품질속성 기반 설계방법을 적용한 소프트웨어 아키텍처 설계 사례연구

서 용 석[†] · 홍 석 붕[††] · 김 현 수[†††]

요 약

소프트웨어 개발에 있어서 구현에 앞서 아키텍처를 설계하는 일은 프로젝트의 성공을 위해 필수적이다. 본 논문은 한국원자력연구소 내에서 가동 중인 하나로 원자로의 방사선감시시스템 소프트웨어 개발과정에서 품질속성 기반 설계방법을 적용하여 소프트웨어 아키텍처를 설계한 사례를 보여준다. 품질속성 기반 설계방법은 Bass[1]가 제시한 속성 기반 설계방법을 변형한 것이다. 이는 먼저 시스템의 기능요건 및 품질요건을 아키텍처 드라이버(driver)로서 도출하고, 이를 만족하기 위한 전술(tactic)을 선택하고, 선택된 전술에 근거하여 아키텍처를 결정하고, 결정된 아키텍처를 구현 및 검증하는 과정으로 이루어진다. 하나로 원자로 방사선감시시스템의 개발요건으로부터 가용성, 유지보수성, 호환성과 같은 품질요건이 추출되었으며, hot-standby 서버 이중화와 약결합의 모듈화와 같은 전술이 선택되었으며, 이중화 서버에 다수의 클라이언트가 연결되는 클라이언트-서버 구조와 객체지향적 데이터 처리 구조가 방사선감시시스템을 위한 아키텍처로 결정되었다. 상용도구인 Adroit를 이용하여 아키텍처가 구현되었으며, 아키텍처 검증은 기능 중심의 시험을 통해 이루어졌다.

적은 예산과 단기간 내에 완수해야 하는 방사선감시시스템 개발에 품질속성 기반 설계방법을 적용함으로써, 보다 효율적으로 과제를 성공시킬 수 있었다. 방사선감시시스템 개발에서 설계된 아키텍처는 한국원자력연구소 내 다른 설비의 방사선감시시스템 개발에 재사용할 예정이다. 추가적으로 방사선감시시스템 아키텍처를 정량적으로 평가하는 작업이 필요하다.

키워드 : 아키텍처, 소프트웨어 아키텍처, 소프트웨어 품질, 품질속성 기반 설계, 원자력 방사선감시시스템

A Case Study of Software Architecture Design by Applying the Quality Attribute-Driven Design Method

Yong-Suk Suh[†] · Seok-Boong Hong[††] · Hyeon-Soo Kim[†††]

ABSTRACT

In a software development, the design of architecture prior to implementing the software is essential for the success. This paper presents a case that we successfully designed a software architecture of radiation monitoring system (RMS) for HANARO research reactor currently operating in KAERI by applying the quality attribute-driven design method which is modified from the attribute-driven design (ADD) introduced by Bass[1]. The quality attribute-driven design method consists of following procedures: eliciting functionality and quality requirements of system as architecture drivers, selecting tactics to satisfy the drivers, determining architectures based on the tactics, and implementing and validating the architectures. The availability, maintainability, and interchangeability were elicited as quality requirements, hot-standby dual servers and weak-coupled modulization were selected as tactics, and client-server structure and object-oriented data processing structure were determined as architectures for the RMS. The architecture was implemented using Adroit which is a commercial off-the-shelf software tool and was validated based on performing the function-oriented testing.

We found that the design method in this paper is an efficient method for a project which has constraints such as low budget and short period of development time. The architecture will be reused for the development of other RMS in KAERI. Further works are necessary to quantitatively evaluate the architecture.

Key Words : Architecture, Software Architecture, Software Quality, Attribute-Driven Design (ADD), Nuclear Radiation Monitoring System (RMS)

1. 서 론

현재 한국원자력연구소 내에는 연구용 원자로인 하나로가 가동 중에 있으며, 이 연구로에서 누출될 우려가 있는 방사선을 감시하고 통제하는 방사선감시시스템(RMS, Radiation Monitoring System)이 설치되어 있다. 이 RMS에는 하나로 내·외부 24곳에서 방사선 누출여부를 현장에서 감시하는 24개의 현장 감시기가 있었으며, 현장 감시기로부터 데이터를 수집하여 종합적으로 감시정보를 운전원에게 제공하는 4곳에 원격컴퓨터가 설치되어 있었다[2]. 이 RMS는 90년대 중반 미국 Victoreen회사가 SCO™ 운영체제와 Dataview™ 그래픽도구로 최초로 개발

† 정 회 원 : 한국원자력연구소 계측제어 · 인간공학연구부 책임연구원
†† 정 회 원 : 한국원자력연구소 계측제어 · 인간공학연구부 책임기술원
††† 정 회 원 : 충남대학교 전기정보통신공학부 부교수
논문접수 : 2006년 8월 3일, 심사완료 : 2006년 12월 19일

한 시스템이다. 한국원자력연구소는 1999년에 Y2k 문제해결을 위한 과제를 통해, 이 시스템을 국내의 Visiontech 회사에 Y2k 문제해결을 요청하였다. Visiontech는 이 시스템의 소프트웨어를 MS-SQL™과 Delphi™를 이용하여 개정하였으나, 이후 운영과정에서 이 시스템은 유지보수가 용이하지 못하다는 문제점을 안게 되었다. 2005년도에 하나로 구역 내에 핵연료시험시설이 추가되면서 6개의 현장감시기가 추가될 필요성과, 기존의 RMS의 유지보수성 문제에 대한 개선 요구사항이 대두되어 하나로 통합 RMS개선 과제가 11월에 시작되었다[3]. 이때, 주어진 요건으로는 개선될 RMS는 유지보수가 용이해야 하며, 기존의 RMS보다 우수한 기능을 제공해야 한다는 것이다. 예를 들어, 저장된 데이터의 편리한 검색기능, 자동 백업기능, 정확한 정보의 명료한 전달이 가능한 사용자 인터페이스 등이 우수한 기능의 예라고 할 수 있다. 문제는 우수하다는 기능을 개발자가 판단하여 제공하여야 한다는 것인데, 이러한 추상적인 의미의 요건들은 개발자에게 구현의 융통성이 주어지는 장점이 있지만, 얼마나 구현해야 사용자를 만족할 수 있겠는가에 대한 판단의 어려움도 있다. 이렇게 요건들이 초기에 확정되지 않은 채 시작되는 소프트웨어 개발 프로젝트는 무수하다. RMS 개발과제 중 소프트웨어 개발에 할당된 인원은 4명이며, 개발 기간은 6개월이다. 이 과제의 시작단계에서 기존 RMS를 역공학을 통하여 개선하는 방법과, 새로 개발하는 방법에 대한 결정이 필요하였다. 기존 RMS의 원시코드는 약 만줄 정도이고, 죽은 코드가 산재되어 있었으며, 개발 문서가 빈약하여 해석이 어려운 상태였다. 이러한 상황은 RMS를 확장하거나 유지보수에 많은 어려움을 초래하고 있었다. 따라서 기존 RMS를 역공학을 통해 개선하는 것보다 새로 개발하는 전략의 선택이 불가피하였다. 원자력관련 소프트웨어는 분석, 설계, 구현, 시험과정을 준수하여 개발하도록 규정되어 있다[4]. 추상적인 요건과 짧은 개발기간이 주어진 RMS 소프트웨어 개발을 위와 같은 개발과정을 준수하며 성공시킬 수 있는 전략이 필요하였다. 이를 위해 프로토타이핑 전략과 이와 병행하여 아키텍처를 조기에 결정하는 전략을 고려하였다. RMS 개발에서 프로토타이핑은 필수적이었다. 프로토타이핑은 사용자가 요건을 추상화 한 경우 이를 구체화 하는 최선의 방법 및 과정으로 이용된다[5]. 일반적으로 사용자가 자신이 원하는 것을 구체적으로 명시할 수 없는 경우, 동작하는 실체를 보여주면 그때서야 자신이 무엇을 원하였는지 응답하는 경향이 있다. 이러한 경향은 RMS 개발에서도 나타났다. 따라서 개선된 RMS 기능의 만족성을 판단하는 방법으로 사용자로 하여금 프로토타입을 통해 결정하는 방법이 불가피하였다. 프로토타입은 사용자 중심의 기능 시현에 중점을 두므로 시스템의 실체라고 볼 수는 없다. 시스템의 실체를 결정하는 방법으로 아키텍처 기반 설계방법을 선택하였다. 아키텍처 기반 설계방법은 조기에 시스템 설계를 결정하는 것에 목표를 둔다[1, 6]. 이러한 목표는 단기간의 개발기간이 주어진 RMS 개발에 적합하다.

