한국산학기술학회논문지
Vol. 11, No. 12 pp. 5072-5082, 2010

인터페이스와 뷰 분석을 이용한 소프트웨어 아키텍처 설계방법

궁상환[1*]
[1]백석대학교 정보통신학부

Software Architecture Design based on Interface and View Analysis

Kung, Sang Hwan[1*]
[1]**Division of Information and telecommunication, Baekseok University**

요 약 본 논문은 소프트웨어 아키텍처의 설계를 위한 방법론을 제안한다. 논문의 제안하는 아이디어는 시스템에 존재하는 액터(actor)나 모듈, 통신요소 간의 인터페이스를 식별하면, 이 인터페이스에 연결된 모듈을 단계적으로 탐색해 나가는 방식을 활용한다. 그리고 설계범위의 인터페이스와 모듈을 점차 확장해 나감으로써 아키텍처의 설계가 완료될 때까지 설계를 점진적으로 진행해 간다. 이 방법은 기본적인 아키텍처 설계의 접근을 인터페이스로부터 출발하여, 이 인터페이스의 입력을 기다리는 모듈, 그리고 다시 이 모듈로부터 출력 인터페이스를 확인하는 방법을 통해, 설계의 원인과 결과가 서로 연관되게 함으로써 자연스러운 설계를 유도한다고 할 수 있다. 또한, 제안 방법은 아키텍처 설계에 필요한 5개의 아키텍처 뷰를 정의하고, 설계과정에서 아키텍처 패턴의 활용하도록 하고 있다. 특히, 이 방법은 아키텍처 설계를 처음 습득하는 설계자들에게 설계방법에 대한 구체적인 이해를 도모한다는 장점을 제공한다.

Abstract The Paper describes the methodology for Software Architecture Design. The key idea is to find the interfaces between the actors, modules, and communicating entities, and use them to identify the software design elements. The identified interfaces and modules are further used to find new modules and interfaces until the every design elements are found and located in the software architecture. This method starts the architecture design with finding the interfaces and enables the natural design procedure by relating the cause and results of the design. It also makes use of not only 5 architectural views for analysis and design of the software, but also concept of architecture patterns in design procedure. Especially, this method is also useful for the novice of the software architecture design.

Key Words : Software Architecture, Software Analysis and Design, Software Architecture Pattern

1. 서론

소프트웨어의 활용이 증가함에 따라, 소프트웨어의 대형화와 소프트웨어 개발의 신속성이 중요한 이슈가 되고 있다. 또한, 최근 소프트웨어에 대한 요구가 소프트웨어를 사용하는 조직의 구조 변경, 업무의 확장 및 변경에 따라 즉각적인 대응을 필요로 한다.

이러한 요구는 안정적이고, 유연성 있는 소프트웨어 아키텍처의 확립을 강조하며, 결국 체계적인 아키텍처를 개발하는 방법론을 도입해야 한다는 결론을 맺게 된다.

논문에서는 먼저 소프트웨어 아키텍처의 개발에 활용되고 있는 잘 알려진 방법론을 소개한다. 그리고 논문에서 제안하는 인터페이스 기반의 아키텍처 설계방법론을 설명한다. 이 과정에서는 설계단계에서 각각의 인터페이스를 식별하고 이 인터페이스를 이용한 설계 방법을 설명한다. 또한 아키텍처를 위한 분석 및 설계단계에서 고려해야 하는 다양한 아키텍처의 관점 즉, 아키텍처 뷰와 그 활용방법에 대해 설명한다.

아울러, 이러한 제안 방법론을 구체적으로 적용하기 위한 사례로 우리에게 친숙한 응용인 기상정보시스템을 이용하여 아키텍처의 설계과정을 적용한다.

끝으로, 제안 방법론을 기존에 제안된 다른 방법론들

*교신저자 : 궁상환(kung@bu.ac.kr)
접수일 10년 10월 08일 수정일 (1차 10년 12월 09일, 2차 10년 12월 16일) 게재확정일 10년 12월 17일

과 비교하여 장점을 살펴보고, 또한 향후 연구의 방향에 대해서도 정리한다.

2. 관련 연구

소프트웨어의 아키텍처를 개발하는 방법에는 응용의 유형이나 규모에 따라 다양한 방법들이 활용되어 왔다. 이러한 방법의 예에는 전통적으로 잘 알려진 DFD(Data Flow Diagram) 기반의 구조적 분석방법(Structured Analysis)이나 유한상태기계(Finite State Machine), 객체모델링기법(Object Modeling Technique), 도메인 분석기법(Domain Analysis) 등이 있다[12].

최근 아키텍처의 중요성이 부각되면서 소프트웨어 아키텍처의 설계에 가장 많이 참조가 되는 방법이라면, 카네기멜론 대학에서 개발한 ADD(Attribute Driven Design)이나 ABD(Architecture Based Design), 그리고 POADM(Pattern Oriented Architecture Design Methodology) 방법을 들 수 있다. 이 방법은 설계요소(Design Element)를 품질속성을 고려하면서 단계적으로 분해하여 가는 설명하고 있다[4,6,7,8,10]. 이 방법은 특별히 설계요소를 분해하면서 분해된 요소들의 배치를 위해 설계패턴 또는 스타일을 참조한다. 이러한 아키텍처 패턴이나 스타일의 정의 및 활용방법에 대한 분야는 별도의 특화된 분야로써 자리 잡아 이미 많이 연구를 통해 아키텍처 설계에 많은 도움을 주고 있다[1,5,9,13]. 그러나 이러한 방법들은 하나의 설계요소가 다른 요소를 포함하는 계층적 관계를 중심으로 한 설계에 치중되고 있다. 즉, 시스템을 절차적으로 분석하여, 특정한 모듈이 왜 필요하게 되는 지에 대해 설명하는 부분은 미흡하다.

한편, 최근 구축되는 많은 비즈니스 응용들은 컴포넌트 개발 방법론(CBD)을 통해 아키텍처의 설계가 수행된다. 이 방법은 2가지 인터페이스를 이용하는 데, 즉 시스템 인터페이스와 비즈니스 인터페이스를 식별하여 컴포넌트의 도출을 진행한다. 그러나 이 방법은 컴포넌트 뷰를 설계하는 데는 유용하지만, 동시성 뷰를 위한 구체적인 방법은 제시하지 못한다.

최근 아키텍처 설계에 있어서의 또 다른 관심은 아키텍처의 표현방법이라고 할 수 있다. 과거에는 많은 아키텍처의 산출물들이 비정형적인 다이어그램을 이용하여 만들어졌지만, 요즈음은 UML(Unified ModelIng Language)을 사용한다[2,3].

