 시민들과 회사들 그리고 관료들이 같은 정보를 반복하여 제공할 필요가 없고 점검할 필요가 없게, 정부는 서로의 자료들을 사용할 수 있어야 하고 기존의 분산되어 있는 서비스나 어플리케이션을 전자정부를 통해 통합할 수 있어야 한다. 덴마크는 정부차원에서 전자정부를 지원하기 위해 주요 정보 아키텍처에 XML을 접목시켰다. 또한 서비스지향 아키텍처를 이용해 조화된 서비스 개발과 재사용으로 기업의 역할을 하고자 하였다.

1.2 이탈리아

이탈리아 정부는 비즈니스 커뮤니티를 만들기 위해 비즈니스 커뮤니티 서비스 기반구조(Business Community Service Infrastructure : BCSI)라 불리는 네트워크 서비스 아키텍처를 설계하는 프로젝트를 수행하였다. 이는 공통된 시장을 공유하고, 서로 다른 역할을 수행하는 각 경영자들에

게 필요한 기반구조를 구축하기 위해서 비즈니스 커뮤니티의 협력 프레임워크를 정의하는 것이다. 기존의 네트워크 서비스와 새롭게 개발될 서비스의 통합을 원활히 하기 위해서 서비스지향 아키텍처를 기반으로 XML과 웹 서비스 기술을 도입하여 BCSI를 구축하고자 하였다.

1.3 국내의 현황

우리나라는 그간 지속적으로 추진해 온 공공부문의 정보화 성과에 힘입어, 전자정부 준비지수 면에서 세계 13위를 차지하였으나, 2004년, 2005년 연속 5위를 차지하면서 글로벌한 전자정부 선진국으로서 입지를 굳히고 있다.

현재 우리나라는 전자정부의 발전단계 모형 중 2단계 '온라인화' 단계를 마치고 차기 전자정부는 3단계인 '통합'의 단계로 나아가고 있다. 우리나라를 포함한 선진국형 전자정부는 기관별 전산화 단계에서 이음새 없는 서비스 제공을 통한 업무혁신 단계로 진화하는데 전자정부 사업 예산을 투자하고 있으며, 많은 사업들이 부처 간의 협업과 연계에 초점을 맞추고 추진되고 있다.

2. 서비스지향 아키텍처 S/W 품질시험 동향

최근 서비스지향 아키텍처에 대한 관심이 급증하면서 이를 기반으로 한 소프트웨어 제품 개발이 활발해지는 추세에 맞춰 소프트웨어 품질평가 분야에서도 새로운 트렌드에 대응하기 위한 노력이 활발해지고 있다.

많은 소프트웨어 기업들이 비즈니스 요구사항과 연계된 솔루션을 구현하기 위한 방법으로 서비스지향 아키텍처 기법을 사용하고 있고, 관련 업계의 기술 개발에 따라 서비스지향 아키텍처 소프트웨어의 사용화가 확산되고 있다. 이에 따라 서비스지향 아키텍처 기법을 사용한 소프트웨어 제품이 급증하고, 향후 지속적인 성장이 예상되는 서비스지향 아키텍처 관련 SW를 테스트할 수 있는 평가모델 개발이 시급한 상황이다. 최근 비즈니스프로세스관리(BPM: Business Process Management), 전사적자원관리(ERP : Enterprise Resource Planning)를 비롯해 많은 소프트웨어 분야에서 서비스지향 아키텍처를 기반으로 컴포넌트 사용 재사용이 가능하도록 제품을 개발하는 추세가 빠르게 확산되고 있다.

3. 서비스지향 아키텍처 관련 시장 동향

3.1 서비스지향 아키텍처 소프트웨어 시장 동향

주목할 만한 국내시장의 현상은 단순히 서비스지향 아키텍처를 지원하는 제품의 수가 증가하는 것이 아니라, 서비스지향 아키텍처를 효율적으로 구현할 수 있는 새로운 제품들이 계속해서 출시되고 있다는 점이다. 이런 추세에 맞추어 관련 솔루션 공급업체의 제품군도 다양해지고 있다. 국내의 대표적인 서비스지향 아키텍처 플랫폼 공급업체 및 이들이 제공하는 주요 제품유형을 보면 다음과 같다.

- 한국 IBM, BEA시스템코리아에서 ESB(En- terprise Service Bus) 기반의 서비스지향 아키텍처 미들웨어 제공
- SOA코리아, 시벨시스템즈코리아에서 서비스지향 아키텍처를 지원하는 비즈니스 어플리케이션 제공
- 한국 CA, 한국 HP에서 서비스지향 아키텍처 인프라 관리 제품 제공
- 마이크로소프트, 오라클, 썬 등에서 어플리케이션 플랫폼 제공

이 외에도 국내업체로는 티맥스소프트에서 미들웨어 중심의 서비스지향 아키텍처 제품을 제공하고 있다.

국외에서는 중소규모에서 엔터프라이즈 규모의 시장뿐만 아니라 통신, 금융, 공공, 교육, 제조, 병원 등 대부분의 업종에서 서비스지향 아키텍처 적용을 하고 있다. 이에 따라 많은 수의 IT 벤더들의 서비스지향 아키텍처 시장 진출이 활발하다.