본 논문의 2장 연구의 필요성에서는 국내외에서 발표되었던 아키텍처 기반 설계방법에 대한 연구내용을 알아보고, 이 설계방법을 RMS 개발에 적용할 필요성에 대해 언급한다. 3장에서는 Bass가 제시한 속성 기반 설계(ADD, Attribute-Driven Design)방법[1]을 RMS 개발에 부여된 제한사항에 맞게 현실적인 측면에서 수정한 품질속성 기반 설계방법을 제시한다. 본 논문의 4장에서는 품질속성 기반 설계방법을 적용하여 수행한 RMS 아키텍처 설계과정을 적용사례로서 제시한다. 5장에서는 설계된 RMS 아키텍처를 자체 평가한 결과에 대해 언급한다.

2. 연구의 필요성

아키텍처의 사전적인 의미는 "art, science, method, or style of building"이라고 merriam-webster에 정의되어 있고, Bass는 소프트웨어 아키텍처를 "소프트웨어 요소들(elements)과, 요소들의 가시적 특성들(visible properties)과, 그러한 요소들과 특성들의 관계(relationships)들로 이루어진 시스템의 구조(structure)"로 정의하고 있다[1]. 이와 같이 아키텍처는 "어떤 원칙에 입각하여 구축된 구조물"이라고 요약할 수 있다. 소프트웨어 공학에서 아키텍처에 대한 관심은 1968년 Dijkstra에 의해 소프트웨어 구조가 개발과 유지보수의 용이성에 영향을 준다는 지적과 함께 시작되었다[7]. 구현 테크닉 중심의 소프트웨어 개발에서 초래되는 프로젝트의 실패를 보완하는 방법 중의 하나로 아키텍처 기반 설계과정이 제시되고 있으며, 아키텍처의 재사용성을 도모함으로써 생산성이 향상될 수 있다는 점에서 아키텍처의 가치가 부각되고 있다. 아키텍처 기반설계과정은 가능한 조기에 사용자 요구사항을 만족하는 아키텍처 수립에 중점을 둔다. 아키텍트는 목적 시스템이 추상적으로 표현된 아키텍처를 설계하며, 아키텍처의 상세한 구현은 구현자에게 전가한다. 따라서 아키텍처는 추상적이면서도 구현방향이 뚜렷하게 표현되어야 한다. 아키텍처는 이해당사자(발주자, 사용자, 설계자, 구현자, 감리자 등)들에게 목적 시스템의 형상이 동일하게 이해될 수 있도록 간결하고 명료하게 표현된 모델이어야 한다. 아키텍처 모델의 구성요소로는 컴포넌트, 인터페이스, 제약사항 등이 있다. 아키텍처의 가치는 이해당사자가 목적 시스템에 대해 의견을 교환하는 수단으로 활용되었을 때 인정된다. 아키텍처의 성공은 이해당사자들이 아키텍처를 보고 목적 시스템을 얼마나 동일하게 이해하고 있는가에 달려있다. 아키텍처의 사실성은 개념과 구현을 일치시킬 수 있는 가교 역할의 수준이어야 한다. 이것은 아키텍처 표현수준을 추상화와 상세화 사이에서 결정해야 하는 기준이 된다.

아키텍처에 대한 연구는 CMU/SEI (카네기멜론대학/소프트웨어공학연구소, 미국방성이 지원하고 있음)에서 활발하게 이루어지고 있다. 이 연구소와 미해군이 협력하여 A-7E 항공기 조종실의 위치추적시스템[8], 훈련시스템[9] 등을 개발하는데 아키텍처 기반 설계방법을 적용하였다. 이러한 사례에서 강조하는 점은 도메인 분석이 제대로 되어야 올바른 아키텍처가 도출될 수 있다는 점이다. A-7E 사례에서는 지상의 상태를 실시간적으로 처리하여 조종사에게 알려주어야 하는 성능요건과, 항공기에 탑재될 수 있는 다양한 무기를 제어하기 위해 소프트웨어를 수정해야 하는 유지보수성요건을 만족하는 소프트웨어 아키텍처를 제시하였다[8]. 훈련시스템 사례에서는 다양한 모델을 반영할 수 있는 유지보수성요건 및 훈련자에게 친숙한 인

터페이스를 제공하는 사용성요건을 만족하는 소프트웨어 아키텍처를 구조적 모델링과 객체지향적 소프트웨어 설계를 중심으로 제시하였다[9].

국내에서는 컴포넌트 기반 소프트웨어 개발환경에서 사용될 수 있는 소프트웨어 아키텍처 뷰 모델을 제시한 연구가 있었으며[10], 아키텍처 기반 소프트웨어 개발에서 소프트웨어 아키텍처 변형을 지원하기 위한 방법이 연구되었다[11]. 우리나라 국방 정보화 청사진을 위한 아키텍처 개발 프레임워크 내에는 아키텍처 개발절차가 프로젝트 정의, 아키텍처 동기 분석, 현재 아키텍처 분석, 기술동향 분석, 목표 아키텍처 정의, 적용계획 수립, 운영 및 성장관리와 같이 7단계로 구분하여 제시되었다[12]. 아키텍처 접근성을 향상시키는 방법으로 아키텍처를 설계하는 과정 중에 품질속성 기반으로 아키텍처를 평가하여 아키텍처를 개선함으로써, 아키텍처를 완성한 후의 평가과정에서 발견되는 결함으로 인해 아키텍처를 재설계해야 하는 오류를 줄이는 방안이 제시되었다[13]. 또한, 서비스 로봇과 같은 시스템이 자신에게 주어진 임무수행을 위해 최적의 아키텍처를 주변 환경의 변화에 따라 스스로 학습에 근거하여 실시간적으로 변경할 수 있도록 하는 소프트웨어 프레임워크 개발연구도 제시되었다[14]. 이러한 연구내용을 종합해볼 때, 아키텍처에 대한 연구는 진화과정에 있으며, 국내의 경우 아키텍처 기반의 소프트웨어 설계사례가 많지 않다.

아키텍처 기반 설계방법으로 Bachmann이 제시한 ABD (Architecture Based Design) 방법[6]과 Bass가 제시한 ADD 방법[1]이 있다. 이 두 방법은 Bass가 지적한 "소프트웨어 품질속성과 아키텍처 결정은 상호 의존하는 관계가 있다[15]."는 관점을 같이 한다. 최종 제품에 부여된 품질요건은 아키텍처 결정에 영향을 준다. 또한, 아키텍처를 결정하기 위하여 여러 가지 기술적 요소들을 고려할 수 있는데, 이들은 아키텍처의 특정 품질속성 달성과 연관이 있다. 예를 들어, 캡슐화(encapsulation) 기술로 변경성(modifiability)이라는 속성을 달성할 수 있고, 복사본(replication)으로 신뢰성(reliability)을 달성할 수 있다는 것이다. ABD 방법은 설계 초기단계에서 상위수준의 설계내용을 결정하는 것이며, 아래와 같은 단계로 구성된다.

① 아키텍처 드라이버(driver)를 도출한다.
② 기능성(functionality)을 분해한다.
③ 아키텍처 스타일을 선택한다.
④ 기능성을 스타일에 할당한다.
⑤ 템플릿을 정제한다.
⑥ 기능성을 검증한다.
⑦ 동시성(concurrency) 뷰를 생산한다.
⑧ 배치(deployment) 뷰를 생산한다.
⑨ 품질 시나리오를 검증한다.

ABD 방법을 HIS(Home Information System)을 설계하는데 실험적으로 적용한 사례가 있다[16]. 이 사례는 실제로 개발된 것이 아니라 강의수준에서 제시된 사례이다. 반면에 ADD 방법의 배경을 설명하면 다음과 같다. 아키텍처는 최종 제품을 기대하는 여러 이해당사자의 관점이 합의되는 과정에서 결정된다. 아키텍처 결정에 영향을 주는 관점 또는 요소를 아키텍처 드라이버라 하며, 개발목표, 제한사항, 기능요건, 품질요건 등이 이에 속한다. 이 가운데 고객이 요구하는 품질요건이 아키텍처 결정에 큰 영향을 준다. 고성능을 중시하는 고객을 위한 아키텍처와 고가용성을 중시하는 아키텍처는 다를 수 있기 때문이다. 아키텍처 드라이버가 도출되고 나면, 이를 만족하기 위한 전술이 결정된다. 전술은 아키텍처 드라이버를 만족하는 아키텍처를 형성하기 위해 필요로 하는 설계요소를 결정하고 선택하는 일이다. 하나의 아키텍처를 결정하기 위해서는 많은 전술들이 필요할 수 있다. 이러한 전술들의 집합을 아키텍처 전략이라 한다. 또한, 모여진 전술들을 패키지화하면 아키텍처 패턴이 형성된다. 아키텍처 패턴은 아키텍처 스타일이라고도 한다. 아키텍처 스타일의 예로서 "pipe-and-filter"를 들 수 있다 [17], [18]. 결론적으로 Bass는 품질요건이 아키텍처를 결정하는데 있어서 주요 드라이버라고 간주하고, 이를 기반으로 아키텍처를 결정하는 아래와 같은 과정을 ADD 방법이라고 정의한 것이다[1].