아키텍처는 단순히 하나의 관점이 아니라 복수의 관점에서 소프트웨어의 구조를 분석한 결과물이다. 이 중 하나의 관점에서 아키텍처를 설계한 것을 뷰라고 부른다. 즉, 소프트웨어 아키텍처는 복수의 뷰로 표현된 문서라고 할 수 있다. 잘 알려진 아키텍처 뷰는 4+1 뷰라고 불리는 모델이다. 이 4+1 뷰라는 의미는 1개의 사용자 관점과 4개의 설계 관점을 의미한다. 정리하면, 5개의 뷰를 통하여 소프트웨어 아키텍처를 설계한다는 개념을 의미한다[10].

본 연구에서는 이 5개 뷰를 다시 분석에서 필요한 2개의 뷰와 설계에서 필요한 3개의 뷰로 구분하여 다시 제안하고 있다.

3. 인터페이스 기반 아키텍처 설계 방법(IOSAD)

인터페이스와 복수의 뷰를 이용한 아키텍처 설계방법(IOSAD)은 크게 두 가지의 특징을 갖는다. 한 가지는 아키텍처의 설계요소를 식별하기 위해 인터페이스를 사용한다는 것이고, 다른 한 가지는 분석 및 설계과정에 효과적인 몇 가지 뷰를 활용해 나간다는 것이다. 이러한 뷰는 분석 및 설계 과정에서 작성되어 궁극적으로는 최종적인 아키텍처 설계의 결과물이 된다.

3.1 인터페이스를 통한 설계요소의 식별 및 설계

인터페이스의 식별을 통해 설계요소를 도출하하는 방법을 설명하기에 앞서, 먼저 소프트웨어에서의 인터페이스 유형에 대해 살펴보기로 한다. 소프트웨어에서의 인터페이스는 데이터 인터페이스와 콘트롤 인터페이스로 구분해 볼 수 있다.

- 데이터 인터페이스 : 사용자, 통신, 다른 모듈(버퍼), 다른 시스템 등이 있다. 또한 데이터 인터페이스는 단방향 통신과 양방향 통신 등으로 구분된다.
- 콘트롤 인터페이스 : 데이터의 전달보다는 모듈의 통제를 위해 사용된다. 이러한 유형에는 다른 모듈을 실행시키기 위한 Invoke(기동) 인터페이스, 다른 모듈을 접근하여 활용하는 Access(접근) 인터페이스 등이 있다.

이러한 인터페이스를 이용하여 그 유형에 따라 새로운 설계요소(모듈)를 발견하는 기본적인 방법은 다음과 같다.

- 입력정보 : 입력되는 정보를 처리할 모듈을 찾는다.
- 입력파일 : 파일을 생성한 선행모듈을 찾는다.
- 출력파일 : 출력된 파일을 사용할 후속모듈을 찾는다.

- 이외에, 프로세스나 쓰레드를 기동시키는 생성자 모듈을 찾는다.

또한, 식별된 인터페이스를 이용하여 아키텍처를 설계하는 절차는 다음과 같다.

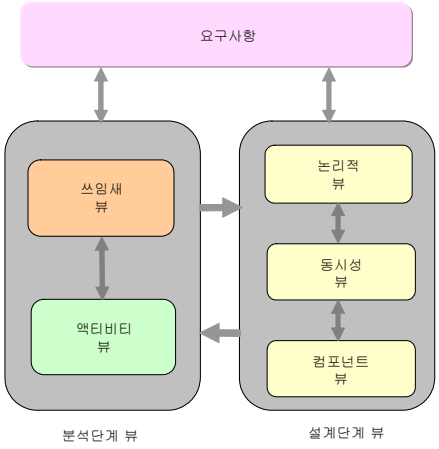
- 중요한 인터페이스를 먼저 식별한다.
  . 사용자 명령이나 외부 입력 메시지 등의 인터페이스를 식별한다.
- 이 인터페이스를 처리할 모듈을 설계한다.
  . 먼저, 이 설계 모듈의 또 다른 입력자원이 있는지 식별한다.
  . 만약 있다면, 이 입력자원에 대한 출력을 식별한다.
  . 이 설계 모듈로부터의 새로운 출력을 식별한다.
  . 이 출력된 자원을 활용할 새로운 모듈을 식별한다.

아키텍처 설계에서 이 인터페이스를 이용하는 방법은 한 마디로 인터페이스를 중심으로 결과와 원인을 파악하여 설계모듈을 식별하는 방식이라고 할 수 있다.

한편, 인터페이스 중심의 설계방식은 실제 설계사례에서는 설계패턴을 이용하는 설계방식과 병행적으로 사용되는 데, 이 설계패턴은 한번에 하나의 모듈이 아니라 복수의 모듈을 동시에 설계에 도입될 수 있도록 해 준다.

3.2 복수의 뷰를 통한 점진적 설계

IOSAD에서 사용하는 뷰는 크게 5가지이다. 시스템의 분석을 위한 뷰로 쓰임새 뷰와 액티비티 뷰가 있으며, 시스템의 설계를 위한 뷰로 논리적 뷰와, 동시성 뷰, 컴포넌트 뷰가 있다. 시스템이 활용하는 프레임워크에 따라서는 특정한 뷰가 생략될 수도 있다.



[그림 1] 5가지 아키텍처 뷰

기본적으로 이러한 뷰는 통상 일정한 순서로 작성이 되지만, 설계의 진행과정에서 각 뷰 간의 동기화 등을 위해서는 필요한 뷰들이 임의적으로 보완 및 설계된다.

3.2.1 시스템 분석 뷰의 작성

1) 쓰임새 뷰(Use Case View)

아키텍처의 설계는 비즈니스 컨텍스트와 사용자 요구사항을 토대로 시스템을 분석하는 과정에서 출발한다. 시스템의 분석은 크게 두 가지 분석을 통해 수행된다. 하나는 사용자 관점에서 기능을 도출하는 것이고, 다른 하나는 시스템의 절차를 분석하는 것이다. 쓰임새 뷰는 사용자 기능을 분해해 가면서, 계층적인 분석을 통해 작성된다. 이 때, 경우에 따라서는 다음 단계의 기능을 직관적으로 분석할 수가 없는 경우도 있다. 따라서 이 과정은 활동 뷰의 분석과 병행적으로 수행하여 보완할 수 가 있다. 이 뷰의 인터페이스에는 시스템 사용자나 시스템 관리자, 또는 다른 시스템 등이 될 수 있다.