이렇게 시장이 활성화되는 현 시점에서 서비스지향 아키텍처 플랫폼의 주요 영역인 ISE (Integrated Service Environment : 비즈니스 프로세스 실행과 서비스지향 아키텍처 기반의 개발도구) 시장에서는 전통적으로 서비스지향 아키텍처 플랫폼 시장을 주도해 오던 IBM, 마이크로소프트, 오라클, SAP 등이 여전히 선도하고 있다.

3.2 서비스지향 아키텍처 및 플랫폼 시장 전망

시장에서는 서비스지향 아키텍처에 대해 매우 긍정적으로 바라보고 있으며 이에 대한 긍정적인 전망들을 내놓고 있다.

- 서비스지향 아키텍처는 새로 개발되는 주요 업무 어플리케이션 및 비즈니스 프로세스 설계에 50% 이상 사용되고 2010년경에는 80% 이상 사용될 것이다.
- 2010년경 2006년에 5% 미만으로 사용되던 레지스트리가 서비스지향 아키텍처 프로젝트에서 40% 이상 사용될 것이다.
- 2011년경 인프라 소프트웨어 80% 이상이 ESB를 채용할 것이다.

〈표 1〉에서 보면 어플리케이션 서버와 트랜잭션 처리 모니터는 마이너스 성장을 기록할 것으로 예상하지만, 통합슈트, 포털, APS는 5% 이상의 연평균 성장률을 기록할 것으로 예측된다.

표 1. 어플리케이션 통합과 미들웨어 부분 시장예측(백만달러 (Source:Garner Dataquest 2005.6)
Table 1. Market Forecast of Application Integration & Middleware Part

Subsegment	2007	2008	2009	CAGR (%)
Application Servers	1,079.0	1,035.8	984.0	-2.5
Integration Suites	1473.6	1,547.3	1,616.9	5.1
Portal Products	923.5	992.8	1,062.3	8.4
APS(Stand Alone)	417.3	450.7	487.7	6.5
Message-Oriented Middleware	533.7	544.4	549.8	3.0
Transaction Processing Monitors	1,329.0	1,262.5	1,136.3	-1.4
Object Request Brokers, Adapters and Others	1,226.3	1,263.1	1,288.3	3.1
Total	**6982.3**	**7,096.5**	**7,125.3**	**2.7**

III. 서비스지향 아키텍처 S/W의 특성

이 절에서는 서비스지향 아키텍처 소프트웨어의 특성을 분석하여 서비스지향 아키텍처 소프트웨어가 갖추어야 할 품질 요구사항을 확립하고자 한다.

1. 통합비용 절감

시스템 간의 통합은 일회성(one-off), 포인트-투-포인트(point-to-point) 접속 방식으로 수행된다. 이 솔루션은 한 시스템의 특정 데이터 세트를 다른 시스템으로 이동할 때 발생하는 문제를 해결한다. 이는 전용 API 와 재사용할 수 없는 특수 코드로 작성된 데이터를 사용한다. 예를 들어, CICS 데이터는 메인프레임 상에 COBOL 카피북(copybook)을 생성하고, FTP 를 사용하여 데이터베이스 호스트로 전송하고, 이를 쉼표로 구분된 파일로 변환한 다음 Oracle 임포트 툴을 사용하여 임포트함으로써 Oracle 메인프레임에서 Oracle 데이터베이스로 이동할 수 있다. 이는 가장 간단한 방법이며 여러 방법 중에서 이러한 특정 통합 문제를 해결하기 위한 최적의 설계 방법이다. 그러나 재사용이나 개방성 측면에서는 취약하다는 단점을 가지고 있다. 이 통합 포인트는 다른 시스템이 표준화하고 재사용할 수 없다.

일정한 수준의 복잡성을 안고 있을 뿐 아니라 사전에 전사적인 정보 통합을 요구하기 때문에 이는 결국 문제를 발생시키게 된다. 예를 들어, 기업이 고객 데이터에 액세스해야 하는 영업 포털을 구현하고 있다고 가정해 보자. 포털은 영업 담당자들에게 고객 정보를 제공하기 위해 CICS 애플리케이션에 액세스해야 한다. 위에서 설명한 통합 솔루션은 그다지 도움이 되지 않는다. 아마도 거의 처음부터 다시 솔루션을 구현해야 할 것이다. 통합해야 하는 N 개의 시스템이 있을 경우 연결이 필요한 총 수는 N/2*(N-1)이다. 이들 각 연결은 다른 연결에서 거의 재사용하지 않거나 아예 없는 포인트 솔루션이 될 것이다. 대신 각 시스템에 대한 표준 기반 서비스 인터페이스를 구축하고 여기에 모든 시스템이 연결되도록 한다면, 각 시스템에 이러한 통합 작업을 단 한 번만 수행하면 되기 때문에 시스템 수가 늘어나더라도 비용을 절감하게 된다.

서비스지향 아키텍처의 이러한 이점은 가장 분명하고 가장 쉽게 눈으로 확인할 수 있지만 서비스지향 아키텍처를 구현하는 가장 중요한 이유는 아니다. 실제로 강력히 결합된 모놀리식 엔터프라이즈 정보 시스템의 복잡성을 관리하는 문제가 구현 비용 문제보다 더욱 심각한 사례가 자주 발생하고 있다.