① 분해(decomposition)할 모듈을 선정한다. 이때, 모듈에 대한 제한사항, 기능요건, 품질요건 등이 결정되어 있어야 한다.
② 모듈을 아래 과정을 통해 정제(refinement)한다.
   ⓐ 아키텍처 드라이버를 선정한다.
   ⓑ 전술을 기반으로 아키텍처 패턴을 선정한다.
   ⓒ 모듈을 인스턴스화 한다.
   ⓓ 자식(child) 모듈 간의 인터페이스를 정의한다.
   ⓔ 모듈을 검증하고 정제한다.
③ 분해가 계속 필요한 모듈은 ①, ②과정을 반복한다.

위와 같이 제시된 아키텍처 기반 설계방법이 특정한 제한사항이 주어진 소프트웨어 개발과제에 적절하게 적용될 수 있는지 의문을 가질 수 있다. 예를 들어, 적은 예산과 단기간의 개발기간이 주어진 RMS 개발에 위의 설계방법이 원활히 적용될 수 있는가에 대한 의문이 제기되었다. 그것은 RMS 개발과제에 아키텍처 기반 설계방법을 적용하는 과정을 통해 해답을 얻을 수 있다. 본 논문은 아키텍처 기반 설계방법이 RMS와 같은 특수한 개발과제에 어떻게 적용되었는가를 하나의 사례로서 제시하는 것이다.

3. 품질속성 기반 설계방법

소프트웨어 설계방법은 크게 구조적, 객체지향적, 컴포넌트 기반, 아키텍처 기반 설계방법으로 전개되어 왔다. 이러한 방법은 소프트웨어 품질과 생산성의 향상이라는 목표를 만족하기 위해 발전된 것이다. 소프트웨어 품질속성은 국제표준인 ISO/IEC 9126에 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성이라는 6가지의 대표적인 속성과 각 속성별로 다시 다수의 세부속성이 분류 및 정의되어 있다[19]. 이 표준은 소프트웨어에서 이와 같은 품질속성이 확인될 수 있다면 그 소프트웨어는 좋은 품질의 소프트웨어로 인정 또는 개선될 수 있다고 정의한다. 모든 소프트웨어가 위의 6가지 품질속성을 모두 만족할 수는 없다. 소프트웨어가 동작하는 환경요건에 따라 만족하여야 할 품질속성의 우선순위가 결정된다. RMS 개발요건 분석결과, 가용성, 유지보수성, 데이터 호환성이라는 품질속성이 우선

적으로 만족되어야 함을 알 수 있었다. 이와 같이 품질속성의 만족성을 우선적으로 고려하여야 하는 과제에서는 ABD 방법보다는 ADD 방법을 적용하는 것이 바람직하다. 일반적인 관점에서도 소프트웨어는 품질을 고려하여 개발되어야 하므로 ADD 방법을 채택하는 것은 타당하다. 그러나 ADD 방법은 모듈을 반복적으로 분해하고, 그 분해과정에서 품질속성을 만족하는 아키텍처 패턴을 찾아 인스턴스화 하는 상세수준의 설계과정이다. 이 방법은 시스템 기능을 소프트웨어 모듈로 분해하는 과정에서 아키텍처가 형성되는 원리에 기반을 둔 것이다. 아키텍처가 모듈의 상세함을 어느 수준까지 요구할 것인가에 따라 이 방법의 적용측면에서 변화는 불가피하다. 예를 들어, 상용 컴포넌트 중심의 소프트웨어 개발에서는 모듈 내부의 상세한 구조보다는 컴포넌트 인터페이스 정의가 아키텍처 형성에 핵심적인 요소이다. RMS 개발은 짧은 개발기간으로 인해 코딩의 최소화가 필요하고, 프로토타이핑과 병행할 수 있는 아키텍처를 설계 초기단계에서 신속히 결정하여야 하는 목표를 두고 있다. 이러한 목표를 만족하기 위해 RMS 개발에서는 ADD 방법을 아래와 같이 수정하여 적용하였다. 본 논문에서는 아래와 같은 설계과정을 품질속성 기반 설계방법이라 명명한다.

① 분석단계에서 아키텍처 드라이버를 도출한다.
② 설계단계에서 아래 과정을 통해 아키텍처를 결정한다.
  ⓐ 아키텍처 드라이버를 선정한다.
  ⓑ 아키텍처 전술을 선택한다.
③ 구현단계에서 아키텍처를 구현한다.
④ 시험단계에서 아키텍처가 드라이버를 만족하는지 검증한다.
⑤ 아키텍처의 정제가 필요하면 ②의 과정부터 반복한다.

위의 품질속성 기반 설계방법은 ADD 방법에 비해 간단하면서 소프트웨어 개발 프로세스가 단계적으로 구분된 방법이다. 즉, Bass의 기본 아이디어인 속성 기반 설계방법은 유지하되 RMS 제한사항을 고려하여 현실적인 방법으로 간략화 할 필요가 있었다. 이것은 RMS 아키텍처를 개발방향이 나타나는 수준의 정보표현이 아키텍트, 구현자, 사용자가 공통적으로 이해할 수 있는 상위수준의 정보로서 충분함을 원칙으로 설정했기 때문이다. 아키텍처를 표현하는 방법으로 Bass는 아키텍처를 Module 구조, Component-and-Connector 구조, Allocation 구조로 구분하여 표현하는 방법을 제시하였으며, Kruchten[20]은 Design 관점, Process 관점, Component 관점, Deployment 관점, Use Case 관점으로 구분하여 표현하는 방법을 제시하였다. 국내에서는 UML을 확장하여 일반적인 아키텍처 명세언어 개발 연구도 있었다[21]. 본 논문은 가급적 UML 표기법을 사용하되 최상위 수준의 아키텍처 표현을 위해 필요한 요소만 이용하여 RMS 아키텍처를 표현하는 것을 원칙으로 설정했다. 품질속성 기반 설계방법을 적용한 RMS 아키텍처 설계과정은 본 논문의 4장에서 설명한다.

4. 품질속성 기반 설계방법을 적용한 RMS 아키텍처 설계과정

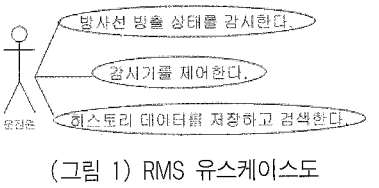
본 논문의 3장에서는 아키텍처를 설계하는 방법으로 Bass가 제시한 ADD 방법을 RMS 개발 제약조건에 부합하도록 변형한 품질속성 기반 설계방법을 제시하였다. 본 장에서는 품질속성 기반 설계방법 적용사례로써 RMS 소프트웨어 아키텍처 설계과정을 분석, 설계, 구현, 시험단계로 구분하여 제시하고자 한다.

4.1 아키텍처 드라이버 도출

분석단계에서는 사용자가 제시한 개발요건을 분석하여 기능요건과 품질요건을 아키텍처의 드라이버로서 도출하는 업무가 수행된다. RMS의 기능요건으로는 일반적인 모니터링 시스템이 제공해야 할 기능과 유사하며, 이에 대해 유스케이스도(use case diagram)를 (그림 1)과 같이 간략하게 나타내었다. (그림 1)에 나타내지 않았지만 세부적인 기능으로는 경보, 평균계산, 로그, 백업, 이벤트, 설정치 변경, 기기작동 명령 등의 기능들이 있다.