2) 활동 뷰(Activity View)

이 뷰는 특정한 하나의 쓰임새 즉, 단위 기능을 분석하는 뷰이다. 따라서 이 뷰의 인터페이스는 각 기능(쓰임새)을 사용하거나 필요로 하는 사람이나 모듈이 인터페이스가 된다. 이 뷰는 절차적으로 작업내용이 분석되며, 마치 구조적 방법론의 DFD(Data Flow Diagram) 작성과정과 유사한 과정이다. 특히, 이 뷰는 쓰임새 뷰에 포함된 더 작은 쓰임새의 도출을 위해 사용되기도 한다.

3.2.2 시스템 설계 뷰의 작성

1) 논리적 뷰의 작성

논리적 뷰의 설계는 크게 2가지 관점에서 이루어진다. 한 가지는 시스템을 논리적 구성요소로 분해하는 것이고, 다른 하나는 이 구성요소간의 접근성을 표시하는 것이다. 시스템의 분해는 수직적 관점과 수평적 관점에서 이루어진다. 수직적 관점의 분해(Layering)는 클라이언트나 표현(presentation) 계층, 비즈니스 계층, 통합 계층, 자원 계층과 같이 상하로 계층을 나누는 것을 말한다. 한편, 수평적 관점의 분해(Partitioning)는 기능의 유사성에 의해 기능을 그룹핑하는 것을 말한다. 비즈니스 채널이나, 정보의 유통 흐름과 같은 것들도 여기에 포함된다.

이와 같이 분해된 구성요소들은 필요에 따라 서로 다른 구성요소를 접근하거나 또는 다른 요소로부터의 접근을 허락한다.

논리적 관점에서 시스템의 분해를 통한 설계를 수행할 때, 기본적인 입력은 시스템의 분석단계에서 작성한 쓰임새 다이어그램이 된다. 즉, 시스템은 서브시스템으로, 그

리고 그 이하의 기능블록들은 단계적으로 쓰임새의 <<include>>와 <<extend>>의 관계를 기초로 식별된다. 이때 쓰임새 뷰에서 다른 작은 쓰임새를 포함하는 모듈은 논리적 뷰에서 상위모듈로, 그리고 포함되는 모듈은 하위모듈로 설계된다.

그러나 논리적 뷰의 초기 설계에서는 동시적인 관점이나 배치적인 관점에서의 고려할 수 없기 때문에, 모든 모듈을 논리적 뷰로 표현하는 것은 불가능하다. 따라서 쓰임새를 기반으로 하여 논리적 모듈들을 구조화 한 후에는, 동시성 뷰와 배치 뷰의 설계에서 발견된 새로운 요소들을 다시 논리적 뷰에 반영해 나가도록 하는 것이 필요하다. 이러한 과정은 일종의 뷰간 동기화(View Synchronization)를 이루는 과정이라고 할 수 있다.

예를 들어, 동시성 뷰는 논리적 뷰에 설계된 모듈이 다시 동작 관점으로 표현하기 때문에, 이 과정에서 새로운 모듈이 식별될 수도 있다. 또한, 기존의 논리적 뷰에서 설계한 모듈을 일부 보완해야 할 경우도 발생하게 된다.

이외에도, 논리적 뷰는 여러 모듈이 공통으로 사용하는 라이브러리 모듈이나 외부 장치와의 인터페이스 모듈, 내부 모듈간의 인터페이스 모듈들이 지속적으로 식별하면서 설계와 조정을 거치게 된다.

2) 동시성 뷰의 작성

동시성 뷰의 설계는 논리적 뷰의 구조를 토대로 출발하게 된다. 다만, 동시성 관점에서 모듈의 구성과 동작을 관찰하여 설계하게 된다. 동시성 관점에서 모듈을 설계하는 과정을 순서적으로 정리해 보면 다음과 같다.

o 중요한 입력정보를 중심으로 설계모듈을 식별한다.
  - 또한, 이 설계모듈을 생성(invoke)시키는 모듈을 식별한다.
    . 필요시 생성모듈을 다시 생성하는 모듈을 식별한다.
    . 이 과정은 최초의(제일 상위의) 모듈이 식별될 때까지 반복된다.
o 동일한 기능을 가진 복수의 입력모듈을 식별한다.
  - 모듈의 기능은 동일하나 분산처리 등의 이유로 실행되어야 하는 모듈이 별도로 필요한 경우를 말한다.
  - 각각의 모듈에 대해 별도의 실시간 실행모듈이 필요하게 된다.
o 입력모듈의 통신패턴을 확인한다. 통신패턴에 따라 실시간 실행모듈이 설계된다.
  - 양방향 통신 : 송신과 수신이 동시에 이루어지는 경우이다. 자료를 특정한 소스로부터 획득하여 지속적으로 송신함과 동시에 수신측으로부터의 송신자료도 송신한다. 예를 들어, 파일전송에서 수신측의 수신확인 메시지를 보내는 경우를 예로 들 수 있다. 이 경우는 자료송신을 위한 실시간 실행모듈 외에도 자료의 수신을 전담할 실시간 실행모듈이 필요하게 된다.
  - 단방향 수신 : 수신을 위한 실시간 실행모듈이 필요하다. 만약, 수신정보를 처리할 모듈이 실시간 실행모듈이라면, 별도로 통신을 위한 실시간 실행모듈은 필요로 하지 않는다.(통신모듈 + 처리모듈)
  - 순차적 응답 : 송신메시지를 수신측에서 수신한 후 그에 대한 처리의 결과를 송신하는 경우이다. 이때는 송신과 수신이 동시에 이루어지지는 않기 때문에 송신 및 수신을 위해 하나의 별도의 실시간 실행모듈만을 필요로 한다.

3) 컴포넌트 뷰의 설계

일반적으로 컴포넌트 설계방법론은 2가지 인터페이스를 통해 컴포넌트를 식별하는 데, 하나는 시스템 인터페이스이며 다른 하나는 비즈니스 인터페이스이다. 시스템 인터페이스란 하나의 쓰임새 당 하나의 인터페이스를 식별하는 것이다. 즉, 유사한 기능을 수행하는 모듈(예: 객체)의 그룹을 컴포넌트로 식별한다. 한편, 비즈니스 인터페이스는 엔티티(핵심 타입) 또는 관련 엔티티의 그룹 당 하나의 인터페이스를 식별한다. 즉, 관련된 정보를 관리하는 기능을 별도의 컴포넌트로 둔다는 것이다. 예를 들어, 특정한 자료나 또는 자료의 집합에 대한 추가, 삭제, 변경, 조회 등이 하나의 컴포넌트에 조합된다.