2. 투명성과 자율성

많은 수의 IT 시스템이 서로 다른 기술을 사용하는 얽힌 연결로 서로 묶여 있다면, 종속성 문제로 인해 시스템 일부를 변경하는 것이 매우 어렵다.

일반적으로 백로그가 길어지거나 비용이 증가하거나 시스템을 변경할 수 없는 등의 결과가 초래된다. 변경의 영향을 완벽하게 분석할 수 있다 하더라도 변경 작업에 소요되는 비용이 그 이점을 무의미하게 만들 수 있다.

독립 시스템은 투명성이나 유연성이 없는 하나의 모놀리식 블록이 되었다. 이를 해결하는 유일한 방법은 시스템을 상호 연결하는 방식을 변경하는 것이다. 완벽하게 정의된 계약을 사용하여 다른 시스템과 상호 작용하면서 독자적으로 운영되도록 시스템을 서비스에 매핑해야 한다. 이는 중앙 집중식 시스템 및 자율 시스템 측면 모두에서 이익을 실현할 수 있다.

전체 인프라를 이러한 방식으로 세분화하면 시스템 간의 종속성을 훨씬 쉽게 추적할 수 있다. 가입자 정보를 이용해 서비스 레지스트리에 표준 기반 서비스 인터페이스의 카탈로그를 만들어 전체 인프라의 종속성 관리를 수행할 수 있다. 또 다른 이점은 IT 그룹이 훨씬 자율적인 방식으로 다른 그룹에서 제공하는 서비스를 사용할 수 있다는 것이다.

3. 서비스 수준의 업계표준 준수

서비스지향 아키텍처의 또 다른 장점은 업계 표준의 준수이다. 이 장점은 두 가지 범주로 나눌 수 있다.

IT 인프라가 표준 기반 계약을 사용하여 상호 작용하는 독립적이고 자율적인 서비스의 집합체일 경우 다른 비즈니스와

의 작업을 통합하는 것이 훨씬 쉬워진다. 사실, 점차 많은 서비스가 보급되면서 내부 서비스를 더 저렴하게 제공하는 비즈니스 파트너의 서비스로 교체하거나 내부 서비스를 시장에서 경쟁력 있는 서비스로 전환할 때 이 IT 변환이 새로운 비즈니스 기회를 창출할 것이다.

또한 업계 표준은 다른 측면에서도 이점을 제공한다. 모든 유형의 서비스를 구축하기 위해서는 툴과 프레임워크가 필요하다. 대부분의 업종이 XML 과 웹 서비스를 기반으로 하는 서비스지향 아키텍처의 공통된 단일 접근 방식으로 통합되고 있기 때문에 서비스 네트워크를 구축하고 운영하는 데 필요한 툴과 프레임워크가 더 일반화되고 널리 사용할 수 있게 되었다. 따라서 제품 비용이 절감되고 사용할 수 있는 숙련자를 더 많이 확보할 수 있다.

4. 이기종 환경에서 적절한 동작

서비스지향 아키텍처는 대부분의 기업에서 발견되는 '모든 것을 갖고 있는' IT 환경을 통합하는 일을 더 쉽게 해 준다. 이것이 서비스지향 아키텍처가 제시하는 가장 큰 가치 가운데 하나다. 다시 말해 서비스지향 아키텍처는 이기종 환경에서 아주 잘 작동한다.

서비스지향 아키텍처를 통하면, 개발자들은 애플리케이션들을 연결하는 새로운 코드를 작성하기 위해 과도한 시간을 낭비할 필요가 없어진다. 대신 개발자들은 웹서비스 같은 표준 프로토콜을 사용할 수 있다. 그리고 서비스지향 아키텍처 코드의 상당 부분은 재사용이 가능하기 때문에 개발 비용도 줄어든다. 서비스지향 아키텍처는 CIO가 기존에 리거시에 투자했던 것(SAP, 시벨, 오라클 등)을 한데 묶어 더 잘 활용할 수 있게 해 준다.

5. 기존 포트폴리오의 활용

서비스지향 아키텍처가 좋은 점은 기존의 포트폴리오를 활용할 수 있게 해 준다는 것이다. CIO는 기존 시스템을 제거하고 새로운 시스템으로 대체할 필요가 없어진다. 기존 시스템의 기능들을 파악한 후 그것들을 활용함으로서, CIO는 위험을 최소화하면서 기존 IT 투자의 가치를 극대화할 수 있다.

또 서비스(예를 들면, 단순객체접근프로토콜(SOAP: simple object access protocol)과 웹서비스기술언어(WSDL: Web services description language) 등을 사용하는 서비스)를 구축하면, 내부 프로세스가 부드러워질 뿐만 아니라, 고객과 비즈니스 파트너들과는 회사의 방화벽을 넘어서 더 쉽게 정보를 공유할 수 있게 된다.

6. 자율성

자율성에 대한 서비스지향의 원칙은 서비스가 가능한 한 독립적이고 내부로직을 제어하는 기능을 자체적으로 포함하고 있어야 한다는 것이다. 이것은 메시지 수준의 독립성을 통해 구현되는데, 서비스를 오가는 메시지들은 충분히 지능적이며 이 메시지를 수신하는 서비스가 메시지를 처리하는 방식을 제어할 수 있다.