RMS 데이터 처리 및 저장 주기는 10초로 요구되었다. 이는 방사선이 환경에서 감지되는 순간이 즉각적이라기보다는 일정시간 동안의 데이터를 평균적으로 계산하여 방사선 농도를 감지하기 때문이다. RMS 성능문제는 심각하지 않다. 반면에 데이터 손실은 최소화 되어야 한다. 이는 방사선 감시와 관련된 법규에 의해 감시 데이터는 영구 보존되어 주기적으로 관련기관으로부터 감사를 받아야 한다. 따라서 RMS는 실시간 성능요건보다는 시스템 가용성요건이 더 중요함을 알 수 있다. 기존의 RMS는 잦은 소프트웨어적 오류가 발생하였고 오류의 원인을 찾아내는데 어려움이 있었다. 이것은 RMS의 가용성을 줄이는 원인이었다. 또한, 새로운 감시기의 추가와 같은 요구사항에 적절히 대응할 수 없는 상태였다. 따라서 오류의 원인이 용이하게 발견되고 수정되거나, 확장이 용이할 수 있는 소프트웨어 구조가 요구되었다. 향후 RMS의 데이터를 활용하고자 하는 시설에서 RMS 데이터를 요구할 경우, 이러한 설비와 용이하게 데이터를 호환할 수 있는 데이터 호환성 요구가 있었다. 이와 같은 요구사항을 RMS 아키텍처의 드라이버로서 도출되었으며, 이 가운데 중요성을 고려하여 우선순위가 높은 순서로 품질요건을 요약하면 아래와 같다.

① 가용성요건: 시스템 동작불능 상태의 최소화 및 히스토리 데이터의 손실이 최소화 되어야 한다.
② 유지보수성요건: 오류 발견 및 수정이 용이하여야 하며, 성능 개선 또는 확장이 용이하여야 한다.
③ 데이터 호환성요건: 다른 설비의 시스템과 데이터 호환이 가능하여야 한다.



(그림 1) RMS 유스케이스도

4.2 아키텍처 전술 선정

설계단계에서는 아키텍처 결정을 위한 전술이 선택된다. 즉, 분석단계에서 도출한 아키텍처 드라이버를 만족하는 전술이 선정되며, 이를 근거로 적절한 아키텍처를 결정하는 업무가 수행된다. 본 논문은 시스템, 하드웨어, 소프트웨어를 모두 포괄

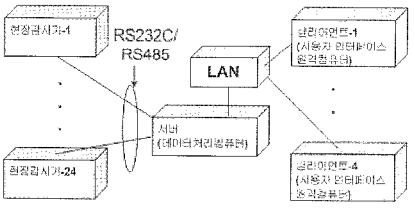
하는 상위수준의 아키텍처 설계에 관점을 둔다. 아키텍처 설계 과정의 하나인 전술 선택은 도메인 특성에 의해 제약을 받을 수 있다. 도메인 특성은 다양한 전술 가운데 특정 전술 선택의 기준으로 작용한다. RMS를 구성하는 컴퓨터들은 지역적으로 분산되어 있다. 이러한 시스템들이 통신망으로 연결되어야 하는 구조는 불가피하다. 이러한 도메인 특성은 RMS 전술 선택에 큰 영향을 준다. 대표적인 RMS 도메인 특성은 아래와 같다.

① 현장감시기와 원격컴퓨터들이 하나로 연구로 주변에 지역적으로 분산되어 있다.
② 현장감시기로부터 수신된 데이터는 하나로 연구로 안전관리실에 영구 보관된다.
③ 사용자는 필요시 현장감시기 또는 원격컴퓨터를 추가하길 원한다.
④ 향후 특정 외부 시스템과 하나로 연구로 RMS가 데이터를 교환할 수 있어야 한다.

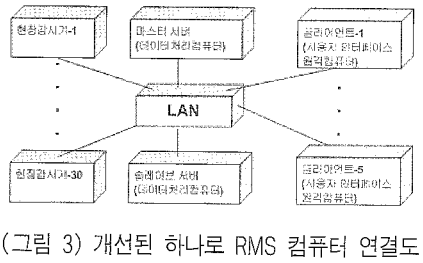
4.2.1 가용성 전술 선정

RMS 가용성요건은 시스템이 고장으로 인해 정지되는 시간을 최소화 하고, 데이터 손실의 최소화를 요구하는 요건이다. 일반적으로 다중화(redundancy)가 고가용성을 보장하는 대표적인 전술로 알려져 있다. 예를 들어, 90%의 신뢰도를 갖는 컴퓨터시스템을 이중화 하였을 경우 신뢰도는 99%로 높아진다. 기존의 하나로 RMS 시스템은 (그림 2)와 같이 단일 서버로 구성되어 있었기 때문에 서버의 이중화가 필요하였다.

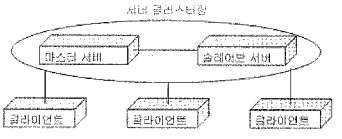
이에 따라 (그림 2)의 구조에서 현장감시기 6대와 원격컴퓨터 1대 추가 요구사항을 반영하고 서버를 이중화 하여 (그림 3)과 같이 RMS 구조를 개선하였다. 특히, (그림 2)의 RS232C/RS485 통신을 LAN 구조로 통일함으로써 확장성이 용이하도록 하였다. (그림 3)에서 서버의 이중화 전술로는 hot-standby 이중화 전술이 채택되었다. 이 전술을 위해서는 (그림 4)와 같은 서버 클러스터링이 필요하다. 클러스터링은 서버들을 그룹화 함으로써 외부에서는 논리적으로 하나의 서버가 동작하는 것처럼 보이게 하는 것이다. 정상 시에는 마스터가 모든 동작을



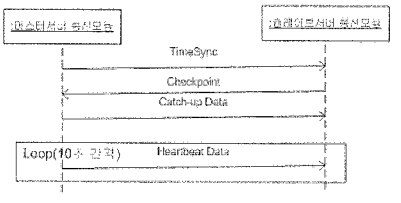
(그림 2) 기존의 하나로 RMS 컴퓨터 연결도



(그림 3) 개선된 하나로 RMS 컴퓨터 연결도



(그림 4) 서버 클러스터링 개념도



(그림 5) hot-standby 이중화 시퀀스도

관장하므로 클라이언트는 마스터와 통신하며, 슬레이브는 동작 중 대기(hot-standby)상태에 있게 된다. 동작 중 대기상태인 슬레이브는 마스터의 고장을 감지하면 즉시 자신이 마스터 역할을 수행한다.

이중화 전술을 채택한 후 고장감지, 고장복구, 고장방지 측면에서 세부전술(sub-tactic)을 고려할 필요가 있다. 이 가운데 고장방지는 고장을 예측하거나 진단하는 기술적 난이도와 예산부족으로 인해 RMS에서는 채택되지 않았으며, 고장감지와 고장복구 측면에서 채택된 세부전술은 다음과 같다. 고장감지는 서버의 고장을 어떻게 감지할 것인가를 고려하는 것이다. 고장감지에 해당되는 세부전술로 ping/echo와 heartbeat가 있으며, 이 가운데 RMS에서는 heartbeat가 채택되었다. ping/echo는 한 쪽에서 ping신호를 보낸 후 응답으로 echo신호를 일정시간 동안 기다린 후 응답이 없으면 상대의 고장을 감지하는 방식으로 보통 클라이언트가 서버의 건전성을 확인하는데 적합하다. RMS에는 ping/echo전술보다는 아래와 같은 이유로 heartbeat가 더 적합하다.

① heartbeat는 마스터가 주기적으로 heartbeat를 슬레이브에게 보내기만 한다. 슬레이브는 일정시간 동안 heartbeat가 오지 않으면 마스터의 고장을 감지한다. 이러한 방식은 구현이 간단하다는 장점이 있다.
② 마스터와 슬레이브의 데이터 일치성을 heartbeat를 이용하여 보장한다. 마스터와 슬레이브의 히스토리 데이터 즉 로그는 항상 일치되어야 한다. 이를 위해 마스터와 슬레이브의 시간은 초기 동작 시 동기화 된다. 마스터는 매 주기마다 현장 감시기로부터 취득한 데이터를 자신의 로그에 저장함과 동시에 heartbeat에 담아 슬레이브에게 전송한다. 슬레이브는 마스터로부터 수신된 데이터를 로그에 저장함으로써 마스터의 히스토리 데이터와 일치된다. 만일 마스터에 고장이 발생하여 heartbeat를 못 받게 되면 슬레이브는 자신을 마스터로 선언하여 RMS 동작을 이어간다.