IOSAD에서는 일반적인 컴포넌트 설계와 같은 방법으로 컴포넌트를 도출하지는 않는다. 왜냐하면, 이미 쓰임새나 활동 뷰의 작성을 통해 컴포넌트의 도출을 수행하였기 때문이다. 따라서 제안에서는 논리적 뷰를 출발점으로 하고, 이 논리적 관점에서의 설계요소를 하나의 컴포넌트로 설계한다. 그런 후, 설계된 컴포넌트가 적합한 컴퓨넌트의 요건을 가지고 있는 지를 분석한다.

콤포넌트의 자격이 될 수 있는 요건을 정리하면 다음과 같다.

- 하나의 단위로 기동된다. (수집 스케쥴러, 기상정보수집, 기상정보저장)
- 하나의 단위로 접근된다. (하나의 인터페이스를 갖는다.)
- 하나의 단위로 재사용된다. (다른 컴포넌트로의 대체의 가능성을 고려한다.
- 하나의 패키지로 관리된다. (동일한 디렉토리에 설치된다.)

- 하나의 단위로 배치된다.

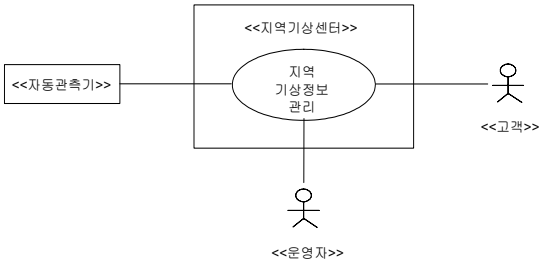
4. 사례 시스템의 설계

제안된 설계방법(IOSAD)을 구체적으로 설명하고, 또한 절차나 방법의 검증을 위하여 지역기상센터의 시스템을 예제로 분석과 설계를 수행하기로 한다.

4.1 요구사항의 정의

4.1.1 비즈니스 컨텍스트(Business Context)

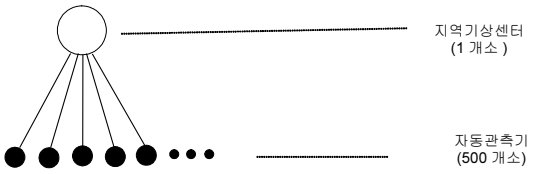
설계할 목표시스템은 특정한 지역(도 또는 도시)의 기상정보를 수집하고 분석하는 기상정보시스템이다. 이 시스템은 일정한 시간 간격별로 자동관측기로부터 기상에 관한 측정정보를 수집하여 DB에 저장한다. 그리고 중요한 경보에 대한 실시간 표시와 기상통계의 분석, 기상예측 무를 수행한다.



[그림 2] 비즈니스 컨텍스트

기상정보의 수집체계와 측정정보의 수집 및 처리방식을 살펴보면 다음과 같다.

o 기상정보시스템의 구성은 다음과 같다.
  - 지역기상센터 : 1개
  - 자동관측기 : 500개
o 자동관측기는 가장 말단의 노드로써 기상정보를 측정하는 역할을 담당한다.
  - 측정되는 정보는 기온, 풍량, 풍속, 강수량, 강설량 등에 대한 정보이다.
  - 이 정보를 수록한 측정 메시지의 크기는 256바이트이다.
o 지역기상센터는 자동관측기에 5분 간격으로 기상정보에 대한 요청(polling) 메시지를 보낸다.
o 자동관측기는 수집된 기상정보를 지역기상센터로 송신한다.
  - 수집정보의 양 : 256Byte/5분 x 500(자동관측기) = 128,000Byte/5분 = 128KB/5분



[그림 3] 기상정보시스템의 체계

4.1.2 아키텍처 요구사항

아키텍처 설계를 위한 요구사항은 크게 기능적 요구사항과 비기능적 요구사항으로 구분된다. 여기서 아키텍처의 설계는 지역기상정보센터와 기상관측소로 국한하고자 한다. 따라서 요구사항도 이 두 구성요소에 대하여 정의한다.

1) 기능적 요구사항
o 관할 구역 내의 자동관측기들로부터 5분마다 측정요청 메시지를 송신하고 측정결과를 수신한다.
o 자동관측기로부터 수집한 기상정보를 DB에 저장한다.
o 정보유형에 따라 정의된 한계 값(threshold)을 초과한 기상정보는 별도의 모니터에 관련 정보(시간, 위치, 관측기, 측정정보)와 함께 출력한다.
o 운영자는 DB로부터 주기별, 유형별 기상통계와 기상예측정보를 확인한다.
o 운영자는 자동관측기의 추가, 삭제, 변경(측정정보의 유형)이 발생하면, 이를 즉시 처리한다.

2) 비기능적 요구사항
o 성능요구사항
  - 기상자료 수집성능
    . 관할 구역 내의 자동관측기로부터 5분 간격으로 기상정보를 수집하고 저장할 수 있어야 한다.
o 변경가능성
  - 자동관측기의 추가 및 삭제가 가능하여야 한다.
    . 이는 시스템의 운영 중에도 가능하여야 한다.

4.2 시스템의 분석 및 설계

4.2.1 시스템의 분석

시스템의 분석을 위해 2가지의 UML 도구를 활용한다. 하나는 시스템의 절차를 분석하기 위한 활동 다이어그램이며, 다른 하나는 사용자 관점의 뷰를 표현하기 위

해 사용하는 쓰임새 다이어그램이다.

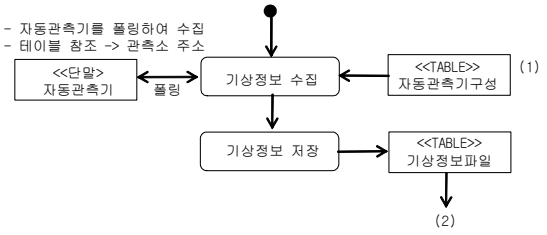
1) 시스템 절차의 분석

지역기상센터의 중요한 인터페이스 중 하나는 자동관측기로부터 전송되는 기상정보를 수신하는 데이터 인터페이스이다.

이 인터페이스는 임의의 모듈 즉, 기상정보수집 모듈로부터 기동되는 폴링과 그에 대한 응답으로 이어진다. 따라서 이 인터페이스는 양방향 통신이 된다. 그러나 요청에 대한 응답한 후에야 비로서 그 다음의 요청처리가 진행된다.