서비스지향 아키텍처는 솔루션 환경과 기업시스템 전반에 걸쳐 자율성의 개념을 적용하도록 촉진하여 이 원칙을 더 확장시킨다. 예를 들면, 자율적인 서비스로 구성된 애플리케이션은 그 자체로 서비스지향적인 통합 환경에서 스스로를 제어할 수 있는 서비스로 간주될 수 있다.

7. 공개표준 기반

웹서비스의 가장 주목할 만할 특징은 데이터의 교환이 공개표준에 따라 결정된다는 사실이다. 하나의 웹서비스가 다른 웹서비스로 메시지를 전송하는 방식은 전세계적으로 표준화되어 있고 이러한 표준화에 준하는 일련의 프로토콜을 통해 전송한다.

좀 더 나아가, 메시지 그 자체도 구성형식이나 적재된 메시지 내용을 나타내는 방식으로 모두 표준화되어 있다. SOAP, WSDL, XML 스키마의 사용으로 메시지는 메시지 내부에 완전히 그 내용을 포함할 수 있고 표준화 되어 있으므로 서비스는 커뮤니케이션하는데 있어 다른 서비스의 명세정보 그 이상을 필요로하지 않게 되었다. 공개 표준 메시지 모델을 사용하면 의존적인 정보를 공유하기 위한 별도 서비스 로직이 필요 없고 느슨한 결합 패러다임을 적용할 수 있다. 최신 서비스지향 아키텍처는 벤더 중립적인 공개 커뮤니케이션 프레임워크 〈그림 1〉을 전적으로 지원하며 그 영향을 강화하고 있다. 서비스지향 아키텍처는 구현에 있어서 개별 벤더 기술의 역할을 제한하고 서비스로 캡슐화된 애플리케이션 로직에 접근하도록 한다. 그래서 서비스 상호 간의 커뮤니케이션에는 항상 선택의 자유가 있다.

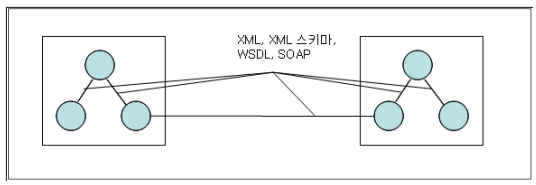

그림 1. 공개표준기술은 솔루션영역 안팎으로 사용
Fig. 1. Open Standard Technology is used within and without of solution scope

Ⅳ. 품질특성 체계와 모델의 구축

이 절에서는 서비스지향 아키텍처 소프트웨어의 다양한 특징과 요구사항을 바탕으로 서비스지향 아키텍처 소프트웨어의 신뢰성에 관한 특성 체계와 모델 구축에 관해 기술하였다.

신뢰성은 명세된 조건에서 사용될 때, 성능 수준을 유지할 수 있는 소프트웨어 제품의 능력으로서 서비스지향 아키텍처 소프트웨어에 대해 신뢰성에 관한 품질특성을 정리하면 〈표 2〉와 같다.

표 2. 신뢰성에 관한 품질특성
Table 2. Quality Characteristics about Reliability

품질특성	항목	내용
신뢰성	예외 처리	통합된 특정한 비즈니스 적업들을 보장하기 위해 트랜잭션 처리를 수행함에 있어 작업의 실패 시 예외 처리 로직의 실행을 보장해야 한다.
	결함 회피	서비스지향 아키텍처 소프트웨어를 일정 시간 운용하는 동안 결함 발생이 최소화되어야 한다.
	서비스 인터페이스 추상화	각 서비스를 추상화를 통해 블랙박스처럼 동작하게 함으로써 다른 서비스로 인한 결함으로부터 자유로울 수 있어야 한다.
	완전한 제어	서비스는 서로 독립적이어야 하며 서비스를 사용하는 시스템들과도 독립적으로 전개되고 수정되고 유지보수되어야 한다.
	다운 회피	서비스지향 아키텍처 소프트웨어를 운용하는 동안 발생되는 결함 중, 소프트웨어의 재시동이 필요한 결함의 발생은 최소화되어야 한다.
	자기 치유	서비스지향 아키텍처에 의해 개발된 소프트웨어가 사용하고 있는 특정 서비스가 임의의 원인에 의해 정상적인 기능을 할 수 없는 경우, 같은 기능의 서비스를 새로 찾아 바인딩 함으로써 자신의 내부에 있는 오류의 원인을 제거할 수 있어야 한다.
	…	…

1. 서비스지향 아키텍처 S/W 시험모델

시험모듈은 품질평가를 위한 평가 메트릭에 대해 ISO/IEC 14598-6의 형식에 의거하여 평가를 위한 제반 사항을 문서로서 정의한 것이다. 서비스지향 아키텍처 소프트웨어의 시험을 위한 모듈에 대해 기본적인 사항에 대해 기술한다. 측정유형은 메트릭의 계산식을 구성하는 측정값들이 가질 수 있는 값의 형태를 의미하며, 시험유형이란 메트릭의 결과값이 가질 수 있는 값의 형태를 말한다. 본 시험모듈에서 사용하는 측정 유형의 종류는 〈표 3〉과 같다.