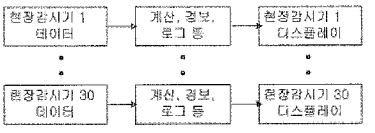
고장복구는 서버가 고장 났을 경우 어떠한 대책으로 시스템의 동작 연속성을 보장할 것인가를 고려하는 전술이다. 이에 대해서는 고장에 대비하는 전술과 고장 후 복구하는 전술로 구분

된다. 고장에 대비하는 전술로는 voting, spare, active 다중화, passive 다중화 등이 있다. RMS에서 voting은 요구되지 않으므로 배제하며, spare는 고장 발생 시 데이터 손실이 장기화될 우려가 있으므로 배제되었다. active 다중화는 여러 서버가 한 이벤트에 대해 동시에 응답하는 전술인데 현장감시기가 여러 서버와 동시에 통신해야 하는 오버헤드가 있어 배제되었다. passive 다중화는 하나의 서버만이 이벤트에 응답하고 나머지에게는 응답상황을 알려주는 전술이다. RMS에서는 고장에 대비하는 전술로서 passive 다중화 전술이 선택되었으며, 마스터만이 이벤트에 응답하는 hot-standby 이중화 구조가 결정되었다. 고장 후 복구를 위한 전술은 서버가 고장 난 후 고장이 수리되어 서버클러스터에 복귀될 때, 그 서버가 취해야 할 행위를 선택하는 것이며, shadow 동작, 상태 resynchronization, checkpoint/rollback 등이 있다. shadow 동작은 동작이 완벽한지 예비동작을 일정시간 동안 수행하여 점검하는 전술로서 RMS에서는 사전에 시험과정을 통해 점검하므로 이 전술을 채택하지 않았다. resynchronization은 서버들의 상태를 일치시키는 것이며, checkpoint/rollback은 고장기간 동안의 데이터를 복구하기 위해 표시점을 이용하여 복구하는 전술이다. RMS에서는 resynchronization과 checkpoint/rollback을 반영한 catch-up 전술이 채택되었다. catch-up 전술은 두 서버가 동작중인 경우에는 마스터가 슬레이브에게 변화된 데이터를 주기적으로 heartbeat로 전송하며, 고장 난 서버가 수리되어 서버클러스터에 연결되면 다음과 같이 동작되는 전술이다. 고장이 수리된 서버는 클러스터에 슬레이브로서 합류되며 이때, 마스터와 시간 및 상태가 resynchronization된다. 슬레이브는 고장기간 동안 저장하지 못한 시점을 checkpoint로써 서버에 보낸다. 마스터는 이 요청을 받은 즉시 자신의 로그에 저장된 히스토리 데이터를 슬레이브에게 전송한다. 이와 같이 고장복구 전술로서 채택한 catch-up 전술이 동작한다. 이상과 같이 가용성요건 만족을 위해 채택된 전술을 통해 hot-standby 이중화 서버를 위한 소프트웨어 아키텍처의 시퀀스도는 그림 5와 같다.

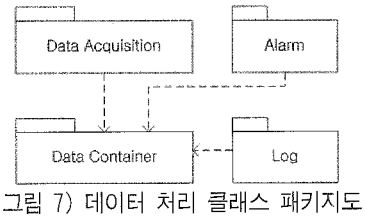
4.2.2 유지보수성 전술 선정

RMS 유지보수성요건은 변경의 용이성 및 변경으로 인한 부효과(side-effect) 발생의 최소화를 요구하는 요건이다. 유지보수성을 보장하는 대표적인 전술로, 높은 응집도와 낮은 결합도를 갖는 모듈화가 알려져 있다. 모듈간의 의존성을 최소화 하는 것이 유지보수성을 달성할 수 있는 대표적인 전술이다. 이 전술을 효과적으로 적용하기 위한 선행 작업으로 RMS 데이터 처리 특성 분석이 필요하다. 데이터 처리 분석과정을 통해 모듈화 방향을 결정할 수 있기 때문이다. RMS는 30곳에 설치된 현장감시기가 독자적으로 방사선 감시를 수행하고, 이러한 감시정보가 취합되어 원격에서 종합적으로 감시되어야 한다. (그림 6)과 같이 RMS 30개의 현장감시기로부터 발생하는 데이터는 상호 영향이 없다.

감시기마다 공통적으로 처리되어야 하는 부분 예를 들어, 평균값 계산, 경보처리, 로그 등은 자신의 내부에서 처리될 부분이지 다른 감시기로부터 어떠한 영향도 받지 않는다. 각 감시기는 처리된 결과를 독립적으로 운전원에게 사용자 인터페이스



(그림 6) RMS 데이터 처리 및 디스플레이 과정

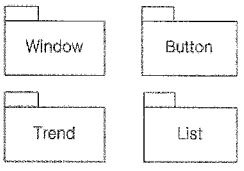


(그림 7) 데이터 처리 클래스 패키지도

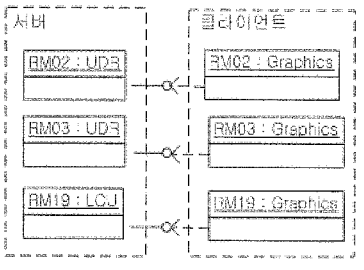
형식으로 전달하면 된다. 이와 같이 감시기 간의 계산이나 처리에 의존성이 없다면 높은 응집도를 갖는 모듈화가 가능하다. 높은 기능적 응집도를 얻게 되면 상대적으로 낮은 결합도를 얻을 수 있다.

RMS 데이터의 다른 특성으로는 30개의 현장감시기가 UDR (Universal Data Ratemeter)과 LCU(Local Control Unit)의 두 가지의 감시기 타입으로 분류된다는 것이다. 이것은 모든 현장감시기를 두 가지의 공통적인 타입으로 분류하여 처리하는 것이 효율적임을 보여준다. 이와 같은 분석결과로 낮은 결합도를 갖는 모듈화가 가능함을 알 수 있다. 모듈화 방식으로 구조적 방식과 객체지향적 방식을 고려할 수 있다. 일반적으로 객체지향적 방식이 정보은닉을 더 자연스럽게 구현할 수 있다. 구조적 방식도 정보은닉을 구현할 수 있겠으나 결합도가 높아질 수 있다는 단점이 있다. 따라서 RMS 모듈화는 객체지향적 방식이 전술로 채택되었다. 객체지향적 방식에서도 클래스간의 연관이 복잡하게 얽혀 있다면 유지보수성이 떨어진다. 단일 클래스에 단일 책임만을 부여하는 원칙을 적용한 RMS 데이터 처리를 위한 개략적 패키지도는 (그림 7)과 같다. Data Acquisition 패키지는 현장감시기로부터 데이터를 취득하는 역할을 수행하고, Data Container 패키지는 감시기와 관련된 모든 데이터 타입을 정의하며 가장 최신의 값을 저장하는 저장소(repository)역할을 수행하고, Alarm 패키지는 경보를 저장 및 처리하는 역할을 수행하고, Log 패키지는 히스토리 데이터를 저장하는 역할을 수행한다. 모든 패키지는 Data Container 패키지에 의존되므로 Data Container의 변경에 영향을 받을 수 있다. 이것은 변경을 집중화 및 변경에 의한 영향의 최소화를 위한 최선의 전술이다. 그 밖의 다른 패키지들은 서로 의존성 없이 독립적인 기능을 수행할 수 있도록 설계하였다.

유지보수가 많이 발생하는 부분이 사용자 인터페이스 화면의 변경이다. 대부분 사용자 인터페이스 화면은 "drag-and-drop"와 "look-and-feel" 방식으로 디자인된다. RMS에서도 이와 같은 객체지향적 그래픽 디자인 방식을 채택한다. 그래픽 객체들은 서로 독립적으로 동작하며, 이를 위한 템플릿(클래스라 할 수 있음)이 (그림 8)과 같이 정의되어 있다. 이 템플릿은 아이콘화 되어 있어서 아이콘을 캔버스에 끌어다 놓기를 하면 캔버스에 그래픽 객체가 생성된다. 객체는 다양한 속성을 갖고 있는데 예를 들어, 데이터표시, 색깔변화, 위치변경과 같은 애니메이션 효과를 위한 속성을 갖고 있다.



(그림 8) 그래픽 처리 클래스 패키지도



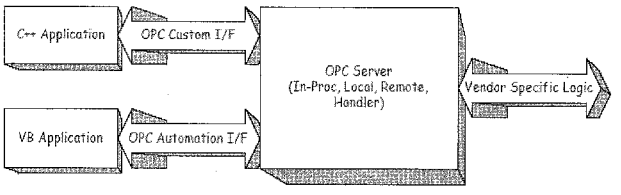
(그림 9) 서버와 클라이언트의 객체 통신도

서버와 클라이언트는 결합도를 최소화하기 위한 전술이 필요한데 이것은 역할을 분리하는 전술이 적절하다. 서버는 데이터를 처리 및 저장하는 생산자 역할을 부여하고, 클라이언트는 데이터를 운전원에게 효율적으로 제공하기 위해 데이터를 사용하는 소비자 역할을 부여한다. 클라이언트와 서버의 의존성은 데이터 사용 유무에만 한정되어 있으며, 나머지 동작에는 서로 영향을 주지 않는다. (그림 9)에서 보는바와 같이 서버에는 UDR 데이터 처리 타입과 LCU 타입의 클래스로부터 RM02, RM03, RM19 등과 같이 총 30개의 객체가 생성된다. 클라이언트에는 현장감시기의 데이터를 운전원에게 보여주기 위한 객체들이 생성되며, 이들은 서버의 객체로부터 데이터를 가져온다. 클라이언트와 서버에는 통신을 위한 인터페이스가 존재한다. 이는 서버에서 발생하는 변경이 인터페이스에만 영향을 미치고 클라이언트 객체의 동작에는 영향을 주지 않는다. 이 인터페이스는 Ethernet 통신을 포함하며 클라이언트 객체와 서버 객체간의 메시지 교환이 가능하도록 중계역할(intermediary)을 수행함으로써 유지보수성을 높일 수 있다.