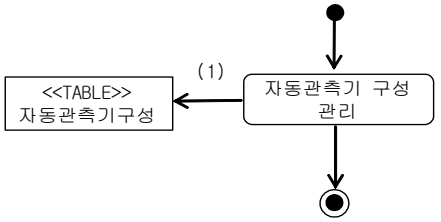
이때, 기상정보수집 모듈이 자동관측기로 정보를 폴링하기 위해서는 자동관측기에 대한 접근정보를 알고 있어야 하는 데, 이를 위하여 자동관측기 구성테이블이 유지되어야 한다.

폴링을 통해 수집된 기상정보는 기상정보파일에 저장되며, 이 자료는 추후 기상정보의 분석에 활용된다.



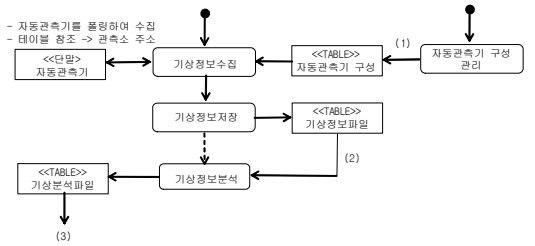
[그림 4] 폴링에 의한 기상정보수집

한편, 앞서 식별한 자동관측기 구성 테이블은 어떤 모듈에 의해 초기화되고, 변경되어야 한다. 여기서, 우리는 이 인터페이스를 담당할 모듈로 기상관측소 구성관리 모듈이 필요함을 발견하게 된다.

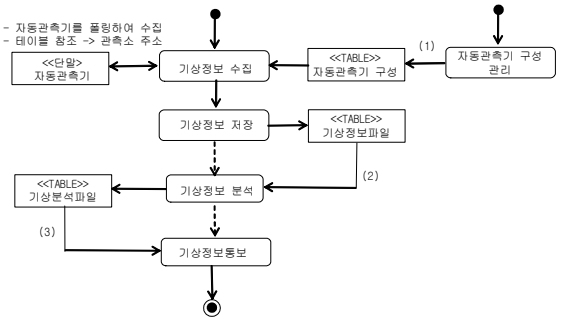


[그림 5] 자동관측기 구성관리 액티비티

또한, 저장된 기상정보는 다양한 통계분석에 이용이 된다. 주별, 월별, 분기별, 연도별과 같은 기간별 기상현황의 분석 뿐 아니라, 기상정보의 유형에 따른 다양한 분석도 가능하다. 물론, 이 정보를 이용하여 미래의 기상을 예측하는 작업도 수행할 수가 있다.



[그림 6] 기상정보분석 액티비티

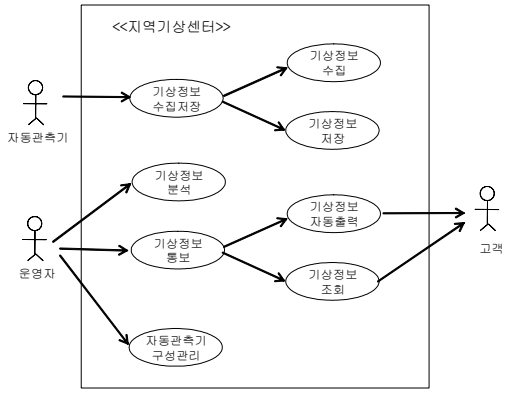


[그림 7] 지역기상센터의 액티비티 다이어그램

2) 시스템 기능의 분석

쓰임새 다이어그램은 지역기상센터의 업무절차를 사용자 관점의 뷰로 표현하는데 이용된다. 이 다이어그램에서는 기능간의 포함관계(<<include>>)를 중요하게 생각한다. 이 포함관계는 기능의 계층구조를 의미하기도 한다.

먼저 외부의 행위자(actor)를 발견하고, 행위자에 의한 시스템의 인터페이스를 중심으로 사용자 관점에서의 시스템 기능을 설계한다. 시스템에 포함된 1차 레벨의 쓰임새는 기상정보 수집저장과 기상정보 분석, 기상정보 통보, 그리고 자동관측기에 관한 정보관리와 같은 쓰임새를 포함한다. 기상정보 수집저장 쓰임새는 자동관측기와 인터페이스하며, 동시에 폴링을 담당하는 기상정보 수집과 이 정보를 DB에 저장하는 기상정보 저장 쓰임새를 포함한다. 기상정보분석은 운영자에 사용되는 쓰임새이며, 수집된 기상정보를 통계와 통보의 목적에 맞도록 가공하는 역할을 수행한다. 한편, 기상정보통보 쓰임새는 이러한 가공정보의 출력을 담당하는 기상정보 자동출력 쓰임새와, 요청에 의한 기상조회 서비스를 제공하는 기상정보 조회 쓰임새로 분해된다.

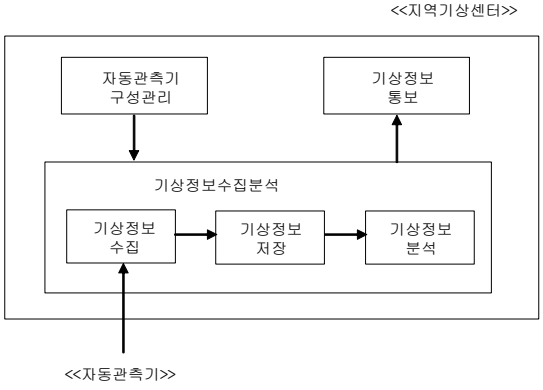


[그림 8] 시스템의 쓰임새 뷰

4.2.2 시스템의 설계

1) 논리적 뷰(Logical View)의 설계

논리적 뷰는 크게 2가지 관점에서의 설계라고 할 수 있는 데, 그것은 수직적 관점의 분해(Layering)와 수평적 관점의 분해(Partitioning)를 말한다. 수직적 관점은 쓰임새 다이어그램의 포함관계로부터 출발한다고 할 수 있다. 이 관계에서 상위계층과 하위계층의 모듈이 서로 구분된다. 하위계층의 모듈은 기능의 차이로 인해 다시 수평적 분해를 유도하며, 다시 다른 모듈로부터의 접근을 허락하여 특정한 서비스를 제공하는 역할을 담당한다.