표 3. 측정 유형의 종류
Table 3. The Kind of a Measurement Type

측정유형	측정단위	표시기호
측정유형 1	Y : 만족, N : 불만족 NA : 적용 불가능	(Y/N/NA)
측정유형 2	비율	Scale
측정유형 3	숫자	Number
측정유형 4	시간	Time

측정유형 1은 측정값이 Y/N의 형태로 판정되는 경우이며 NA(Not Applicable)는 측정 대상이 미비하거나 적용하기 곤란한 경우를 의미한다. 측정유형 2는 0부터 1사이의 비율 값으로 나타나는 경우이며, 측정유형 3은 개수를 측정하는 경우와 같이 정수값의 형태로 측정값이 나타나는 경우이다. 측정유형 4는 정해진 시간에 일어나는 사건(오류발생, 결함의 복구 등)에 대해 측정하는 메트릭의 경우에 필요한 시간값을 나타낸다. 본 시험모듈에서 사용하는 시험 유형의 종류는 〈표 4〉와 같다.

시험유형 1은 메트릭에 대한 결과가 Y/N의 형태인 경우이며 NA인 경우는 메트릭을 적용할 대상이 미비하여 적용할 수 없는 경우이다. 시험유형 2는 메트릭의 결과가 비율 형태의 0과 1사이의 값으로 나타나는 경우이다.

표 4. 시험 유형의 종류
Table 4. The Kind of a Test Type

시험유형	측정단위	표시기호
시험유형 1	Y : 만족, N : 불만족 NA : 적용 불가능	(Y/N/NA)
시험유형 2	비율	Scale

2. 시험모듈의 체계와 개발 내역

2.1 시험모듈의 체계

시험모듈은 품질시험에 관한 전반적인 사항을 정리하여 문서화한 것으로 시험의 개요, 기법, 메트릭에 대한 상세 내용, 적용 절차, 결과에 대한 해석 등을 포함하고 있으며 품질평가 프로세스에 관한 국제표준인 ISO/IEC 14598-6의 평가모듈 형식에 근거하여 작성하였다. 품질시험 모듈의 체계는 〈표 5〉와 같다.

표 5. 품질시험 모듈의 체계
Table 5. The System of Quality Test Module

구성 항목		내용
개요	메트릭의 개념	평가모듈의 기본 개념
	측정 목적	평가모듈의 측정 측정 목적
	메트릭 범주	메트릭이 속하는 소속을 기술
	용어 설명	메트릭의 개념과 목적 관련 용어 설명
적용 범위	적용대상	메트릭을 적용해야 할 문서나 소프트웨어 등의 대상을 기술
	필요자원	메트릭 적용에 필요한 도구/자원
	기법	적용할 수 있는 시험 기법
	적용 고려사항	평가모듈 적용시 고려해야할 관련 정보
참조문서		메트릭이 도출된 관련 문서
메트릭	측정 항목	측정할 데이터 항목
	측정 방법	메트릭을 구성하는 측정 항목에 대한 구체적인 측정 방법의 기술
	계산식	데이터 항목을 이용한 계산식 정의
적용절차		시험을 수행하는 구체적인 절차와 방법에 대한 기술
결과해석 및 보고	측정치의 매핑	메트릭 결과에 대한 판정으로 값으로 나타날 경우, 값의 범위
	측정 결과의 해석	측정 결과에 대한 해석 방법에 대해 지침 제시
	보고 사항	측정 결과에 대해 문서로서 보고해야 할 사항에 대한 명시

2.2 시험모듈 개발 내역

본 연구를 통해 서비스지향 아키텍처 소프트웨어의 신뢰성 품질 평가를 위해 신뢰성의 부특성인 성숙성, 결함허용성, 회복성, 준수성에 관한 11개의 메트릭을 개발하였다.

표 6. 시험모듈 내역
Table 6. Details of Test Module

품질특성	부특성	시험모듈 개발 내역	계
신뢰성	성숙성	〈예외처리〉 외 4개	14
	결함허용성	〈브레이크다운회피율〉 외 2개	
	회복성	〈자기치유〉 외 3개	
	준수성	〈신뢰성수준준수율〉 외 1개	
계	4		14

3. 품질검사표

품질검사표는 시험모듈에 정의된 메트릭을 기준으로 실제 품질 시험을 수행하는 과정에서 편리하게 활용할 수 있도록 필요한 핵심적인 사항들을 추출하여 정리한 표로서 메트릭명과 개념, 측정항목, 메트릭의 계산식, 결과의 영역, 결과값, 문제점 기술 부분 등으로 구성되어 있다. 이러한 품질검사표의 예를 〈표 7〉에 나타내었다. "서비스 인터페이스 추상화"란 서비스가 외부에 세부적인 내용을 숨기고 블랙박스처럼 동작할 수 있도록 인터페이스를 구성하는 것을 의미한다.

품질검사표에는 기본적으로 메트릭명과 메트릭이 측정하고자 하는 내용에 대한 문장이 포함되어 있다. 측정항목은 계산식을 통해 메트릭을 구성하는 요소로 1개 이상의 요소로 구성되며 항목 개요와 측정 방법에 대한 기술을 포함한다.

결과 영역은 계산식에 의해 산출되는 값이 나타날 수 있는 영역으로 메트릭들은 전체적으로 0과 1사이의 값으로 사상될 수 있도록 정의하였다.