대부분의 사용자 인터페이스 처리는 그 특성상 MVC (Model-View-Controller) 패턴을 벗어나지 않는다. RMS의 경우 model은 서버에 존재하는 데이터 처리 객체이며 view는 클라이언트에 존재하는 그래픽객체이며 controller 역할은 인터페이스가 담당한다. 이상과 같이 유지보수성요건 만족을 위해 채택된 전술을 통해 약결합의 객체지향적 아키텍처를 유도할 수 있었다.

4.2.3 데이터 호환성 전술 선정

RMS 데이터 호환성요건은 이 기종 시스템과 데이터 호환을 요구하는 요건이다. 이것은 일반적인 호환성 정의와 다소 차이가 있다. RMS 개발에서 호환성요건을 만족하기 위해 OPC 국제산업표준을 채택하였다. OPC™는 "OLE (Object Linking and Embedding) for Process Control"의 약어이다[22]. OPC는 Microsoft사의 OLE/COM (Component Object Model) 기술을 기초로 한다. 플랜트 제어 및 감시 시스템 분야에서 구현되는

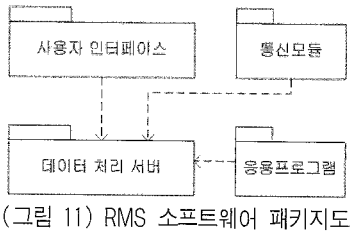


(그림 10) OPC 인터페이스 개념도

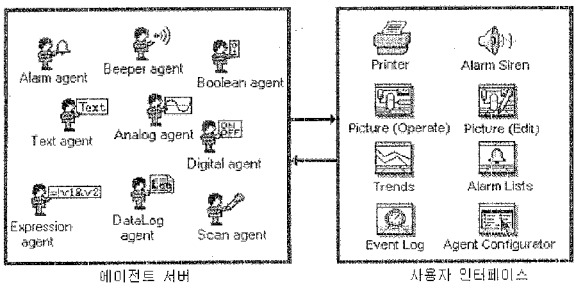
많은 SCADA (Supervisory Control And Data Acquisition) 또는 DCS (Distributed Control System) 응용프로그램들이 Microsoft사의 운영체제에서 동작되는 추세이며, 이들 프로그램들이 Basic, C, C++, Pascal 등 다양한 프로그래밍언어와 도구들로 개발됨에 따라 상호 데이터 호환에 문제가 발생하자, 산업체들이 지원하여 설립된 OPC Foundation에서 Microsoft사의 OLE COM / DCOM (Distributed Component Object Model)을 기반으로 데이터 호환을 위한 표준 인터페이스 사양서(specification)를 발행하게 되었다. (그림 10)에서 보는바와 같이 OPC Foundation에서 발행한 인터페이스 표준을 따르는 OPC 서버가 동작되는 환경에서 그 인터페이스 표준을 따르는 응용프로그램들은 OPC 표준 하에서 데이터교환이 가능하다. 범용 데이터베이스와의 연결은 Microsoft사의 ODBC (Open Database Connectivity) 또는 OLE DB 방식을 이용하여 데이터교환이 가능하도록 하였다.

4.3 아키텍처 구현

구현단계에서는 설계단계에서 결정된 아키텍처를 구현하는 업무가 수행된다. 좋은 아키텍처는 구현업무를 명확히 구분할 수 있고 업무 간의 인터페이스를 간결하게 한다. RMS 아키텍처 구현업무는 (그림 11)과 같이 명확히 구분되며, 각각의 업무는 인터페이스를 고려한 상태에서 병렬적으로 수행하였다. 구현의 중심은 데이터 처리 서버 구축에 있다. 데이터 처리 서버의 수정은 통신모듈, 응용프로그램, 사용자 인터페이스에 영향을 준다. 데이터 처리 서버에 전체적인 데이터 처리를 위한 기본 골격을 형성하고, 통신모듈, 응용프로그램, 사용자 인터페이스와의 인터페이스를 정의한다. 데이터 처리 서버 구현업무에서는 감시기 타입별 클래스 설계와 감시기 수만큼의 객체를 생성한다. 각 객체는 서로 의존하지 않고 독립적으로 동작한다. 통신모듈 구현업무에서는 서버와 현장 감시기의 데이터 교환, 서버 간의 heartbeat와 catch-up 데이터 전송을 구현한다. 응용프로그램은 복잡한 계산 알고리즘 수행을 위해 개발한다. 응용프로그램은 계산 결과를 주고받는 서버 객체와 인터페이스를 일치시킨 후 ActiveX 또는 DLL (Dynamic Link Library)를 이용하여 독립적으로 구현한다. 사용자 인터페이스는 많은 그래



(그림 11) RMS 소프트웨어 패키지도



(그림 12) Adroit에서 에이전트 서버와 사용자 인터페이스의 역할이 분리된 개념도

픽 객체를 설계하는 업무이며, 각 객체는 데이터 처리 서버에 의해 동작되는 해당 데이터 객체와 인터페이스를 갖는다.

RMS 개발 과제는 단 시간 내에 개발되어야 하는 제약사항 때문에 프로그래밍을 최소화 하는 전략이 불가피하다. 이를 위해 SCADA 시스템 구축 도구인 Adroit™[23]가 선택되었다. Adroit는 분산 실시간 시스템을 객체지향적으로 구축할 수 있는 상용도구이다. 이 도구는 본 논문에서 채택한 전술을 만족하는 것으로 평가되었다. 이 도구를 이용하여 RMS 데이터 처리 서버와 사용자 인터페이스를 구현하였다. Adroit는 에이전트(agent) 개념을 제공한다. 에이전트는 클래스와 유사하나 이보다 확장된 개념으로 자율적(autonomous)인 행동과 에이전트 간의 상호작용이 가능한 독립적인 개체(entity)로 정의된다. Adroit가 제공하는 에이전트로는 (그림 12)와 같이 아날로그, 디지털, 알람, 로그, 익스프레션, 커스텀, 스캔 등이 있다. 개발자는 Adroit가 제공하는 에이전트를 이용하거나 커스텀 에이전트를 이용하여 새로운 에이전트 설계가 가능한데, RMS에서는 (그림 9)의 UDR과 LCU 클래스를 커스텀 에이전트를 이용하여 설계하였다.

Adroit의 에이전트 서버는 에이전트로부터 생성된 객체를 동작시키고 외부로부터 객체 인터페이스를 연결해 준다. Adroit는 디스플레이 화면을 "look-and-feel" 방식으로 설계하도록 사용자 인터페이스 설계 기능을 제공한다. Adroit는 (그림 12)와 같이 다양한 그래픽 클래스를 제공하며, 개발자는 이러한 클래스를 "drag-and-drop" 기능을 이용하여 캔버스에 배치함으로써 그래픽 객체를 생성한다. Adroit는 사용자 인터페이스에 있는 그래픽 객체와 에이전트 서버에 있는 데이터 처리 객체의 연결을 위한 인터페이스 기능을 제공한다. 이러한 작업은 그래픽 객체 생성 시 데이터 처리 객체의 이름을 입력하는 방법으로 간단하게 처리할 수 있다.

4.4 아키텍처 검증

시험단계에서는 RMS 아키텍처 드라이버에 대한 아키텍처의 만족도를 검증하는 업무가 수행된다. 검증 방법으로는 기능시험이 채택되었다. 기능요건을 추상적으로 명시한 RMS 소프트웨어 요건명세서를 기반으로 기능중심의 알파 테스트를 수행하였으며, 사용자에게 프로토타입을 시연 및 동작하게 함으로써 사용성을 검증하였다. RMS 개발에서 시간적으로 전술의 구현 가능성을 판단하는 일과 기능요건을 확정하는 일이 병행될 필요가 있었는데 이를 위해 신속한 프로토타이핑은 필연적이었다. 가용성 검증은, 마스터 서버와 슬레이브 서버의 실시간 데이터 일치성 시험과, 마스터 서버에 고장을 발생시켜 슬레이브 서버에 의한 운전 연속성 시험과, 고장이 복구된 서버가 클러스터에 연결되었을 때, 마스터 서버의 데이터가 슬레이브 서버에 완전히 복구되는 시험 등을 통해 이루어졌으며, 이에 대한 개략적 시나리오는 <표 1>과 같다.