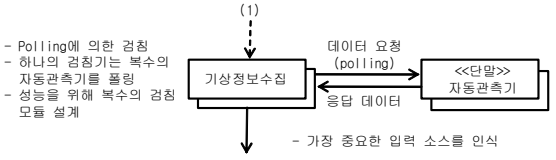
[그림 9] 시스템의 논리적 뷰

2) 동시성 뷰(Concurrency View)의 설계

이 뷰의 목적은 설계 모듈을 컴퓨팅 환경이 제공하는 복수의 CPU에 프로세스나 쓰레드로 분산하여 성능을 개선하는 것이다. 따라서 이 관점에서는 동시에 수행되어야 할 모듈을 파악하고, 이 모듈을 프로세스나 쓰레드로 설계한다. 또한, 하나의 모듈이 외부의 하드웨어나 다른 소프트웨어 모듈에 중복해서 인터페이스하는 모듈을 발견하고, 이 모듈들을 병렬화(프로세스나 쓰레드로 설계)한다. 복수의 정보소스로부터 데이터가 집중화되는 경우에 이 방법은 효과적인 설계결정이 된다.

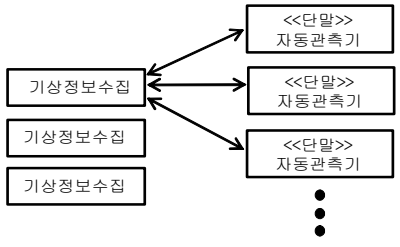
동시성 뷰의 설계를 단계적으로 설명하면 다음과 같다.

o 기상정보검침 모듈의 배치(Thread for Work 패턴)
- 자동관측기로부터 기상정보를 검침하기 위해 사용하는 방법은 폴링(polling)이다. 폴링은 일정한 시간간격별로 자동관측기로부터 측정한 데이터를 수집해 오기 위해 사용된다.



[그림 10] 자동관측기의 기상정보 폴링

- 이때, 하나의 쓰레드가 모든 자동관측기를 대상으로 측정한 정보를 검침해 오도록 설계할 수가 있다. 그러나 모든 자동관측기를 하나의 쓰레드가 폴링을 하게 된다면, 순서적으로 폴링하는 방법 밖에 없고, 이것은 기상정보의 수집시간을 지연시키게 된다.
- 이를 위한 대안은 일정한 자동관측기의 그룹에 대해 한 개의 쓰레드를 할당하는 것이 다. 이 경우, 하나의 검침 쓰레드가 몇 개의 자동관측기를 담당하도록 하느냐가 성능의 관건이 된다.
- 따라서, 자동관측기와 기상정보검침 모듈은 'm대n'의 매칭관계로 구성된다. 즉 일정한 개수를 초과하는 자동관측기에 대해 기상정보검침 모듈은 복수의 쓰레드를 할당하여 효율성을 높인다.

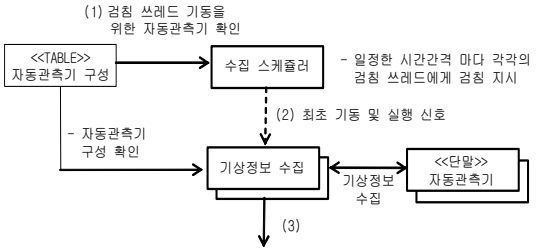


[그림 11] 복수의 기상정보수집 모듈

o 수집 스케쥴러의 설계(참조패턴: Event Notification Pattern)
- 수집 스케쥴러는 일정한 시간 간격(1분 단위)별로,

모든 기상정보검침 모듈에 인터럽트 신호를 송출한다. 이 수집 스케쥴러는 타이머에 의해 지속적으로 운영되고, 또한 별도의 쓰레드로 동작된다.

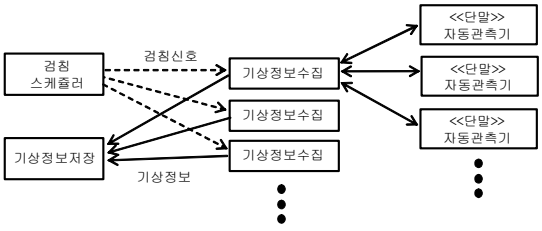
- 기상정보수집 모듈은 인터럽트를 받으면, 기상정보의 수집기능을 수행하지만, 그렇지 않은 경우는 대기(wait) 상태에 있게 된다.
- 한편, 수집 스케쥴러는 자동관측기 구성 테이블의 내용을 토대로, 적절한 수의 기상정보수집 모듈을 기동시킨다.



[그림 12] 수집 스케쥴러의 설계

o 기상정보저장 모듈의 설계(참조패턴: Data Collection Pattern)

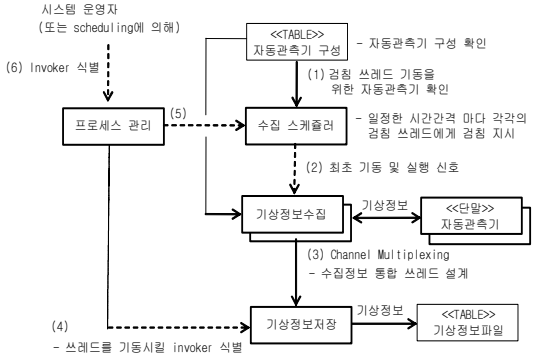
- 기상정보수집 모듈은 복수의 쓰레드로 동작되므로, 각각의 모듈이 수집한 데이터는 다시 하나의 모듈에 의해 통합되어야 한다. 이 자료를 통합하기 위해 read-only 모드로 동작하는 검침정보저장 모듈을 설계한다.
- 이 모듈은 모든 기상정보수집 모듈이 데이터를 전달하므로 많은 부하가 예상되는 모듈이 된다.



[그림 13] 기상정보저장 모듈의 설계

o 수집 스케쥴러, 기상정보수집, 기상정보저장 모듈의 기동(참조패턴: Task Delegation Pattern)

- 기상정보의 수집저장을 위해 운영되는 복수의 쓰레드는 하나의 통제기에 의해 기동되고 통제되어야 한다. 이 역할을 담당하는 모듈이 프로세스관리 쓰레드이다. 이 쓰레드는 다른 쓰레드들의 연관관계를 고려하여 필요한 쓰레드를 순서적으로 기동시킨다.
- 우선, 앞서 설명한 바와 같이, 기상정보수집 모듈은 검침 스케쥴러에 의하여 기동이 된다.
- 기상정보수집 모듈은 다시 기상정보저장 모듈로 검침한 정보를 전달한다. 이를 위해서는 기상정보수집 모듈에게 기상정보저장 모듈의 주소가 미리 통보되어 있어야 한다.
- 따라서, 프로세스관리 모듈은 기상정보저장 모듈을 제일 먼저 기동하고, 검침 스케쥴러가 기동할 때 이 주소를 통보해 준다. 수집 스케쥴러는 다시 기상정보수집 모듈을 기동할 때 이 주소를 통보하여, 수집된 기상정보의 저장을 요청할 때 사용하도록 한다.