표 7. 품질검사표의 예(서비스 인터페이스 추상화)
Table 7. An Example of Quality Test Table

메트릭명		서비스 인터페이스 추상화는 서비스가 외부에 세부적이 내용을 숨기고 블랙 박스처럼 동작할 수 있게 하는 원칙이다. 각 서비스를 추상화를 통해 블랙박스처럼 동작하게 함으로써 다른 서비스로 인한 결함으로부터 자유로울 수 있도록 구성되어 있는가?		
서비스 인터페이스 추상화				
측정 항목	A	시스템을 구성하는 전체 서비스 항목의 수		
		서비스 항목의 분류 기준이 요구됨		
	B	다른 서비스의 영향을 받지 않도록 추상화된 서비스의 수		
		추상화 여부를 검토할 수 있는 방법 필요		
계산식		서비스 인터페이스 추상화 = B / A		
결과영역		0≤서비스 인터페이스 추상화≤1	결과값	
문제점				

4. 점검표

점검표는 품질검사표를 이용하여 측정항목에 대한 측정을 수행하기 위해 작성된 테스트 케이스의 시험 목록이다. 예를 들어 〈표 8〉의 점검표는 기능점검표를 나타내고 있다. 기능점검표란 소프트웨어에서 제공하고 있는 기능 요소를 명시하고 이에 대해 필요한 기능설명이나 정보를 명확하게 제공하고 있는지 확인하기 위한 점검표라고 할 수 있다.

표 8. 점검표의 예(결함점검표)
Table 8. An Example of the Check Table

순번	결함발생 내역	결함정도				
		다운	고장		사소한 결함	
		치명 결함	중대 결함	경결 함	단순 결함	권고 결함
1						
각 결함수		0	0	0	0	0
결함회피율 =1-min(1, B/A) A=운용시간, B=총결함수		1.00				
결함발생평균시간=min(1, (B/A)/C) A=발견된 결함수, B=운용시간, C=결함발생평균시간의 한계값		1.00				
다운회피율=1-B/A A=발견된 결하수, B=다운회수		1.00				
고장회피율=1-B/A A=발견된 고장수, B=고장회수		1.00				
이용가능률=A/(A+B) A=운용시간, B=오류회복시간		1.00				
평균복구시간=1-min(1,(B/A)/C) A=복구회수, B=총복구시간, C=복구시간의 한계값		1.00				

5. 시험결과서

점검표의 테스트 케이스를 사용하여 품질검사표에 대한 측정이 수행되면 각 메트릭별 측정 결과가 산출될 수 있다. 이 결과들을 품질특성, 부특성에 대한 메트릭별로 〈표 9〉와 같은 시험결과서로 정리된다.

표 9. 시험결과서의 예
Table 9. An Example of the Test Result

제품설명서 및 사용자 문서			
품질특성	부특성	메트릭	측정값
신뢰성	성숙성	예외 처리	0.85
		결함회피율	0.93
		서비스인터페이스 추상화	0.85
		완전한 제어(Autonymy)	0.85
		모듈독립성	0.73
	결함허용성	브레이크다운회피율	0.78
		고장회피율	0.95
		오조작회피율	0.84
	복구성	자기치유(self-healing)	0.82
		데이터회복정보제공	0.89
		데이터회복률	0.93
		복구기능률	0.88
	준수성	신뢰성수준 정보제공	0.78
		신뢰성수준 준수율	0.83

6. 시험성적서

시험성적서에서는 〈표 10〉과 같이 문제점 기록서 및 시험결과서를 바탕으로 하여 발견된 문제점에 대해 품질특성의 관점에서 전반적인 문제점을 제시한다. 〈표 10〉에서는 품질특성 수준에서 개략적인 문제점을 제시하였으나 필요한 경우 품질부특성 수준 및 메트릭 수준에서 상세 문제점을 제시할 수도 있다.

표 10. 시험성적서의 예
Table 10. An Example of the Test Grade

시험 항목별 결과 내역	
시험 대상 : S/W에 대한 기능설명, 매뉴얼, 프로그램	
품질특성	결과
기능성	▪ 기능수행 방법이 명확치 않은 부분이 있음 ▪ 상호운용 가능기기가 부족
신뢰성	▪ 부분적으로 데이터의 신뢰도 저하 ▪ 결함 발생으로 시스템 정지 상황 발생
사용성	▪ S/W의 복잡성에 비해 매뉴얼의 설명이 미비함
효율성	▪ 자원 사용이 최적화되어 있지 못함
유지 보수성	▪ 자체적인 시스템 점검 기능 미흡 ▪ 결함 발견이나 복구가 용이하지 않음
이식성	▪ 설치 과정에 복잡한 단계로 구성되어 있으며 설치방법의 기술이 미비함

V. 품질측정과 평가 사례

본 평가 사례에서는 서비스지향 아키텍처 소프트웨어를 대상으로 신뢰성에 관한 평가를 수행하여 품질을 측정하고 평가한 사례를 통해 평가 방법에 대해 소개하고자 한다.

1. 품질평가 계획

품질특성 중 본 평가에 포함되는 항목과 제외되는 항목에 대해 〈표 11〉에 나타내었다.