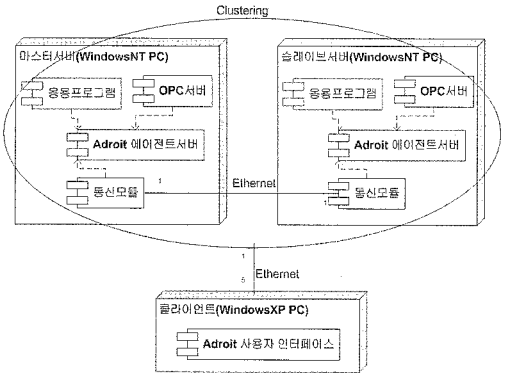
유지보수성에 대한 만족성은 임의의 고장 또는 변경 사례를 만들어 그 임무가 최소한 24시간 이내에 완료됨으로 검증하였다. 호환성에 대한 만족성은 이기종으로 개발한 외부 시스템과의 OPC 서버를 통해 데이터 교환 시험과 MS-SQL™ DB와의 호환성 시험을 통해 검증하였다. 확장성에 대한 만족성은 새로운 화면 추가, 현장감시기 추가, 원격컴퓨터 추가 등의 시험을 통해 검증하였다.

〈표 1〉 가용성 검증 시나리오

서버 동작 상태		확인 내용
서버-1	서버-2	
마스터	슬레이브	▪ 초기 동작 상태임 ▪ 시간동기가 일치함
동작중	동작중	▪ 모든 현장감시기/원격컴퓨터가 서버-1과 통신함 ▪ 서버-2의 로그가 서버-1과 일치함
고장	동작중	▪ 서버-2가 마스터로 전환됨 ▪ 모든 현장감시기/원격컴퓨터가 서버-2와 통신함
복귀	동작중	▪ 서버-1은 슬레이브가 됨 ▪ 모든 현장감시기/원격컴퓨터가 서버-2와 통신함 ▪ 서버-1의 로그가 서버-2와 일치함
동작중	고장	▪ 서버-1이 마스터로 전환됨 ▪ 모든 현장감시기/원격컴퓨터가 서버-1과 통신함
동작중	복귀	▪ 서버-2는 슬레이브가 됨 ▪ 모든 현장감시기/원격컴퓨터가 서버-1과 통신함 ▪ 서버-2의 로그가 서버-1과 일치함

5. RMS 아키텍처 평가

시스템 개발의 최종 목표는 사용자가 요구하는 품질을 만족하는 것과 생산물의 재사용을 통해 생산성 향상을 추구하는 것이다. 특정 품질요건에 한정된 특정 아키텍처가 정해져 있는 것은 아니다. 동일한 품질요건에 대해 아키텍트마다 다양한 아키텍처를 제시할 수 있다. 이것은 아키텍처 설계 시, 다양한 전술 적용으로 다양한 아키텍처가 결정될 수 있음을 의미한다. 다양한 전술 가운데 특정 전술을 선택하는 tradeoff는 정책, 예산, 기술의 난이도, 사업성, 품질요건 등에 의해 좌우된다. 이러한 기준들의 우선순위 또는 상관관계에 의해 특정 전술이 선택된다. 이와 같이 특정 전술을 최종적으로 선택하는 일은 간단하지 않다. 또한, 특정 전술이 아키텍처에 어떠한 영향을 미칠 것인가를 정량적으로 평가하거나 예측하는 기술은 발전 중에 있다. 아키텍처의 정량적 평가방법으로 ATAM (Architecture Tradeoff Analysis Method) 및 CBAM (Cost Benefit Analysis Method)이 제시되어 있다[1]. 이러한 방법들이 분석결과를 정량적으로 제시하려는 노력은 보이지만, 분석항목 선정이나 항



(그림 13) RMS 소프트웨어 아키텍처

〈표 2〉 RMS 소프트웨어 아키텍처 평가결과

가용성 품질속성 평가결과	
평가항목	평가결과
기존의 단일 서버 가용성을 향상하였는가?	서버를 hot-standby 이중화하여 만족함
기존의 직렬 통신 가용성을 향상하였는가?	Ethernet 통신으로 개선함
기존의 데이터 저장 가용성을 향상하였는가?	디스크를 RAID화하여 만족함
고장감지 방법을 적용하였는가?	Heartbeat 방법으로 만족함
고장복구 방법을 적용하였는가?	Catch-up 방법으로 만족함
고장방지 방법을 적용하였는가?	예산상의 문제로 적용하지 못함
OS의 가용성을 확인하였는가?	사실상 산업표준인MS-Windows를 사용함
상용 소프트웨어 도구의 가용성을 확인하였는가?	Adroit 도구를 무정지 동작을 통해 확인함
사용자의 데이터 접근 가용성을 확인하였는가?	사용자가 시험하여 만족함
유지보수성 품질속성 평가결과	
평가항목	평가결과
현장감시기의 추가/삭제가 용이하였는가?	비 개발자가 유지보수매뉴얼에 따라 수행함을 만족함
클라이언트의 추가/삭제가 용이하였는가?	Adroit의 클러스터링 셋업으로 가능함
디스플레이 정보의 변경이 용이하였는가?	Adroit의 사용자 인터페이스 기능으로 만족함
모듈의 약결합성을 확인하였는가?	객체를 템플릿 방식으로 추가/삭제가 가능함으로 만족함
응용 프로그램의 추가/삭제가 용이하였는가?	ActiveX 또는 DLL 방식으로 만족함
데이터 호환성 품질속성 평가결과	
평가항목	평가결과
이기종 시스템과 데이터 호환이 가능하였는가?	OPC 인터페이스를 통해 가능하므로 만족함
MS-SQL DB와 데이터 호환이 가능하였는가?	ODBC를 통해 가능하므로 만족함

목별로 부여하는 값 또는 가중치들은 다분히 주관적인 경험이나 정성적 판단에 근거하므로 엄밀한 의미에서의 정량적 분석 방법이라 볼 수 없다. 본 논문을 통해 도출한 RMS 소프트웨어 아키텍처는 (그림 13)과 같다. 이 아키텍처에 대한 3자(third-party) 평가 등의 객관적 평가는 수행되지 않았다. 다만, 아키텍처 평가방법으로는 주어진 예산과 기간 내에서 결정한 아키텍처를 사용자와 비공식적으로 약 6회, 공식적으로 3회 회의를 통해 설명하고 동의를 구하는 절차가 수행되었으며, 공식적인 3회 회의에는 프로토타입을 제공하여 사용자가 평가하는 절차가 포함되었다. 이러한 절차를 통해 도출된 RMS 소프트웨어 아키텍처의 평가결과를 <표 2>를 통해 요약하였다. <표 2>의 평가이외에 사용성 품질속성을 사용자의 편리성을 중심으로 평가하였으며, 성능의 만족성을 평가하였으며, 보안성을 평가하였다.

아키텍처 기반 설계방법의 장점으로는 재사용이 가능한 인자를 규명하고 획득하는 과정을 통해 소프트웨어 생산성을 향상시킬 수 있다는 점이다. 특히, 제품계열을 구축하기 위해서는 재사용 인자의 규명은 필수적이다. Bass는 재사용 가능한 인자의 범위를 요건서, 아키텍처, 설계요소, 모듈, 시험, 과제계획서, 도구, 개발자, 프로토타입, 결함제거 사례 등까지 광범위하게 설정하고 있다. 그러나 RMS 개발에서는 재사용이 가능한 공통 인자가 아래와 같이 도출되었다.

① LAN 기반 분산 시스템 아키텍처
② hot-standby 이중화 서버 아키텍처
③ 클라이언트-서버 아키텍처

위와 같은 재사용 가능한 인자는 미래의 RMS 개발에 있어서 참조(reference) 아키텍처가 될 수 있으며, 높은 가용성과 유지보수성을 요구하는 다른 RMS 개발에 사용될 것이다. 클라이언트-서버 아키텍처에서 채택된 객체지향 설계 및 처리 구조는 재사용 가능한 인자이다. 그러나 구현에 사용된 Adroit 도구, 통신모듈, 응용프로그램, 사용자 인터페이스는 가변인자이다. RMS 개발과제를 통해 축적된 엔지니어링 산출물의 프레임워크도 재사용이 가능한 인자이다.