[그림 14] 프로세스 관리 모듈의 설계

o 관측기 구성제어(Blackboard pattern)

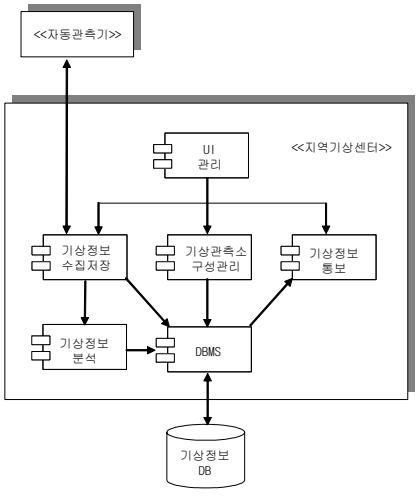
- 자동관측기 정보는 운영자에 의해 직접 관리된다. 각각의 자동관측기 정보는 관측기 구성 테이블에 등록되어 엔트리가 생성되고, 데이터 수집 제어기의 기동시에 참고가 된다.

3) 컴포넌트 뷰(Component View)의 설계

컴포넌트 뷰는 특정한 컴포넌트가 어떠한 컴퓨팅 자원에 탑재되어 실행되느냐를 보여준다. 다양한 사용 환경이 전제가 되는 응용 소프트웨어는 고객의 여건에 따라 복수의 컴퓨팅 환경을 가정해야만 한다. 동일한 소프트웨어 패키지라 할지라도, 고객에 따라 하나의 컴퓨터에 탑재하여 운영하기를 원하는 경우도 있으나, 클라이언트-서버와 같이 분산된 컴퓨터에 탑재하기를 원하는 경우도 있다. 또한, 경우에 따라서는 동일한 소프트웨어를 여러 개의 컴퓨터에 탑재시켜 동시에 처리되는 서비스를 통해 효율성을 높이고자 하는 경우도 있을 것이다.

컴포넌트의 배치를 위해서는 두 가지 고려된다. 하나는 배치를 위한 개별 컴포넌트를 인식하는 것이고, 또 다른 하

나는 이 컴포넌트를 적절한 컴퓨팅 노드에 배치하는 것이다. 배치 뷰의 컴포넌트는 동시성 뷰에 나타나는 컴포넌트에 비해 규모가 크다고 할 수 있다. 이것은 배치관점에서의 컴포넌트는 동작관점에서가 아니라 재사용 관점에서 컴포넌트가 구성되어야 하고, 특히 사용자의 환경에 맞는 패키징 관점에서 컴포넌트를 구성해야 하므로, 보다 큰 컴포넌트가 실용성이 있다. 따라서 배치를 위해서는 먼저 적절한 규모의 컴포넌트로 패키징하는 것이 필요하다.



[그림 15] 시스템의 컴포넌트 뷰

5. 평가 및 결론

본 논문에서는 소프트웨어 설계요소간의 인터페이스와 복수의 아키텍처 뷰를 이용하여 소프트웨어 아키텍처를 효율적으로 설계하는 방법을 다룬다. 논문의 중심 아이디어는 기존의 설계방법의 핵심이라고 할 수 있는 시스템의 분해와 아키텍처 패턴의 적용에 있어서 시스템이 포함하는 중요한 인터페이스를 식별하여 활용하는 것이다. 이 방법은 기본적인 아키텍처 설계의 접근을 인터페이스로부터 출발하여 설계의 원인과 결과가 서로 연관되게 함으로써 자연스러운 설계를 가능하게 한다. 특히, 아키텍처 설계를 처음 습득하는 설계자들에게 설계방법에 대한 구체적인 이해를 도모한다는 장점도 제공한다.

아키텍처 설계 방법론의 평가를 위한 객관적 지표로 활용할 수 있는 직접적인 표준이나 지표는 없으므로, 연구에서는 ISO 9126의 품질모델에서 명시하고 있는 품질 속성의 유형을 아키텍처 방법론의 관점에서 필요한 속성으로 변환하여 사용하고자 한다. 여기서 아키텍처 관점에서 필요한 속성은 기존의 방법론들이 중요한 설계절차의 이슈로 고려하는 사항이며, 이것은 결국은 각 방법론들이 설계의 효율성을 위해 해결해야 할 과제가 된다.

ISO 9126에서는 시스템의 품질을 6가지 특성으로 구분하는 데, 이 특성에는 기능성, 효율성, 신뢰성, 사용용이성, 유지보수성, 이식성이 포함된다. 이러한 특성을 토대로, 제안 방법론의 평가를 위한 기본적인 지표로 기능성(설계절차의 구체성), 효율성(패턴 및 컴포넌트 활용), 신뢰성(뷰의 활용, 뷰간 동기화), 사용용이성(이해성 및 학습성, 모듈식별의 용이성), 유지보수성(방법론의 테일러링), (이식성)적용능력을 설정하고, 이러한 세부지표를 통해 각각의 방법론을 비교, 분석토록 하여 보았다. 최종적인 분석평가의 결과는 표 1에 정리되어 있다.

[표 1] 아키텍처 설계 방법론간 비교

특성	세부 특성	ADD/ ABD	CBD	POADM	IOSAD
기능성	설계절차의 구체성	하(아이디어 레벨)	중(제한된 도메인)	하(아이디어 레벨)	상(모듈에서 모듈 찾기)
효율성	패턴 활용	활용	활용 안함	활용	활용
	컴포넌트 활용	활용 안함	활용 (재사용 개념 제시)	활용 안함	활용 (재사용 개념 없음)
신뢰성	뷰의 활용	3개 뷰 -논리 -동시성 -배치	2개 뷰 -사용자 -컴포넌트	4개 뷰 -사용자 -논리 -동시성 -배치	5개 뷰 -사용자 -활동 -동시성 -컴포넌트
	뷰간 동기화	없음	없음	있음	있음
사용용이성	이해성 및 학습성	하 (많은 경험 요구)	상 (세부절차 있음. 단, 특정 분야에 제한)	하 (세부 절차 없음)	상 (세부절차 있고, 사용용이)
	모듈식별의 용이성	분해 기반	분해기반, 인터페이스 기반(외부 인터페이스, 비즈니스 인터페이스 기반)	분해 기반, 패턴 기반	분해 기반, 패턴 기반, 인터페이스 기반(모든 내부 및 외부 인터페이스 기반)
유지보수성	방법론의 테일러링 능력	하	중	하	상 (응용에 맞추어 뷰와 절차의 조정 가능)
이식성	적용분야	제한 없음	비즈니스 응용	제한 없음	제한 없음