신뢰성의 경우, 부특성인 성숙성에 대해 품질특성 필수 대상으로 분류하고 결함허용성, 복구성, 준수성에 대해 품질특성 선택 대상으로 분류하였다. 성숙성을 필수 대상으로 분류한 이유는 소프트웨어 신뢰성의 경우 결함으로 인한 문제를 피해가는 특성이나 문제발생 이후에 해결하는 특성보다는 문제 발생을 사전에 예방하는 것이 상대적으로 중요한 특성을 생각할 수 있기 때문이다.

표 11. 품질특성 범위
Table 11. The Range of Quality Characteristics

품질특성)	부특성 구분	품질측정 대상 여부
신뢰성	성숙성	○
	결함허용성	△
	복구성	△
	준수성	△

● : 품질측정 필수 대상, △ : 품질측정 선택 대상, X : 품질측정 대상 제외

2. 메트릭의 산정

본 평가사례에서 선정한 메트릭은 〈표 12〉와 같다. 선정은 평가 대상 소프트웨어의 특성을 고려하여 중요성이 낮거나 평가 대상이 준비되어 있지 않거나 적용하기에 적합하지 않은 것들은 제외하였다.

표 12. 평가명세 과정에서의 메트릭 선정
Table 12. Metric Selection of Evaluation Specification Process

품질특성	부특성	메트릭	필수여부
신뢰성	성숙성	예외 처리	●
		결함회피율	●
		서비스인터페이스 추상화	●
		완전한 제어(Autonymy)	●
		모듈독립성	●
	결함허용성	브레이크다운회피율	●
		고장회피율	●
		오조작회피율	●
	복구성	자기치유(self-healing)	■
		데이터회복정보제공	●
		데이터회복률	●
		복구가능률	●
	준수성	신뢰성수준 정보제공	●
		신뢰성수준 준수율	●

●: 품질측정 필수 대상, ■: 품질측정 선택 대상, X: 품질측정 범위 아님

3. 메트릭에 대한 측정 결과

평가는 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성에 대해 수행하였으며 〈표 13〉에서는 기능성에 대한 측정 결과를 나타내었다.

표 13. 신뢰성에 대한 측정 사례
Table 13. An Measurement Example about Reliability

특성	평가항목	측정값	비고
신뢰성	성숙성(예외처리)	0.93	
	성숙성(결함회피율)	0.95	
	성숙성(서비스인터페이스 추상화)	0.91	
	성숙성(완전한 제어)	0.96	
	성숙성(모듈독립성)	0.94	
	결함허용성(다운회피율)	1	
	결함허용성(고장회피율)	1	
	결함허용성(오조작회피율)	1	
	복구성(자기치유)	NA	평가제외
	복구성(데이터복구정보제공)	1	
	복구성(데이터회복률)	1	
	복구성(복구가능률)	1	
	준수성(신뢰성수준 정보제공)	1	
	준수성(신뢰성수준 준수율)	0.87	

측정 결과를 통해 각 메트릭에 대한 결과를 알 수 있고 상대적으로 취약한 특성을 파악할 수 있다. 기능구현완전성, 접근제어성 등이 우수한 결과를 나타내고 있으며 발견용이성, 독립적 단위 등이 상대적으로 낮은 값으로 나타났다.

메트릭의 측정결과에 대해 0.6미만은 매우미흡, 0.6이상-0.7미만은 미흡, 0.7이상-0.8미만은 보통, 0.8이상-0.9미만은 우수, 0.9이상은 매우우수 등으로 분류하여 레벨을 부여할 수 있으나 지속적인 평가의 축적을 통해 분석된 결과를 바탕으로 타당성 있는 기준에 따라 레벨을 부여할 필요가 있다.

4. 품질부특성과 품질특성의 결과 집계

〈표 14〉는 품질부특성에 대한 집계 결과를 나타낸 것이다. 품질부특성의 집계는 〈표 14〉의 메트릭 결과로부터 각 부특성에 대한 메트릭값의 합계를 평균한 것이다. 이때 Y/N로 측정되는 메트릭의 경우, Y를 1로, N을 0으로 기재하였으며 NA(Not Applicable)가 나오는 경우에는 평가 대상이 되는 문서가 준비되지 않았거나 미비에 기인하는 것일 수도 있고 평가 대상 소프트웨어 제품의 특성상 평가 대상에서 제외되는 것이 f수도 있으므로 0의 값을 부여하여 계산하거나 계산 대상에서 제외한다. 결과를 통해 각 품질특성별로 취약한 결과를 보이고 있는 부특성들을 확인할 수 있다.

표 14. 품질부특성에 대한 집계표
Table 14. The Totalization Table of Quality Subcharacteristics

특성	부특성	결과
신뢰성	성숙성	0.94
	결함허용성	1.00
	복구성	1.00
	준수성	0.94
평균	0.97	

5. 문제점의 제시

점을 분석하여 개발자에게 제시함으로써 품질을 향상시키는 것이 목적이므로 〈표 15〉와 같이 품질특성 수준에서 평가 대상 소프트웨어에 나타난 문제점을 예시하였다.