6. 결 론

Booch는 "객체지향방법론을 적용하여 성공한 프로젝트에서 발견된 공통점은 반복적이고 점진적인 개발방법과 강인한 아키텍처 수립을 도입하였다는 점이다"고 언급하였다[7]. 반면에 아키텍처 또는 패턴 중심의 개발에 치우치면 소프트웨어 개발 공정들이 생략될 우려도 있다. 본 논문은 소프트웨어 공학적 개발 프로세스를 준수하면서 단시간 내에 개발을 완료해야 하는 RMS 개발과제에 Bass가 제시한 속성 기반 설계방법을 변형한 품질속성 기반 설계방법을 적용한 아키텍처 기반 설계방법이 성공적으로 적용된 사례를 제시하였다. 이로써 아키텍처 기반 설계방법이 적절히 적용된 사례가 제시되었다. 본 논문이 제시한 품질속성 기반 설계방법은 시스템의 품질요건이 아키텍처 드라이버로서 개발초기에 먼저 분석되고, 이를 우선순위화 하여 우선순위에 따라 적절한 전술을 선택하는 과정을 밟음으로써 아키텍처가 자연적으로 결정되는 아이디어로부터 비롯된 것이다. 이것은 소프트웨어 개발목표를 수립한 후 각각의 목표를 달성하기 위한 전술을 수립하여 아키텍처를 결정하는 과정이 소프트웨어를 구현에만 집중하였을 때 발생할 수 있는 과제의 실패 위험을 방지하는 효과가 있다. 국내에서는 아키텍처 기반 설계사례가 많이 제시되고 있지 않으나 본 논문을 통해 제시된 RMS 개발이 아키텍처 기반 설계방법을 따른 성공사례로써 소프트웨어 공학적인 의미가 있다. 특히, 본 논문이 제시한 전술 선정과정은 다른 응용분야에서도 참조할 수 있는 내용이다.

RMS 개발사례를 통해 아래와 같은 사항이 발견되었다.

① 사용자는 문서나 도면으로 시스템을 이해하기보다는 시스템의 동작을 보고 이해하길 원한다. 따라서 개발과정에서 조기에 시스템의 동작이나 형상을 사용자에게 보여주는 것은 과제의 성공을 위해 필수적이다.

② 프로토타이핑과 전술 선택과정은 상호보완적이다. 따라서 전술 선택의 확신은 프로토타이핑을 통해 얻을 수 있었다.

③ RMS의 경우, 품질요건 중에 아키텍처 결정에 영향을 주는 주요 드라이버로는 가용성요건과 유지보수성요건이었다.

④ Bass의 설계방법을 변형한 본 논문의 설계방법이 아키텍처를 신속하게 결정, 전달, 이해할 수 있는 방법임을 확인하였다.

⑤ 아키텍처의 추상화 또는 상세수준은 도메인 특성, 개발 제한사항 및 전략, 이해당사자의 요구에 따라 가변적인 특성이 있다.

단기간 내에 개발을 완수해야 하는 방사선감시시스템을 아키텍처 중심의 설계방법을 적용하여 개발함으로써 보다 효율적으로 과제를 성공시킬 수 있었으며, 이 아키텍처를 한국원자력연구소 내 다른 설비의 방사선감시시스템 개발에 재사용할 예정이다. 본 논문에서 제시한 아키텍처를 정량적으로 분석하여 아키텍처의 타당성을 검증하는 연구가 추가적으로 필요할 것이다.

참 고 문 헌

[1] L. Bass, P. Clements, R. Kazman, "Software Architecture in Practice", 2nd Ed., Addison Wesley, 2003.

[2] 김정택 외, "Y2k 문제 해결을 위한 하나로 및 부속설비의 방사선감시계통 전산설비 개선", KAERI/RR-2059/99, 한국원자력연구소, 2000.

[3] 안성호, "Technical Specification: FTL/RX RMS Integration System", HAN-FL-678-DT-K001, 한국원자력연구소, 2005.

[4] KINS-G-001, "경수로형 원전 안전심사지침", 개정 2, 한국원자력안전기술원, 1999.

[5] 윤청, "패러다임 전환을 통한 소프트웨어 공학", 생능출판사, pp. 82-83, 2000.

[6] F. Bachmann, et al., "The Architecture Based Design Method", CMU/SEI-2000-TR-001, CMU, 2000.

[7] P. Clements, L. Northrop, "Software Architecture: An Executive Overview", CMU/SEI-96-TR-003, CMU, 1996.

[8] K. Britton, D. Parnas, "A-7E Software Module Guide." Naval Res. Lab., Washington DC, NRL Memorandum Report 4702, 1981.

[9] G. Chastek, L. Brownsword, "A Case Study in Structural Modeling", CMU/SEI-96-TR-035, CMU, 1996.

[10] 박준석, 문미경, 염근혁, "컴포넌트 기반 소프트웨어 개발을 지원하는 소프트웨어 아키텍처 뷰 모델", 정보과학회논문지, 제30권 제6호, pp.515-528, 2003.

[11] 최희석, 염근혁, "아키텍처 기반 소프트웨어 개발에서 소프트웨어 아키텍처 변형을 지원하기 위한 방법", 정보과학회논문지, 제32권 제1호, pp.10-21, 2005.

[12] 최남용, 진종현, 송영재, "국방아키텍처프레임워크의 개발", 한국정보처리학회논문지 D, 제11-D권 제2호, pp.407-414, 2004.

[13] 고현희, 궁상환, 박재년, "소프트웨어 아키텍처 설계 단계에서 아키텍처 접근법 선정을 위한 평가 방법", 한국정보처리학회논문지 D, 제12-D권 제4호, pp.617-626, 2005.

[14] 박수용 외, "서비스 로봇을 위한 Self-Managed 소프트웨어 프레임워크 개발", 정보과학회지 제24권 제3호, pp.35-42, 2006.

[15] M. Klein, F. Bachmann, "Quality Attribute Design Primitives", CMU/SEI-2000-TN-017, CMU, 2000.

[16] F. Bachmann, L. Bass, M. Klein, "An Application of the Architecture-Based Design Method to the Electronic House", CMU/SEI-2000-SR-009, CMU, 2000.

[17] J. Georgas, E. Dashofy, R. Taylor, "Architecture-Centric Development: A Different Approach to Software Engineering", www.acm.org/crossroads/xrds12-4/arqcentric.html

[18] R. Monroe, et al., "Architectural Styles, Design Patterns, and Objects", IEEE Software, pp. 43-52, JANUARY 1997.

[19] ISO/IEC 9126-1, "Software engineering - Product quality - Part 1: Quality model", 2004.

[20] R. Kruchten, "The 4+1 View Model of Architecture", IEEE Software 12(6), 1995.

[21] 노성환 외, "UML 2.0 기반의 Generic ADL 정의", 정보과학회논문지, 제33권 제2호, pp.167-185, 2006.

[22] "OPC Overview", Version 1.0, OPC Foundation, 1998. (www.opcfoundation.org)

[23] www.adroit.co.za

서 용 석

e-mail : yssuh@kaeri.re.kr
1987년 광운대학교 전자계산학과(학사)
1996년 충남대학교 컴퓨터공학과(공학석사)
2006년 충남대학교 컴퓨터공학과(공학박사과정)
1987년~현재 한국원자력연구소 계측제어·인간공학연구부 책임연구원
관심분야 : 소프트웨어공학(소프트웨어프로세스, 소프트웨어 확인검증, 소프트웨어 아키텍처) 등

홍 석 붕

e-mail : boong@kaeri.re.kr
1979년 성균관대학교 전자공학과(학사)
1982년 성균관대학교 전자공학과(공학석사)
2002년 성균관대학교 전자공학과(공학박사)
1987년~현재 한국원자력연구소 계측제어·인간공학연구부 책임기술원
관심분야 : 지능시스템, 원자력계측기기, 잡음제어 등

김 현 수

e-mail : hskim401@cnu.ac.kr
1988년 서울대학교 계산통계학과(학사)
1991년 한국과학기술원 전산학과(공학석사)
1995년 한국과학기술원 전산학과(공학박사)
1995년~1995년 한국전자통신연구원 Post Doc.
1996년~2001년 금오공과대학교 조교수
1999년~2000년 Colorado State University 방문교수
2001년~현재 충남대학교 전기정보통신공학부 부교수
관심분야 : 소프트웨어공학(소프트웨어 테스팅, 소프트웨어 재공학, 소프트웨어 아키텍처), 컴포넌트 기반 소프트웨어 공학(컴포넌트 모델링, 컴포넌트 마이닝), 분산 컴퓨팅(J2EE/EJB, .NET, Web Services, Service Oriented Architecture) 등