다른 방법론과 비교해서, 제안된 IOSAD가 제시하는 방법론의 특징을 요약해 보면 다음과 같다. 우선 제안 방법론은 아키텍처 설계절차를 매우 상세하게 제시하고 있다. 따라서 이러한 절차를 따를 때 설계가 용이할 뿐 아니라, 안정된 설계결과를 산출할 수가 있다. 또한, 패턴을 활용하여 잘 모르는 분야의 설계도 효율적으로 수행할

수가 있다. 또한, 아키텍처의 분석과 설계과정에서 각각 2개와 3개의 뷰, 총 5개의 뷰를 사용함으로써 다른 방법론에 비해 다양한 관점에서 아키텍처를 정의할 수가 있다. 이것은 제안 방법론이 다른 방법에 비해 보다 높은 신뢰성을 제공한다고도 말할 수가 있겠다. 또 다른 장점은 응용분야에 대한 사전 지식이 없어도 설계가 용이하다는 점과 특히 설계모듈을 식별하는 방법이 다른 방법들에 비해 매우 우수하다는 점을 들 수 있다. 즉, 시스템을 분해하고 설계할 때, 단순한 분해와 아키텍처 패턴을 활용하는 것 외에도, 다양한 인터페이스를 명시적으로 식별하여 설계요소를 확인하는 것이 용이하도록 되어 있다. 한편, 본 제안에서 제시하는 뷰는 응용분야에 따라 생략이 가능하다. 이것은 방법론의 테일러링이 용이하다는 것을 의마한다. 또한, 적용범위에 있어 제한이 없어, 이점은 CBD에 우선한다고 볼 수 있다. 아울러, 대규모의 실시간 시스템이나 또는 임베디드 시스템에 적용하는 데에도 무리가 없다.

한편, 본 제안을 예제 시스템으로 사용한 기상정보시스템의 설계과정에서 다른 방법과 비교하여 어떠한 특징이 있나를 살펴보면 다음과 같다. 우선, 아키텍처의 설계작업은 분석과 설계가 상호 유기적으로 연계되므로, 분석의 결과가 설계의 입력이 되면 더욱 안정성있는 설계가 가능하게 된다. ADD/ABD, CBD, POADM과 같은 경우는 절차적 관점에서 기상정보시스템을 분석하는 과정이 미흡하기 때문에, 쓰임새 뷰를 작성하고 사용자 기능을 분석하였다고 하여도, 도출기능이 정확히 분석되었는 지의 신뢰성이 높을 수가 없다. 그것은 사용자 관점의 쓰임새 뷰는 기능의 분해 만 있지, 활동절차를 분석하지 않기 때문에 명확한 기능을 도출하기가 어렵다.

또한, 기상정보시스템에서는 최초 기상관측기로부터 정보를 수집하는 방법이 시스템의 성능에 중요한 영향을 미치지만, 기존의 CBD 방법과 같은 경우는 동시성 뷰에 대한 언급이 없기 때문에, 기상정보의 수집이나 저장, 그리고 프로세스 관리와 같은 실시간 프로세스의 설계가 불가능하다. 한편, ADD/ABD나 POADM은 절차적이고 상세한 설계방법을 제안하고 있지 않고 있다. 그러나 본 제안에서는 이러한 실시간 모듈 에 대해서도 기상정보의 폴링, 수집, 수집 스케쥴러 등 단계적인 설계를 수행하면서, 최종적으로 프로세스관리 모듈까지를 설계하기 때문에, 최종적인 설계 산출물은 아키텍처의 품질을 크게 향상시켜 준다.

이러한 장점 이외에도, IOSAD의 인터페이스 중심적인 설계방법은 최근 활용이 급격히 증가하는 스마트 폰의 어플개발에도 유용한 도구로 활용될 수 있을 것으로 사료된다.

참고문헌

[1] 궁상환, 품질속성을 고려한 소프트웨어 아키텍처 패턴의 정의, 한국산학기술학회 논문지, 제8권 제1호, 2007. 2.

[2] 궁상환, UML을 응용한 GLORY 소프트웨어 아키텍처의 표현, 제10권 제8호, 2009. 8.

[3] D. Benyon and S. Skidmore, "Towards a Tool Kit for the Systems Analyst", The Computer Journal, Vol.30, No.1, pp.2-7, 1987.

[4] Felix Bachmann, Len Bass, Gay Chastek, Patric Donohoe, Fabio Perzzi, Architecture Based Design Method, *Technical Report CMU/SEI-2000-TR-001*, CMU Software Engineering Institute, 2000.

[5] Frank Buschmann, Regine Meunier, Hans Rohnert, Perter Sommerlad, Michael Stal, Pattern-Oriented Software Architecture Volume 1 : A System of Patterns, John Wiley & Sons, July, 2001.

[6] L. Bass and R. Kazman, "*Architecture- Based Development",* Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-99-TR-007, ESC-TR-99-007, 1999.

[7] Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice*, Addison Wesley, 1998.

[8] Less Bass and Rick Kazman, Architecture-Based Development, *CMU Software Engineering Institute, Technical Report CMU/SEI-99-TR-007*, ESC-TR-99-007, 1999.

[9] Mark Klein and Rick Kazman, Attribute-Based Architecture Style, *Technical Report CMU/SEI-99-TR-022*, CMU Software Engineering Institute, 1999.

[10] Philippe B. Kruchten, "4+1 View model of Architecture", IEEE Software 12(6), pp.42-50, Rational Software(IBM), Nov. 1995.

[11] Rob Wojcik and et al, Attribute-Driven Design(ADD), Version 2.0, *Technical Report CMU/SEI-2006-TR-023*, CMU Software Engineering Institute, 2006.

[12] Robert Krut, Nathan Zalman, "Domain Analysis Workshop Report for the Automated Prompt and Response System Domain", TR CMU/SEI-96-SR-001, CMU/SEI, May 1996.

[13] S. Kung, Pattern Oriented Software Design Approach for USN Middleware, *The International Conference of Ubiquitous Information Technology*, 2007.

궁 상 환(Kung, Sang Hwan) **[정회원]**

- 1977년 2월 : 숭실대학교 전자계산학과 졸업(이학사)
- 1983년 8월 : 고려대학교 경영대학원 전자정보처리학과 졸업(경영학석사)
- 1988년 2월 : 충북대학교 대학원 전자계산학과 졸업(이학박사)
- 1981년 7월 ~ 1998년 2월 : 한국전자통신연구원 책임연구원
- 2001년 3월 ~ 현재 : 백석대학교 정보통신학부 교수

<관심분야>
소프트웨어아키텍처, 분산시스템