표 15. 문제점 예시의 일부
Table 15. A Part of Problem Presentation

시험결과 내역	
시험대상 : 제품설명서 및 사용자 문서	
신뢰성	- 소프트웨어의 신뢰성 수준에 관한 규정이나 수준에 관한 정보를 제공하고 있으나 일부 준수되지 못하는 규정들이 있음 - 트랜잭션 처리 수행시 작업의 실패에 대비한 예외처리 로직이 준비되어 있지 못한 경우가 일부 있음

Ⅵ. 결론

서비스지향 아키텍처는s IT의 아키텍처를 서비스 중심의 비즈니스 구조에 맞게 재구조화시키려는 움직임이라고 할 수 있다. 즉, 중복요소를 제거하여 공유할 수 있는 객체를 식별하여 재사용성을 높이고, 비즈니스 내의 서비스 기능 단위로 IT 재구조화하며 객체간의 연결을 사용자와 제공자 사이의 계약관계에 입각하여 최대한 느슨하게 연결함으로써 비즈니스 서비스화에 따른 변화에 최대한 빠르게 대응하여 개발할 수 있게 한다는 것이다.

이를 통해 새로운 비즈니스 업무 프로세스에 대해 기존에 구축된 서비스를 유연하게 연동해 새로운 업무처리 지원을 신속히 제공할 수 있다.

이러한 장점을 지닌 서비스지향 아키텍처 소프트웨어에 대해 특성을 수용한 평가 방법이나 체계가 미비한 것이 사실이다. 본 연구에서는 ISO/IEC 12119를 기반으로 하여 서비스지향 아키텍처 소프트웨어 평가를 위한 평가모델을 개발하고 평가 과정에서 활용할 수 있는 품질검사표를 개발하였다. 또한, 품질평가 방법을 구축하여 품질 수준 제고와 대외 경쟁력을 향상시키기 위한 연구를 수행하고 품질평가 사례를 제시하여 구체적인 평가절차에 대해 소개하였다.

본 연구를 통해 개발된 품질평가 방법을 활용하여 서비스지향 아키텍처 소프트웨어의 품질수준 향상과 시험체계 구축을 위한 기반 연구로 활용할 수 있을 것으로 기대한다.

향후 연구과제로 서비스지향 아키텍처 소프트웨어의 평가기술에 대한 실질적인 적용을 통해 평가기술의 객관성과 타당성을 제고하는 연구를 지속적으로 추진하고 평가 사례의 축적을 통해 평가방법에 대한 적합성 검증을 추진할 필요가 있다.

참고문헌

〔1〕 양해술, "*SOA 기반 소프트웨어 품질모델 개발*," 한국정보통신기술협회 최종보고서, 2007년 11월

〔2〕 권수갑, "*SOA 개념과 동향*," 전자부품연구원, 2005. 11.

〔3〕 토마스 얼, "*SOA: 서비스 지향 아키텍처(XML과 웹서비스 통합을 위한 필드 가이드)*" 성안당, 2007년

〔4〕 ISO/IEC 9126, "*Information Technology - Software Quality Characteristics and metrics*"

〔5〕 ISO/IEC 14598, "*Information Technology - Software product evaluation - Part 1-6*".

〔6〕 ISO/IEC 12119, "*Information Technology - Software Package - Quality requirement and testing*"

〔7〕 N. F. Schneidewind, "Methodology for Validating Software Metrics" IEEE Trans. on SE. Vol. 18,

No. 5, May 1992.
〔8〕 Moller, K. H. and Paulish, D. J., *"Software Metrics"* Chapmen & Hall(IEEE Press), 1993.
〔9〕 Wallmuller, E., *"Software Quality Assurance A practical approach"* Prentice Hall, 1994.
〔10〕 J. Boegh, S. De Panfilis, B. A. Kitchenham, A. Pasquini, "A Method for software Quality Planning, Control, and Evaluation," IEEE Software, Vol. 16, No. 2, Mar./Apr. 1999.
〔11〕 水野幸男, "ソフトウエアの綜合的品質管理," 日科技連出版, 1993.
〔12〕 더크 크래프지그, 칼 방케, 더크 슬라마, "*엔터프라이즈 SOA(서비스 지향 아키텍처 베스트 프랙티스)*"태극미디어, 2006년 11월

저 자 소 개

최 인 용(In-Yong Choi)

- 2005년 : 호서대학교 벤처전문대학원 정보경영학과 졸업(석사)
- 2005년-현재 : 서울벤처정보대학원 대학교컴퓨터응용기술학과 박사과정 수료
- 1975-84년:한진중공업 전산실
- 1984-99년:기아정보시스템(주) SI 사업부장
- 1999-현재:유니온정보시스템(주) 대표이사
- 2009년-현재:한국소프트웨어전문기업협회 회장

〈관심분야〉 : 소프트웨어공학, 소프트웨어 품질감리 및 평가, 컴포넌트 기반 개발방법론

양 해 술(Hae-Sool Yang)

- 1975년 : 홍익대학교 전기공학과 졸업(학사)
- 1978년 : 성균관대학교 정보처리학과 졸업(석사)
- 1991년 : 日本 오사카대학 정보공학과 S/W공학 전공(공학박사)
- 1975년~79년 : 육군중앙경리단 전자계산실 시스템분석장교
- 1980년~95년 : 강원대학교 전자계산학과 교수
- 1986년~87년 : 日本 오사카대학교 객원연구원
- 1995년~02년 : 한국소프트웨어품질연구소 소장
- 1999년~현재 : 호서대학교 벤처전문대학원 교수

〈관심분야〉 : S/W공학(특히, S/W 품질보증과 품질평가, 품질감리 및 컨설팅), S/W 프로젝트관리, 품질경영